Kony Visualizer

API Developers' Guide

Release V8 SP4

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.

© 2013 by Kony, Inc. All rights reserved

Copyright © 2013 Kony, Inc.

All rights reserved.

September, 2021

This document contains information proprietary to Kony, Inc., is bound by the Kony license agreements, and may not be used except in the context of understanding the use and methods of Kony, Inc., software without prior, express, written permission. Kony, Empowering Everywhere, Kony Fabric, Kony Nitro, and Kony Visualizer are trademarks of Kony, Inc. MobileFabric is a registered trademark of Kony, Inc. Microsoft, the Microsoft logo, Internet Explorer, Windows, and Windows Vista are registered trademarks of Microsoft Corporation. Apple, the Apple logo, iTunes, iPhone, iPad, OS X, Objective-C, Safari, Apple Pay, Apple Watch, and Xcode are trademarks or registered trademarks of Apple, Inc. Google, the Google logo, Android, and the Android logo are registered trademarks of Google, Inc. Chrome is a trademark of Google, Inc. BlackBerry, PlayBook, Research in Motion, and RIM are registered trademarks of BlackBerry. SAP® and SAP® Business Suite® are registered trademarks of SAP SE in Germany and in several other countries. All other terms, trademarks, or service marks mentioned in this document have been capitalized and are to be considered the property of their respective owners.

Revision History

Date	Document Version	Description of Releases and Updates
09/13/2021	1.13	Document updated for the V8 Service Pack 4 Fixpack 138 release. The following items have been added/ enhanced: • Added support for the psp parameter in the kony.media.createFromFile and kony.media.createFromUri APIs.

Date	Document Version	Description of Releases and Updates
08/10/2020	1.12	Document updated for V8 SP4 FP98 release. The following items have been added/enhanced: • Added the following APIs as a part of Android TargetSDK API level 29 enhancements: • kony.io.FileSystem.getExternalCacheDir • kony.io.FileSystem.getExternalCacheDirs • kony.io.FileSystem.getExternalFilesDir • kony.io.FileSystem.getExternalFilesDirs • kony.io.FileSystem.getExternalStorageState • kony.io.FileSystem.getFileSystemStats • kony.io.FileSystem.getNoBackupFilesDir • kony.io.FileSystem.isExternalStorageEmulated • kony.io.FileSystem.isExternalStorageLegacy • kony.io.FileSystem.isExternalStorageRemovable • Added support for the HandleRecoverableException parameter in the writeToMediaGallery API. • Added support for the requireBackgroundAccess parameter in the kony.location.getCurrentPosition and kony.location.watchPosition APIs. Additionally, updates have been made to include Android-specific Error Codes.
07/06/2020	1.11	Document updated with the identifierForVendor property of the DeviceInfo object for the V8 SP4 FP 93 release.

Date	Document Version	Description of Releases and Updates
06/22/2020	1.10	Document updated with the cookieFormat parameter in the kony.net.getCookies API for the V8 SP4 FP 90 release.
06/02/2020	1.9	Document updated with the kony.net.setCookies API for the V8 SP3 FP 73 and V8 SP4 FP 83 releases.

10-07-2019	1.8	The following items have been added/enhanced:
		• Added support to the dismissSIPinCallbacks key for the kony.application.setApplicationBehaviors API.
		• Added the following APIs as a part of Android enhancements:
		  ◦ kony.application.startForegroundService
		  ◦ kony.application.updateForegroundNotification
		  ◦ kony.application.stopForegroundService
		  ◦ kony.application.requestPermissionSet
		• Updated the following APIs as a part of Android enhancements:
		  ◦ kony.application.requestPermission
		• Added userDataLocation for kony.application.setApplicationBehaviors API.
		• Added the following APIs as a part of kony.phone namespace.
		  ◦ kony.phone.clearSMSListeners
		  ◦ kony.phone.generateAppHashKey
		  ◦ kony.phone.retrieveSMS

Date	Document Version	Description of Releases and Updates
10-01-2019	1.7	The following item has been added/enhanced: Added support for applicationAppearanceStyle parameter in the kony.application.getSettingValue API, to support the Dark mode feature of iOS 13.
07-22-2019	1.6	Document updated for V8 SP4 FP28 release. The following items have been added/enhanced: • Added support for launching Kony library without UI using the invokeInHeadlessMode API and the kony.application.setLibraryHeadlessModeCallback function.
06-10-2019	1.5	Document updated for V8 June FP release. The following items have been added/enhanced: • Added the supported view types for the Automation API. Added note for Calender, CheckboxGroup, ListBoxGroup, RadioButtonGroup, and SegmentedUI.

Date	Document Version	Description of Releases and Updates
04-25-2019	1.4	Document updated for V8 SP4 FP12 release. The following items have been added/enhanced: • Added the kony.notificationsettings.pickTitleAndDescriptionFromPushPayload API as part of the kony.notificationsettings Namespace. • Added the kony.types.RawBytes.getTempPath API as part of the kony.types Namespace. • Added the kony.os.detectDynamicInstrumentation function as part of the kony.os Namespace. • Added ScrollToWidget and WaitFor for the kony.automation.namespace API.

Date	Document Version	Description of Releases and Updates

02/04/2019	1.3	Document updated for V8 SP4 release. The following items have been added/enhanced:
		• kony.push.setCallbacks
		  ◦ New Callback
		  ◦ Registering the Callback in kony.push.setCallbacks API
		• Added iOS support for these Cryptography APIs:
		  ◦ kony.crypto.asymmetricEncrypt
		  ◦ kony.crypto.asymmetricDecrypt
		  ◦ kony.crypto.generateAsymmetricKeyPair
		  ◦ kony.crypto.retrieveAsymmetricPublicKey
		• Added a new key in kony.contact.add API to store website links in the contact details of a mobile device.
		• Added the kony.notificationsettings.setShowBadge API and overview to enable or disable notification badges for push/remote or local notifications on Android devices.
		• Added the rotate Method for image Object to rotate an image Object by a required degree.
		• Improvements made to the following APIs for the storage of child apps in App Viewer:
		  ◦ kony.i18n Database
		  ◦ Data Store APIs
		  ◦ Local Storage APIs
		  ◦ Network APIs
		  ◦ kony.crypto.readKey
		  ◦ kony.crypto.saveKey

Date	Document Version	Description of Releases and Updates

02/04/2019	1.3	• Added the following APIs to provide users with a safer browsing experience (by using Google Safe Browsing) in Android Browser widget:
		  ◦ kony.ui.BrowserSettings.getSafeBrowsingPrivacyPolicyUrl
		  ◦ kony.ui.BrowserSettings.setOnSafeBrowsingInitializedCallback
		  ◦ kony.ui.BrowserSettings.setSafeBrowsingWhitelist
		• Support of Runtime Permission APIs for all dangerous permissions:
		  ◦ checkPermission API
		  ◦ requestPermission API
		• Updated the kony.map.searchRoutes API.
		• Added the following two APIs to the kony.application Namespace:
		  ◦ addApplicationCallbacks
		  ◦ removeApplicationCallbacks
		• Added the kony.web.WebAuthenticationSession API.
		• Added the following two APIs as part of Desktop Web enhancements:
		  ◦ kony.application.isImageTurnedOff
		  ◦ kony.application.isPopupBlocked
		• Added the isI18nLayoutConfigEnabled parameter for kony.application.setApplicationBehaviors API.
		• Added Desktop Web enhancements for Push Notifications and kony.push Namespace.

Date	Document Version	Description of Releases and Updates
02/04/2019	1.3	• Added the openURLAsync API as part of kony.application Namespace. • Added Desktop Web and SPA support for the deviceID key of DeviceInfo Object. • Added the kony.types.RawBytes.getTempPath as part of the kony.types Namespace. • Added the kony.os.detectDynamicInstrumentation function as part of the kony.os Namespace. • Added a new key in kony.contact.add API to store website links in the contact details of a mobile device. • Added the kony.notificationsettings.setShowBadge API and overview to enable or disable notification badges for push/remote or local notifications on Android devices. • Added the rotate Method for image Object to rotate an image Object by a required degree.

Date	Document Version	Description of Releases and Updates

02/04/2019	1.3	• Added the kony.i18n.setLocaleLayoutConfig API.
		• Added the Communication APIs for React Native App and its associated elements:
		    ◦ kony.reactNative Namespace
		    ◦ React Native APIs
		• Improvements made to the following APIs for the storage of child apps in App Viewer:
		    ◦ kony.i18n Database
		    ◦ Data Store APIs
		    ◦ Local Storage APIs
		    ◦ Network APIs
		    ◦ kony.crypto.readKey
		    ◦ kony.crypto.saveKey

Date	Document Version	Description of Releases and Updates

02/04/2019	1.3	• Added the following APIs to provide users with a safer browsing experience (by using Google Safe Browsing) in Android Browser widget:
		  ◦ kony.ui.BrowserSettings.getSafeBrowsingPrivacyPolicyUrl
		  ◦ kony.ui.BrowserSettings.setOnSafeBrowsingInitializedCallback
		  ◦ kony.ui.BrowserSettings.setSafeBrowsingWhitelist
		• Support of Runtime Permission APIs for all dangerous permissions:
		  ◦ checkPermission API
		  ◦ requestPermission API
		• Updated the kony.map.searchRoutes API.
		• Added the following two APIs to the kony.application Namespace:
		  ◦ addApplicationCallbacks
		  ◦ removeApplicationCallbacks

Date	Document Version	Description of Releases and Updates
02/04/2019	1.3	• Added the kony.web.WebAuthenticationSession API. • Added the following two APIs as part of Desktop Web enhancements: ◦ kony.application.isImageTurnedOff ◦ kony.application.isPopupBlocked • Added the isI18nLayoutConfigEnabled parameter for kony.application.setApplicationBehaviors API. • Added the kony.i18n.setLocaleLayoutConfig API.
02/04/2019	1.3	• Added Desktop Web enhancements for Push Notifications and kony.push Namespace. • Added the Communication APIs for React Native App and its associated elements: ◦ kony.reactNative Namespace ◦ React Native APIs • Added the openURLAsync API as part of kony.application Namespace. • Added Desktop Web and SPA support for the deviceID key of DeviceInfo Object.

Date	Document Version	Description of Releases and Updates
09/24/2018	1.2	Document updated for V8 SP3 release. The following items have been added/enhanced: • FileSystem API • getDataDirectoryPath • getDatabaseDirectoryPath • getSupportDirectoryPath • Communication APIs for Native Library • Initialize Library API • Start Library API • kony.application.sendLibraryResultToNativeApp • kony.application.exitLibrary

Date	Document Version	Description of Releases and Updates
09/24/2018	1.2	• Phone APIs ◦ kony.contact.find ◦ kony.phone.addCalendarEvent ◦ kony.phone.removeCalendarEvent ◦ kony.phone.getRemoveEventOptions ◦ kony.phone.startVibration ◦ kony.phone.cancelVibration ◦ kony.phone.hasVibratorSupport ◦ kony.phone.performHapticFeedback ◦ rawbytes.getRawbytesType API and livePhotoResources Property of kony.phone.openMediaGallery API
09/24/2018	1.2	• Cryptography APIs ◦ kony.crypto.saveKey ◦ kony.crypto.readKey ◦ kony.crypto.generateSecureRandom ◦ kony.crypto.generateAsymmetricKeyPair ◦ kony.crypto.asymmetricEncrypt ◦ kony.crypto.asymmetricDecrypt ◦ kony.crypto.retrieveAsymmetricPublicKey

Date	Document Version	Description of Releases and Updates
09/24/2018	1.2	• Keychain API ◦ kony.keychain.save • Notification Settings API ◦ presentationOptions parameter of kony.notificationsettings.createCategory API • HTTP Integrity Checking • Callback Event APIs for Device Settings Change: kony.application.registerOnSettingsChangeCallback API and kony.application.getSettingValue API • kony.nosql APIs
04/23/2018	1.1	Document updated for V8 SP2 release. The following items have been added/enhanced: • Alert API • Updated the availability of kony.net.setIntegrityCheck function for Windows platform. • Added the postAccessibilityNotification function as part of the kony.application Namespace. • Added Map Styling API. • Added createUUID and getDeviceId APIs as part of the kony.os Namespace.

Date	Document Version	Description of Releases and Updates
04/23/2018	1.1	• Added the isInMultiWindowMode function and the onmultiwindowmodechanged callback (in general and Android-specific callback list) as part of the kony.application Namespace. • Added six new Local Notifications Properties. • Added Payment API, kony.payment Namespace, and its associated functions. • Added Request App Review API and its associated requestReview function. • Updated the Keychain API and kony.keychain functions for Android.
09-19-2018	1.0	Document updated for V8 release. The following items have been added/enhanced: • Updated the createHash function as part of the kony.crypto Namespace. • Updated the remove function as part of the kony.keychain Namespace. • Added success/error codes for all functions of the kony.keychain Namespace. • Added the Application Shortcuts feature as part of ForceTouch API. • Added the disableQuickActionItems, enableQuickActionItems, and getPinnedQuickActionItems functions and updated all other functions of the kony.forcetouch Namespace.

Table of Contents

1. Preface

Kony Visualizer is an *integrated development environment* (IDE) for rapid development and deployment of mobile applications. The Kony Visualizer IDE offers you an easy to learn and use visual development environment to design and develop the mobile applications. Kony Visualizer also provides a group of APIs that you can use to do such tasks as access the device's camera, perform common cryptography tasks, and internationalize your application for multiple locales. Collectively, this group of APIs is known as the Kony Visualizer API and the entire group is documented in this guide.

All the Kony supported devices, development languages, and browsers are listed at, Supported Devices and Browsers.

1.1 Purpose

This document explains the Application Program Interface (API) used in Kony Visualizer and provides information that enables you to use it in your apps.

1.2 Intended Audience

This document is targeted at developers who use Kony Visualizer to develop applications.

1.3 Formatting Conventions

The following are the formatting conventions used throughout the document:

Click here

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

1.4 Related Documents

Document	Purpose
Kony Server Installation Guides	This guide will help you understand the server installation procedure.
Kony Visualizer Installation Guide	This guide will help you understand the Kony Visualizer installation procedure.
Kony Widget User Guide	This guide will help you understand the widgets and their functionality in Kony Visualizer.

1.5 Contact Us

We welcome your feedback on our documentation. Write to us at techpubs@kony.com.

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

API Developers' Guide

Kony Visualizer is available in two editions: Visualizer and Visualizer Classic (formerly Enterprise). This API guide provides documentation for both Kony Visualizer variants.

The Visualizer edition enables you to design, develop, and distribute applications for Native and Web platforms by using low-code tooling. This type of low-code tooling includes the ability to generate and distribute Native and Web applications by using the Cloud Build service and new Enterprise App Store (EAS) features. Visualizer also requires a more lightweight installation, without the heavyweight installation of Eclipse that the Visualizer Classic carries. This makes the Visualizer edition more performant and robust.

The Visualizer Classic edition includes support for legacy/deprecated features as well as additional advanced features used by a much smaller subset of app use cases, such as Cordova SDK integration. For all new users of Kony Quantum and for existing users who want to start new projects, we recommend that you utilize only the Kony Visualizer edition to create your application. You should use the Visualizer Classic edition only if any legacy features are required, as described in this Kony Base Camp article.

For information on revision history, click Revision History.

Kony Visualizer APIs comprises a collection of APIs that you can use in apps written with Kony Visualizer. The APIs enable you to do such tasks as performing various operations on tables, manipulating strings, or invoking a service. The Kony Visualizer API library consists of the following:

- **Accelerometer API**: The Accelerometer APIs allows you to capture the device motion acceleration in X, Y, Z directions and also provide out of the box mechanisms to register for shake gestures event. The chapter comprises of all the APIs with the name **kony.accelerometer**.

- Action Sheet API for iOS: The Action Sheet API provides support for Apple's Action Sheets on iOS apps.
- Alert API: The **kony.ui** namespace includes a function that enables your app to pop up an alert dialog box to display important message to the app's user.
- Animation API: The Animation API functions help your app to add animations to the rows of SegmentedUI widgets.
- App Extension API for iOS: The Kony Visualizer API provides support for App Extensions in iOS apps.
- Application API: During the lifecycle of an application, the mobile device usually triggers several events. The APIs in this chapter allow you to listen to these events and override them with application specific functionality. The chapter comprises all the APIs with the namespace **kony.application**.
- Application Settings API: The Application Settings API enables your apps to control application level settings so that the end users of an application can modify configurations and change the application behavior on iOS.
- Automation API: The Automation API provides you a more convenient way to test your application across various platforms.
- Background Agent API: The Background Agent API enables your Windows app to execute code in the background.
- Badge API: Your apps use the Badge API to display icon badges for iOS apps.
- Battery API: The Battery API provides a standard interface that can be used across multiple hardware platforms for checking the current state of a device's battery.
- Beacon API: The Beacon API enables your app to use iOS beacons.

- Bookmark and Refresh API: The Bookmark and Refresh API enables developers to add context to the URL so that when the end user bookmarks it or shares it, the URL carries necessary parameters so that the application is rendered accordingly.
- Caching API: The Caching API provides a standard interface that can be used on iOS platform to manage network cache.
- Camera API: The Camera API helps your app manage the data captured by the camera widget.
- Charm Setting API: The Charm Setting API provides your app with the ability to access and use Windows charms.
- Client Authentication API: The Client Authentication API provides your apps with the ability to authenticate clients that want to access an HTTPS servers.
- Communication API for React Native App: Communication APIs for React Native allow you to communicate between the ReactNative app and a Kony app.
- Communication API for Native Library: Communication APIs for Native Library allow you to communicate from Native apps to the Kony library. You can use these APIs to launch the Kony library from a Native app.
- Cryptography API: *Cryptography* is the process of securing the information. It can be defined as the conversion of data into scrambled text (concealing its readability and meaning) and deciphering it using a key. This data can be sent across safely over public and private networks. The chapter comprises all the APIs with the namespace **kony.crypto**.
- Drag and Drop API: The Drag and Drop API enables you to share data, such as, images, text (by using JSON), and files, between different apps by dragging items from an app and dropping it into another app.
- ForceTouch API: The ForceTouch API provides functions to support 3D Touch features.
- Functional Modules API: The Functional Modules API enables your app to load functional modules into your app.

- Geolocation API: The GeoLocation API defines a high-level interface to location information, such as latitude and longitude associated with the mobile device.
- Gesture API: The Gesture API gives your app access to the underlying gesture reading capabilities of mobile devices.
- Image API: The Image API provides your app with image processing tools.
- Input and Output API: Use the Input and Output API to access the device's underlying file system.
- Internationalization API: The i18n APIs enable you to design or develop an application in such a way that it supports various languages and regions. The namespace for internationalization is **kony.i18n**.
- Keychain API: The Keychain API provides your app a mechanism to store chunks of user data (such as passwords) in an encrypted database. This credential information is saved in the device's keychain, and your app can retrieve and/ or remove the data if required.
- Local Authentication API: The **kony.localAuthentication** namespace contains functions that enable your apps to do Touch ID biometric fingerprint identification.
- Language API: The Language API is implemented for exceptions that are not handled using try/catch blocks.
- Live Tiles API: Live Tiles enable you to represent an application as a tile on the Start Screen of your device. You can launch the application using a Live Tile. This chapter comprises of all the APIs that enable you to add Live tiles. The namespace of all the APIs in this chapter is **kony.application**.
- Map API: The **kony.map** namespace provides constants and functions that are used in conjunction with the Map widget.
- Map Styling API: The Map Styling API helps provide various custom styling options for maps.
- Math API: The Math API has functions that you can use to perform mathematical operations.
- Media API: The Media API enables your app to play and record audio files.

- Network API: The Network APIs enable you to invoke service calls or cancel network calls. The chapter comprises all the APIs with the namespace **kony.net**.
- Notifications API: The notification system allows users to keep informed about relevant and timely events in your app.
- Offline Data Access API: This API allows you to store data onto the device data store persistently.
- Operating System API: This chapter comprises all the APIs with the namespace **kony.os**.
- Passbook API: The Passbook API is used to keep things like airline boarding passes, movie tickets, and gift cards all in one place.
- Payment API: The Payment API facilitates online transactions in various Kony applications.
- Phone API: The functions in this API provide you the ability to access the default applications of the underlying platform on the mobile device and perform operations. The namespace for Phone API is **kony.phone**.
- Request App Review API: The Request App Review API enables you to ask the users of your app for ratings and written reviews. You can then respond to the provided feedback in order to improve your app’s discoverability, encourage downloads, and build a rapport with the app users.
- Runtime Permission API: With the function in this API, your app can obtain permissions at runtime.
- Shared App Group Container API for iOS: The Shared App Group Container API contains functions that your app can use to share data on iOS. Each app group is associated with a data container that is located in persistent storage outside of the application sandbox of any app on the iOS device.
- Standard Kony API: The generic functions with the namespace **kony**.
- Streaming API: *Streaming* is a feature that is capable of listening to a continuous data stream and use the data. The data is available on Streaming Servers. The common use of streaming is to utilize the multimedia from an external source (like audio or video) in your application.

Streaming data uses different protocols to fetch the data. The chapter comprises all the APIs with the namespace **kony.stream**.

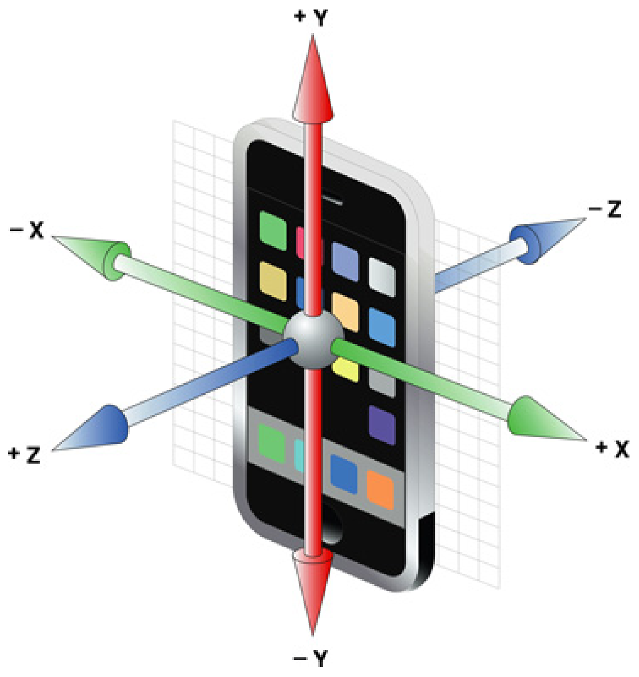
- String API: The string Library has APIs that you can use to manipulate strings. The namespace is **kony.string**.
- Sync API: The Sync API enables your app to perform various operations on Kony Sync.
- Theme API: The Theme API lets you specify common skins for widgets in different states
- Threading API: The Threading API helps JavaScript bindings work on main thread.
- Timer API: The Timer API functions provide you the ability to schedule the execution of a function block at regular intervals. The namespace for Timer API is **kony.timer**.
- Toast API: The Toast API implements toast messages in your Kony Visualizer apps..
- Worker Thread API: The functions in the Worker Thread API provide your apps with a means to execute different tasks in multiple parallel contexts of execution in a concurrent manner.
- File Sharing in Android with other Apps: Android, file sharing from one app with another app happens in the form of content URI. So, you must make your app generate content URIs of files that you want to share with other apps.

2. Accelerometer API

The Accelerometer API allows you to capture the device motion acceleration in 3D space using an X, Y, Z coordinate system, and also provides out-of-the-box mechanisms to register for shake gesture events.

Using the Accelerometer API, you can register various gesture events (such as shake), and then detect the change in orientation of a device along three orthogonal axes x, y, and z.

The following diagram shows the standard coordinate system that the Kony Accelerometer API uses



This coordinate system is independent of any particular type of device. The following table describes the coordinate system to be used for the accelerometer API:

Axis	Description	Value	Direction

X	Floating point value indicating the magnitude of the acceleration force along the X-Axis	positive	Towards the right when the device is facing you
		negative	Towards the left when the device is facing you
Y	Floating point value indicating the magnitude of the acceleration force along the Y-Axis	positive	Towards the top of the top of the device
		negative	Towards the bottom of the device
Z	Floating point value indicating the magnitude of the acceleration force along the Z-Axis	positive	The axis coming out from the device screen (towards you) when the device is facing you
		negative	The axis coming out from the device back (away from you) when the device is facing you

The Accelerometer API uses `kony.accelerometer Namespace` and helps you manage and retrieve data from the device's accelerometer. It comprises of the following API elements.

Function	Description
`kony.accelerometer.registerAccelerationEvents`	Registers event handlers for acceleration events, such as 'shake'.

Function	Description
kony.accelerometer.retrieveCurrentAcceleration	Sets callback functions for retrieving the current device acceleration.
kony.accelerometer.startMonitoringAcceleration	Starts monitoring the device's acceleration on a continuous basis.
kony.accelerometer.stopMonitoringAcceleration	Stops the device monitoring activity if it is active.
kony.accelerometer.unregisterAccelerationEvents	Unregisters event handlers for the specified acceleration event types.

When you use the Acceleometer API, any changes in gestures are registered by invoking the `kony.accelerometer.registerAccelerationEvents` function. After the change is registered, the current acceleration of the device is determined by using the `kony.accelerometer.retrieveCurrentAcceleration` function. Moniter the acceleration of the device by using the `kony.accelerometer.startMonitoringAcceleration` function. To stop monitoring the device, use the `kony.accelerometer.stopMonitoringAcceleration` function. To unregister event handlers for the specified acceleration event types, you can use the `kony.accelerometer.unregisterAccelerationEvents` function.

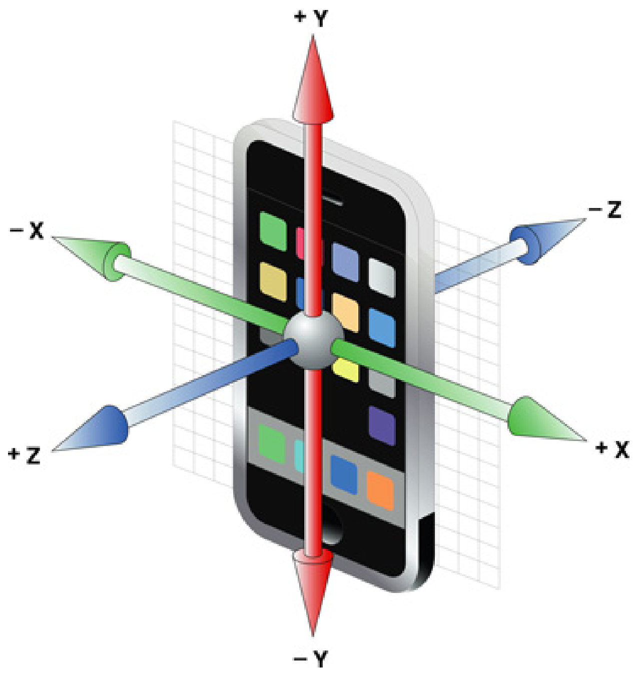
> ***Note:*** The accelerometer APIs are applicable only when `os.platform().hasaccelerometer` returns true.

To view the functionality of the Accelerometer API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

2.1 Overview

The following diagram shows the standard coordinate system that the Kony Accelerometer API uses,



This coordinate system is independent of any particular type of device. The following table describes the coordinate system to be used for the accelerometer API:

Axis	Description	Value	Direction
X	Floating point value indicating the magnitude of the acceleration force along the X-Axis	positive	Towards the right when the device is facing you
		negative	Towards the left when the device is facing you
Y	Floating point value indicating the magnitude of the acceleration force along the Y-Axis	positive	Towards the top of the top of the device
		negative	Towards the bottom of the device

Z	Floating point value indicating the magnitude of the acceleration force along the Z-Axis	positive	The axis coming out from the device screen (towards you) when the device is facing you
		negative	The axis coming out from the device back (away from you) when the device is facing you

X, Y, Z are measured in terms of G, where G denotes the standard gravitational force of the Earth. Therefore, G = 9.80665 m/s^2 (meters per seconds squared). For example, if x=1, its actual value is 1G, i.e., 9.80665 m/s^2. A value of 2 is equal to 2G or 19.6133 m/s^2, and so on.

When a device is laying still with its back on a horizontal surface, each acceleration event has approximately the following values:

- X = 0
- Y = 0
- Z = -1

These values that the device is not undergoing acceleration in the X and Y directions, but the force of gravity, which is equal to -1G, is holding it down on the tabletop. X, Y, Z can also have negative values based on the direction in which the device is positioned.

2.2 kony.accelerometer Namespace

The kony.accelerometer namespace provides the following API elements for managing and retrieving data from the device's accelerometer.

2.2.1 Accelerometer Functions

The kony.accelerometer namespace provides the following functions.

kony.accelerometer.registerAccelerationEvents

Registers event handlers for acceleration events, such as 'shake'.

Syntax

```
kony.accelerometer.registerAccelerationEvents(
    events)
```

Input Parameters

Parameter	Description
events	An object that specifies a hash table containing the events. This table has the key-value pairs that specify the event and the event handler.

Example

```
//Defining the shake event handler function
function onshake() {
    kony.print("Shake called");
}

function registerAccelerationEvents() {

    // Register acceleration events.

    //Define the event object.
    var events = {
        shake: onshake
    };

    //Register the shake event handler function.
    kony.accelerometer.registerAccelerationEvents(events);
}
```

Return Values

None.

Remarks

Event handlers that you set with this function are called asynchronously whenever an accelerometer event occurs. The event handler is triggered only at the end of the event. For example, a `shake` event handler is called after the shaking has stopped, indicating that device motion has occurred. The precision with which a `shake` event can be recognized may vary per platform and depends on the device capability.

The table contained in the *event* parameter has the following format : `{<event>:<event-handler-function>}` where `<event>` is the name of the event and `<event-handler-function>` is the name of the event handler function for that specific event. For example, to set an event handler for the `shake` event, the table would look similar to the following.

```
{shake:onshake}
```

> ***Note:*** Currently, shake is the only event that is supported.

Platform Availability

Available on all platforms except SPA, Desktop Web, Windows 7/Kiosk, and Mobile Web.

kony.accelerometer.retrieveCurrentAcceleration

Sets callback functions for retrieving the current device acceleration.

Syntax

```
kony.accelerometer.retrieveCurrentAcceleration (
    onSuccessCallback,
    onFailureCallback);
```

Input Parameters

Parameter	Description
onSuccessCallback	The callback function that is executed when retrieving the current device acceleration is successful. For details, see the **Remarks** section below.
onFailureCallback	The callback that is executed when an error occurs while retrieving the current device acceleration. For details, see the **Remarks** section below.

Example

Example 1:

```
// Retrieve the current acceleration data

// onsuccesscallback
// This method accepts an 'accelerometerdata' object, which contains the
current device acceleration values
function onsuccesscallback(accelerometerdata) {
    kony.print("X: " + accelerometerdata.x + "Y: " + accelerometerdata.y + "Z:
" + accelerometerdata.z + "Timestamp: " + accelerometerdata.timestamp);
}
// onfailurecallback

function onfailurecallback(error) {
    kony.print("code: " + error.code + "message: " + error.message);
}

function retrieveCurrentAcceleration() {
    // Set the callbacks for getting the acceleration information.
    kony.accelerometer.retrieveCurrentAcceleration(onsuccesscallback,
onfailurecallback);
}
```

Example 2:

```
//Displays the accelerometer data in form frmAclMeter1. This is callback
function is set by the kony.accelerometer.retrievecurrentacceleration function
and invoked automatically by the Kony Visualizer API framework when the
retrieval of the current device acceleration is successful.

function onsuccesscallbackretCurrentAcc(accelerometerdata) {
    frmAclMeter1.lblX.text = accelerometerdata.x;
    frmAclMeter1.lblY.text = accelerometerdata.y;
    frmAclMeter1.lblZ.text = accelerometerdata.z;
    frmAclMeter1.lblT.text = accelerometerdata.timestamp;
}

//To display an error alert if retrievecurrentacceleration fails. This
callback function is set by the accelerometer.retrievecurrentacceleration
function and invoked automatically by the Kony Visualizer API Framework when
the retrieval of the current device acceleration is unsuccessful/failed.

function onfailurecallbackretCurrentAcc(error) {
    alert("Accelerometer is not supported in the device.");
}

//Calls the accelerometer.retrievecurrentacceleration. Function to retrieve
the current device acceleration.
function retrieveCurrentAcceleration()
{
    try
    {
        kony.accelerometer.retrieveCurrentAcceleration(
            onsuccesscallbackretCurrentAcc,
            onfailurecallbackstartmonitoringAcc);
            frmAclMeter1.btnStopAcc.setVisibility(false);
    }
    catch(e)
    {
        alert("Accelerometer not supported.");
    }
}
```

Return Values

None.

Remarks

This function sets twp callback that your app uses for retrieving the current acceleration information from the device. One is called if the acceleration was successfully retrieved, while the other is called if it was not.

The *onSuccessCallback* parameter contains a callback function that is invoked upon success has the following syntax:

```
onSuccessCallback(accelerometerdata);
```

The *accelerometerdata* parameter to the `onSuccessCallback` function is a table containing key-value pairs, as explained in the following.

Key	Type	Description
x	Floating Point Number	The acceleration in the X direction.
y	Floating Point Number	The acceleration in the Y direction.
z	Floating Point Number	The acceleration in the Z direction.
timestamp	Floating Point Number	The number of milli seconds elapsed since the start of the Unix Epoch. The standard Unix Epoch is 00:00:00 UTC on 1 January 1970. The timestamp does not reflect the frequency at which the device can retrieve the accelerometer data, because the device capability (in terms of frequency) can vary from one platform to the other.

The *onFailureCallback* parameter contains callback function that is invoked if an error occurs has the following syntax:

```
onFailureCallback();
```

The `onFailureCallback` function has no parameters. It enables your app to handle the error however you want it to.

The two callback functions are invoked asynchronously and this function returns the value immediately without waiting for actual retrieval of the device acceleration data.

Platform Availability

Available on all platforms except SPA, Desktop Web, Windows 7/Kiosk, and Mobile Web.

kony.accelerometer.startMonitoringAcceleration

Starts monitoring the device's acceleration on a continuous basis.

Syntax

```
kony.accelerometer.startMonitoringAcceleration(
    onSuccessCallback,
    onFailureCallback,
    configData);
```

Input Parameters

Parameter	Description
onSuccessCallback	The callback function that is executed when retrieving the current device acceleration is successful. For details, see the **Remarks** section below.

Parameter	Description
onFailureCallback	The callback that is executed when an error occurs while retrieving the current device acceleration. For details, see the **Remarks** section below.
configData	A JavaScript object that specifies the configuration parameters for the monitoring operation. For details, see the **Remarks** section below.

Example

Example 1:

```
//Start monitoring acceleration

// onSuccessCallback
// This function accepts an 'accelerometerdata' object, which contains the
current device acceleration values
function onSuccessCallback(accelerometerdata) {
    kony.print("X: " + accelerometerdata.x + "Y: " + accelerometerdata.y +
"Z:" + accelerometerdata.z + "Timestamp: " + accelerometerdata.timestamp);
}

//onFailureCallback callback
function onFailureCallback(error) {
    kony.print("code: " + error.code + "message: " + error.message);
}
```

```
function startMonitoringAcceleration() {
    // Start monitor acceleration.
    kony.accelerometer.startMonitoringAcceleration(onSuccessCallback,
onFailureCallback, {
        frequency: 10,
        onchange: false
    });
}
```

Example 2:

```
//To display the accelerometerdata in form frmAclMeter1. This callback
function is set by the kony.accelerometer.startmonitoringacceleration function
and invoked automatically by the Kony Visualizer API Framework when the
retrieval of the current device acceleration is successful and there is a
change in the device acceleration values because the device has moved.

function onsuccesscallbackstartmonitoringAcc(startmonitoringdata) {
    frmAclMeter1.lblX.text = startmonitoringdata.x;
    frmAclMeter1.lblY.text = startmonitoringdata.y;
    frmAclMeter1.lblZ.text = startmonitoringdata.z;
    frmAclMeter1.lblT.text = startmonitoringdata.timestamp;
}

//To display an error alert if startMonitoringAccelerationfails. This callback
function is set by the kony.accelerometer.startMonitoringAcceleration function
and invoked automatically by the Kony Visualizer API Framework when the
retrieval of the current device acceleration is unsuccessful/failed.

function onfailurecallbackstartmonitoringAcc(error) {
    alert("Accelerometer is not supported in the device.");
}

// To call accelerometer.startmonitoringacceleration API to start monitoring
the device acceleration or motion.
```

```
function startmonitoringAcc() {
    try {
        kony.accelerometer.startMonitoringAcceleration(
            onsuccesscallbackstartmonitoringAcc,
            onfailurecallbackstartmonitoringAcc, {
                frequency: 200,
                onChange: true
            });
        frmAclMeter1.btnStopAcc.setVisibility(true);
    } catch (e) {
        alert("Accelerometer is not supported.");
    }
}
```

Return Values

None.

Remarks

By calling this function, your app can start monitoring the device acceleration or motion continuously. When there is a change in the device acceleration values because the device moves, the callback functions passed in through this function's parameters are invoked asynchronously. This function returns immediately without waiting for the device initialization for accelerometer.

The *onSuccessCallback* parameter contains a callback function that is invoked upon success has the following syntax:

```
onSuccessCallback(accelerometerdata);
```

The *accelerometerdata* parameter to the `onSuccessCallback` function is a table containing key-value pairs, as explained in the following.

Key	Type	Description
x	Floating Point Number	The acceleration in the X direction.
y	Floating Point Number	The acceleration in the Y direction.

Key	Type	Description
z	Floating Point Number	The acceleration in the Z direction.
timestamp	Floating Point Number	The number of milli seconds elapsed since the start of the Unix Epoch. The standard Unix Epoch is 00:00:00 UTC on 1 January 1970. The timestamp does not reflect the frequency at which the device can retrieve the accelerometer data, because the device capability (in terms of frequency) can vary from one platform to the other.

The callback function that is invoked if an error occurs has the following syntax:

```
onFailureCallback();
```

The `onFailureCallback` function has no parameters. It enables your app to handle the error however you want it to.

The two callback functions are invoked asynchronously and this function returns the value immediately without waiting for actual retrieval of the device acceleration data.

When your app invokes the `kony.accelerometer.startMonitoringAcceleration` function, the third parameter that your app must pass is *configData*. The *configData* parameter contains a JavaScript object that holds a set of key-value pairs that must be in the following format.

Key	Type	Description
frequency	Floating Point Number	The time interval, in milliseconds, in which accelerometer data needs to be retrieved. The default value of `frequency` must be "200" milliseconds minimum. Any negative value specified in the frequency reverts to the default value i.e., 200ms.
onchange	Boolean	A value that determines whether or not to trigger an event whenever the device is moving regardless of the value specified in `frequency`. If `onchange` is set to `true`, the number set in *frequency* is not respected and the *onSuccessCallback* event is invoked whenever the device is in motion. If this value is set to *false*, the *onSuccessCallback* event is invoked in the time interval specified in the frequency parameter is used. The default value for `onchange` is *false*.

> ***Note:*** If you set `onchange` to `false`, it is necessary to specify a `frequency` value or the *onSuccessCallback* function is never invoked.

Platform Availability

Available on all platforms except SPA, Desktop Web, Windows 7/Kiosk, and Mobile Web.

kony.accelerometer.stopMonitoringAcceleration

Stops the device monitoring activity if it is active.

Syntax

```
kony.accelerometer.stopMonitoringAcceleration();
```

Example

```
function stopMonitoringAcceleration() {
    // Stop the device monitoring activity if it is active.
    kony.accelerometer.stopMonitoringAcceleration();
}
```

Input Parameters

None.

Return Values

None.

Remarks

If your app has been continuously monitoring the device's motion, it calls the `kony.accelerometer.stopMonitoringAcceleration` to stop. Apps can start monitoring the device motion using the accelerometer.startmonitoringacceleration function.

Platform Availability

Available on all platforms except SPA, Desktop Web, Windows 7/Kiosk, and Mobile Web.

kony.accelerometer.unregisterAccelerationEvents

Unregisters event handlers for the specified acceleration event types.

Syntax

```
kony.accelerometer.unregisterAccelerationEvents(
    eventTypes);
```

Input Parameters

Parameter	Description
eventTypes	An array of events.

Example

```
function unregisteraccelerationevents() {
    // Unregister for acceleration events.
    kony.accelerometer.unregisteraccelerationevents({
        "shake"
    });
}
```

Return Values

None.

Remarks

After this function returns, the event specified in the *eventTypes* parameter no longer have event handlers registered for them. As a result, your app no longer receives notifications of those events.

Platform Availability

Available on all platforms except SPA, Desktop Web, Windows 7/Kiosk, and Mobile Web.

3. Action Sheet API for iOS

Action sheet is a pop-up menu that consists of a list of options for a user to complete an action. It can also be used for notification dialog boxes or alert boxes. The Action Sheet API provides support for Apple's Action Sheets on iOS apps.

To implement an Action Sheet, you must create an ActionSheet object by using the kony.ui.ActionSheet function. To add an item to the Action Sheet, you must create an `ActionItem` object by invoking the kony.ui.ActionItem function. Each `ActionItem` object has a callback function that is automatically invoked whenever the user taps the Action Sheet choice represented by the `ActionItem`. The callback function processes the user's input as needed.

The process of implementing an Action Sheet involves two steps:

- Creating an **ActionSheet** object.
- Adding an item to the Action Sheet by creating an **ActionItem** object.

The Action Sheet API for iOS uses `kony.ui Namespace` and the following API elements.

Function	Description
`kony.ui.ActionItem`	Constructs an ActionItem object for use in an ActionSheet object.
`kony.ui.ActionSheet`	Constructs an ActionSheet object that represents an iOS Action Sheet.

- ActionItem Object

Property	Description
`enable`	Enables or disables the ActionItem object.

- ActionSheet Object

Method	Description
`addAction`	Adds an ActionItem object to the ActionSheet object.
`setAnchorConfiguration`	Sets the anchor configuration information on iPads.
`show`	Shows the Action Sheet on the display.

To create an **ActionSheet** object, use the `kony.ui.ActionSheet` function. Then add an item to the Action Sheet, create an **ActionItem** object by invoking the `kony.ui.ActionItem` function. After you add an action item, the action sheet can be displayed by using the `show` method. Each **ActionItem** object has a callback function that is automatically invoked whenever the user taps the Action Sheet choice represented by the **ActionItem**. The callback function processes the user's input as needed.

To view the functionality of the Action Sheet API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

3.1 ActionItem Object

An `ActionItem`object is used together with the `ActionSheet` object to implement Action Sheets on iOS. It functions as a menu item in a group of menu items.

3.1.1 Overview

You create an `ActionItem` object by using the kony.ui.ActionItem function. At the time you create the `ActionItem` object, you specify a callback function that is automatically invoked when the user selects the particular `ActionItem`.

After you call the `kony.ui.ActionItem` function and instantiate an `ActionItem` object, you can enable or disable it with the `ActionItem` object's enable property.

3.1.2 Properties

The `ActionItem` object provides the following properties.

enable

Enables or disables the `ActionItem` object.

Syntax

```
enable
```

Example

```
// Disable an action item.
myActionItem.enable = false;
```

Type

Boolean

Read/Write

Read+Write

Remarks

Set this property to `true` to enable the `ActionItem` object. To disable it, set this property to `false`.

Platform Availability

iOS

3.2 kony.ui.ActionItem Function

The details of the kony.ui.ActionItem function, which is part of the kony.ui Namespace, are as follows.

Constructs an `ActionItem` object for use in an ActionSheet object.

Syntax

```
new kony.ui.ActionItem(actionItemParams)
```

Input Parameters

actionItemParams

A JavaScript object containing key-value pairs that define the configuration parameters for the action item. This object must contain the following keys.

Constant	Description
title	A string that specifies the title for the action item.
style	A value from the Action Item Style Constants that selects the style of the action item.
actionCallback	A JavaScript function that handles user selections from the action item. For more information, see **Remarks** below.

Example

```
var actionItem1 = new kony.ui.ActionItem({
    "title": "alert title",
    "style": constants.ACTION_STYLE_DEFAULT,
    "action": myActionFunction
});
```

```
//Creating the Action Item Object
  setActionSheet: function(){
    var actionItem = new kony.ui.ActionItem({
    "title": "Open Basecamp",
    "style": constants.ACTION_STYLE_DEFAULT,
    "action": function(){
     kony.application.openURL("https://basecamp.kony.com/s/");
    }
```

Return Values

Returns an `ActionItem` object that can be added to an Action Sheet.

Remarks

The `actionCallback` function, which is passed into this function through the *actionSheetParams* parameter, must have the following prototype.

```
actionItemCallback(actionSheetObject, actionItem1);
```

where actionSheetObject is a handle to the `ActionSheet` object that the `ActionItem` object is associated with, and *actionItem1* is a handle to the `ActionItem` object that the user selected.

Platform Availability

iOS

3.3 ActionSheet Object

The ActionSheet object implements Apple's Action Sheets for iOS apps. They are not supported on other platforms. The ActionSheet object comprises of the following elements:

3.3.1 Methods

The `ActionSheet` object contains the following methods.

addAction

Adds an `ActionItem` object to the `ActionSheet` object.

Syntax

```
addAction(
    actionItem1)
```

Parameters

Parameter	Description
actionItem1	An `ActionItem` object to add to the Action Sheet.

Example

```
var actionItem1 = new kony.ui.ActionItem({
    "title": "alert title",
    "style": constants.ACTION_STYLE_DEFAULT,
    "action": myActionFunction
});
actionSheetObject.addAction(actionItem1);
```

```
//Creating the Action Item Object
setActionSheet: function() {
    var actionItem = new kony.ui.ActionItem({
        "title": "Open Basecamp",
        "style": constants.ACTION_STYLE_DEFAULT,
        "action": function() {
            kony.application.openURL("https://basecamp.kony.com/s/");
        }
    });
    //Adding action to the Action Sheet object
    actionSheetObject.addAction(actionItem);
}
```

Return Values

None.

Platform Availability

iOS only

setAnchorConfiguration

Sets the anchor configuration information on iPads.

Syntax

```
setAnchorConfiguration(
    configParams)
```

Parameters

Parameter	Description
configParams	A JavaScript object containing key-value pairs that specify the anchor configuration parameters for the Action Sheet. The following keys are required. <ul><li>`direction` : A constant from the Action Sheet Anchor Direction Constants that specifies the side of the widget that the Action Sheet attaches to.</li><li>`widget`: The widget that the Action Sheet attaches to.</li></ul>

Example

```
var configInfo = {
    "direction": constants.ANCHOR_DIRECTION_LEFT,
    "widget": frmWidgetName
};
myActionSheet.setAnchorConfiguration(configInfo);
```

Return Values

None.

Remarks

This method is only used on the iPad.

Platform Availability

iPad only

show

Shows the Action Sheet on the display.

Syntax

```
show();
```

Example

```
actionSheetObject.show();
```

Parameters

None.

Return Values

None.

Platform Availability

iOS only

3.4 kony.ui.ActionSheet Function

The details of the kony.ui.ActionSheet function, which is part of the kony.ui Namespace, are as follows.

kony.ui.ActionSheet

Constructs an `ActionSheet` object that represents an iOS Action Sheet.

Syntax

```
kony.ui.ActionSheet(actionSheetParams)
```

Input Parameters

actionSheetParams

A JavaScript object containing key-value pairs that define the configuration parameters for the Action Sheet. This object must contain the following keys.

Key	Description
title	A string that specifies the title for the Action Sheet.
message	A string containing the action sheet's message to display to the user.
showCompletionCallback	A callback function that is invoked after the display of the action sheet. For details, see the **Remarks** section below.

Example

```
var actionSheetObject = new kony.ui.ActionSheet({
        "title":"some title",
        "message":"some message",
        "showCompletionCallback": myCompletionCallback
    });
```

```
//Creating the Action Sheet Object
  var actionSheetObject = new kony.ui.ActionSheet({
        "title":"Kony Basecamp",
        "message":"Welcome to Kony Base Camp! Explore. Learn. Develop.
Share.",
        "showCompletionCallback": function(){
        }
    });
```

Return Values

Returns an `ActionSheet` object.

Remarks

The *actionSheetParams* parameter is an object containing key-value pairs. When your app uses the `showCompletionCallback` key, it specifies a callback function that is automatically invoked after your app displays the action sheet. The callback function must have the following signature.

```
showCompletionCallback();
```

In an Action Sheet, only one action item can have the style `constants.ACTION_ITEM_STYLE_CANCEL`.

Platform Availability

iOS

4. Alert API

Kony Alert API enables you to configure and send alerts to an application. Alert APIs draw the attention of the user to take some action or to provide the user some information.

There are three types of Alert messages:

- **Confirmation** type alerts display confirmation messages with Yes and No options on the screen.
- **Info** type alerts display informative messages on the screen. This message can in turn be a warning or a success message.
- **Error** type alerts display error messages on the screen.

When alerts appear in an app, the user cannot proceed with other UI operations unless they dismiss the alert.

When an alert appears, the user must take some action on the alert. The user cannot proceed with any other operation without responding to the alert first. Alert API uses `kony.ui Namespace` and contains the following function.

Function	Description
`kony.ui.Alert`	Provides you the ability to add alerts in the application.

To view the functionality of the Alert API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

4.1 kony.ui.Alert Function

The details of the kony.ui.Alert function, which is part of the kony.ui Namespace, are as follows.

kony.ui.Alert

This API provides you the ability to add alerts in the application. The alerts are of the following types:

- **information** - an informative message is displayed on the screen. This message can be in turn a *warning* or a *success* message.
- **confirmation** - a confirmation message with **Yes** and **No** options is displayed on the screen.
- **error** - an error message is displayed on the screen.

All the alerts are modal in nature, i.e., the user cannot proceed with other UI operations unless the alert is dismissed.

Syntax

```
kony.ui.Alert(basicConfig, pspConfig)
```

Input Parameters

basicConfig

basicConfig is an object with the following configuration properties.

Property	Description
message [String] - Mandatory	The message to be shown when an alert is thrown.
alertType [Number] - Mandatory	Denotes the type of the alert. The possible values are as follows: ◦ ALERT_TYPE_CONFIRMATION ◦ ALERT_TYPE_ERROR ◦ ALERT_TYPE_INFO
alertTitle [String] - Optional	Title of the alert.
yesLabel [String] - Optional	Text to be displayed for the Yes label. If the text for the Yes label is not provided, individual platforms display default values.
noLabel [String] - Optional	Text to be displayed for the No label. If the text for the No label is not provided, individual platforms display default values.
alertIcon [String / image Object] - Optional	Icon to be displayed to visually indicate the type of alert, such as, Info, Error, Confirmation. This parameter is not supported on iPhone. You can create an image Object by using kony.image Namespace functions.
alertHandler [Read / Write Event] - Mandatory	JavaScript function that should get called when alert is dismissed either through "yes" label button or through "no" label button.

pspConfig

pspConfig is an object with platform specific configuration properties.

Property	Description

ondeviceback [Write Event] - Optional	JavaScript function that should get called when alert is open and the device back button is pressed.
	
	Note: Supported on Windows Phone 8, Windows Phone 7.5 (Mango) channels and not supported on Windows Kiosk and Windows 8 channels.
	
	`var pspConf = {`
	`    ondeviceback: func1`
	`};`
	`var confirmationAlert =`
	`kony.ui.Alert(basicConf,`
	`pspConf);`
	
	`function func1() {`
	`    kony.print("Example`
	`function on device back");`
	`}`
	
	Note: The configuration properties should be passed only in the respective configuration objects otherwise they are ignored.

Property	Description
contentAlignment	• Used to align content of an alert. Following are the values of this property: ◦ constants.ALERT_ CONTENT_ALIGN_LEFT // default ◦ constants.ALERT_ CONTENT_ALIGN_ CENTER ◦ constants.ALERT_ CONTENT_ALIGN_RIGHT
iconPosition	It is used to align and alert title icon. Following are the values of this property: ◦ constants.ALERT_ICON_ POSITION_LEFT // default ◦ constants.ALERT_ICON_ POSITION_RIGHT ***Note:*** Refer the example given below to create an alert using `contentAlignment` and `iconPostion` parameters.

Example

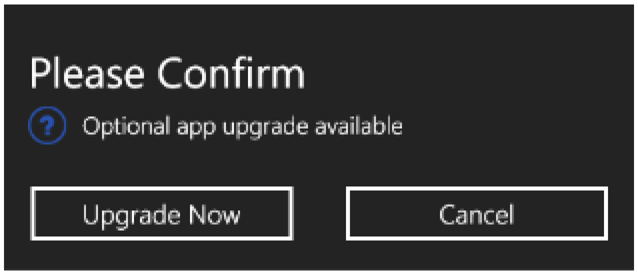
```
//Defining basicConf parameter for alert
var basicConf = {
    message: "This is an info alert",
    alertType: constants.
    ALERT_TYPE_INFO,
    alertTitle: "infoWithoutImage",
    yesLabel: "yes",
    noLabel: "no",
    alertHandler: handle2
};

//Defining pspConf parameter for alert
var pspConf = {};


//Alert definition
var infoAlert = kony.ui.Alert(basicConf, pspConf);

function handle2(response) {
    frm6.show();
}
```

```
var pspConfig = {
    "iconPosition": constants.ALERT_CONTENT_ALIGN_CENTER,
    "contentAlignment": constants.ALERT_ICON_POSITION_LEFT
};

var alert = kony.ui.Alert({
        "message": "Hello Alert",
        "alertType": constants.ALERT_TYPE_ERROR,
        "alertTitle": "I'm Alert",
        "yesLabel": "Yes",
        "noLabel": "",
        "alertIcon": "",
        "alertHandler": null
    },
    pspConfig);
```

```
  confirmationAlert: function(){
   //Creating the basicConfig object
  var basicConf = {
    message: "This is an confirmation alert",
    alertType: constants.ALERT_TYPE_CONFIRMATION,

};
   //Creating the pspConfig object
  var pspConfig = {
    "contentAlignment": constants.ALERT_CONTENT_ALIGN_CENTER
};
   kony.ui.Alert(basicConf, pspConfig);
 },

   informationAlert: function(){
     //Creating the basicConfig object
  var basicConf = {
    message: "This is an info alert",
    alertType: constants.ALERT_TYPE_INFO,

};
     //Creating the pspConfig object
  var pspConfig = {
  "contentAlignment": constants.ALERT_CONTENT_ALIGN_LEFT
};
   kony.ui.Alert(basicConf, pspConfig);
 },

   errorAlert: function(){
   //Creating the basicConfig object
  var basicConf = {
    message: "This is an error alert",
    alertType: constants.ALERT_TYPE_ERROR,

};
     //Creating the pspConfig object
  var pspConfig = {
  "contentAlignment": constants.ALERT_CONTENT_ALIGN_RIGHT
};
```

```
  kony.ui.Alert(basicConf, pspConfig);
},
```

Alerts are displayed on some platforms as follows:

Windows Phone Mango	
 Please Confirm Optional app upgrade available	

Return Values

None.

Remarks

Invoking this API multiple times in the same action sequence leads to an erroneous behavior.

This API should be invoked at the end of a function as a best practice.

The following are the behavioral aspects of alerts on various platforms:

RichClient

In all native implementations alert is non blocking, i.e. the execution of any logic defined after the alert definition continues without the alert confirmation.

> ***Important:*** Alert images are not supported on Windows 8 tablet.

Android

Android platform supports display of multiple alerts each time the **kony.ui.Alert** API is invoked. On device back, the alert gets dismissed and also the alert handler is raised.

For CONFIRMATION type alert, the alert callback is invoked with cancel flag. For example, false Boolean argument.

For INFO & ERROR type alert, the alert callback is invoked with true argument.

iPhone

iPhone does not support displaying image icons based on the alert types: info,confirmation, error. *"\n"* as a newline character in the alert messages supported for iPhone Platform

Mobile Web/SPA/Desktop Web

- For advanced Mobile Web devices like iPhone, Android, and Palm Pre, alerts are displayed as popups. For basic devices, the alerts are displayed in a new page.
- For all platforms you cannot customize alert icons/yes or no labels/ look and feel.

- Titles of the alerts are provided by the browser and you cannot modify them. The alert title attribute does not apply for Mobile Web. Usually, the title on the alert will be the IP address or the domain name of the application.
- Confirmation alerts in the basic devices will be displayed in another form.
- If the alert message is *nil*, alert is not displayed.
- In case of all platforms, the execution of the logic defined after alert is blocked until the user clicks "yes" or "no".

For JavaScript conversion, an alert has to be created using another variant constructor,i.e an *Indexed argument* constructor.

```
kony.ui.Alert(message, alertHandler, alertType, yesLabel, noLabel,
alertTitle, pspConf);
```

Platform Availability

Available on all platforms.

5. Animation API

5.1 Overview

These functions create animations and transforms that are then passed to the methods of widgets that are capable of performing animations. For example, the SegmentedUI widget methods addAll, addDataAt, and addSectionAt all perform animations on UI elements in the rows.

In addition, you can use the animation API to do 3D transformations and animations on all widgets that support animations. To do so, your app creates a transform object, call the appropriate transformation methods to set the transformation's properties, and then store the transform object into the transform property of the widget you want to perform the transformation on.

Using the Animation API, you can create animations and 3D transformations on widgets that support animations. You can transform objects, define animations, and configure various properties of an animation. To associate an animation to a widget, the animation configuration object, the animation definition object, and the callbacks are passed to the methods of the widgets such as **addAll**, **addDataAt**, and **addSectionAt**.

The Animation API contains the following Namespaces and API elements:

AnimationConfiguration Object

Key	Description
`delay`	This key defines when the animation will start.

Key	Description
`direction`	This key defines whether the animation must play in reverse on some or all cycles.
`duration`	This key defines the time in seconds that an animation takes to complete one cycle.
`fillMode`	This key defines what values are applied to the widget state by the animation outside the time it is executing.

Key	Description
`iterationCount`	This key specifies the number of times an animation cycle is played.

kony.ui Namespace

Function	Description
`kony.ui.createAnimation`	Creates an object that defines an animation.
`kony.ui.makeAffineTransform`	Creates a transformation object that can be used in an animation definition.

kony.anim Namespace

Constant	Description
Animation Effect Constants	These constants are used to select what type of animation will take place.
Animation Fill Mode Constants	Specifies the fill mode being used when performing widget animations.

Transform Object

Method	Description
rotate	Returns an affine transformation matrix constructed by rotating receivers affine transform.
rotate3D	Rotates the widget by angle on the unit directional vector formed by rx, ry, and rz.
scale	Returns an affine transformation matrix constructed by scaling receivers affine transform.

Method	Description
`scale3D`	Scales a widget in three dimensions (x, y, z) coordinate system.
`setPerspective`	Sets the perspective and sets the vanishing point at the center of the widget.
`translate`	Returns an affine transformation matrix constructed by translating receivers affine transform.
`translate3D`	Translates the widget from present location to new location by x, y, z amount.

Transform Objects: Create a transform object by using the `kony.ui.makeAffineTransform` function. Using the transform object, you can set 2D and 3D transformations to a widget. You can rotate the widget using `rotate` and `rotate3D` methods; scale the widget using `scale` and `scale3D` methods; and translate the widget using `translate` and `translate3D` method. You can also set the perspective and the vanishing point using the `setPerspective` method.

Animation Definition Object: Create an animation definition object by using the `kony.ui.createAnimation` function. The animation definition object defines the state of the widget at any specified point of time. Then configure properties of the animation by using the animation configuration object.

Animation Configuration Object: The animation configuration object has various key-value pairs that determine properties of the animation. You can set the animation `delay`, `duration`, `direction`, `iterationCount`, and the `fillMode` properties that determine the state of the widget at the end of the animation.

To view the functionality of the Animation API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

5.2 AnimationDefinition Object

The `AnimationDefinition` object contains key-value pairs and defines the state of a widget at any specified point of time. You can perform transformation, rotation, and scaling operations on widgets to move them, rotate them, or make them larger or smaller.

5.2.1 Key Frame Animation

Using the Key frame animation, you can specify the transformations that must happen at specified locations. The Key frame definition consists of steps (key frames) that contain properties of a widget along with their values at specified points of time. Each step from initial value to the final value can be configured with an animation behavior.

For example, you could specify that a Label widget moves +5 in the y direction, then move -6 in the x direction, and then rotate 90 degrees. Each of these three steps can be performed by specifying three key frames.

When your app performs Key frame animation, it must define the key frames to use in an animation definition object. The animation definition object is a JavaScript object that your app creates and passes to the `kony.ui.createAnimation` function.

Following is the syntax of a Key frame:

```
{ <step>:
{ <widget_property> : <value> ,
 <widget_property> : <value> , …
stepConfig : { … } //optional parameter

}
```

Example

```
var animDefinition = {
    "0": {
        "top": "0dp",
        "left": "0dp"
    },
    "100": {
        "top": "50dp",
        "left": "50dp"
    }
};
animDef = kony.ui.createAnimation(animDefinition);
```

For more information on Key frame animation, refer Widget Animation Using Flex forms.

5.3 AnimationConfiguration Object

The `AnimationConfiguration` object is a JavaScript object that your app builds to perform animations on widgets. The `AnimationConfiguration` object contains key-value pairs. The following keys are supported.

delay

This key defines when the animation will start. It allows an animation to start executing after it is applied. This is specified in seconds and fractional values are allowed.

A delay value of zero (0) means the animation will execute as soon as it is applied. Otherwise, the value specifies an offset from the moment the animation is applied, and the animation will delay execution by that offset. The default value is zero and any negative or invalid values will default this property to zero.

This is used for the overall widget animation configuration and not the step-level configuration.

direction

This key defines whether the animation must play in reverse on some or all cycles. If an animation is played in reverse, the timing functions are also reversed. For example, when played in reverse an ease-in animation would appear to be an ease-out animation.

Following are the possible predefined values:

- kony.anim.DIRECTION_NONE (default)

All iterations of the animation are played as specified.

- kony.anim.DIRECTION_ALTERNATE

The animation cycle iterations that have odd counts are played in the normal direction, and the animation cycle iterations that are even counts are played in a reverse direction.

This is used for the overall widget animation configuration and not the step-level configuration.

Values will be specified as a string containing one of the above values. Any other values will be ignored and the default is applied or the behavior is undefined, depending on the underlying hardware implementation.

duration

This key defines the time in seconds that an animation takes to complete one cycle. This is used for the overall widget animation configuration and not the step-level configuration.

Possible values include all the positive float numbers with a precision of three and the default value is zero, which indicates that the animation is instantaneous. However, there will not be visible animation changes for a value of zero. However, technically the animation occurs and all animation callbacks get triggered.

Negative values will be treated as zero or may lead to undefined behavior.

fillMode

This key defines what values are applied to the widget state by the animation outside the time it is executing. The value for this key is a member of the Animation Fill Mode Constants.

iterationCount

This key specifies the number of times an animation cycle is played. Default value is one (1), meaning the animation will play from beginning to end. A value of zero (0) will cause the animation to repeat forever until the view is live in the current hierarchy.

Possible values include all the positive integer numbers. Any invalid values such as negative values are ignored or may lead to undefined behavior.

Common Example

```
var animDef = {
    "delay": 0.01,
    "iterationCount": "2",
    "fillMode": kony.anim.FILL_MODE_FORWARDS,
    "duration": 0.25,
    "direction": kony.anim.DIRECTION_ALTERNATE
};
```

5.4 kony.anim Namespace

The kony.anim namespace is part of the Animation API. It provides the following API elements.

5.4.1 Constants

The `kony.anim` namespace provides the following constants.

Animation Effect Constants

These constants are used to select what type of animation will take place.

Constant	Description
kony.anim.ANIMATION_EFFECT_AUTOMATIC	Use the default animation style.
kony.anim.ANIMATION_EFFECT_BOTTOM	Add the new data at the bottom.
kony.anim.ANIMATION_EFFECT_FADE	Fade new data into the current location.
kony.anim.ANIMATION_EFFECT_LEFT	Add the new data in from the left.
kony.anim.ANIMATION_EFFECT_MIDDLE	Move existing data below the add point downward and insert the new data into the middle.

Constant	Description
kony.anim.ANIMATION_EFFECT_NONE	No animation effect.
kony.anim.ANIMATION_EFFECT_TOP	Add the new data from the top.

The following constants are used to define the velocity of animation.

Constant	Description
kony.anim.EASE	Ensures that the timing of your animations matches that of most system animations.
kony.anim.EASE_IN	Animation begins slowly and then speeds up as it progresses.
kony.anim.EASE_OUT	Animation begins quickly and then slows down as it progresses.
kony.anim.EASE_IN_OUT	Animation begins slowly, accelerates through the middle of its duration, and then slow again before completing.
kony.anim.Linear	Animation to occur evenly over its duration.

Remarks

These constants are usable with the following widget methods, which are documented in the Kony Widget Programmer's Guide. These methods are available on all widgets that support animation.

- addDataAt
- addAll
- addSectionAt
- removeAt
- removeAll
- removeSectionAt
- setDataAt
- setSectionAt
- SetData

Currently, animations are supported for the SegmentedUI widget.

Example 1

```
animation = kony.anim.ANIMATION_EFFECT_LEFT;
form.segments.addAt(data, sectionIndex, rowIndex, animation);
```

Example 2

```
function animateWidget() {
    <Widget>.animate(kony.ui.createAnimation({
        "100": {
            "stepConfig": {
                "timingFunction": kony.anim.EASE_IN_OUT
            },
            "width": "20%",
            "height": "5%"
        }
    }), {
        "delay": 0,
        "iterationCount": 1,
        "fillMode": kony.anim.FILL_MODE_FORWARDS,
        "duration": 1.5
    }, {
        "animationEnd": null
    });
}
```

Animation Fill Mode Constants

Specifies the fill mode being used when performing widget animations.

Constant	Description
kony.anim.FILL_MODE_BACKWARDS	The values configured in the first step of animation definition are applied to the widget at the beginning of the animation (even before the delay ends). At the end of animation, values are reset to the values, that were there before the start of the animation. Therefore, the widget returns to its starting point after the animation ends.
kony.anim.FILL_MODE_BOTH	The animation is applied twice on the widget. First at the beginning of the animation, before the animation delay with the values configured in the first step of the animation, and second at the end of the animation, with the values configured in the last step of the animation definition.

Constant	Description
kony.anim.FILL_MODE_FORWARDS	The values configured in the last step of animation definition are the final values that are applied to the widget at the end of animation. So the widget stays where it is at the end of the animation.
kony.anim.FILL_MODE_NONE	The values in animation definition are never set to the actual widget. In this case, the widget comes back to original state after the animation is completed. This is the default.

Example

```
function animConfig() {
    var config = {
        "duration": 1,
        "iterationCount": 1,
        "delay": 0,
        "fillMode": kony.anim.FILL_MODE_FORWARDS
    };
    return config;
}
```

5.5 kony.ui.createAnimation Function

The details of the kony.ui.createAnimation function, which is part of the kony.ui Namespace, are as follows.

kony.ui.createAnimation Function

Creates an object that defines an animation.

Syntax

```
kony.ui.createAnimation(
    animationDefinition);
```

Input Parameters

Parameter	Description
animationDefinition	An object that defines the transformations to perform during the animation.

Example

```
var transformObject = kony.ui.makeAffineTransform();
transformObject.translate(10, 0);
transformObject.scale(0.1, 1);
animationDef = {
    100: {
        "transform": transformObject
    }
};
animationConfig = {
    duration: 0.3,
    fillMode: kony.anim.FILL_MODE_FORWARDS
};
animationDefObject = kony.ui.createAnimation(animationDef);
```

Return Values

Returns an instantiated `animation` object.

5.6 kony.ui.makeAffineTransform Function

The details of the kony.ui.makeAffineTransform function, which is part of the kony.ui Namespace, are as follows.

kony.ui.makeAffineTransform

Creates a transformation object that can be used in an animation definition.

Syntax

```
kony.ui.makeAffineTransform()
```

Example

```
/*****************************************************************
 *     Name    : createAnimation
 *     Author  : Kony
 *     Purpose : To call makeAffineTransform API and createAnimation API on
widgets.
 *****************************************************************/
function animation() {

    // Creates a transformation object that can be used in an animation
definition.
    var transformObject = kony.ui.makeAffineTransform();

    // Add a translation and a scale.
    transformObject.translate(10, 0);
    transformObject.scale(0.1, 1);

    // Create the animation definition.
    animationDef = {
        100: {
            "transform": transformObject
        }
    };

    //Create the animation configuration.
    animationConfig = {
        duration: 0.3,
        fillMode: kony.anim.FILL_MODE_FORWARDS
    };

    // Creates an object that defines an animation.
    animationDefObject = kony.ui.createAnimation(animationDef);
    Form0bf93c59bdc404d.Button00aaa01360b0349.animate(animationDefObject,
animationConfig);

}
```

Input Parameters

None.

Return Vales

An object that can be used to specify a transformation.

5.7 transform Object

Use the transform Object to attach 2D and 3D transformations to a widget.

5.7.1 Methods

The transform Object provides the following methods.

rotate

This method returns an affine transformation matrix constructed by rotating receivers affine transform. Angle is a number in degrees and always measured from x-axis as shown.

Syntax

```
rotate(angle)
```

Input Parameters

Parameter	Description
angle [Number]	A number represents the angle, in degrees, by which this matrix rotates the coordinate system axes. A positive value specifies counterclockwise rotation and a negative value specifies clockwise rotation.

Example

```
//Sample code of animation
function animDeftranslate() {
    var transformProp1 = kony.ui.makeAffineTransform();
    transformProp1.translate(100, 100);
    var transformProp2 = kony.ui.makeAffineTransform();
    transformProp2.scale(2, 2);
    var transformProp3 = kony.ui.makeAffineTransform();
    transformProp3.rotate(90);
    var animDefinitionOne = {
        0: {
            "transform": transformProp1
        },
        50: {

            "transform": transformProp2
        },
        100: {

            "transform": transformProp3
        }
    }
    animDef = kony.ui.createAnimation(animDefinitionOne);
    return animDef;

}

Function getParent() {

    var result = this.parent;
}
```

Return Values

Returns an affine transformation matrix constructed by rotating receivers affine transform.

Exceptions

WidgetError

Remarks

Default value is 0, if transform was never applied to the widget. The rotation does not result in any layout changes to parent or peer widgets. This is also applicable for widgets placed inside horizontal or vertical flex containers.

For example, if you want to rotate a widget in 360 degrees, you can follow the below sequence of steps:

step1: Rotate the widget from 0 - 120

step1: Rotate the widget from 120 - 240

step3: Rotate the widget from 240 - 360

Any value greater than 180 degrees may lead to shortest path rotation from its current position. For cross platform values, for example 190 degrees will make the object rotate -170 (190-360) in negative direction, as 170 is shortest path compared to 190.

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

rotate3D

This method rotates the widget by angle on the unit directional vector formed by rx, ry, and rz.

Syntax

```
rotate3D(
    angle,
    rx,
    ry,
    rz)
```

Input Parameters

Parameter	Description
angle	Specify the angle, by which a widget to be rotated around rx, ry, and rz axises.
rx	Specify the x-axis value on which rotation to happen.
ry	Specify the y-axis value on which rotation to happen.
rz	Specify the z-axis value on which rotation to happen.

Example

```
var newTransform = kony.ui.makeAffineTransform();
newTransform.rotate3D(45, 1, 0, 1); //rotates by 45degrees in x and z Axis.
widget.transform = newTransform;
```

Exceptions

Error Code	Description
100	Invalid input
101	Incomplete input

Remarks

The value of angle should be in degrees and the range should be in between 180º to -180º. Any value greater or lesser than range will result into platform-specific behavior. Positive values of angle will rotate the widget in anti-clockwise direction and vice versa.

The values of rx, ry, and rz should be in the range of 0 - 1. If the (0,0,0) vector is specified, the behavior is platform-specific.

In the Android platform, the values between 0 - 1 are not accepted. Only '0' or '1' is accepted.

All the input parameters need to be specified. If any parameter found missing will result in an exception 101.

Availability

Available in the IDE

iOS

Android

SPA

scale

This method returns an affine transformation matrix constructed by scaling receivers affine transform. It is a JSObject with keys sx and sy and allow numbers only.

Syntax

```
scale (
   sx,
   sy)
```

Input Parameters

Parameter	Description
sx [Number]	The factor by which to scale the x-axis of the widget coordinate system.
sy [Number]	The factor by which to scale the y-axis of the widget coordinate system.

Default values are {"sx":1, "sy":1}, if the transform was never applied to the widget.

Example

```
//Sample code of animation
function animDeftranslate() {
    var transformProp1 = kony.ui.makeAffineTransform();
    transformProp1.translate(100, 100);
    var transformProp2 = kony.ui.makeAffineTransform();
    transformProp2.scale(2, 2);
    var transformProp3 = kony.ui.makeAffineTransform();
    transformProp3.rotate(90);
    var animDefinitionOne = {
        0: {
            "transform": transformProp1
        },
        50: {

            "transform": transformProp2
        },
        100: {

            "transform": transformProp3
        }
    }
    animDef = kony.ui.createAnimation(animDefinitionOne);
    return animDef;

}

Function getParent() {

    Var result = this.parent;
}
```

Return Values

Returns an affine transformation matrix constructed by scaling receivers affine transform.

Exceptions

WidgetError

Remarks

Scaling does not result in any layout changes to parent or peer widgets. This is applicable to the widgets placed inside horizontal or vertical flex containers. Negative values for sx and sy will make the widget flip in that direction.

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

---

scale3D

Scales a widget in three dimensions (x, y, z) coordinate system.

Syntax

```
scale3D(
    sx,
    sy,
    sz)
```

Input Parameters

Parameter	Description
sx [Number]	Specify the value to be scaled in the x direction.

Parameter	Description
sy [Number]	Specify the value to be scaled in the y direction.
sz [Number]	Specify the value to be scaled in the z direction.

Example

```
var newTransform = kony.ui.makeAffineTransform();
newTransform.scale3D(2, 0.5, 1);
//scales by 200% in xDirection, 50% in yDirection and no scale happening in
zDirection.
widget.transform = newTransform;
```

Exceptions

Error Code	Description
100	Invalid input
101	Incomplete input

Remarks

The default values of the sx, sy, and sz directions are (1, 1, 1). Any value with in the 0 - 1 range scales down the widget and the value greater than '1' scales up in the specified directions. As all the widgets are not 3D meshes, this function may not be applicable for z-axis and may have platform-specific behavior. The scale3D method should not be applied on zero dimension widgets. If applied, the behavior is undefined.

All the input parameters need to be specified. If any parameter found missing will result in an exception 101.

Availability

Available in the IDE

iOS

SPA

setPerspective

This method sets the perspective and sets the vanishing point at the center of the widget.

Syntax

```
setPerspective(
    distanceOfViewerToPlane)
```

Input Parameters

Parameter	Description
distanceOfViewerToPlane	The distance between the viewer and object. Always the value of this parameter should be greater than zero. Otherwise results an exception 100.

Example

```
var newTransform = kony.ui.makeAffineTransform();
newTransform.setPerspective(1000.0);
//Sets the perspective as such this will have no effect until it is combined
with other transformation matrix.
newTransform.rotate3D(45, 1, 0, 1);
//rotates by 45degrees in x and z Axis. Now the perspective can be observed
widget.transform = newTransform;
```

Exceptions

Error Code	Description
100	Invalid input
101	Incomplete input

Remarks

The perspective has to be set in combination with other transforms. The perspective set by itself will not have any effect. If perspective is set to transform in any key frame, the perspective will be applied to that particular key frame itself in the KeyFrameAnimation.

The perspective is platform dependent so that each platform has different perspective of a view for same value. The default perspective on the Android platform is 1280. Any perspective less than 1280 makes the camera perspective closer to the view and greater than 1280 makes perspective far from the view.

In the Android platform, when perspective is not specified, the default perspective is applied.

For the iOS platform, the value of the distanceOfViewerToPlane parameter should be greater than max (width, height) values of the widget view's frame. For example, if the value of (width, height) is (100, 50), the parameter value should be greater than 100. The effect of this parameter vary visually on different platforms for the same value. The units of the distanceOfViewerToPlane parameter is platform-specific.

All the input parameters need to be specified. If any parameter found missing will result in an exception 101.

Availability

Available in the IDE

iOS

SPA

translate

This method returns an affine transformation matrix constructed by translating receivers affine transform. It is a JavaScript object with keys tx and ty and allow numbers in dp.

Syntax

```
translate (
   tx,
   ty)
```

Input Parameters

Parameter	Description
tx [Number]	The value by which to move the x-axis of the widget coordinate system.
ty [Number]	The factor by which to move the y-axis of the widget coordinate system.

Default values are {"tx":0, "ty":0} if the transform was never applied to the widget.

Example

```
//Sample code of animation
function animDeftranslate() {
    var transformProp1 = kony.ui.makeAffineTransform();
    transformProp1.translate(100, 100);
    var transformProp2 = kony.ui.makeAffineTransform();
    transformProp2.scale(2, 2);
    var transformProp3 = kony.ui.makeAffineTransform();
    transformProp3.rotate(90);
    var animDefinitionOne = {
        0: {
            "transform": transformProp1
        },
        50: {

            "transform": transformProp2
        },
        100: {

            "transform": transformProp3
        }
    }
    animDef = kony.ui.createAnimation(animDefinitionOne);
    return animDef;

}

Function getParent() {

    Var result = this.parent;
}
```

Return Values

Returns an affine transformation matrix constructed by translating receivers affine transform.

Exceptions

WidgetError

Remarks

Translate does not result in any layout changes to parent or peer widgets. This is applicable to the widgets placed inside horizontal or vertical flex containers.

Note: Values cannot be specified using percentage and pixels.

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

translate3D

Translates the widget from present location to new location by x, y, z amount.

Syntax

```
translate3D(
    tx,
    ty,
    tz)
```

Input Parameters

Parameter	Description
tx	Specify the value to be moved in the x direction from present location.
ty	Specify the value to be moved in the y direction from present location.
tz	Specify the value to be moved in the z direction from present location.

Example

```
var newTransform = kony.ui.makeAffineTransform();
newTransform.translate3D(223, 12, 56); //translates by 223 xAxis,12 in
yAxis,56 in zAxis
widget.transform = newTransform;
```

Exceptions

Error Code	Description
100	Invalid input
101	Incomplete input

Remarks

The values of tx, ty, and tz should be floating numbers. If the setPerspective method is not used, the widget moving in the z direction will not have any visual effect.

All the input parameters need to be specified. If any parameter found missing will result in an exception 101.

Availability

Available in the IDE

iOS

SPA

6. App Extension API for iOS

Kony Visualizer facilitates the process of creating app extensions for iOS apps. An app extension adds extended functionality to the app in which the extension is embedded. Apps use app extensions to perform only specific tasks that take small amounts of time to execute. For example, you can create an app extension to that lets users edit images that are embedded within another app, such as a document viewer or text editor.

The App Extension API for iOS contains the following Namespaces and API elements:

- kony.actionExtension Namespace

Function	Description
`kony.actionExtension.setExtensionsCallbacks`	Sets an Action Extension with callbacks for app extension state changes.

Property	Description
`kony.actionExtensions.view`	Holds the current extension view.

- kony.iMessageExtensions Namespace

Function	Description
`kony.iMessageExtensions.setExtensionsCallbacks`	Sets an iMessage extension functionality with various states as callback events.

Property	Description
`kony.iMessageExtensions.view`	Holds the current extension view.

- kony.intentExtension Namespace

Function	Description
`kony.intentExtension.setExtensionsCallbacks`	Sets an iMessage extension functionality with various states as callback events.

- kony.notificationContentExtension Namespace

Function	Description
`kony.notificationContentExtension.setExtensionsCallbacks`	Sets a notification content extension with various states as callback events.

Property	Description
`kony.notificationContentExtension .view`	Holds the current extension view.

- kony.shareExtensions Namespace

Function	Description
`kony.shareExtensions.popConfigurationViewController`	Dismisses the current configuration view controller.
`kony.shareExtensions.pushConfigurationViewController`	Displays a configuration view controller.
`kony.shareExtensions.setExtensionsCallbacks`	Allows your app to set callback event handlers for a Share extension.

Property	Description
`kony.shareExtensions.charactersRemaining`	Sets an initial value to be displayed as the number of characters remaining in the placeholder.
`kony.shareExtensions.contentText`	Contains the text from the current textView.
`kony.shareExtensions.extensionContext`	Returns the current extension context.
`kony.shareExtensions.placeholder`	Sets the text for the share app extension in the placeholder.
`kony.shareExtensions.view`	Holds the current extension view.

- kony.today Namespace

Function	Description
`kony.todayExtension.setExtensionsCallbacks`	Sets a Today extension with callbacks for app extension state changes.

Property	Description
kony.todayExtension.view	Stores the current extension view.

6.1 Overview

Kony Visualizer provides support for the following Apple App Extensions.

- Action Extensions
- iMessage Extensions
- Intent Extensions
- Notification Content Extensions
- Share Extensions
- Today Extensions

For more information about what app extension are and what you can do with them, refer Apple's App Extension Programming Guide.

Kony Visualizer Enterprise allows you to create app extensions directly in its workspace. Each app extension within the workspace will be treated as a Visualizer Enterprise project within the project and stored in a folder called `appextensions`.

6.1.1 Creating App Extensions

App extensions must be embedded in a container app. The app extension enhances the functionality of the container app. To build an app extension in Kony Visualizer, you write your app extension using the Kony App Extension API for iOS and with native iOS calls that your app invokes through the Kony Native Function API.

To learn about creating an app extension in Kony Visualizer, follow these steps:

1. Open an existing Kony Visualizer project. This is your container app.
2. From Kony Visualizer's main menu, select **File** > **New Project**.

3. In the dialog box that appears, choose **App Extension**.
4. In the dialog box that appears, choose the type of app extension that you would like to create and then click **Add**.
5. The **Configure App Extension** dialog box appears. Type in a name for your app extension.
6. Select the template to use for the extension. Your extension can either have a UI or not, depending on which template you select.
7. Click **Finish**.

Kony Visualizer creates an empty view that you can add JavaScript code to. For more information, refer create an iOS application extension.

You can then import an existing app extension into your project. To learn about importing an app extension, refer Import an application extension.

6.1.2 Sharing Data between an Extension and Its App

Even though the app extension is packaged within the containing app bundle, you must explicitly set up a share container so that the app extension and the containing app can share data among themselves.

The shared container can be created using the App Group API and the Native Function API. For more about sharing data, refer to the **Sharing Data with Your Containing App** section in the Apple documentation. Also, see the App Group API overview in the Kony Visualizer API Developer's Guide.

6.1.3 Deleting an App Extension from a Project

You can delete an app extension from your project by right-clicking the extension in the **Project** pane of Kony Visualizer and selecting Delete. Visualizer will remove the extension and its folder from the project.

6.1.4 Enabling and Disabling an App Extension within a Project

You can enable or disable an app extension that is in a project by right-clicking the extension in the **Project** pane of Kony Visualizer. If the app extension is enabled, **Disabled** will appear in the context menu and you can select that to disable the app extension. Disabled app extensions are not compiled or added to the final app when you do a build.

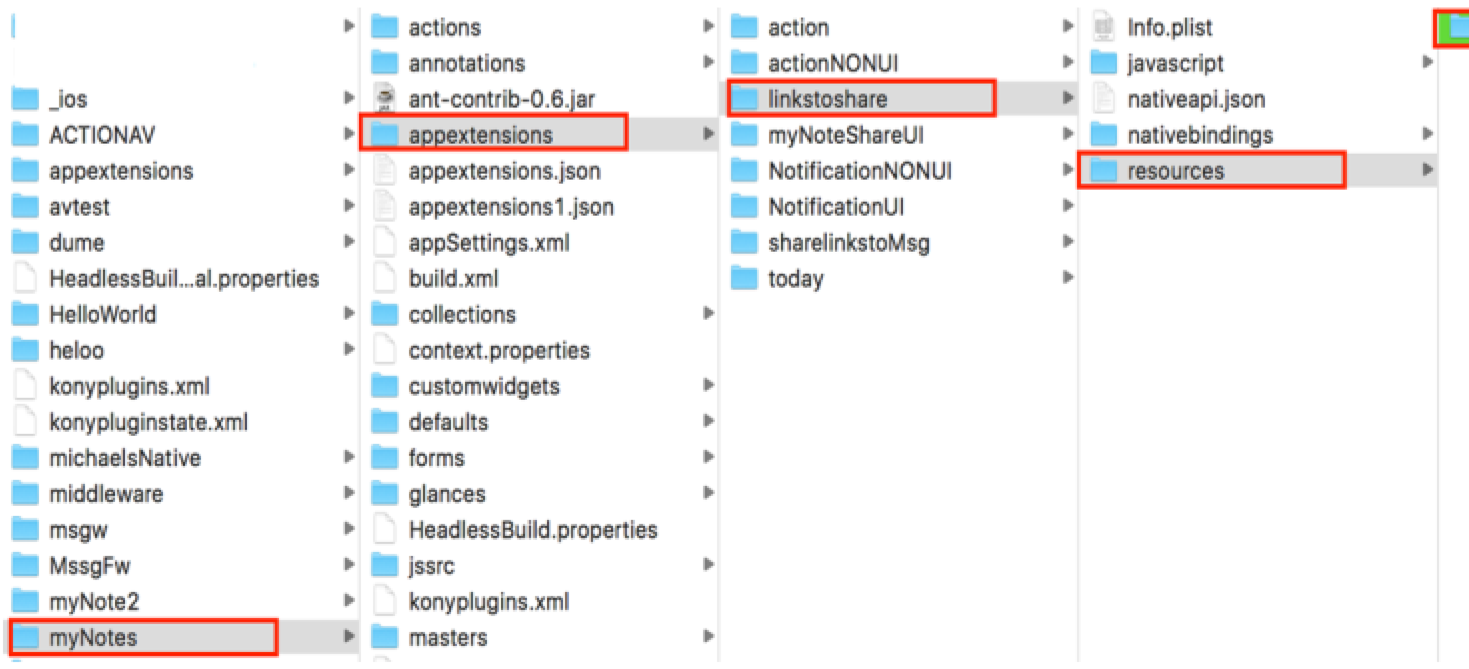
If the app extension is disabled, the context menu contains **Enable**. Select that to enable the app extension.

6.1.5 Adding an Icon to an App Extension

You can add a custom icon to your app extension that represents the function it performs. To do so, you must create a template image that iOS uses as a mask when it generates the final icon. To ensure that the template image looks good in the final UI, please follow the guidelines below.

- DO use a black and white image with the appropriate level of transparency.
- DON'T include a drop shadow.
- DO use antialiasing when you create your template image.
- DO use a transparent background for your template image.
- DON'T use a solid white background for the image.

After you have created the image, place it in the AppIcon folder, which is shown in the illustration below.



Apple requires you to provide your template image in specific sizes. The following table shows the required template image files.

Name	Size
Icon-App-20x20@1x.png	20 × 20
Icon-App-20x20@2x.png	40 × 40
Icon-App-20x20@3x.png	60 × 60
Icon-App-29x29@1x.png	29 × 29
Icon-App-29x29@2x.png	58 × 58
Icon-App-29x29@3x.png	87 × 87
Icon-App-40x40@1x.png	40 × 40
Icon-App-40x40@2x.png	80 × 80
Icon-App-40x40@3x.png	120 × 120
Icon-App-57x57@1x.png	57 × 57
Icon-App-57x57@2x.png	114 × 114
Icon-App-60x60@1x.png	60 × 60
Icon-App-60x60@2x.png	120 × 120
Icon-App-60x60@3x.png	180 × 180
Icon-App-72x72@1x.png	72 × 72
Icon-App-72x72@2x.png	144 × 144
Icon-App-76x76@1x.png	76 × 76
Icon-App-76x76@2x.png	152 × 152
Icon-App-76x76@3x.png	228 × 228
Icon-App-83.5x83.5@2x.png	167 × 167
Icon-Small-50x50@1x.png	50 × 50
Icon-Small-50x50@2x.png	100 × 100

6.2 kony.actionExtension Namespace

The kony.actionExtension Namespace provides support for the iOS Action extension, which is a kind of iOS app extension.

An Action extension helps users view or update content originating in a host app. For example, an Action extension might help users edit an image in a document that they're viewing in a document editor. For more information about what Action extensions are and what you can use them for, please see the Apple developer documentation.

You add an Action extension to your app in the same way that you add any other type of iOS app extension. For more details, refer App Extension API for iOS.

Before your Action extension can be used, your app must set the callback functions that provide the Action extension with its functionality. It does this by invoking the kony.actionExtension.setExtensionsCallbacks function.

The kony.actionExtension Namespace contains the following API elements.

6.2.1 Properties

The kony.actionExtensions Namespace provides the following properties.

kony.actionExtensions.view

Holds the current extension view.

Syntax

```
kony.actionExtensions.view;
```

Example

```
//Sample code
var myView = kony.actionExtensions.view;
myView.addSubView(button);
```

Type

UIView

Read/Write

Read only.

Platform Availability

iOS.

6.2.2 Functions

The kony.actionExtension namespace contains the following functions.

kony.actionExtension.setExtensionsCallbacks

Sets an Action Extension with callbacks for app extension state changes.

Syntax

```
kony.actionExtension.setExtensionsCallbacks(callbacks)
```

Input Parameters

callbacks

Contains an object with key-value pairs where the key specifies the extension state and the value is a callback function. The following are the possible keys.

Key	Description
beginRequestWithExtensionContext	The user has selected the action.
loadView	Loads a view that the controller manages.
viewDidAppear	A view was just displayed.

Key	Description
viewDidDisappear	A view just removed from the view hierarchy.
viewDidLoad	The view's controller was loaded into memory.
viewWillAppear	A view is about to be displayed.
viewWillDisappear	A view is about to be removed from the view hierarchy.

Example: beginRequestWithExtensionContext

```
function beginRequestWithExtensionContext(var ExtensionContext) {
    // Native bindings code
}
kony.actionExtension.setExtensionsCallbacks({
    "beginRequestWithExtensionContext": beginRequestWithExtensionContext
});
```

Example: loadView

```
function loadView() {
    // Native Function API code
}

kony.actionExtension.setExtensionsCallbacks({
    "loadView": loadView
});
```

Example: viewDidAppear

```
function viewDidAppear() {
    // Native Function API code
}

kony.actionExtension.setExtensionsCallbacks({
    "viewDidAppear": viewDidAppear
});
```

Example: viewDidLoad

```
function viewDidLoad() {
    // Native Function API code
}

kony.actionExtension.setExtensionsCallbacks({
    "viewDidLoad": viewDidLoad
});
```

Example: viewDidDisappear

```
function viewDidDisappear() {
    // Native Function API code
}

kony.actionExtension.setExtensionsCallbacks({
    "viewDidDisappear": viewDidDisappear
});
```

Example: viewWillAppear

```
function viewWillAppear() {
    // Native Function API code
}

kony.actionExtension.setExtensionsCallbacks({
    "viewWillAppear": viewWillAppear
});
```

Example: viewWillDisappear

```
function viewWillDisappear() {
    // Native Function API code
}

kony.actionExtension.setExtensionsCallbacks({
    "viewWillDisappear": viewWillDisappear
});
```

Return Values

None.

Platform Availability

iOS only

6.3 kony.iMessageExtensions Namespace

The kony.iMessageExtensions Namespace enables you to add support for iMessage app extensions in your iOS app. With iMessage app extensions, users can send content text, stickers, and media files to each other. For more information about what iMessage app extensions are and what you can use them for, please see the Apple developer documentation.

With the kony.iMessageExtensions namespace, your iOS apps can leverage the power of the iMessage app to add interactive conversations and file sharing to your app's functionality.

You add an iMessage extension to your app in the same way that you add any other type of iOS app extension. For more details, refer App Extension for iOS.

Before your iMessage extension can be used, your app must set the callback functions that provide the iMessage extension with its functionality. It does this by invoking the kony..iMessageExtension.setExtensionsCallbacks function.

6.3.1 Properties

The kony.iMessageExtensions Namespace provides the following properties.

kony.iMessageExtensions.view

Holds the current extension view.

Syntax

```
kony.shareExtensions.view
```

Example

```
//Sample code
var myView = kony.iMessageExtensions.view;
myView.addSubView(button);
```

Type

UIView

Read/Write

Read only.

Platform Availability

iOS.

6.3.2 Functions

The kony.iMessageExtensions namespace provides the following function.

kony.iMessageExtensions.setExtensionsCallbacks

Sets an iMessage extension functionality with various states as callback events.

Syntax

```
kony.iMessageExtensions.setExtensionsCallbacks(callbacks)
```

Input Parameters

callbacks

Contains an object with key-value pairs where the key specifies the extension state and the value is a callback function. The following are the possible keys.

Key	Description
didBecomeActiveWithConversation	The extension is about to present the UI.
didCancelSendingMessageConversation	The user deleted the message without sending it.

Key	Description
didReceiveMessageCconversation	A message has arrived that was generated by another instance of this extension on a remote device.
didStartSendingMessageConversation	The user tapped the Send button.
didTransitionToPresentationStyle	The extension has just transitioned to a new presentation style.
loadView	Loads a view that the controller manages.
viewDidAppear	A view was just displayed.

Key	Description
viewDidDisappear	A view just removed from the view hierarchy.
viewDidLoad	The the view controller has loaded its view hierarchy into memory.
viewWillAppear	A view is about to be displayed.
viewWillDisappear	A view is about to be removed from the view hierarchy.
willResignActiveWithConversation	The extension is about to change from the active to inactive state.

Key	Description
willTransitionToPresentationStyle	The extension is about to transition to a new presentation style.

Example: didBecomeActiveWithConversation

```
function didBecomeActiveWithConversation() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "didBecomeActiveWithConversation": didBecomeActiveWithConversation
});
```

Example: didCancelSendingMessageConversation

```
function didCancelSendingMessageConversation() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "didCancelSendingMessageConversation": didCancelSendingMessageConversation
});
```

Example: didReceiveMessageCconversation

```
function didReceiveMessageConversation() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "didReceiveMessageConversation": didReceiveMessageConversation
});
```

Example: didStartSendingMessageConversation

```

function didStartSendingMessageConversation() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "didStartSendingMessageConversation": didStartSendingMessageConversation
});
```

Example: didTransitionToPresentationStyle

```
function didTransitionToPresentationStyle() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "didTransitionToPresentationStyle": didTransitionToPresentationStyle
});
```

Example: loadView

```
function loadView() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "loadView": loadView
});
```

Example: viewDidAppear

```
function viewDidAppear() {
    // Native Function API code
}
```

```
kony.iMessageExtensions.setExtensionsCallbacks({
    "viewDidAppear": viewDidAppear
});
```

Example: viewWillAppear

```
function viewWillAppear() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "viewWillAppear": viewWillAppear
});
```

Example: viewDidDisappear

```
function viewDidDisappear() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "viewDidDisappear": viewDidDisappear
});
```

Example: viewWillDisappear

```
function viewWillDisappear() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "viewWillDisappear": viewWillDisappear
});
```

Example: willResignActiveWithConversation

```
function willResignActiveWithConversation()
```

```
{
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks
({"willResignActiveWithConversation": willResignActiveWithConversation});
```

```
function willResignActiveWithConversation() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "willResignActiveWithConversation": willResignActiveWithConversation
});
```

Example: willTransitionToPresentationStyle

```
function willTransitionToPresentationStyle() {
    // Native Function API code
}

kony.iMessageExtensions.setExtensionsCallbacks({
    "willTransitionToPresentationStyle": willTransitionToPresentationStyle
});
```

Return Values

None.

Platform Availability

iOS.only

6.4 kony.intentExtension Namespace

The kony.intentExtension namespace provides you with the ability to add Siri-related functionality For more information about what iMessage app extensions are and what you can use them for, refer the Apple developer documentation.

Intent extensions enable your iOS app to interface with Siri. An Intent contains the data that Siri gathered from the user. The callback functions that your app sets with the setExtensionsCallbacks Function process the input in the Intent.

By default, an Intent extension in Kony Visualizer does not have a UI. However, Visualizer also enables you to create an IntentUI extension that does provide a UI. When you create an IntentUI with Kony Visualizer, it generates an empty view. Whether you use an Intent extension or an IntentUI extension, you add your business logic to the extension's callback functions. Your JavaScript code accesses native functionality on iOS devices by using objects and invoking functions in the Native Functions API.

Kony Visualizer supports following functionality in the Intent (or IntentUI) extensions.

- VoIP calling
- Messaging
- Payments
- Photo
- Workouts
- Ride booking

6.4.0.1 App Vocabulary

It is often the case that your app will have specific vocabulary words associated with it that you will need to teach to Siri when your app is installed. Some vocabulary words are needed by all users of your app. Others are user-specific. Your app can set the user-specific vocabulary by invoking the kony.vocabulary.setVocabularyStrings function. Your app can remove vocabulary words with the kony.vocabulary.removeAllVocabularyStrings function.

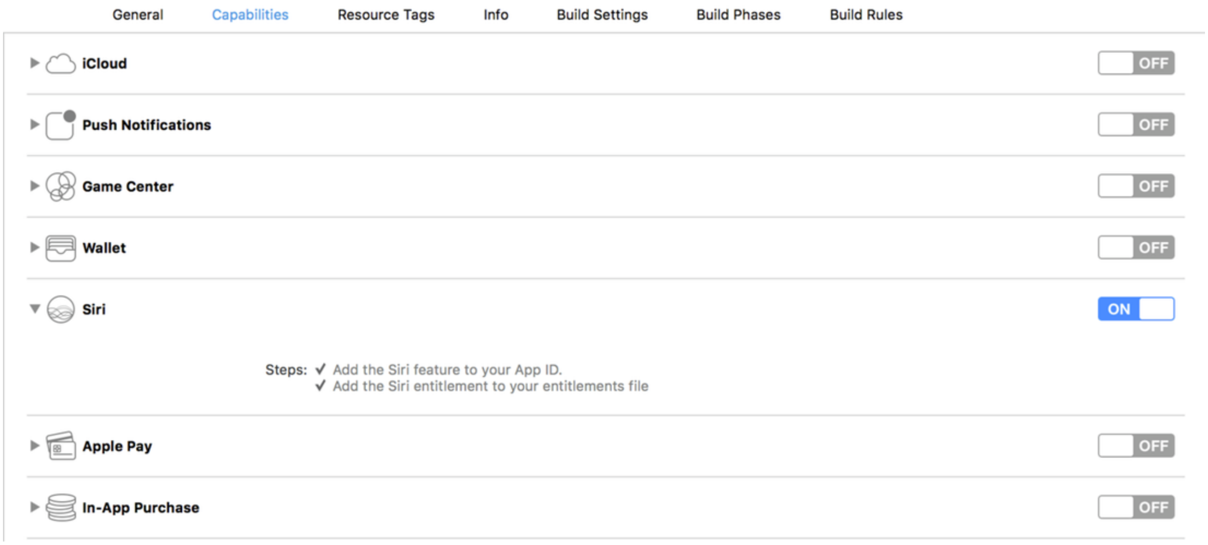
For vocabulary words that are needed by all of your users, you register a global vocabulary file when you install your app. For information on registering custom vocabulary words using a vocabulary file, refer the Apple developer documentation.

6.4.0.2 Granting Permissions

Apple specifies that users must manually grant apps permission to use the Apple SiriKit. For your app to request permission from the user, it must do the following.

1. Include the `NSSiriUsageDescription` key in your iOS app's Info.plist file. The value for this key is a string that describes what information your app shares with SiriKit. For example, a workout app might set the value to the string "Workout information will be sent to Siri." Inclusion of this key in your Info.plist is required.

2. Enable Siri under the Capabilities tab for Main app, as shown in the following illustration.



iCloud	OFF
Push Notifications	OFF
Game Center	OFF
Wallet	OFF
Siri	ON
Apple Pay	OFF
In-App Purchase	OFF

Steps: ✓ Add the Siri feature to your App ID.
✓ Add the Siri entitlement to your entitlements file

3. Provide runtime permissions for your app by invoking functions in the Runtime Permissions API. An example of this is provided in the following sample code.

```
var result =
    kony.application.checkPermission(kony.os.RESOURCE_SIRI,
null);

// If the app does not have the required permissions ...
if (result.status == kony.application.PERMISSION_DENIED) {
    // See if the app can request permission.
    if (result.canRequestPermission) {
        kony.print("Requesting Permission");
        kony.application.requestPermission(
            kony.os.RESOURCE_SIRI,
            permissionStatusCallback,
            null);
    } else {
        kony.ui.Alert(
            "PERMISSION DENIED: Open Device Settings.",
            null,
            "Siri Error",
            null,
            null);
    }
} else if (result.status == kony.application.PERMISSION_GRANTED)
{
    kony.print("Permission Granted");
} else if (result.status == kony.application.PERMISSION_
RESTRICTED) {
    kony.print("Permission Restricted");
}

function permissionStatusCallback(response) {
    if (response.status == kony.application.PERMISSION_GRANTED) {
        kony.print("Permission Granted");
    } else if (result.status == kony.application.PERMISSION_
DENIED) {
        kony.print("Permission Denied");
    }
}
```

6.4.0.3 Configuring Your Project for Intent Extensions

To configure your project for Intent extensions, you **must** perform the following **mandatory** steps.

Step 1: Enable 'Siri' under Capabilities for KRelease, KDebug Target in your Xcode Project.

Step 2: Include the `NSSiriUsageDescription` key in your iOS app's Info.plist file for authorization.

Step 3: Specify the intents that your extension supports.

1. n Xcode, select the Info.plist file of your Intents extension.
2. Expand the `NSExtension` and `NSExtensionAttributes` keys to reveal the `IntentsSupported` and `IntentsRestrictedWhileLocked` keys.
3. In the `IntentsSupported` key, add a String item for each intent that the extension handles. Set the value of each item to the class name of the intent.This key is required. You can support all of the intents in a given domain or only some of them, and a single extension can support multiple domains.
4. In the `IntentsRestrictedWhileLocked` key, add a String item for each intent for which you require the device to be unlocked. Set the value of each item to the class name of the intent. This key is optional. Some intents, such as those involving financial transactions, always require the user's device to be unlocked. You can use this key to augment the default list with intents that do not require an unlocked device by default.

6.4.1 Functions

The kony.intentExtension namespace provides the following function.

kony.intentExtension.setExtensionsCallbacks Function

Sets an iMessage extension functionality with various states as callback events.

Syntax

```
kony.intentExtension.setExtensionsCallbacks(callbacks)
```

Input Parameters

callbacks

Contains an object with key-value pairs where the key specifies the extension state and the value is a callback function. The following are the possible keys.

Key	Description
configureWithInteractionContextCompletion	The configuration is complete for the given interaction, the hosted view controller should call the completion block with its view's desired size. This size will be constrained between hostedViewMinimumAllowedSize and hostedViewMaximumAllowedSize of the extension context. Used with IntentUI extensions only.
handlerForIntent	An intent has arrived for the app. Used with Intent extensions only.
loadView	Loads a view that the controller manages. Used with IntentUI extensions only.
viewDidAppear	A view was just displayed. Used with IntentUI extensions only.
viewDidDisappear	A view just removed from the view hierarchy. Used with IntentUI extensions only.

Key	Description
viewDidLoad	The the view controller has loaded its view hierarchy into memory. Used with IntentUI extensions only.
viewWillAppear	A view is about to be displayed. Used with IntentUI extensions only.
viewWillDisappear	A view is about to be removed from the view hierarchy. Used with IntentUI extensions only.

Example: configureWithInteractionContextCompletion

```
function configureWithInteractionContextCompletion({
    "configureWithInteractionContextCompletion":
configureWithInteractionContextCompletion
}) {
    // Native Function API code
}

kony.intentExtension.setExtensionsCallbacks(interaction, uicontext,
completion);
```

Example: handlerForIntent

```
function handlerForIntent(intent) {
    // Native Function API code
}

kony.intentExtension.setExtensionsCallbacks({
    "handlerForIntent": handlerForIntent
});
```

Example: loadView

```
function loadView() {
    // Native Function API code
}

kony.intentExtension.setExtensionsCallbacks({
    "loadView": loadView
});
```

Example: viewDidAppear

```
function viewDidAppear() {
    // Native Function API code
}

kony.intentExtension.setExtensionsCallbacks({
    "viewDidAppear": viewDidAppear
});
```

Example: viewWillAppear

```
function viewWillAppear() {
    // Native Function API code
}

kony.intentExtension.setExtensionsCallbacks({
    "viewWillAppear": viewWillAppear
});
```

Example: viewDidDisappear

```
function viewDidDisappear() {
    // Native Function API code
}

kony.intentExtension.setExtensionsCallbacks({
    "viewDidDisappear": viewDidDisappear
});
```

Example: viewWillDisappear

```
function viewWillDisappear() {
    // Native Function API code
}

kony.intentExtension.setExtensionsCallbacks({
    "viewWillDisappear": viewWillDisappear
});
```

Return Values

None.

Remarks

When setting the callback function for `handlerForIntent`, the callback function takes a parameter named `intent`. This parameter contains an intent object of type INIntent class that encapsulates the request coming from Siri.

When setting the callback function for `configureWithInteractionContextCompletion`, the callback function takes the following parameters.

interaction

An object that contains the intent and response objects. Use the information in this object to configure the content of your view controller's view. For some types of interactions, only an intent object is available.

uiContext

An object that holds context in which your view controller is displayed. Use this parameter to determine whether your view controller is to be displayed in the Maps or Siri interface. You can customize your view controller accordingly for each interface.

completion

The block to execute when you finish configuring your view controller. You must execute this block at some point in your implementation of this method. This block has no return value and takes a parameter named *desiredSize* that sets the size you want applied to the view controller's view. Specify a value that is between the allowed minimum and maximum size, which you can get from the view controller associated extension object. Specify CGRectZero to hide your view controller's content altogether

Platform Availability

iOS.only

6.5 kony.notificationContentExtension Namespace

The kony.notificationContentExtension Namespace enables you to add support for Notification Content app extensions in your iOS app. With Notification Content app extensions, your app can display a custom user interface for its notifications. For more information about what Notification Content app extensions are and what you can use them for, refer the Apple developer documentation.

You can add a Notification Content extension to your app in the same way that you add any other type of iOS app extension. For more details, refer App Extension for iOS.

Before your Notification Content extension can be used, your app must set the callback functions that provide the Notification Content app extension with its functionality. It does this by invoking the kony.notificationContentExtension.setExtensionsCallbacks function.

6.5.1 Properties

The kony.notificationContentExtension Namespace contains the following properties.

kony.notificationContentExtension .view

Holds the current extension view.

Syntax

```
kony.notificationContentExtension.view;
```

Example

```
//Sample code
var myView = kony.notificationContentExtension.view;
myView.addSubView(button);
```

Type

UIView

Read/Write

Read only.

Platform Availability

iOS.

6.5.2 Functions

The kony.notificationContentExtension Namespace provides the following function.

kony.notificationContentExtension.setExtensionsCallbacks

Sets a notification content extension with various states as callback events.

Syntax

```
kony.notificationContentExtension.setExtensionsCallbacks(
    callbacks);
```

Input Parameters

callbacks

Contains an object with key-value pairs where the key specifies the extension state and the value is a callback function. The following are the possible keys.

Key	Description
didReceiveNotification	The app received a notification.
didReceiveNotificationResponse	The user tapped one of the notification's actions.
loadView	Loads a view that the controller manages.
viewDidAppear	A view was just displayed.
viewDidDisappear	A view just removed from the view hierarchy.
viewDidLoad	The controller has loaded its view hierarchy into memory.

Key	Description
viewWillAppear	A view is about to be displayed.
viewWillDisappear	A view is about to be removed from the view hierarchy.

Example: didReceiveNotification

```
function didReceiveNotification() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "didReceiveNotification": didReceiveNotification
});
```

Example: didReceiveNotificationResponse

```
function didReceiveNotificationResponse() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "didReceiveNotificationResponse": didReceiveNotificationResponse
});
```

Example: loadView

```
function loadView() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "loadView": loadView
});
```

Example: viewDidAppear

```
function viewDidAppear() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "viewDidAppear": viewDidAppear
});
```

Example : **viewDidLoad**

```
function viewDidLoad() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "viewDidLoad": viewDidLoad
});
```

Example: viewWillAppear

```
function viewWillAppear() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "viewWillAppear": viewWillAppear
});
```

Example: viewDidDisappear

```
function viewDidDisappear() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "viewDidDisappear": viewDidDisappear
});
```

Example: viewWillDisappear

```
function viewWillDisappear() {
    // Native Function API code
}

kony.notificationContentExtension.setExtensionsCallbacks({
    "viewWillDisappear": viewWillDisappear
});
```

Return Values

None.

Platform Availability

iOS.

6.6 kony.shareExtensions Namespace

The kony.shareExtensions Namespace implements the iOS Share extension, which is a type of app extension.

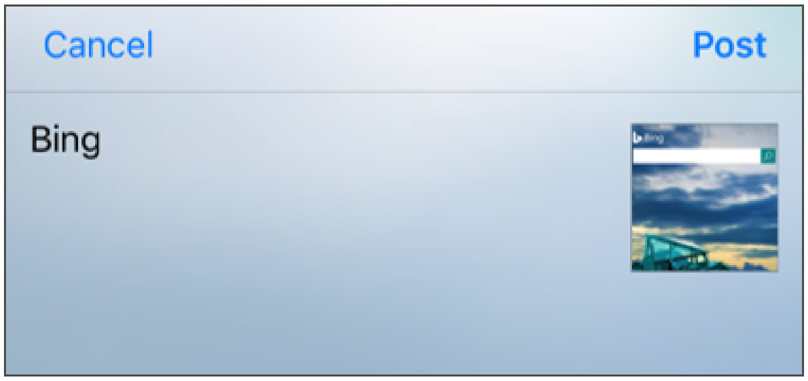
The Share app extension is one of the types of app extensions provided by Apple for iOS apps. The Share app extension allows the app users to share information from the current context with other entities, such as social media, apps, and services. For example, a Share app extension can be used to share photos directly from the Image Gallery with the social media.

The share extension app can be accessed by tapping an action button provided in an app to display the activity view. The activity view contains extensions relevant to the current context.

Kony Visualizer provides integrated support for developing Share app extensions for iOS apps. You develop Share extensions and package them into your app in the same way you do for other types of app extensions. For more information, refer App Extension API for iOS.

For more information about what the Share app extension is and what you can do with it, refer the Apple developer documentation.

A Share extension can use the default UI provided by Apple or a custom UI that you create. When you create a share extension using the Default GUI, a standard compose view UI is used as shown in the figure below.



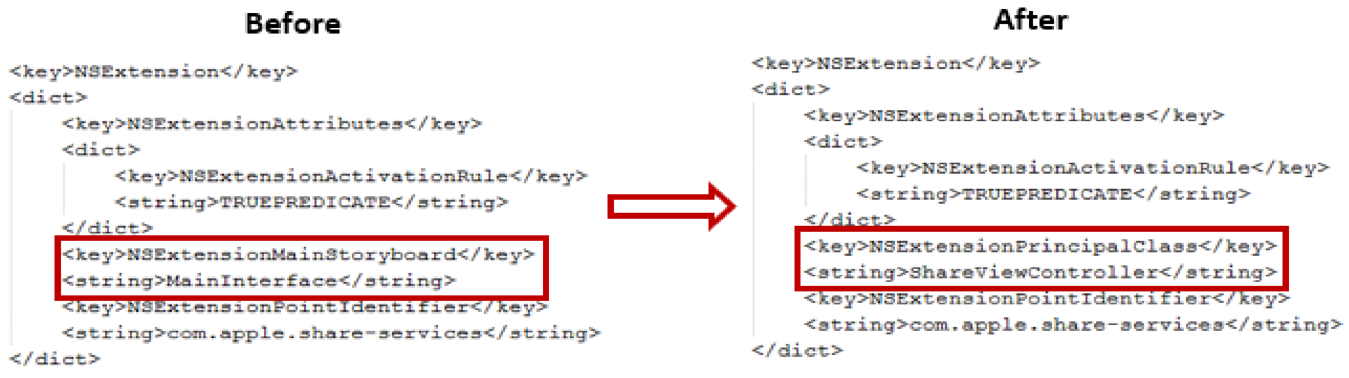
The implementation of the kony.shareExtensions Namespace provides you with what you need to add your own business logic using the Native Function API in callback handlers.

If you create your own custom UI, you must use the Native Function API and then add your business logic according to your needs. The Info.plist file generated when you create a Share app extension from Xcode is configured to use the Default UI by default. To make use of the custom UI for your app extension, you need to perform the following steps.

1. Open the Info.plist file of the Share app extension for which you want to develop a custom UI.
2. Find and replace the `NSExtensionMainStoryboard` key with

]

NSExtensionPrincipalClass, and also replace its value, MainInterface with ShareViewController. The figure below shows difference in the Info.plist file before and after modification.


Before

```
<key>NSExtension</key>
<dict>
    <key>NSExtensionAttributes</key>
    <dict>
        <key>NSExtensionActivationRule</key>
        <string>TRUEPREDICATE</string>
    </dict>
    <key>NSExtensionMainStoryboard</key>
    <string>MainInterface</string>
    <key>NSExtensionPointIdentifier</key>
    <string>com.apple.share-services</string>
</dict>
```

After

```
<key>NSExtension</key>
<dict>
    <key>NSExtensionAttributes</key>
    <dict>
        <key>NSExtensionActivationRule</key>
        <string>TRUEPREDICATE</string>
    </dict>
    <key>NSExtensionPrincipalClass</key>
    <string>ShareViewController</string>
    <key>NSExtensionPointIdentifier</key>
    <string>com.apple.share-services</string>
</dict>
```

3. Save and close the file.

When you develop a Share extension, you put your business logic into a specific set of callback functions. Your app must set these callback function by invoking the kony.shareExtensions.setExtensionsCallbacks function.

6.6.1 Properties

The kony.shareExtensions Namespace provides the following properties.

kony.shareExtensions.charactersRemaining

Sets an initial value to be displayed as the number of characters remaining in the placeholder.

Syntax

```
kony.shareExtensions.charactersRemaining;
```

Example

```
//Sample code
kony.shareExtensions.charactersRemaining = 100;
```

Type

Number

Read/Write

Read and write.

Remarks

This property is accessible only in the default GUI mode.

Platform Availability

iOS

kony.shareExtensions.contentText

Contains the text from the current textView.

Syntax

```
kony.shareExtensions.contentText;
```

Example

```
//Sample code
var text = kony.shareExtensions.contentText;
```

Type

String

Return Values

text

Remarks

This property is only available in the default GUI mode.

Platform Availability

iOS

kony.shareExtensions.extensionContext

Returns the current extension context.

Syntax

```
kony.shareExtensions.extensionContext;
```

Example

```
//Sample code
var Context = kony.shareExtensions.extensionContext;
Context.extensionContext.completeRequestReturningItemsCompletionHandler([], );
```

Type

Object

Read/Write

Read only

Platform Availability

iOS

kony.shareExtensions.placeholder

Sets the text for the share app extension in the placeholder.

Syntax

```
kony.shareExtensions.placeholder;
```

Example

```
//Sample code
kony.shareExtensions.placeholder = "write here";
```

Type

String

Read/Write

Read and write.

Remarks

The API works only in default GUI mode.

Platform Availability

iOS

kony.shareExtensions.view

Holds the current extension view.

Syntax

```
kony.shareExtensions.view;
```

Example

```
//Sample code
var myView = kony.shareExtensions.view;
myView.addSubView(button);
```

Type

UIView

Read/Write

Read only.

Platform Availability

iOS

6.6.2 Functions

The kony.shareExtensions Namespace provides the following function.

kony.shareExtensions.popConfigurationViewController

Dismisses the current configuration view controller.

Syntax

```
kony.shareExtensions.popConfigurationViewController()
```

Example

```
//Sample code
kony.shareExtensions.popConfigurationViewController();
```

Parameters

None.

Return Values

None.

Remarks

The function works only in the default GUI mode.

Platform Availability

iOS.

kony.shareExtensions.pushConfigurationViewController

Displays a configuration view controller.

Syntax

```
kony.shareExtensions.pushConfigurationViewController(UIVController)
```

Input Parameters

Parameter	Description
UIViewController	A UIViewController that your app creates using the Native Functions.

Example

```
var UIVC = //native bindings code to create UIViewController
 kony.shareExtensions.pushConfigurationViewController(UIVC);
```

Return Values

None.

Remarks

The function works only in the default GUI mode. The configuration view controller is called from a configuration item's tabHandler. Only one configuration view controller is allowed at a time. The pushed view controller should set `preferredContentSize` appropriately. The `SLComposeServiceViewController` observes changes to that property and animates sheet size changes as necessary.

Platform Availability

iOS.

kony.shareExtensions.setExtensionsCallbacks

Allows your app to set callback event handlers for a Share extension.

Syntax

```
kony.shareExtensions.setExtensionsCallbacks(
    callbacks)
```

Input Parameters

callbacks {Object}

Contains an object with key-value pairs where the key specifies the extension state and the value is a callback function. The following are the possible keys.

Key	Description
configurationItems	Enables the user to add configuration options via table cells at the bottom of the sheet, Returns an array of SLComposeSheetConfigurationItem objects. Only available in default GUI mode.
didSelectCancel	The user clicked the Cancel button. Only available in default GUI mode.
didSelectPost	The user clicked the Post button. Only available in default GUI mode.
isContentValid	Determines whether or not the content is valid. Only available in default GUI mode. Invalid content disables the Post button. Valid content enables it.
loadView	Loads the view into memory.
presentationAnimationDidFinish	The sheet presentation animation is finished. Only available in default GUI mode.
viewDidAppear	The view was just displayed.
viewWillAppear	The view controller's view is about to be added to a view hierarchy.

Key	Description
viewDidDisappear	The view has just been removed from the view hierarchy.
viewWillDisappear	The view is about to be removed from the view hierarchy.

Example

```
var callbackEvents = {
    didSelectCancel: function() {
        kony.shareExtensions.extensionContext.cancelRequestWithError
(undefined);
    },
    isContentValid: function() {
        var input = kony.shareExtensions.contentText;
        if (input.length < 100) {
            kony.shareExtensions.charactersRemaining = 100 - input.length;
            return true;
        } else {
            return false;
        }
    },
    configurationItems: function() {
        return [ConfigurationItem1, ConfigurationItem2];
    },
    viewDidLoad: function() {
        kony.shareExtensions.charactersRemaining = 100;
    }
};

kony.shareExtensions.setExtensionsCallbacks(callbackEvents);
```

Example: configurationItems

```
function configurationItems() {
    var returnarray = native bindings code to
    return array of SLComposeSheetConfigurationItem
    return returnarray;
}
kony.shareExtensions.setExtensionsCallbacks({"
    configurationItems": configurationItems
});
```

Example: didSelectCancel

```
function didSelectCancelcallback() {
    // native bindings code
}
kony.shareExtensions.setExtensionsCallbacks({"
    didSelectCancel": didSelectCancelcallback
});
```

Example: didSelectPostcallback

```
function didSelectPostcallback() {
    // native bindings code
}

kony.shareExtensions.setExtensionsCallbacks({"
    didSelectPost": didSelectPostcallback
});
```

Example: isContentValid

```
function isContentValid() {
    if ( //check the validity of the input using native
        {
            return true; //will enable the post button.
        }
        return false; //will disable the post button.
    }
    kony.shareExtensions.setExtensionsCallbacks({‟
        isContentValid”: isContentValid
    });
```

Example; loadView

```
function loadView() {
    //native bindings code
}
kony.shareExtensions.setExtensionsCallbacks({‟
    loadView”: loadView
});
```

Example: presentationAnimationDidFinish

```
function presentationAnimationDidFinish() {
    //native bindings code
}
kony.shareExtensions.setExtensionsCallbacks({‟
    presentationAnimationDidFinish”: presentationAnimationDidFinish
});
```

Example: viewDidAppear

```
function viewDidAppear() {
    //native bindings code
}
kony.shareExtensions.setExtensionsCallbacks({‟
    viewDidAppear”: viewDidAppear
});
```

Example: viewWillAppear

```
function viewWillAppear() {
    //native bindings code
}
kony.shareExtensions.setExtensionsCallbacks({‟
    viewWillAppear”: viewWillAppear
});
```

Example: viewDidDisappear

```
function viewDidDisappear() {
    //native bindings code
}
kony.shareExtensions.setExtensionsCallbacks({"
    viewDidDisappear": viewDidDisappear
});
```

Example: viewWillDisappear

```
function viewWillDisappear() {
    //native bindings code
}
kony.shareExtensions.setExtensionsCallbacks({"
    viewWillDisappear": viewWillDisappear
});
```

Return Values

None.

Platform Availability

iOS.

6.7 kony.todayExtension Namespace

The kony.todayExtension Namespace provides support for the iOS Today extension, which is a type of iOS app extension.

The Today extensions in Today view are called widgets. Widgets give users quick access to information that is important at the current moment. For example, users open the Today view to check items such as current stock prices or that day's weather forecast. For more information on Today extensions and how you can use them, refer the relevant Apple developer documentation.

You can add a Today extension to your app in the same way that you add any other type of iOS app extension. For more information on how to add iOS app extensions, refer App Extension API for iOS.

Before your app can use the Today extension, your app must set the callback functions that provide the Today extension with its functionality. The app does this by invoking the kony.todayExtension.setExtensionsCallbacks function.

The kony.todayExtension Namespace contains the following API elements.

6.7.1 Properties

The kony.todayExtension Namespace contains the following property.

kony.todayExtension.view

Stores the current extension view.

Syntax

```
kony.todayExtension.view;
```

Example

```
//Sample code
var myView = kony.todayExtension.view;
myView.addSubView(button);
```

Type

UIView

Read/Write

Read only

Platform Availability

- iOS

6.7.2 Functions

The kony.todayExtension Namespace contains the following function.

kony.todayExtension.setExtensionsCallbacks

Sets a Today extension with callbacks for app extension state changes.

Syntax

```
kony.todayExtension.setExtensionsCallbacks(
    callbackEvents);
```

Input Parameters

callbackEvents

Contains an object with key-value pairs where the key specifies the extension state and the value is a callback function. The possible keys of this parameter are as follows.

Key	Description
loadView	Loads a view that the controller manages.
viewDidLoad	The view's controller was loaded into the memory.
viewDidAppear	A view was just displayed.
viewWillAppear	A view is about to be displayed.
viewDidDisappear	A view was just removed from the view hierarchy.
viewWillDisappear	A view is about to be removed from the view hierarchy.

Key	Description
widgetPerformUpdate	The system calls this key at opportune times for the widget to update its state; both when the Notification Center is visible as well as when the Notification Center is in the background.
widgetActiveDisplayModeDidChangeWithMaximumSize	Called when the active display mode changes. It has the following arguments: • displayMode. It can take two values: NCWidgetDisplayModeCompact and NCWidgetDisplayModeExpanded. • NSValue. It contains one value: CGSizeValue.

Example

```
//Sample Code
function loadViewSample() {
    //native bindings code
}

function viewDidLoadSample() {
    //native bindings code
}

function viewWillAppearSample() {
    //native bindings code
}
```

```
function viewDidAppearSample() {
    //native bindings code
}

function viewWillDisappearSample() {
    //native bindings code
}

function viewDidDisappearSample() {
    //native bindings code
}

function widgetPerformUpdateSample() {
    //native bindings code
    return NCUpdateResult;
}

function widgetActiveDisplayModeDidChangeWithMaximumSizeSample(var
vardisplayMode,
    var maxsize) {
    var preferredContentSize;
    if (activeDisplayMode == NCWidgetDisplayModeCompact) {
        preferredContentSize = {
            width: maxsize.CGSizeValue.width,
            height: 300
        };
    } else {
        preferredContentSize = {
            width: maxsize.CGSizeValue.width,
            height: 800
        };
    }
    return preferredContentSize;
}
//Setting Extensions Callbacks: Example 1kony.
```

```
todayExtension.setExtensionsCallbacks(
{
    "loadView": loadViewSample,
    "viewDidLoad": viewDidLoadSample,
    "viewWillAppear": viewWillAppearSample,
    "viewDidAppear": viewDidAppearSample,
    "viewWillDisappear": viewWillDisappearSample,
    "viewDidDisappear": viewDidDisappearSample,
    "widgetPerformUpdate": widgetPerformUpdateSample,
    "widgetActiveDisplayModeDidChangeWithMaximumSize":
widgetActiveDisplayModeDidChangeWithMaximumSizeSample
}); //Setting Extensions Callbacks: Example 2var callbackEvents={
viewDidLoad: function() {
    var myView = kony.todayExtension.view;
    myView.addSubView(button);
}
};
kony.todayExtension.setExtensionsCallbacks(callbackEvents);
//end of code
```

Return Values

None

Platform Availability

- iOS

7. Application API

The Application API provides functions that enable you to control the application-level events and behaviors of your app.

The Application API uses `kony.application Namespace` and the following API elements:

Constants

Constant Type	Description
Application Constants	Identifies the location from where the app is launched.
Breakpoint Constants	Checks if the current browser window size has gone beyond highest value of breakpoints list defined.
Runtime Permissions Constants	Report the status of runtime permissions.

Functions

Function	Description
`kony.application.addApplicationCallbacks`	Helps you to register multiple callbacks for the same event.
`kony.application.addBMState`	Adds a specified key and value to the parameter list of the URL of the form.
`kony.application.addGestureRecognizerForAllForms`	Enables the developers to set a gesture recognizer for the specified gesture of the specified widget.
`kony.application.addSettingsMenuItemAt`	Enables you to add a menu item at a given index in the Charm settings menu.
`kony.application.beginBackgroundTask`	Used when you want to run a long running or the asynchronous task in the background of the phone app.

Function	Description
`kony.application.checkPermission`	Checks and returns the permission status of one or more resources.
`kony.application.createSettingsMenu`	Enables you to create a Charm settings menu for an application.
`kony.application.dismissLoadingScreen`	Provides you the ability to dismiss the loading screen displayed earlier
`kony.application.destroyForm`	Destroys the target form.
`kony.application.disableZoomedOutView`	Enables you to disable a zoomed out view set for an application using the previous API.
`kony.application.endBackgroundTask`	Invoked when you are done with an execution of long running tasks in the background.
`kony.application.exit`	Terminates the application.

Function	Description
kony.application.exitLibrary	Provides you the ability to exit the library. After exiting the library, the control moves to the Native app UI.
kony.application.getApplicationBadgeValue	Enables you to read the badge value (if any) attached to the given application icon.
kony.application.getApplicationMode	Enables you to get the application mode.
kony.application.getApplicationState	Checks whether the app is running in the background or not to make UI updates.
kony.application.getAppMenuBadgeValue	Enables you to read the badge value (if any) attached to the specified app menu item.

Function	Description
`kony.application.getAppWindow`	Returns a handle to an AppWindow object.
`kony.application.getBMState`	Retrieves the list of parameters attached to a URL using the above add, set APIs.
`kony.application.getCurrentBreakpoint`	Returns the current breakpoint value.
`kony.application.getCurrentForm`	Returns a handle to the current form.
`kony.application.getCurrentSettingsMenu`	Returns the unique identifier of the current menu that is set through getCurrentSettingsMenu.
`kony.application.getPreviousForm`	Returns a handle to the previous form.
`kony.application.getPreviousSessionParams`	Retrieves the previous session parameters in the application life cycle.

Function	Description
`kony.application.getSettingValue`	Retrieves the current device settings.
`kony.application.invalidateSession`	Invalidates a session on Mobile Web.
`kony.application.isImageTurnedOff`	Gets the status of image settings, which are defined by a particular user, in a web browser.
`kony.application.isInMultiWindowMode`	Returns true if the application is in multi-window mode, and the function returns false if the application is in full-screen mode.
`kony.application.isPopupBlocked`	Gets the status of pop-up settings, which are defined by a particular user, in a web browser.

Function	Description
`kony.application.launchApp`	Launches the application specified by the input URL.
`kony.application.openApplicationSettings`	Opens the application-specific settings or device-level application settings.
`kony.application.openMediaURL`	Launches the native media player and starts playing the media (audio or video) at the specified URL.
`kony.application.openURL`	Opens the web page at the specified URL in the native browser of the mobile device.
`kony.application.openURLAsync`	Opens the web page at the specified URL in the native browser of the mobile device.

Function	Description
kony.application.postAccessibilityNotification	Posts a notification to "assistive" applications
kony.application.registerForIdleTimeout	Specifies if the application must timeout after a defined period of inactivity and also specifies the action after the timeout interval.
kony.application.registerOnKeyPress	Connects an event handler function to a key press event.
kony.application.registerOnSettingsChangeCallback	Listens if any settings have been changed in Native settings applications.
kony.application.removeApplicationCallbacks	Helps you to clear callback functions associated with the specified appstates.

Function	Description
kony.application.removeBMState	Removes a specified key from the parameter list of the URL of the form.
kony.application.removeGestureRecognizerForAllForms	Enables you to remove a specified gesture recognizer for all Forms.
kony.application.removeSecondaryTile	Enables you to remove and unpin a specified secondary tile which was created earlier.
kony.application.removeSeoDataReadyFlag	Clears the flag that caches forms for SEO.
kony.application.removeSettingsMenuItemAt	Enables you to removes the specified App Menu item based on the index.
kony.application.requestPermission	Requests for the end-user consent to access a particular resource.

Function	Description
`kony.application.requestPermissionSet`	Sends a request for a set of permissions. The status of the request is sent back to the user through a callback.
`kony.application.requestReview`	Requests users to provide a rating and to write a review for an app.
`kony.application.resetBMState`	Resets the state associated with the URL of a form.
`kony.application.sendLibraryResultToNativeApp`	Enables you to send an acknowledgment to a Native app that launched this library.
`kony.application.setApplicationBehaviors`	Enables your app to configure its response to various events.

Function	Description
kony.application.setApplicationBadgeValue	Enables you to set a badge value to an application icon on the mobile desktop at the top-right corner of the application icon.
kony.application.setApplicationCallbacks	Captures the callback events for various states of the application
kony.application.setApplicationInitializationEvents	Configures all initialization events such as, preappinit, postappinit, init, appservice, showstartupform and so on.
kony.application.setApplicationLayout	Specifies if the application must have a layout from "left to right" or "right to left".
kony.application.setApplicationMode	Enables you to set the application mode to Native, Hybrid, or Wrapper.

Function	Description
kony.application.setApplicationProperties	Enables you to set properties at the application level.
kony.application.setAppMenuBadgeValue	Enables you to set a badge value to the specified app menu item on the top-right corner of the app menu item.
kony.application.setAppTile	Enables you to set the data for an application tile.
kony.application.setBMState	Sets the bookmark state to the URL.
kony.application.setCheckBoxSelectionImageAlignment	Used to set the alignment of the checkBox selection image.
kony.application.setCurrentSettingsMenu	Uses the unique identifier which represents the Charm settings menu and sets it as current settings menu.

Function	Description
`kony.application.setCurrentAppMenuFont`	Sets the font name and font size of various app menu items in the current app menu.
`kony.application.setDefaultListboxPadding`	Customizes the default paddings applied for a ListBox.
`kony.application.setDefaultTextboxPadding`	Customizes the default paddings applied for a Textbox.
`kony.application.setLibraryHeadlessModeCallback`	Registers a listener or a callback that receives request from a Native app to launch Kony library without UI or in headless mode.

© 2013 by Kony, Inc. All rights reserved

Function	Description
`kony.application.setRespectImageSizeForImageWidgetAlignment`	Sets the ImageWidget width to minimum or maximum according to available width or image width in absence of reference width
`kony.application.setSecondaryTile`	Enables you to create or update data for a secondary tile for an application.
`kony.application.setSeoDataReadyFlag`	Sets a flag indicating that the current form is ready to be cached for search engine optimization.
`kony.application.setZoomedOutView`	Enables you to set a form to be shown to the user when a zoom out gesture is performed.

© 2013 by Kony, Inc. All rights reserved

Function	Description
kony.application.showLoadingScreen	Enables you to display a loading screen (following a certain color schema) to the user while another action is in progress.
kony.application.startForegroundService	Defines the notifications for an app that is running in the background, i.e, an app with which the user is not interacting directly.
kony.application.stopForegroundService	Enables you to stop the foreground service for an application that is running in the background.

Function	Description
kony.application.unregisterForIdleTimeout	Specifies that the application must not timeout after a defined period of inactivity (time difference between the current device time and the last time you clicked on any user interface component).
kony.application.updateForegroundNotification	Enables you to customize and update the existing notifications shown by the foreground service.
kony.application.zoomIn	Enables you to zoom in on an application programmatically.

7.1 Overview

This functions in the Application API handle such tasks as setting application callbacks, starting and stopping background tasks, showing and dismissing the loading screen, and so forth.

Note: All of the Application API functions are in the kony.application namespace. However, the `kony.application namespace` contains some functions that are not part of the Application API. Only the functions listed above are part of the Application API.

During the lifecycle of an application, the mobile device usually triggers several events. The functions in the Application API allow you to listen for these events and override them with application-specific functionality. Your app should register for application events during the application load event of the project or *masterdataload* event of the startup form.

7.1.1 Deep Linking

Applications can use deep linking and the kony.application.launchApp function to launch target apps from the current source app. To use deep linking, first perform the following steps.

Step 1: Register for deep linking in Kony Visualizer.

Register for deep linking by doing the following.

1. Load the project for the target app.
2. From the Kony Visualizer **Project** pane, choose **Project Settings**.
3. In the dialog box that appears, click the **Native** tab.
4. On Windows Phone, select the **Windows Phone** tab. On Windows Tablet, choose the **Windows Tablet** tab.
5. On Windows Phone, select the **Common** tab. On Windows Tablet, click the **Packaging** tab.
6. Enter the deep linking schema name in the **Deeplink URL Scheme** text box. The schema name should be a string that uniquely identifies your app from all other apps. It must be 2-39 characters in length, and it can contain numbers, lowercase characters, periods ('.'), plus signs ('+'), or hyphens ('-'). The string cannot start with a period.

Step 2: In the source app, launch the target app.

Launch the target app from the source app using code similar to the following example.

```
var protocolName = "KonyApp1234567.8";
var data = {
    navigatetoForm ": "
    FrmHome ",
    "
    TexttoShow ": "
    Launched FrmHome by Deeplinking "
};
kony.application.launchApp(protocolName,data);
```

As the example code shows, your source app must use the deep linking protocol name specified in the target app's Kony Visualizer project file. It can also pass whatever data the target app is designed to accept.

Step 3: In the target app, process input parameters as needed.

The target app should process input parameters when it launches, if there are any. To do so, it can use code similar to the following.

```
function AppEvents(eventobject) {
    // Extract the deep linking data from the input parameters.
    var launchParams = eventobject["launchparams"];
    var deeplinkData = launchParams["deeplinkinglaunchparams"];

    // Now deeplinkData contains the same information as the data
    // that was passed to kony.application.launchApp
    var toForm = deeplinkData["navigatetoForm"];
    var frmtext = deeplinkData["TexttoShow"];
    if (toForm == "FrmHome")
        FrmHome.lb1.text = frmtext;
}
```

In this example, the `AppEvents` function extracts the launch parameters from the *eventobject* parameter. The function then retrieves the deep linking data from the launch parameters. Next, it declares two variables to receive the individual pieces of data that the source app passed to the `kony.application.launchApp` function when it launched the target app.

7.2 kony.application Namespace

The kony.application namespace contains the following constants and functions that you can use to manage many aspects of your applications.

7.2.1 Constants

The kony.application namespace provides the following constants.

Application Constants

These application constants are available from V8 SP3 onwards.

Constant	Description
kony.application.APP_LAUNCH_MODE_NORMAL	Constant that identifies if the app was launched in Normal mode.
kony.application.APP_LAUNCH_MODE_PUSH	Constant that identifies if the app was launched from push notifications.
kony.application.APP_LAUNCH_MODE_URL	Constant that identifies if the app was launched from a deep link.
kony.application.APP_LAUNCH_MODE_LIBRARY	Constant that identifies if the app was launched from the library.

Note: The framework sends any one of these application constants in the **launchmode** propertyKey of the object that is passed as a argument to the appService callback .

Breakpoint Constants

Constant	Description
constants.BREAKPOINT_MAX_VALUE	Checks if the current browser window size has gone beyond highest value of breakpoints list defined.

Runtime Permissions Constants

The following constants report the status of runtime permissions.

Constant	Description
kony.application.PERMISSION_DENIED	The app does not have permission to access the resource or file.
kony.application.PERMISSION_GRANTED	The app has permission to access the resource or file.
kony.application.PERMISSION_RESTRICTED	The app has permission to access the resource or file on a restricted basis.

7.2.2 Functions

The kony.application namespace contains the following functions.

kony.application.addApplicationCallbacks

The kony.application.addApplicationCallbacks API helps you to register multiple callbacks for the same event. This API is available from V8 SP4 onwards.

Syntax

```
kony.application.addApplicationCallbacks(callbacksMap)
```

Input Parameters

callbacksMap [Object] - Mandatory

Specifies an Object with key as **appstate** and value as the Map Object (key with value as callback function) for the corresponding appstate. The following appstates are applicable:

App state	Description
isAppLaunchedForInteraction	State which indicates that the application is visible to users for interaction. isAppLaunchedForInteraction is triggered only once in the lifecycle of the app. If the app is already visible to the user by the time the callback is registered, it is immediately triggered irrespective of whether the app is in the foreground or the background. If the same key is used again, you need to throw a "#APP_STATE# callback with #CALLBACK_ID # key already exists" error message. For example, consider a scenario where the isAppLaunchedForInteraction callback with xyz key already exists. If the callbackID is removed or the callback for that respective callbackID is already executed, a duplicate error message is not displayed if the user tries to add the callback with the same callbackID.
onactive	State which indicates that the mobile device is active and the application is running.

App state	Description
oninactive	State which indicates that the mobile device is inactive and the application is running.
onbackground	State which indicates that the application is active and running in the background.
onforeground	State which indicates that the application is active and running in the foreground.
onappterminate	State which indicates that the application has been terminated, and has stopped running.
onkeyboardchange	State which indicates whether a keyboard is deployed for an application.
onpowersourcechange	State which indicates whether a power source is attached to the user's device.

App state	Description
onnetworkchange	This is specific to Mango. State which occurs when there is a change in the following: • **Status**: Indicates the status of the device. The applicable statuses are Connected, Disconnected, Roaming, or Unknown. • **Network**: Indicates the available network on the device. The network statuses are Wireless80211, Ethernet, MobileBroadbandGSM, MobileBroadbandCDMA, or None. • **Date**: Indicates the date on which the event occurs.

Example

```
function functionCallback() {
    kony.print alert("====isAppLaunchedForInteraction callback executed====");
}

var callbacksMapObject = {
    "isAppLaunchedForInteraction": {"
        functionID": functionCallback
    }
};

kony.application.addApplicationCallbacks(callbacksMapObject);
```

Platform Availability

- Android
- iOS
- Windows

kony.application.addBMState

This API adds a specified key and value to the parameter list of the URL of the form.

Syntax

```
kony.application.addBMState(formID, key, value)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form to be bookmarked.
key [String] - Mandatory	Key string representing the LHS of the parameter.
value [String] - Mandatory	Value string representing the RHS of the key-value combination. The value can not be a nested structure.

Example

```
// To set a Bookmark, enter the following
kony.application.addBMState("form1", "About", "page2")
```

```
addbookmark: function() {

    kony.application.addBMState("Form1", "About", "page2");
    alert("A specified key and value are added to the parameter list of the
URL");

},
```

Return Values

None.

Platform Availability

Supported for SPA and Desktop Web.

kony.application.addGestureRecognizer

Using the addGestureRecognizer function, you can set a gesture recognizer for a specified widget.

Syntax

```
kony.application.addGestureRecognizer (gestureType,
gestureConfigParams,onGestureClosure)
```

Input Parameters

gestureType [Number] - Mandatory

Indicates the type of gesture that must be detected on the widget. Following are the possible gestureType values:

- 1 - constants.GESTURE_TYPE_TAP
- 2 - constants.GESTURE_TYPE_SWIPE

- 3 - constants.GESTURE_TYPE_LONGPRESS
- 4 - constants.GESTURE_TYPE_PAN
- 5 - constants.GESTURE_TYPE_ROTATION
- 6 - constants.GESTURE_TYPE_PINCH
- 7 - constants.GESTURE_TYPE_RIGHTTAP

Note:

- RIGHTTAP is applicable only to Windows 8.1 and Windows Desktop/Kiosk platforms.
- ROTATION is not supported on Android.

gestureConfigParams [object] - Mandatory

Specifies a table that has the configuration parameters that are required to setup a gesture recognizer. The configuration parameters vary based on the type of the gesture.

This parameter has the following key-value pairs:

Gesture Type	Configuration Parameter
TAP	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. Default value is 1. • taps [Number] - specifies the maximum number of taps that are allowed for a gesture. The possible values are 1, 2. Default value is 1.

Gesture Type	Configuration Parameter

SWIPE	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. Default value is 1.
	
	• swipedistance [Number] - specifies the distance between the pixel from where the swipe started to the pixel where the swipe stopped (finger is moved up or removed). The default value is 50 pixels.
	
	Note: This parameter is applicable only on Android.
	
	• swipevelocity [Number] - specifies the velocity of the swipe, measured in pixels per second. The default value is 75.
	
	Note: This parameter is applicable only on Android.
	
	• recognizeSimultaneously Boolean - Indicates whether the gesture should recognize simultaneous with other gesture or not. For example, if a swipe gesture is added to a form, and if this form has swipe interaction widgetslike Segment widget. If the gesture config params has recognizeSimultaneously as true, the segment widget when swiped triggers the swipe gesture callback and if the recognizeSimultaneously is false, the segment widget swipe won't trigger the swipe gesture callback.
	
	`For example,`
	`{fingers:1,swipedistance:50,swipevelocity:75}`

Gesture Type	Configuration Parameter
LONGPRESS	• pressDuration [Number] - specifies the minimum time interval (in seconds) after which the gesture is recognized. The default value is 1. This is not applicable to Windows. `For example, {pressDuration:1}`
PAN	• fingers [number] - specifies the minimum number of fingers that are required to recognize this gesture. Default value is 1. • continuousEvents [Boolean] - indicates if callback should be called continuously for every change beginning from the time the gesture is recognized to the time it ends.
ROTATION	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] - indicates if callback must be called continuously for every change beginning from the time the gesture is recognized to the time it ends.
PINCH	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] indicates if callback should be called continuously every change beginning from the time the gesture is recognized to the time it ends.

onGestureClosure [function] - Mandatory

Specifies the function that needs to be executed when a gesture is recognized. This function will be raised asynchronously and has the following signature:

```
onGestureClosure(widgetRef, gestureInfo, context)
```

Parameter	Description
widgetRef	specifies the handle to the widget on which the gesture was recognized.
gestureInfo	Table with information about the gesture. The contents of this table vary based on the gesture type.
context	Table with SegmentedUI row details.

gestureInfo table has the following key-value pairs:

Key	Description
gestureType [number]	Indicates the gesture type
gesturesetUpParams [object]	Specifies the set up parameters passed while adding the gesture recognizer
gesturePosition [number]	Indicates the position where the gesture is recognized. Possible values are: • 1 for TOPLEFT • 2 for TOPCENTER • 3 for TOPRIGHT • 4 for MIDDLELEFT • 5 for MIDDLECENTER • 6 for MIDDLERIGHT • 7 for BOTTOMLEFT • 8 for BOTTOMCENTER • 9 for BOTTOMRIGHT • 10 for CENTER

Key	Description
swipeDirection [number]	Indicates the direction of swipe. Direction is w.r.t the view and not device orientation. This parameter is applicable only if the gesture type is SWIPE. Possible values are: • 1 for LEFT • 2 for RIGHT • 3 for TOP • 4 for BOTTOM
gestureX [number]	specifies the X coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
gestureY [number]	specifies the Y coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
widgetWidth [number]	specifies the width of the widget (in pixels).
widgetHeight [number]	specifies the height of the widget (in pixels).

Key	Description
gestureState[number]	Indicates the gesture state. The gestureState is applicable only for continuous gestures like PAN, ROTATION, and PINCH. • 1 - gesture state begin • 2 - gesture state changed • 3 - gesture state ended
rotation [number]	Rotation of the gesture in degrees since its last change.(Applicable only when gesture type is ROTATION)
velocityX and velocityY	horizontal and vertical component of velocity expressed in points per second. (Applicable only for PAN gesture type)
velocity [number]	velocity of pinch in scale per second. (Applicable only for Pinch gesture)
scale [number]	scale factor relative to the points of the two touches in screen coordinates.

Key	Description
touchType[number]	(Applicable to windows platform only) • 0 - constants.TOUCHTYPE_FINGER • 1 - constants.TOUCHTYPE_PEN • 2 - constants.TOUCHTYPE_MOUSE
translationX and translationY [number]	Cumulative distance as number. (Applicable only for PAN gesture type)

context table has the following key-value pairs:

Key	Description
rowIndex [number]	Row index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)
sectionIndex [number]	Section index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)

Example

```
 addLongPressGesture: function() {
    var getTime = parseInt(this.view.lstbx.selectedKey);

    var longConfig = {
        pressDuration: getTime
    };
    gesturehandle = this.view.flxLongpress.addGestureRecognizer
(constants.GESTURE_TYPE_LONGPRESS, longConfig, this.onLongpressClosure);

}
```

Return Values

String - Reference to the gesture is returned.

Platform Availability

Available on all platforms except Server Side Mobile Web, Windows 7/Kiosk, and Desktop Web.

kony.application.setGestureRecognizerForAllForms

Using the setGestureRecognizerForAllForms function, you can set a gesture recognizer for all the forms.

Syntax

```
kony.application.setGestureRecognizerForAllForms (gestureType,
gestureConfigParams,onGestureClosure)
```

Input Parameters

gestureType [Number] - Mandatory

Indicates the type of gesture that must be detected on the widget. Following are the possible gestureType values:

- 1 - constants.GESTURE_TYPE_TAP
- 2 - constants.GESTURE_TYPE_SWIPE
- 3 - constants.GESTURE_TYPE_LONGPRESS
- 4 - constants.GESTURE_TYPE_PAN
- 5 - constants.GESTURE_TYPE_ROTATION
- 6 - constants.GESTURE_TYPE_PINCH
- 7 - constants.GESTURE_TYPE_RIGHTTAP

Note:

- RIGHTTAP is applicable only to Windows 8.1 and Windows Desktop/Kiosk platforms.
- ROTATION is not supported on Android.

gestureConfigParams [object] - Mandatory

Specifies a table that has the configuration parameters that are required to setup a gesture recognizer. The configuration parameters vary based on the type of the gesture.

Gesture Type	Configuration Parameter
TAP	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. Default value is 1. • taps [Number] - specifies the maximum number of taps that are allowed for a gesture. The possible values are 1, 2. Default value is 1.

Gesture Type	Configuration Parameter
SWIPE	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. Default value is 1. • swipedistance [Number] - specifies the distance between the pixel from where the swipe started to the pixel where the swipe stopped (finger is moved up or removed). The default value is 50 pixels. *Note:* This parameter is applicable only on Android. • swipevelocity [Number] - specifies the velocity of the swipe, measured in pixels per second. The default value is 75. *Note:* This parameter is applicable only on Android. `For example, {fingers:1,swipedistance:50,swipevelocity:75}`
LONGPRESS	• pressDuration [Number] - specifies the minimum time interval (in seconds) after which the gesture is recognized. The default value is 1. This is not applicable to Windows. `For example, {pressDuration:1}`

Gesture Type	Configuration Parameter
PAN	• fingers [number] - specifies the minimum number of fingers that are required to recognize this gesture. Default value is 1. • continuousEvents [Boolean] - indicates if callback should be called continuously for every change beginning from the time the gesture is recognized to the time it ends.
ROTATION	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] - indicates if callback must be called continuously for every change beginning from the time the gesture is recognized to the time it ends.
PINCH	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] indicates if callback should be called continuously every change beginning from the time the gesture is recognized to the time it ends.

onGestureClosure [function] - Mandatory

Specifies the function that needs to be executed when a gesture is recognized. This function will be raised asynchronously and has the following signature:

```
onGestureClosure(widgetRef, gestureInfo, context)
```

Parameter	Description
widgetRef	specifies the handle to the widget on which the gesture was recognized.
gestureInfo	Table with information about the gesture. The contents of this table vary based on the gesture type.
context	Table with SegmentedUI row details.

gestureInfo table has the following key-value pairs:

Key	Description
gestureType [number]	Indicates the gesture type
gesturesetUpParams [object]	Specifies the set up parameters passed while adding the gesture recognizer
gesturePosition [number]	Indicates the position where the gesture is recognized. Possible values are: • 1 for TOPLEFT • 2 for TOPCENTER • 3 for TOPRIGHT • 4 for MIDDLELEFT • 5 for MIDDLECENTER • 6 for MIDDLERIGHT • 7 for BOTTOMLEFT • 8 for BOTTOMCENTER • 9 for BOTTOMRIGHT • 10 for CENTER

Key	Description
swipeDirection [number]	Indicates the direction of swipe. Direction is w.r.t the view and not device orientation. This parameter is applicable only if the gesture type is SWIPE. Possible values are: • 1 for LEFT • 2 for RIGHT • 3 for TOP • 4 for BOTTOM
gestureX [number]	specifies the X coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
gestureY [number]	specifies the Y coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
widgetWidth [number]	specifies the width of the widget (in pixels).
widgetHeight [number]	specifies the height of the widget (in pixels).

Key	Description
gestureState[number]	Indicates the gesture state. The gestureState is applicable only for continuous gestures like PAN, ROTATION, and PINCH. • 1 - gesture state begin • 2 - gesture state changed • 3 - gesture state ended
rotation [number]	Rotation of the gesture in degrees since its last change.(Applicable only when gesture type is ROTATION)
velocityX and velocityY	horizontal and vertical component of velocity expressed in points per second. (Applicable only for PAN gesture type)
velocity [number]	velocity of pinch in scale per second. (Applicable only for Pinch gesture)
scale [number]	scale factor relative to the points of the two touches in screen coordinates.

Key	Description
touchType[number]	(Applicable to windows platform only) • 0 - constants.TOUCHTYPE_FINGER • 1 - constants.TOUCHTYPE_PEN • 2 - constants.TOUCHTYPE_MOUSE
translationX and translationY [number]	Cumulative distance as number. (Applicable only for PAN gesture type)

context table has the following key-value pairs:

Key	Description
rowIndex [number]	Row index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)

Key	Description
sectionIndex [number]	Section index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)

Example

```
//Defining a function
function formGesture(widgetID, gestureInfo) {
    var y = kony.type(gestureInfo); //expected value of y = table
    var z = kony.type(gestureInfo.gesturesetUpParams); //expected values of z
= table
    var a = gestureInfo.gestureType;
    var b = gestureInfo.gesturesetUpParams;
    var c = gestureInfo.gesturePosition;
    var d = gestureInfo.gestureX;
    var e = gestureInfo.gestureY;
    var f = gestureInfo.widgetWidth;
    var g = gestureInfo.widgetHeight;
    kony.print("*******************************************");
    if (kony.os.toNumber(gestureInfo.gestureType) == 2) {
        h = gestureInfo.swipeDirection;
        kony.print("swipe direction is: " + h);
    } else {
        h = "";
    }
    if (kony.os.toNumber(a) == 1) {
        b1 = "fingers: " + gestureInfo.gesturesetUpParams.fingers;
        b2 = "taps: " + gestureInfo.gesturesetUpParams.taps;
        kony.print("" + b1 + "" + b2);
    } else if (kony.os.toNumber(a) == 2) {
        b1 = "fingers :" + gestureInfo.gesturesetUpParams.fingers;
        b2 = "";
        kony.print("" + b1 + "" + b2);
    } else if (kony.os.toNumber(a) == 3) {
        b1 = "pressduration:" + gestureInfo.gesturesetUpParams pressDuration;
        b2 = "";
        kony.print("" + b1 + "" + b2);
    }

    kony.print("widget id is: " + widgetID[id]); //will print the widgetID.
    //To print widgetID use widgetID.id
    kony.print("type of gestureInfo is: " + y);
    kony.print("type of gesturesetUpParams is: " + z);
    kony.print("gestureType is: " + a); //gestureType=1 or 2 or 3
    kony.print("gesturesetUpParams is: " + b.fingers);
  /*gesturesetUpParams
    = {
        fingers = 1, taps = 1
    }
    or {
        fingers = 1, taps = 2
    }
    or {
        fingers = 1
    }
    or {
        pressDuration = 1
    }*/
    kony.print("gesturePosition is: " + c); //gesturePosition=1 or 2 or 3 or
.....9
    kony.print("gestureX is: " + d); //ex: gestureX=30
    kony.print("gestureY is: " + e); //ex: gestureY=100
    kony.print("widgetWidth is: " + f); //ex: widgetWidth=320
    kony.print("widgetHeight is: " + g); //ex: widgetHeight=28
    //gesturePosition, gestureX, gestureY, widgetWidth, widgetHeight params
are not applicable in android
    kony.print("*******************************************");
}

function callbackSingleTapGesture() {
    var x = {
        fingers: 1,
        taps: 1
    };
    try {
        kony.application.setGestureRecognizerForAllForms(1, x,
            formGesture);
    } catch (err) {
        alert(typeof err);
        alert("error in function callbackSingleTapGesture: " + err.message);
    }
}
```

Return Values

String - Reference to the gesture is returned.

Platform Availability

Available on all platforms except Server Side Mobile Web, Windows 7/Kiosk, and Desktop Web.

kony.application.addSettingsMenuItemAt

This API enables you to add a menu item at a given index in the Charm settings menu.

Syntax

```
kony.application.addSettingsMenuItemAt (id, index, menuSettings)
```

Input Parameters

Parameter	Description
id [String] - Mandatory.	Identifier of the Charm setting menu created.
index [Number] - Mandatory.	The index at which the menu item must be added. The index value lies between 0 and n-1. If the index is beyond the current length of the Charm menu items then the item is added to the end.
menuSettings [Hash table] - Mandatory	The menuSettings hash table comprises the following key-value pairs: • **id**: ID of the Charm menu item. • **text**: Name of the menu item. • **onClick**: onclick event to be executed for the menu item.

Example

To add a menu item at a given index, enter the following:

```
//The below function is the callback function for onClickClosure event of app
menu item with id "appmenuitemid3".
function onClickClosure3() {
    //proceed with the logic
}

var settingsMenuItem1 = {
    id: "about",
    text: "About",
    onClick: onClickClosure3
};

//Adding the above app menu item at the index 3.
kony.application.addSettingsMenuItemAt("accountMenu", 3, settingsMenuItem1);
```

Return Values

None.

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.beginBackgroundTask

In some scenarios apps are launched in the background to perform some long running task. A typical example, could be when a watch app requests its parent app (phone app) for some information, the phone app is launched in the background. If the phone app needs to run a long running task in order to serve the requirement of the watch app, it is recommended to start a background task. This API ensures iOS does not suspend the phone app when running in the background that take too much time or resources for execution.

This API is used when you want to run a long running or the asynchronous task in the background of the phone app. When the long running task is completed, you must end the background task using the API `kony.application.endBackgroundTask`.

Syntax

```
kony.application.beginBackgroundTask(taskID, callback)
```

Input Parameters

Parameter	Description
taskID [String] - Mandatory	Specifies the unique identifier for the task.
callback [function] - Optional	Specifies the callback that gets executed in the background before iOS suspends the app. You can use this callback to save information or clean up resources before your app gets suspended.

Example

```
function ExpirationHandlercallback(){
//your logic to clear resources or save data.
} function jsfunction(userInfo, replyObj){

    var taskID = kony.application.beginBackgroundTask("TaskName",
ExpirationHandlercallback);

    //Long running task.

    kony.application.endbackgroundTask(taskID);

}
```

Return Values

This API returns `taskID` as a number that is used as an input for `kony.application.endBackgroundTask` API.

Platform Availability

Available on iPhone and iPad.

kony.application.checkPermission

Checks and returns the permission status of one or more resources.

Syntax

```
kony.application.checkPermission(resourceId[constant/String], options
[JSObject])
```

Input Parameters

Parameter	Description
resourceId [constant/String] - Mandatory	Specify the ID of the resource or name of the permission (only for Android) for which you want to check the status. You can specify either a String (permission name) or an integer (resourceId) value. The feature to specify the name of the permission as a String is applicable only for Android. For instance, you can query a Native Android permission from the AndroidManifest.xml file by specifying the String directly: "android.permission.READ_PHONE_STATE".
options [JSObject] - Optional	Specify the additional option to identify the exact resource of which you want to know the status. This is a platform-specific key. For more information, refer to Resource ID.

Example 1

```
var options = {
    isAccessModeAlways: true
};
var result = kony.application.checkPermission(kony.os.RESOURCE_LOCATION,
options);
if (result.status = = kony.application.PERMISSION_DENIED) {
    kony.application.requestPermission();
} else if (result.status = kony.application.PERMISSION_GRANTED) {
    kony.location.getCurrentPosition();
}
```

Example 2

```
< uses - permission  android: name = "android.permission.READ_PHONE_STATE" / >
var result = kony.application.checkPermission("android.permission.READ_PHONE_
STATE");
if (result.status = = kony.application.PERMISSION_DENIED) {
    kony.application.requestPermission();
} else if (result.status = kony.application.PERMISSION_GRANTED) {
    kony.location.getCurrentPosition();
}
```

Return Values

JSObject

A JS Object contains the authorization status of the requested resource. The returned JSObject contains the following keys:

Return value	Description
status [constant]	Resource status constant which indicates the overall status of the resource authorization. For more information, refer to Permission Status.

Return value	Description
canRequestPermission [Boolean]	Indicates whether you can request for the permissions or not in case the value of the status is PERMISSION_DENIED. In the iOS platform, authorization for a resource can be requested only once. For more information, refer to Permission model in iOS. In the Android platform, the app can request for the permissions even though the status return value is PERMISSION_DENIED or direct the user to app settings to turn on or off the authorization.

Platform Availability

- Android
- iOS
- Windows 10
- SPA

kony.application.createSettingsMenu

This API enables you to create a *Charm settings* menu for an application.

Syntax

```
kony.applicationcreateSettingsMenu (id, menuSettings)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Identifier of the Charm setting menu created.

Parameter	Description
menuSettings [Hash table] - Mandatory	The menuSettings hash table comprises the following key-value pairs: • **id**: ID of the Charm menu item. • **text**: Name of the menu item.

Example

```
//To create a Charm settings menu, enter the following
var settingsMenuItem1 = {
    id: "about",
    text: "About"
};
var settingsMenuItem2 = {
    id: "help",
    text: "Help"
};
var settingsMenu = [settingsMenuItem1, settingsMenuItem2];
kony.application.createSettingsMenu("mysettingsmenu", settingsMenu);
```

Return Values

None.

Special Considerations

If a Charm setting menu is already created with the identifier passed, a new Charm setting menu will be created and the old Charm setting menu will be replaced with the new one. The same holds true for menu items as well.

At least one menu item must be present in the Charm settings menu created. A Charm settings menu with no menu items is invalid.

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.dismissLoadingScreen

This API provides you the ability to dismiss the loading screen displayed earlier. If there is no loading screen, this API has no affect.

Syntax

```
kony.application.dismissLoadingScreen()
```

Example

```
kony.application.dismissLoadingScreen();
```

Input Parameters

None

Return Values

None.

Platform Availability

Available on all platforms except Mobile Web.

kony.application.destroyForm

Destroys the target form.

Syntax

```
kony.application.destroyForm(
    friendlyName);
```

Input Parameters

Parameter	Description
friendlyName	A string containing the friendly name of the form to be destroyed.

Example

```
kony.application.destroyForm("Form1");
```

Return Values

None.

Remarks

The function destroys both the target form and its form controller.

kony.application.disableZoomedOutView

This API enables you to disable a zoomed out view set for an application using the previous API.

Syntax

```
kony.application.disableZoomedOutView()
```

Example

```
//Disabling zoomout on an application
function zoomout() {
    kony.application.disableZoomedOutView();
}
```

Input Parameters

None

Return Values

None.

Platform Availability

Windows 8

kony.application.endBackgroundTask

This API is invoked when you are done with an execution of long running tasks in the background. The return value of the API `kony.application.beginBackgroundTask` is used as the input parameter for this API.

Syntax

```
kony.application.endBackgroundTask()
```

Input Parameters

Parameter	Description
taskID [Number] - Mandatory	Specifies the identifier returned by the `kony.application.beginBackgroundTask` API.

Example

```
function ExpirationHandlercallback(){
    //your logic to clear resources or save data.
}
function jsfunction(userInfo, replyObj)
{
    var taskID = kony.application.beginBackgroundTask("TaskName",
ExpirationHandlercallback);
    //Long running task.
    kony.application.endbackgroundTask(taskID);
}
```

Return Values

None

Platform Availability

Available on iPhone and iPad.

kony.application.exit

This API terminates the application.

Syntax

```
kony.application.exit()
```

Example

```
function exit() {
    try {
        kony.application.exit();
    } catch (Error) {
        alert("Exception While getting exiting the application  : " + Error);
    }
}
```

Input Parameters

None

Return Values

None

Platform Availability

Available on all platforms except for Server side Mobile Web and Desktop Web. SPA supports only iPhone, iPad, and Windows 8 Tablet.

kony.application.exitLibrary

This API provides you the ability to exit the library. After exiting the library, the control moves to the Native app UI. This API is available from V8 SP3 onwards.

For more information on Library mode, click here.

Syntax

```
kony.application.exitLibrary()
```

Example

```
kony.application.exitLibrary();
```

Input Parameters

None

Return Values

None.

Platform Availability

- Android
- iOS

- Windows10
- Windows10 Mobile

Note: The kony.application.exitLibrary API is only applicable when the app is launched in Library mode.

kony.application.getApplicationBadgeValue

This API allows you to read the badge value (if any) attached to the given application icon. If the applications icon does not have any badge value attached to it, this API returns an empty string.

Syntax

```
kony.application.getApplicationBadgeValue()
```

Input Parameters

None

Example

```
function getApplicationBadgeValue(){
       /*Get the ApplicationBadgeValue from the  application icon on the mobile
desktop at the top-right corner of the application icon.*/
       kony.application.getApplicationBadgeValue();
}
```

```
gettingBadge: function() {
    var badge = kony.application.getApplicationBadgeValue();
    alert("The badge value is " + badge);
},
```

Return Values

Return Value	Description
badgeValue [String]	Returns the badge value applied to the application icon If the application icon has no badge value attached to it, it returns null/nil.

Platform Availability

Available only on iPhone and iPad.

kony.application.getApplicationMode

This API enables you to get the application mode.

Syntax

```
kony.application.getApplicationMode()
```

Example

```
function getApplicationMode() {
       kony.application.getApplicationMode();
}
```

Input Parameters

None

Return Values

Integer Constant

- 1: constants.APPLICATION_MODE_NATIVE
- 2: constants.APPLICATION_MODE_HYBRID
- 3: constants.APPLICATION_MODE_WRAPPER

Special Considerations

This API should be called only in preappinit or before the execution of preappinit. If this API is invoked elsewhere in the program, it is invalid and leads to undefined behavior.

Exceptions

Error

Platform Availability

Available on all platforms.

kony.application.getApplicationState

iOS prohibits UI updates when the app is running in the background. Using this API you can check whether the app is running in the background or not to make UI updates. The possible application states are active, inactive, or background.

Syntax

```
kony.application.getApplicationState()
```

Example

```
//Sample code to get the application state
function util() {
    var appState = kony.application.getApplicationState();
    if (appState = constants.APPLICATION_STATE_BACKGROUND) {
        kony.print(appState);
    }
}
```

Input Parameters

None

Return Values

This API returns the following constant values:

- APPLICATION_STATE_ACTIVE
- APPLICATION_STATE_INACTIVE
- APPLICATION_STATE_BACKGROUND

Platform Availability

Available on iPhone and iPad.

kony.application.getAppMenuBadgeValue

This API enables you to read the badge value (if any) attached to the specified app menu item. If the specified app menu item does not have any badge value attached to it, the API returns an empty string.

Syntax

```
kony.application.getAppMenuBadgeValue(appmenuID, menuItemID)
```

Input Parameters

Parameter	Description
appMenuId [String] - Mandatory	If you are setting the badge for an app menu item that was created dynamically, use the same ID that was used to create the app menu item.If you are setting the badge for an app menu item that was created from the IDE, use the ID available in the generated script file.
menuItemId [String] - Mandatory	Identifier of the app menu item from which the badge value is to be read.

Example

```
function getAppMenuBadgeValue() {
    //Get the AppMenuBadgeValue for the menu item with id:"appmenuitemid3"
 kony.application.getAppMenuBadgeValue("accountMenu", "appmenuitemid3");
}
```

```
onClickMenuItem1: function() {
    alert("The Badge Value of Accounts App Menu Item is " +
kony.application.getAppMenuBadgeValue("SampleAppMenu", "appmenuitemid1"));

},
onClickMenuItem2: function() {
    alert("The Badge Value of Examination App Menu Item is " +
kony.application.getAppMenuBadgeValue("SampleAppMenu", "appmenuitemid2"));

},
```

Return Values

Return Value	Description
badgeValue [String]	Returns the badge value applied to the specified app menu. If the specified app menu has no badge value attached to it, it returns an empty string.

Platform Availability

Available only on iPhone and iPad.

kony.application.getAppWindow

This API returns a handle to an AppWindow object. This object allows you to switch between application modes and to query and update properties of the app window.

Syntax

```
kony.application.getAppWindow()
```

Example

```

var appwindow = kony.application.getAppWindow();
alert(appwindow.title);
```

Input Parameters

None

Return Values

A single handle to an AppWindow object.

Platform Availability

Windows 10

kony.application.getBMState

This API retrieves the list of parameters attached to a URL using the above add, set APIs.

Syntax

```
kony.application.getBMState(formID)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form for which the parameters of the URL have to be fetched.

Example

```
// To fetch a list of parameters, enter the following
kony.application.getBMState("form1");
```

```
getbookmark: function() {

    var a = kony.application.getBMState("Form1");
    alert(" The list of parameters attached to the URL are " + JSON.stringify
(a));
},
```

Return Values

A JSON structure representing key-values of various parameters attached to the URL string of the given form.

Platform Availability

Supported for SPA and Desktop Web.

---

kony.application.getCurrentBreakpoint

This API returns the current breakpoint value.

Syntax

```
kony.application.getCurrentBreakpoint()
```

Input Parameters

None

Return Values

This API returns the current breakpoint

If window size exceeds the highest of breakpoints list, returns **constants.BREAKPOINT_MAX_VALUE**.

Platform Availability

Desktop web

kony.application.getCurrentForm

This API returns a handle to the current form. The form handle is useful when you are sending analytics to the analytic engine.

Note: For iOS, kony.application.getCurrentForm() returns the form id of current form. This rule does not apply for form used as cameraOverlay.

Use Case

You can use this API in the following scenarios:

1. When you want to send analytics to the analytic engine.
2. When you want to use the handle to the current form in any other function.

Syntax

```
kony.application.getCurrentForm()
```

Example

```
function getCurrentForm() {
       //Get the current form
       var currentForm = kony.application.getCurrentForm();
       //Alert the current form
       alert("currentForm is::" + currentForm);
}
```

Input Parameters

None

Return Values

currentForm - Object

This API returns the current form.

Rules and Restrictions

The following are the guidelines applicable for the API:

- If you invoke this API in the `preshow` method of the application, it does not have any affect.
- If you invoke this API in the `appinit` method of the application, nil is returned.

UI Behavior

None

Platform Availability

Available on all platforms.

kony.application.getCurrentSettingsMenu

This method returns the unique identifier of the current menu that is set through getCurrentSettingsMenu.

Syntax

```
kony.application.getCurrentSettingsMenu()
```

Input Parameters

None

Example

```
//To get the unique identifier a Charm settings menu, enter the following
kony.application.getCurrentSettingsMenu();

//Alert the Current Charm Settings menu
alert("Current charm menu id is: " + currCharmMenuId);
```

Return Values

Return value	Description
Unique Identifier	Identifier of the Charm setting menu to be set.

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.getPreviousForm

This API returns a handle to the previous form.

Use Case

You can use this API in the following scenarios:

1. When you want to send analytics to the analytic engine.
2. When you want to use the handle to the current form in any other function.

Syntax

```
kony.application.getPreviousForm()
```

Example

```
function getPreviousForm() {
    //Get the Previous form
    var previousForm = kony.application.getPreviousForm();
    //Alert the Previous form
    alert("previousForm is::" + previousForm);
}
```

Input Parameters

None

Return Values

Return Value	Description
previousForm-Object	API returns the previous form handle.

Rules and Restrictions

The following are the guidelines applicable for the API:

- If you invoke this API in `preshow` event of the application, you get unpredictable results.
- If you invoke this API in the `appinit` method of the application, nil is returned.
- If you use the Back button on the mobile web browser, this API does not return the handle to the previous form.
- If you use this API on the first form of the application it returns null/nil.

UI Behavior

None

Platform Availability

Available on all platforms.

kony.application.getPreviousSessionParams

This API retrieves the previous session parameters in the application life cycle.

Important: You must use the *kony.application.invalidateSession("frmName", sessionParams)* API before using *kony.application.getPreviousSessionParams API.*

Syntax

```
kony.application.getPreviousSessionParams()
```

Example

```
//The following function is for getPreviousSessionParams API
//You must use the invalidateSession("frmName", sessionParams) API before
using getPreviousSessionParams API.
function invalidateSession() {
    //invalidating the session
    var sessionParams = {};
    sessionParams.category = "news";
    sessionParams.country = "US";
    kony.application.invalidateSession("frmName", sessionParams);

}

function getPreviousSessionParams() {
    //Collecting all the params in to an object
    var mySessionParams = kony.application.getPreviousSessionParams();
    kony.print("mySessionParams are" + mySessionParams);
}
```

Input Parameters

None.

Return Values

Return Value	Description
listOfParams [Array]	Returns a table of the previous session parameters that is passed from the invalidateSession API.

Platform Availability

Available on Mobile Web

kony.application.getSettingValue

You can use this API to retrieve the current device setting. You must pass the setting that you want to query, in the input parameter of this API.

Syntax

```
kony.application.getSettingValue(setting, args)
```

Parameters

setting[String] - Mandatory

The setting that you want to query must be passed in this parameter. the following settings are supported by various platforms:

- **Android**: "location", "device_locale", "time_zone", and "wifi".

Settings	Description
location	This key is used to identify the current "location" status of the device. Calling the kony.application.getSettingValue API with this key returns a JavaScript object, which has the following key values: • gps_provider (key): This key has value as "true" or "false", which informs that the gps_provider has been enabled or disabled. • network_provider (key): This key has value as "true" or "false", which informs that the network_provider has been enabled or disabled.

Settings	Description
device_locale	This key is used to identify the current locale of the device. Calling the kony.application.getSettingValue API with this key returns a JavaScript object, which has the following key value: • device_locale (key): This key has the current locale value of the device.
time_zone	This key is used to identify the current time zone of the device. Calling the kony.application.getSettingValue API with this key returns a JavaScript object, which has the following key value: • time_zone (key): This key has the current time zone value of the device.

Settings	Description
wifi	This key is used to identify the current wifi state of the device. ***Note:*** To query this setting, you must add the following permission to the AndroidManifest.xml file: **<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />** Calling the kony.application.getSettingValue API with this key returns a JS object, which has the following key values: • wifi_current_state (key): This key has value as current wifi state value of the device. The list of wifi states is as follows: • kony.settings.WIFI_DISABLING: Wifi is disabling. Constant Value: 0 • kony.settings.WIFI_DISABLED: Wifi is disabled. Constant Value: 1 • kony.settings.WIFI_ENABLING: Wifi is enabling. Constant Value: 2 • kony.settings.WIFI_ENABLED: Wifi is enabled. Constant Value: 3 • kony.settings.WIFI_UNKNOWN: Wifi state is unknown. Constant Value: 4

- **Windows**: "color"

Settings	Description
color	This key is used to identify the current system color values. To get the current system color, you must provide the **args** parameter in the kony.application.getSettingValue API. Calling the kony.application.getSettingValue API with this key returns a hexadecimal string value.

- **iOS:** "applicationAppearanceStyle"

Settings	Description
applicationAppearanceStyle	iOS 13 provides support for dark mode in iOS devices. This key enables you to identify the appearance mode of iOS devices. When the kony.application.getSettingValue API is called with this key, any of the following values are returned: • kony.application.APPEARANCESTYLE_DARK: The current mode of appearance is dark mode. • kony.application.APPEARANCESTYLE_LIGHT: The current mode of appearance is light mode. • kony.application.APPEARANCESTYLE_UNKNOWN: The current mode of appearance is unknown. This is applicable from Kony Visualizer V8 SP3 FP 58 and V8 SP4 FP 45. ***Note:*** By default, the dark and light modes are supported for apps that are built on iOS 13 devices. If you want to limit the appearance of your app to a single mode, navigate to: <app_name>/resources/common/infoplist_configuration.json. Then, add the `UIUserInterfaceStyle` key with the dark or light value as required.

args [object [ ]] - Optional

This parameter is applicable for Windows, and is an array that should contain any of the following constants:

- kony.SystemColorType.Background
- kony.SystemColorType.Foreground
- kony.SystemColorType.AccentDark3
- kony.SystemColorType.AccentDark2
- kony.SystemColorType.AccentDark1
- kony.SystemColorType.Accent
- kony.SystemColorType.AccentLight1
- kony.SystemColorType.AccentLight2
- kony.SystemColorType.AccentLight3
- kony.SystemColorType.Complement

Example

Example 1:

```
alert(kony.application.getSettingValue("location"));
```

Example 2:

```
var args = [kony.SystemColorType.Background];
alert(kony.application.getSettingValue("color", args));
```

Example 3:

```
function getAppearenceStyle() {
 var themeApearenceStyle =
kony.application.getSettingValue("applicationAppearanceStyle");
```

```
 switch (themeApearenceStyle) {
 case kony.application.APPEARANCESTYLE_DARK:
 alert("APPEARENCE STYLE : dark mode");
 break;
 case kony.application.APPEARANCESTYLE_LIGHT:
 alert("APPEARENCE STYLE : light mode");
 break;
 case kony.application.APPEARANCESTYLE_UNKNOWN:
 alert("APPEARENCE STYLE : unknown mode");
 break;
 }
}
function callback(params) {
 switch (params.setting) {
 case "applicationAppearanceStyle":
 switch (params.applicationAppearanceStyle) {
 case kony.application.APPEARANCESTYLE_DARK:
 //alert("selected dark mode");
 kony.theme.setCurrentTheme("darkTheme",
onThemeCallback, onThemeCallback);
 break;
 case kony.application.APPEARANCESTYLE_LIGHT:
 //alert("selected light mode");
 kony.theme.setCurrentTheme("lightTheme",
onThemeCallback, onThemeCallback);
 break;
 case kony.application.APPEARANCESTYLE_UNKNOWN:
 //alert("selected unknown mode");
 break;
 }
 break;
 }
 function onThemeCallback() {
 //alert("theme callback");
 }
}
```

```
kony.application.registerOnSettingsChangeCallback(["applicationAppearanceSty
le"], callback);
```

Platform Availability

- Android
- Windows
- iOS

---

kony.application.invalidateSession

This API explicitly invalidates a session on Mobile Web.

> *Important:* This API is applicable only on Mobile Web platforms

Use Case

You can use this API for Financial applications where you want to completely invalidate the session on execution of specific functions. For example, logout function. When the user logs out from the application, you can invoke this API to invalidate the current session and initiate a new session before navigating to any other form. In this case, all the session data is cleared.

Syntax

```
kony.application.invalidateSession(ormName,params)
```

Input Parameters

Parameter	Description
formName [Handle to the form] - Optional	Specifies the form to which the application must navigate after the session is invalidated
params [Function] - Optional	Specifies a table of parameters that you want to pass to the new session

Example

```
//The following function is for invalidateSession API
function invalidateSession() {
       //invalidating the session
       kony.application.invalidateSession();
}
```

Return Values

This API navigates to the specified form or remains on the same form if there is no form specified.

Rules and Restrictions

When the current session is invalidated, the Mobile Web framework will start the application lifecycle again. This API invokes *appinit* and then navigates the user to the startup form or any form that is associated with the logout function.

Platform Availability

Available on Mobile Web.

---

kony.application.isImageTurnedOff Function

This API is used to get the status of image settings, which are defined by a particular user, in a web browser.

Syntax

```
kony.application.isImageTurnedOff(imageCb)
```

Input Parameters

Parameter	Description
imageCb	The kony.application.isImageTurnedOff API takes the **imageCb** function as an argument, and returns either **true** or **false** based on the browser settings for images.

Example

```
function getImageStatus() {
    kony.application.isImageTurnedOff(imageCb);
}

function imageCb(param) {
    if (param === true) {
        DesktopApis.lbl1.text = "Image settings in web browser is disabled";
    } else {
        DesktopApis.lbl1.text = "Image settings in web browser is enabled";
    }

    DesktopApis.forceLayout();

}
```

Return Values

- true: If the user disables the display of images from the browser settings, the kony.application.isImageTurnedOff API returns true.
- false: If the user enables the display of images in the browser, the kony.application.isImageTurnedOff API returns false.

Platform Availability

- Desktop Web

kony.application.isInMultiWindowMode Function

Returns true if the application is in multi-window mode, and the function returns false if the application is in full-screen mode.

Syntax

```
kony.application.isInMultiWindowMode()
```

Example

```
function checkMultiWindowMode() {
var isInMultiWindow = kony.application.isInMultiWindowMode();
kony.print("Multi Window Mode : " + isInMultiWindow);
}
```

Input Parameters

None.

Return Values

Boolean value.

Platform Availability

- Android 7.0 and later

kony.application.isPopupBlocked Function

This API is used to get the status of pop-up settings, which are defined by a particular user, in a web browser.

Syntax

```
kony.application.isPopupBlocked(popupCb)
```

Input Parameters

Parameter	Description
popupCb	The kony.application.isPopupBlocked API takes the **popupCb** function as an argument, and returns either **true** or **false** based on the browser settings for pop-ups.

Example

```
function getPopupStatus() {
    kony.application.isPopupBlocked(popupCb);
}

function popupCb(param) {
    if (param === true) {
        Form1.lbl1.text = "Pop-up blocker is enabled. Please do not allow pop-
ups from this website.";
    } else {
        Form1.lbl1.text = "Popup blocker is turned off.";
    }

    Form1.forceLayout();
}
```

Return Values

Return Value	Description
true	If the user disables the display of pop-ups from the browser settings, the kony.application.isPopupBlocked API returns true.
false	If the user enables the display of pop-ups in the browser, the kony.application.isPopupBlocked API returns false.

Platform Availability

- Desktop Web

kony.application.launchApp Function

Launches the application specified by the input URL.

Syntax

```
kony.application.launchApp(
    protocolName,
    data)
```

Input Parameters

Parameter	Description
protocolName	A string that you provide in Kony Visualizer that specifies the URL scheme . For more information, see Deep Linking.

Parameter	Description
data	A JavaScript object that contains key-value pairs specifying the data that is needed to launch the application. The key-value pairs are defined as whatever is supported by the target app.

Example

```
var protocolName= "KonyApp1234567.8";
var data = {
    "navigatetoForm": "FrmHome",
    "TexttoShow": "Launched FrmHome by Deeplinking"
};
kony.application.launchApp(protocolName,data);
```

Return Values

None.

Remarks

This function uses deep linking to launch an application. For more information, see Deep Linking.

Platform Availability

Available in Windows Phone 8 and later, Windows 8.1 and later, Windows 10 Tablet/Mobile.

---

kony.application.openApplicationSettings

Opens the application-specific settings or device-level application settings.

You may need to direct the end-user to application settings to manually enable or disable a permission for the app to access a particular resource. This function is required when the end-user had denied the permission when the app prompted with a dialog box, and later wants the app to access the resource. For example, if your app wants to access the user's contacts - so the app displayed a dialog box with "Allow" and "Deny" options, asking end-user to grant permission for the first time. The end-user tapped the "Deny" option and the app cannot access the user's contacts. Later, after some point of time, if end-user wants the app access the user's contacts; at that time, you can call the openApplicationSettings API that allows the user to navigate to the application settings screen, and then grant the required permission to the app.

The behavior of the openApplicationSettings API in different platforms:

- **Windows 10**: There is no provision to open the application-level settings. The openApplicationSettings API accepts the resourceid as an optional parameter that helps open the resource-specific settings screen. If the resourceid is not provided, results in unexpected behavior.
- **iOS**: Opens the application-level settings screen showing the access status of the resource. The end-user can turn on or off the access to the resource from the app. The resourceid parameter is ignored in the iOS platform.
- **Android**: Opens the application-level settings screen showing the access status of the resource. The end-user can turn on or off the access to the resource from the app. The resourceid parameter is ignored in the iOS platform.

Syntax

```
kony.application.openApplicationSettings(resourceId[const])
```

Input Parameters

Function	Description
resourceId [constant] - Optional	Specify the resource ID of the resource that you want open its settings. The parameter works only for Windows 10. For more information, refer to Resource ID.

Example

```
kony.application.openApplicationSettings(kony.os.RESOURCE_CONTACTS);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10

kony.application.openMediaURL

This API launches the native media player and starts playing the media (audio or video) at the specified URL. The media server provides the appropriate media content depending upon the device (for example, iPhone, etc). This API is not applicable on SPA.

Use Case

You can use this API when you want to access and use multimedia from an external URL or server.

Syntax

```
kony.application.openMediaURL(URL)
```

Input Parameters

Parameter	Description
URL [String] - Mandatory	Specifies the URL that points to the audio/video file

Example

```
function openMediaURL() {
    /*Launching the native media player and starts playing the media(audio or
video) at the
 URL:"http://www.boisestatefootball.com/sites/default/files/videos/original/01%
20-%20coach%20pete%20bio_4.mp4" */
    kony.application.openMediaURL
 ("http://www.boisestatefootball.com/sites/default/files/videos/original/01%20-
%20coach%20pete%20bio_4.mp4");
}
```

Return Values

None

Implementation Details

This API assumes that the media is available at the specified URL. The API does not check for the availability of the media at the specified location. The responsibility lies on the developer to ensure that appropriate media is available at the referred URL.

Any errors related to the type of media or the availability of the media are handled by the native media player. The errors are not propagated back to the application.

Note: For more information on media formats, see:

- Android: http://developer.android.com/guide/appendix/media-formats.html

  You have to provide an absolute path of the video file in the URL. For example, http://www.boisestatefootball.com/sites/default/files/videos/original/01%20-%20coach%20pete%20bio_4.mp4
  or the URL should be in 'rtsp' format if it is a YouTube video.
- iPhone: Apple Documentation
- Windows Phone (7.5 and 8.0): http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff462087(v=vs.105).aspx
- Windows 8 tablet: Supported File Types

Platform Availability

Available on all platforms except SPA and Windows 7 / Kiosk.

kony.application.openURL

This API opens the web page at the specified URL in the native browser of the mobile device.

Use Case

You can use this API when you want to access an external web page within the application. With this API, you can open a web page without using a browser widget in your application.

Syntax

```
kony.application.openURL(URL)
```

Input Parameters

Parameter	Description
URL [String] - Mandatory	Specifies the URL that points to the external web page

Example

```
//The following function is for openURL API
  function openURL() {
         //Accessing an external web page : http://www.google.com
         kony.application.openURL("http://www.google.com");
       }
```

Return Values

None

Implementation Details

When you use this API, the behavior of the application is as follows on different platforms:

Platform	Behavior
iPhone	• On 4.0 and above versions, the application opens the specified URL in the native browser and the application goes into background • On versions below 4.0, the application opens the specified URL in the native browser and the application exits • For opening the maps application of iOS, please using following URLs: ◦ For iOS 5 :http://maps.google.com/maps ◦ For iOS 6:http://maps.apple.com/maps
Android	The application opens the specified URL in the native browser and the application goes into background
Windows Phone/Windows Kiosk/Mango	The application opens the specified URL in the native browser and the application exits

Platform	Behavior
SPA	• The URL opens in the browser • The application redirects the existing browser instance to the new URL
Symbian	The application opens the specified URL in the native browser and the application goes into background
Palm	The application opens the specified URL in the native browser and the application goes into background

Platform Availability

Available on all Rich Client platforms.

kony.application.openURLAsync

This API opens the web page at the specified URL in the native browser of the mobile device. The openURLAsync API is the asynchronous counterpart of the kony.application.openURL API.

Use Case

You can use this API when you want to asynchronously access an external web page within the application. With this API, you can open a web page without using a Browser widget in your application.

Syntax

```
kony.application.openURLAsync(CONFIG)
```

Input Parameters

Parameter	Description
CONFIG[JSObject] - **Mandatory**	This is the parent encapsulating object that stores these configuration keys: url and callback. **Configuration Keys** *url*[String] - **Mandatory** Specifies the URL that points to the external web page. This is the URL that the user wants to open by using the openURLAsync API. *callback* [Function] - **Optional** The callback function that is triggered with the Success, Failure, or Unknown responses once the openURLAsync API opens the specified URL.

Example

```
var _url = "fb://page/" + FB_PAGE_ID;
var callbackFunction = function(response) {
    if (response == constants.OPEN_URL_SUCCESS) {
        //openURL return successfull from Native Side
    } else if (response == constants.OPEN_URL_FAILURE) {
        //openURL return failed from Native Side
    } else if (response == constants.OPEN_URL_UNKNOWN) {
        // if native don't provide a callback, we need to supply this to the
end user
    }
};

kony.application.openURLAsync({
    url: _url,
    callback: callbackFunction
});
```

Response Type for the Callback Function

```
constants.OPEN_URL_SUCCESS
constants.OPEN_URL_FAILURE
constants.OPEN_URL_UNKNOWN //If the native platform does not provide any
callback, this constant will be passed in the callback by default.
```

Return Values

None

Native Limitations

- Because of the asynchronous nature of the openURLAsync API, some cross-platform inconsistency may occur.

Platform Availability

Available on all Rich Client platforms.

kony.application.postAccessibilityNotification

This API posts a notification to "assistive" applications, i.e., applications that are designed to increase accessibility for blind and low-vision users, as well as for users with dyslexia.

Your application may need to post accessibility notifications if you have user interface (UI) components that change very frequently or ones that appear and disappear. Examples of such UI components include widgets that can be hidden or made visible in the layout on the click of a button.

Every Kony widget contains the VoiceOver accessibility feature, i.e., a voice-over that reads the titles of Kony widgets once a notification is posted.

Accessibility is directly managed by native iOS, and users don't have the control to set the focus, especially in Browser widgets. As a result, whenever a screen changes, the accessibility control sometimes remains in the previous screen. So, in order to set the focus of accessibility to the new screen, this API is used to notify the operating system that the screen has changed.

Syntax

```
kony.application.postAccessibilityNotification(config);
```

Input Parameters

Parameter	Description
config - Mandatory	This is a jsdict. `config[jsdictionary]` **accessibilityNotificationName [const] - Mandatory** Here, the constant (const) specifies the type of notification that the app can send. **constants.ACCESSIBILITY_ SCREENCHANGED_ NOTIFICATION** This is posted by the app when a new view appears that takes up a major portion of the screen.

Example

```
kony.application.postAccessibilityNotification({
accessibilityNotificationName:constants.ACCESSIBILITY_SCREENCHANGED_
NOTIFICATION});
```

Return Values

None

Platform Availability

- iOS

---

kony.application.registerForIdleTimeout

This API specifies if the application must timeout after a defined period of inactivity (time difference between the current device time and the last time you clicked on any user interface component) and also specifies the action after the timeout interval.

Important:

- You must enable the forms for idletimeout by setting the property for **enabledForIdleTimeout** as ***True*** in the IDE. The *idletimeout* event is triggered only when the user is on a form that is enabled for idletimeout.
- When the idletimeout event is triggered, the user is navigated to the home screen with an alert message.
- In BJS devices, to validate registerForIdleTimeout, an event must be triggered once time out is occurred.

Use Case

You can use this API typically in financial applications when you want to log the user out automatically after a specific period of inactivity. This way, you can ensure that there is no un-authorized access to the application or to the sensitive information like account number, credit card numbers, and so on.

Syntax

```
kony.application.registerForIdleTimeout(timeoutValue, callback)
```

Input Parameters

Parameter	Description
timeoutValue [Number] - Mandatory	The timeout value in minutes.
callback [Function] - Mandatory	The function that must be executed after the timeout has occurred. The callback function has the logic to display the current form. The function can be changed during runtime.

Note: Current form is the Form that was being displayed on the device screen when the timeout occurred.

Example

```
function test() {
    //Logic for the callback function registerForIdleTimeout
}

function registerForIdleTimeout() {
    kony.application.registerForIdleTimeout(1, test);
}
```

Return Values

None

Implementation Details

The *idletimeout* event is triggered only once for every call to `registerForIdleTimeout` API and this event is not fired again until the next call to the API. The *idletimeout* event unregisters itself automatically when the event is fired. When the event unregisters itself, the application can register again for idletimeout.

The following are a few scenarios:

Scenario 1

idletimeout event is triggered after the specified time interval, but the user is on a form that is not enabled for idletimeout. In this case, the function associated with the event is executed as soon as the user navigates to the form that is enabled for idletimeout.

Note: In this case, the form that the user is trying to navigate to is never shown as the idletimeout event takes precedence.

Scenario 2

idletimeout event is triggered and the user is on a form enabled for idletimeout. In this case, the associated function is executed immediately.

Scenario 3

The application is registered for idle time out and the user tries to register again for idletimeout. In this case, the latter call to the `registerForIdleTimeout` API has no affect and is ignored.

Scenario 4

The application is registered for idletimeout but none of the forms in the application are enabled for idletimeout. In this case, this API has no affect and the behavior of the application is undefined.

Each underlying platform handles the *idletimeout* event differently:

iPhone

- When the application is running in the background, no logic is executed. However, the timers keep running and the idletimeout event is triggered in the background. But the function is executed immediately when the application comes to foreground.

> ***Note:*** When the application comes to foreground, you will suddenly notice the function of the idletimeout event getting executed.

Android

- When you press the device Back button to go out of the application, the underlying OS destroys the UI activity. In such cases, the idletimeout event is fired only when the application comes to the foreground.
- When you navigate out of the application using the device Home button, the underlying OS pauses the UI activity but does not destroy it. In such cases, the idletimeout event is fired immediately in the background.

Symbian

- The underlying platform runs the timer and triggers the idletimeout event even when the application is in background. The function associated with the event is executed as soon as the application comes to the foreground.

Mobile Web

- In Mobile Web applications, the timer is not active when the application goes into background or sleep mode. When the application comes into foreground, the timer starts running and the `idletimeout` triggers after the specified time interval.

Rules and Restrictions

This API is applicable only when the Form property *Enable Time Out* is set to *true*. For more information about *Enable Time Out* property, see *Kony Widget User Guide*.

Exceptions

An error is thrown if input is invalid or does not follow the expected structure.

102-Invalid input error

Platform Availability

Available on all platforms.

kony.application.registerOnKeyPress

Connects an event handler function to a key press event.

Syntax

```
kony.application.registerOnKeyPress(
    values)
```

Input Parameters

Parameter	Description
values	A JavaScript object that contains key-value pairs. The keys are strings that specify which key on the keyboard the event handler callback function is connected to. The values for each key in the key-value pairs are event handler callback functions.

Example

```
function keyPress(keycode) {
    kony.print("DONE!!!" + keycode);
}

function callRegisterOnKeyPress() {
    var values = {
        "Ctrl+C": keyPress,
        "Ctrl+X": keyPress,
        "Ctrl+V": keyPress,
        "Esc": keyPress,
        "Return": keyPress
    };
    kony.application.registerOnKeyPress(values);
}
```

Return Values

None.

Platform Availability

Windows 7 and later.

kony.application.registerOnSettingsChangeCallback

Customers can use the registerOnSettingsChangeCallback API to listen in Kony applications, if any settings have been changed in Native settings applications. Users can pass the list of required settings in the first parameter. In addition, users can pass the callback, which is invoked when any required setting is changed, in the second parameter. The callback is invoked with parameters, which provide information about the setting that was changed and the values of said setting. If users want to stop listening to changes in the settings, they must successfully register and pass null in the second parameter as well as pass the list of settings that they want to stop listening in the first parameter.

Syntax

```
kony.application.registerOnSettingsChangeCallback(settingsList,callback)
```

Input Parameters

settingsList[JSONObject]- Mandatory

The list of settings that users want to listen must be provided in this parameter. It can have one or more values. Following are the settings supported by various platforms:

- Android: "location", "device_locale", "time_zone", "time", and "wifi".
- Windows: "font", "color", "advancedEffectsEnabled", and "inputLanguage".

callback[Function object] - Mandatory

The callback to be invoked when there is a change in any setting that is passed in the first parameter. This callback must be passed in the second parameter. This is the single function for all the settings. It can have function object or null value. Function object is passed for listening to settings and null value is passed to stop listening to setting changes. If any registered setting changes, the callback is invoked with JS object. This object contains information about the changed setting. Depending on the setting that is changed, the JS object can have different keys. Following are the various keys that JS object can contain:

- Windows

Settings	Description
font (key)	This key has value as "textScaleFactor". It informs that the system Text Size setting has been changed.
color (key)	This key has null value, you must invoke the API to get the values.

Settings	Description
advancedEffectsEnabled (key)	This key has "value". It indicates whether the system Transparency Effects setting has been enabled (True/False).
inputLanguage (key)	This key has "value". It informs when the current input language has been changed.

- Android

Settings	Description
location	• If "location" setting changes, the JS object passed to the callback has the following key values: ◦ setting (key): This key has value as "location", which informs that the "location" setting has been changed. ◦ gps_provider (key) : This key has value as "true"/"false", which informs that the gps_provider has been enabled/disabled. ◦ network_provider (key): This key has value as "true"/"false", which informs that the network_provider has been enabled/disabled.

Settings	Description
device_locale	• If the "device_locale" setting changes, the JS object passed to the callback has the following key values: ◦ setting (key): This key has value as "device_locale", which informs that the "device_locale" setting has been changed. ◦ device_locale (key): This key has new device locale value, which is changed from the application device settings.

Settings	Description
time_zone	• If the "time_zone" setting changes, the JS object passed to the callback has the following key values: ◦ setting (key): This key has value as "time_zone", which informs that the "time_zone" setting has been changed. ◦ time_zone (key): This key has new time zone value, that is changed from the application device settings.
time	• If the "time" setting changes, the JS object is passed to the callback has the following key value: ◦ setting (key): This key has value as "time", which informs that the "time" setting has been changed.

Settings	Description
wifi	If the "wifi" setting changes, the JS object passed to the callback has the following key values: • setting (key): This key has value as "wifi", which informs that the "wifi" setting has been changed. • wifi_current_state (key): This key has value as current wifi state value. • wifi_previous_state (key): This key has value as previous wifi state value. The list of wifi states is as follows: • kony.settings.WIFI_ DISABLING: Wifi is disabling. Constant Value: 0 • kony.settings.WIFI_ DISABLED: Wifi is disabled. Constant Value: 1 • kony.settings.WIFI_ ENABLING: Wifi is enabling. Constant Value: 2 • kony.settings.WIFI_ ENABLED: Wifi is enabled. Constant Value: 3 • kony.settings.WIFI_ UNKNOWN: Wifi state is unknown. Constant Value: 4

Example

```
function callback(params) {
    switch (params.setting) {
        case "location":
            alert(params.gps_provider);
            alert(params.network_provider);
            break;
        case "device_locale":
            alert(params.device_locale);
            break;
        case "time_zone":
            alert(params.time_zone);
            break;
        case "time":
            alert("time is changed in settings");
            break;
        case "wifi":
            alert(params.wifi_current_state);
            alert(params.wifi_previous_state);
            break;
    }
}

//To register for setting changes.
kony.application.registerOnSettingsChangeCallback(["location", "device_
locale", "time_zone", "time", "wifi"], callback);

//To deregister for setting changes.
kony.application.registerOnSettingsChangeCallback(["location", "device_
locale", "time_zone", "time", "wifi"], null);

//To register for setting changes.
kony.application.registerOnSettingsChangeCallback(["location"], callback);

//To register for setting changes.
```

```
kony.application.registerOnSettingsChangeCallback(["device_locale", "time_
zone"], callback);
```

Platform Availability

- Android
- Windows

kony.application.removeApplicationCallbacks

The kony.application.removeApplicationCallbacks API helps you to clear callback functions associated with the specified appstates. This API is available from V8 SP4 onwards.

Syntax

```
kony.application.removeApplicationCallbacks(appstatesMap)
```

Input Parameters

appstatesMap [Object] - Mandatory

Specifies an Object with key as appstate and value as an array of function IDs for the corresponding appstate for which the registered callback functions need to be cleared. The following appstates are applicable:

App state	Description
isAppLaunchedForInteraction	State which indicates that the application is visible to users for interaction.
onactive	State which indicates that the mobile device is active and the application is running.

App state	Description
oninactive	State which indicates that the mobile device is inactive and the application is running.
onbackground	State which indicates that the application is active and running in the background.
onforeground	State which indicates that the application is active and running in the foreground.
onappterminate	State which indicates that the application has been terminated, and has stopped running.
onkeyboardchange	State which indicates whether a keyboard is deployed for an application.
onpowersourcechange	State which indicates whether a power source is attached to the user's device.

App state	Description
onnetworkchange	This is specific to Mango. State which occurs when there is a change in the following: • **Status**: Indicates the status of the device. The applicable statuses are Connected, Disconnected, Roaming, or Unknown. • **Network**: Indicates the available network on the device. The network statuses are Wireless80211, Ethernet, MobileBroadbandGSM, MobileBroadbandCDMA, or None. • **Date**: Indicates the date on which the event occurs.

Example

```
var callbacksToBeRemoved = {
    "isAppLaunchedForInteraction": ["functionID1", "functionID2"]
};
kony.application.removeApplicationCallbacks(callbacksToBeRemoved);
```

Platform Availability

- Android
- iOS

- Windows

kony.application.removeBMState

This API removes a specified key from the parameter list of the URL of the form.

Syntax

```
kony.application.removeBMState(formID,key)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form for which the parameters of the URL have to be removed.
key [String] - Mandatory	Key string representing the key to be removed.

Example

To remove a bookmark for a URL, enter the following:

```
kony.application.removeBMState ("form1", "About");
```

```
removebookmark: function() {
    kony.application.removeBMState("Form1", "About");
    alert("The About key is removed from the parameter list");
},
```

Return Values

None

Platform Availability

Supported for SPA and Desktop Web.

kony.application.removeGestureRecognizerForAllForms

This method allows you to remove a specified gesture recognizer for all Forms.

Syntax

```
kony.application.removeGestureRecognizerForAllForms(uniqueIdentifier)
```

Input Parameters

Function	Description
uniqueIdentifier - Mandatory	Reference to the gesture. The reference to the gesture is returned by the setGestureRecognizerForAllForms.

Example

```
function callbackClearLongPressGesture() {
    try {
        kony.application.removeGestureRecognizerForAllForms(uniqueidentifier);
    } catch (err) {
        alert(typeof err);
        alert("error in function callbackClearLongPressGesture: " +
err.message);
    }
}
```

Platform Availability

Available on all platforms except Server Side Mobile Web, Windows 7/Kiosk, and Desktop Web.

kony.application.removeSecondaryTile

This API enables you to remove and unpin a specified secondary tile which was created earlier.

Syntax

```
kony.application.removeSecondaryTile(id)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Unique identifier of the secondary tile

Example

```
kony.application.removeSecondaryTile("12345");
```

Return Values

None

Platform Availability

Available on Windows Mango, Windows Phone 8, and Windows 8.

---

kony.application.removeSeoDataReadyFlag

Clears the flag that caches forms for SEO.

Syntax

```
kony.application.removeSeoDataReadyFlag()
```

Example

```
kony.application.removeSeoDataReadyFlag();
```

Input Parameters

None.

Return Values

None.

Remarks

For information on SEO, see kony.application.setSeoDataReadyFlag and the Kony Visualizer User Guide.

---

kony.application.removeSettingsMenuItemAt

This API enables you to removes the specified App Menu item based on the index.

Syntax

```
kony.application.removeSettingsMenuItemAt (id, index)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Identifier of the Charm setting menu created.
index [Number] - Mandatory	The index from which the menu item must be removed. The index value lies between 0 and n-1.

Example

To remove a menu item from a given index, enter the following:

```
//Removing a menu item from the index 3.
kony.application.removeSettingsMenuItemAt("charmmenu", 3);
```

Return Values

None

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.requestPermission

Sends a request to the end-user to provide the access to specific resource.

Syntax

```
kony.application.requestPermission(resourceId[constant/String],
statusCallback[Function], options[JSObject])
```

Input Parameters

Parameter	Description
resourceId [constant/String] - Mandatory	Specifies the ID of the resource or name of the permission (only for Android) that you want to access. You can specify either a String (permission name) or an integer (resourceId) value. The feature to specify the name of the permission as a String is applicable only for Android. For instance, you can query a Native Android permission from the AndroidManifest.xml file by specifying the String directly: "android.permission.READ_PHONE_STATE". The available **resourceId** constants are as follows: • kony.os.RESOURCE_CAMERA • kony.os.RESOURCE_LOCATION • kony.os.RESOURCE_PHOTO_GALERY • kony.os.RESOURCE_CONTACTS • kony.os.RESOURCE_CALENDAR • kony.os.RESOURCE_SIRI (iOS-specific) • kony.os.RESOURCE_AUDIO_RECORD • kony.os.RESOURCE_NOTIFICATION (iOS-specific)

Parameter	Description
statusCallback [Function] - Mandatory	A callback function receives the end-user's decision. The statusCallback function receives a JS Object, which contains overall status and permission-specific status that end-user responded on the permission dialog box. `function statusCallback(response);` Here, **response** is a hash map that contains the authorization status of the requested resource. This argument contains the following key: **status [constant]** Resource status constant that indicates the overall status of the resource authorization. The possible values for **status** are as follows: • kony.application.PERMISSION_GRANTED • kony.application.PERMISSION_DENIED • kony.application.PERMISSION_NEVER_ASK_AGAIN

options [JSObject] - Optional	Specifies the additional option to identify the resource for which you want permission. This key is applicable on android only.
	
	To obtain the kony.application.PERMISSION_NEVER_ASK_AGAIN status, you have to set the `getNeverAskAgainStatus` key to true and pass the key in the options object. If the key is not set, and the user selects either the Deny or Never Ask Again options, then the permission status is considered as Kony.application.PERMISSION_DENIED.
	
	`var options = {`
	`    "isVideoCapture": true,`
	`    "getNeverAskAgainStatus": true`
	`}`

Parameter	Description
options [Object] - For Notifications	This is a mandatory parameter for notifications. `{notificationtypes : constants}` The available constants are as follows: • kony.notificationsettings.BADGE • kony.notificationsettings.SOUND • kony.notificationsettings.ALERT

Example 1

```
//< uses - permission  android: name = "android.permission.READ_PHONE_
STATE" >
    kony.application.requestPermission("android.permission.READ_PHONE_
STATE", permissionStatusCallback);

function permissionStatusCallback(response) {
    if (response.status == kony.application.PERMISSION_GRANTED) {
        kony.location.getCurrentPosition();
    } else if (response.status == kony.application.PERMISSION_DENIED)
{
        //Display Application Settings alert by using
kony.application.openApplicationSettings()
    }
}
```

Example 2

```
function requestpermission() {

    var options = {
        "isVideoCapture": true,
        "getNeverAskAgainStatus": true
    }

    kony.application.requestPermission(kony.os.RESOURCE_LOCATION,
permissionStatusCallback, options);

}

function permissionStatusCallback(response) {

    alert("response ::" + JSON.stringify());

    if (response.status == kony.application.PERMISSION_GRANTED) {

        kony.location.getCurrentPosition();

    } else if (response.status == kony.application.PERMISSION_DENIED)
{


        Requestpermission(); /* To show the reason to users for
granting the permission to use the feature and then raise a request.
*/

    } else if (response.status == kony.application.PERMISSION_NEVER_
ASK_AGAIN) {


        kony.application.openApplicationSettings(); /* To show the
reason to users for granting the permission to use the feature and
then open application settings to grant the request. */

    }
```

```
)

}
```

Return Values

Function	Description
JSObject	A JSObject contains the authorization status of the requested resource. The returned JSObject contains the following key: **status [constant]** Resource status constant which indicates the overall status of the resource authorization. For more information, refer to Permission Status. ***Note:*** In the Android platform, the status remains PERMISSION_DENIED if at least one of the permissions associated with the resource is denied by the end-user.

Platform Availability

- Android
- iOS
- Windows 10
- SPA

Kony.application.requestPermissionSet

When invoked, this API sends a request for a set of permissions. The status of the request is sent back to the user through a callback.

Syntax

```
Kony.application.requestPermissionSet(permissions, callback)
```

Input Parameters

Parameter	Description
Permissions	Array of qualified android permission strings.

Parameter	Description
Callback	Function object result will invoke this function. The result is a JSobject where the key is permission string and the value is the permission status.

Example

```
function requestpermission() {

    kony.application.requestPermissionSet(["android.permission.CAMERA",
"android.permission.WRITE_CONTACTS"], permissionStatusCallback);

}

function permissionStatusCallback(response) {

    var camera = "android.permission.CAMERA";

    var contacts = "android.permission.WRITE_CONTACTS";

    for (var i in response) {

        /* iterating through permissionSet key value pair from response
jsObject where 'i' is permission key and result is permission status */
        var result = response[i];

        if (result == kony.application.PERMISSION_DENIED) {

            // show message  and raise request again

        } else if (result == kony.application.PERMISSION_NEVER_ASK_AGAIN) {

            // show message and open settings page

            kony.application.openApplicationSettings();

        }

    }

}
```

Return Values

None.

Platform Availability

Android

kony.application.requestReview

This function requests users to provide a rating and to write a review for an app.

Syntax

```
kony.application.requestReview()
```

Input Parameters

None.

Example

```
kony.application.requestReview();
```

```
  //To request the user for his rating on your app, use the below API
  requestAppReview: function(){
    kony.application.requestReview();
  }
```

Return Values

None.

IDE/CodeGen Requirements

None.

Platform Availability

- iOS
- Android

Error Codes

The Android error codes are as follows:

- **Error code**: 801
  Message: Review Action Not Found
  This error code is displayed if the requested Play Store or browser is not found.
- **Error code**: 802
  Message: Review Error Unknown

kony.application.resetBMState

This API resets the state associated with the URL of a form. It removes all the parameters attached to the form URL

Syntax

```
kony.application.resetBMState(formID)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form for which the parameters of the URL have to be removed.

Example

```
//To reset a bookmark, enter the following
kony.application.resetBMStat("form1");
```

```
resetBookmarkState: function() {
    kony.application.resetBMState("Form1");
    alert("The state is removed from the URL");
}
```

Return Values

None

Platform Availability

Supported for SPA and Desktop Web.

kony.application.sendLibraryResultToNativeApp Function

This API allows you to send an acknowledgment to a Native app that launched this library. This API is available from V8 SP3 onwards.

For more information on Library mode, click here.

Note: Calling this API invokes the onLibraryResult() interface/protocol method, which is passed in the Native Start API of Kony Library.

Syntax

```
kony.application.sendLibraryResultToNativeApp(resultData)
```

Input Parameters

Parameter	Description
resultData	An object with key-value pair elements with key as String and value as primitive data types (int, float, double, String, and Boolean).

Example

```
var resultData = {
    status: "success"
};
kony.application.sendLibraryResultToNativeApp(resultData);
```

Return Values

None.

Platform Availability

- Android
- iOS
- Windows10
- Windows10 Mobile

Note: The kony.application.sendLibraryResultToNativeApp API is only applicable when the app is launched in Library mode.

kony.application.setApplicationBehaviors

The `kony.application.setApplicationBehaviors` function enables your app to configure its response to various events.

Syntax

```
kony.application.setApplicationBehaviors(
    Objectbehaviors);
```

Input Parameters

Objectbehaviors

A JavaScript object that contains key-value pairs which specify the app's response to various events. The object includes the following key-value pairs.

Key	Value
adherePercentageStrictly	By default, this key is set to `true`. This property is supported on SPA and Desktop Web only.
backOnEsc	If set to `true`, allows the user to go back to previous form by pressing Escape key. If the user is on the first form, the operation is ignored. Default value is `false`. If you are using the `onDeviceBack` property on the form, the app will override the `backOnEsc` property and gives priority to `onDeviceBack` property. Available on only Windows 7.0 and later.
blurEffectStyleInBackground	A value from the Blur Effect Constants that blurs the app's most recent screen when it is in the background. This value is ignored if the value of the `enableSplashScreen` key is `true`.
breakpoints	Sets the various breakpoints for the form for responsive web design.

Key	Value
defaultIndicatorColor	Sets the color for the progress indicator. A Hex code or a word can be used to set the color. The words can only be used for two colors (white and grey), and these words are not case sensitive. The `defaultIndicatorColor` key is only supported on Android and Windows. The default color might vary for each platform as it picks from the application's theme, form color or predefined colors.

Key	Value
dismissSIPinCallbacks	Enables users to dismiss the soft inputs such as keyboard and rotating wheel from the mobile screen. Using the `dismissSIPinCallbacks` key, users can dismiss the soft inputs when a registered callback such as onClick or onDownload is invoked for a corresponding widget. This property is available only on the iOS platform. Following are the possible values of the `dismissSIPinCallbacks` key: • DISMISS_SIP_IN_ CALLBACKS_DEFAULT: The native iOS Kony Framework determines if a soft input will be dismissed on invoking a registered callback. • DISMISS_SIP_IN_ CALLBACKS_YES: A soft input is dismissed after a registered callback is triggered for a widget. • DISMISS_SIP_IN_ CALLBACKS_NO: A soft input is not dismissed. Users must close the control such as a keyboard explicitly by clicking on the cancel or done buttons. ***Note:*** Users can set the `dismissSIPinCallbacks` property multiple times in an application.

Key	Value
enableNSURLSession	A Boolean value that, when set to `true`, sets all network requirements to be served with NSURLSession. Setting this to `false` restores the current behavior which uses NSURLConnection.
enableRedrawRegions	A Boolean value that enables redraw regions when set to `true`, or disables them when set to `false`. The default is `true`. Available on only Windows 8.0 and later.
enableSplashScreen	A Boolean value that causes a splash screen to be displayed when the app is in the background when set to `true`, or disables the splash screen when set to `false`. The default is `true`. Available on iOS.

Key	Value

fontScaleFactor	A Boolean value that sets the scale factor in pre/post app init. When configured, fonts in the app will scale accordingly. Configure this only once in the app life cycle to avoid ambiguous behavior. JavaScript developers must add the following snippet in pre/post init.
	
	Example:
	
	`var deviceInfo = kony.os.deviceInfo();`
	`    if (deviceInfo["model"] == "iPhone 6 Plus")`
	`    {`
	` kony.application.setApplicationBehaviors({fontScaleFactor:2.25});`
	`    }`
	`    else if (deviceInfo["model"] == "iPhone 6")`
	`    {`
	` kony.application.setApplicationBehaviors({fontScaleFactor:1.24});`
	`    }`
	`    else if(deviceInfo["model"] == "iPhone 5C")`
	`    {`
	` kony.application.setApplicationBehaviors({fontScaleFactor:0.98});`
	`    }`

Key	Value
hideDefaultLoadingIndicator	A Boolean value that hides the loading indicator when set to `true`, or shows it when set to `false`.
hideDefaultLoadingIndicator	A Boolean value that, if set to `true`, hides the default loading indicator that appears when a background operation is performed. The `hideDefaultLoadingIndicator` key is available only on Android and it is supported from Kony Android plug-in GA 5.0.32 onwards.
hideStatusBar	A Boolean value that hides the status bar when set to `true`, or shows it when set to `false`. The hideStatusBar property is available only on Windows (8 and later versions) and it is supported in Kony Visualizer GA 5.6.4.11 and in GA 6.0.2.3.
invokePreshowPostShowEventsOnDeviceBack	A Boolean value that selects whether the pre-show or post-show events are triggered when the user presses the **Back** option on a device.

Key	Value
isGestureEventsAsync	A Boolean value that specifies that all gesture events are executed asynchronously when your app sets this value to `true`. Otherwise, they are executed synchronously. Available on only Windows 8.0 and later.

Key	Value
isI18nLayoutConfigEnabled	A Boolean value that enables or disables the layout mirroring support for Kony applications. It is a global application-level parameter. This parameter is passed as an input for the kony.application.setApplicationBehaviors API, and is available from V8 SP4 onwards. If you assign the isI18nLayoutConfigEnabled key as **false**, the layout mirroring feature of your application is disabled. If you assign the isI18nLayoutConfigEnabled key as **true**, the layout mirroring feature of your application is enabled. Available on Windows 10, iOS, Android, and SPA. For Windows, if you use the **isI18nLayoutConfigEnabled** key and the **kony.application.setApplicationLayout** API together in a single application, the application does not function as expected.

Key	Value
isLowLevelEventsAsync	A Boolean value that specifies that the `onTouchStart`, `onTouchMove`, `onTouchEnd` , and `onTouchCancel` events are executed asynchronously when your app sets this value to `true`. Otherwise, they are executed synchronously. Available on only Windows 8.0 and later.
isPopupModel	A Boolean value that specifies whether the pop-up is modal. The value set as the property for a pop-up takes precedence over the value set dynamically.
isScrollEventsAsync	A Boolean value that, if set to `true`, indicates that scroll events are executed asynchronously. Kony recommends that you set this value to `false` and execute the scroll events synchronously. Available on only Windows 8.0 and later.
marginsIncludedInWidgetContainerWeight	A Boolean value that applies the margin calculations when set to `true`. Margins are calculated based on width of the parent box.

Key	Value
maxHeight	A numeric value that allows your app to set maximum height of the main window. Available on only Windows 7.0 and later.
maxWidth	A numeric value that allows your app to set maximum width of the main window. Available on only Windows 7.0 and later.
minHeight	A numeric value that allows your app to set minimum height of the main window. Available on only Windows 7.0 and later.
minWidth	A numeric value that allows your app to set minimum width of the main window. Available on only Windows 7.0 and later.
popupAsDialog	A Boolean value that, when set to `true`, forces all the pop-ups to be displayed outside the main window. If set to `false`, pop-ups are displayed embedded in main window. The default value is `false`. Available on only Windows 7.0 and later.

Key	Value

prefersHomeIndicatorAutoHidden (available only from iOS 11)	A Boolean value that specifies to auto hide the virtual home indicator that is introduced in iPhoneX bezel less display. This virtual home indicator is used instead of the physical home button. Apple takes care of when to auto hide the virtual home indicator if the value is set to true. The default value is false.
	
	Sample code:
	
	`function setHomeInidicatorAutoHidden(){`
	
	`kony.application.setApplicationBehaviors(`
	
	`{"prefersHomeIndicatorAutoHidden" : true}`
	`);`
	`}`

restoreformstateondeviceback	A Boolean value that specifies whether the default behavior of form app menu state is retained when the user presses the Back button on the device.

Key	Value
retainSpaceOnHide	A Boolean value that specifies whether the space allocated for a widget is retained when the widget is made invisible. For more information, see the **Remarks** section below.
saveState	If set to `true`, location and size of the main window will be saved between application session. Default is `false`. If the `setApplicationBehaviour` function is called multiple times, latest settings are considered and all old settings are overridden. Available on only Windows 7.0 and later.
settings	A JavaScript Dictionary object that contains key-value pairs. For more information, see the **Remarks** section below. Available on only Windows 8.0 and later.
skinImageScaleMode	A value from the Skin Image Scale Mode Constants.Available on only Windows 8.0 and later.
skipEscapeHTML	A Boolean value that allows HTML tags in Label widgets when set to `true`, or disallows them when set to `false`.

Key	Value
useNativeMSG	A Boolean value that, when set to `true`, enables your app to use native multistep gradients that are scaled relative to the size of the widget. Otherwise, set to `false` to disable this behavior.
userDataLocation	A String value that contains the absolute path where all the files/db related to an application will be saved. Available only on Windows Desktop/ KIOSK platform.

Example

```
//The Application must to call kony.application.setApplicationBehaviors
(appTable)
//in post-appinit() methods.
function setApplicationBehaviors() {
    //Controlling the behaviours by setting invokePreshowPostShow
    //EventsOnDeviceBack as true,isPopupModel as true,and retainSpaceOnHide as
true.
    kony.application.setApplicationBehaviors({
        invokePreshowPostShowEventsOnDeviceBack: true,
        isPopupModel: true,
        retainSpaceOnHide: true,
        "hideDefaultLoadingIndicator": true
    });
}

var obj = {
    windows7: {
        backOnEsc: true,
        minWidth: 300,
        maxWidth: 1000,
        minHeight: 400,
        maxHeight: 800,
        popupAsDialog: true,
        saveState: true,
    }
};

kony.application.setApplicationBehaviors(obj);

var inputParamTable = {};
inputParamTable.isScrollEventsAsync = false;
inputParamTable.isLowLevelEventsAsync = false;
inputParamTable.isGestureEventsAsync = false;
kony.application.setApplicationBehaviors(inputParamTable);
```

Return Values

None.

Remarks

This function enables you to specify whether:

- The pre-show and post-show events are invoked if you press Back option on a mobile device.
- Pop-ups are modal.
- The app menu needs to be updated when the pre-show and post-show events are invoked if you press the Back option on a mobile device.

Your app's menu has to be configured in pre-show/post show of each form if each form has separate app menu buttons or items. But on some platforms, the pre-show and post-show events are triggered when the user navigates back using Back option. On other platforms, those events are not triggered. If the pre-show/post-show event is not triggered, the form cannot update its app menu. In this scenario, ideally, the form must retain its app menu which it set earlier.

To ensure consistency while porting applications across platforms, the `kony.application.setApplicationBehaviors` function enables you to set the default behavior of pre-show and post-show events, a pop-up, and app menu.

> ***Note:*** This function can only be called in the post-appinit event handler.

Parameter Details

The following provides more information about the key-value pairs in the *Objectbehaviors* parameter.

retainSpaceOnHide

This key is applicable to widgets placed in a percentage container.

Limitations in Windows channel: When retainSpaceOnHide is true and widget isVisible property is false, top, and bottom margins will not be considered for invisible widget.When visibility of widget is changed and when retainSpaceOnHide is True, there must not be any change in the width of HBox or % scrollBox.

Height changes to the max height of the visible child widget.When retainSpaceOnHide property is made False, inconsistency among platforms is observed. If any of the widgets placed inside a container are made invisible, then some platforms will retain the space and some platforms will not retain the space. To overcome this inconsistency or to retain the space in all platforms, retainSpaceOnHide property must be configured to True. The default value of retainSpaceOnHide property is True.

settings

The value for this key is a JavaScript object that contains key-value pairs that are used to scale UI elements from Windows to iOS. These properties allow you set the property values provided in Dp metrics to have the same UI view in both iOS and Windows platforms.

Generally, The UI view may vary across platforms when values are provided to the properties of the widgets in Dp. The Dots per Inch (DPI) and Pixels per Inch (PPI) values vary vastly for iOS and Windows. Due to this, for a given Dp value, the resultant UI is smaller on Windows compared to iOS.

The variation is only on the Windows 8/8.1 platform. As the Windows 10 uses the effective pixels concept, other than font size, the widget sizes are appeared similar in iOS and Windows 10 Mobile platforms. If any application migrated from Widnows 8/8.1 to Windows 10 using flex (Dp), there may be differences in the UI.

Currently, the only supported key is `dpScaleFactor`. The value for the `dpScaleFactor` key is a JavaScript object that also contains key-value pairs. The supported keys are as follows.

Constant	Value
referenceFontSize	An optional numeric value that specifies a value to scale the all font sizes. Set this parameter only in the pre-appinit and post-appinit. The parameter is applicable only for Windows 10 (Phone).
referenceWidth	An optional numeric value that specifies the width of the target iOS device to which the Windows device screen has to be scaled. The default value of this parameter is the width of the Windows device screen in Dp.

The resulting code for using the `dpScaleFactor` key would resemble the following.

```
function scaleWindowsToiOS()
{
    var scaleData = {};
    scaleData.referenceWidth = 320;
    scaleData.referenceFontSize = 0.8;
    kony.application.setApplicationBehaviors({“dpScaleFactor”:
scaleData});
}
```

Platform Availability

Available on all platforms

kony.application.setApplicationBadgeValue

This API allows you to set a badge value to an application icon on the mobile desktop at the top-right corner of the application icon. If you pass an empty string as a parameter, the badge applied on the application icon is removed.

Syntax

```
kony.application.setApplicationBadgeValue(badgeValue, tileID)
```

Input Parameters

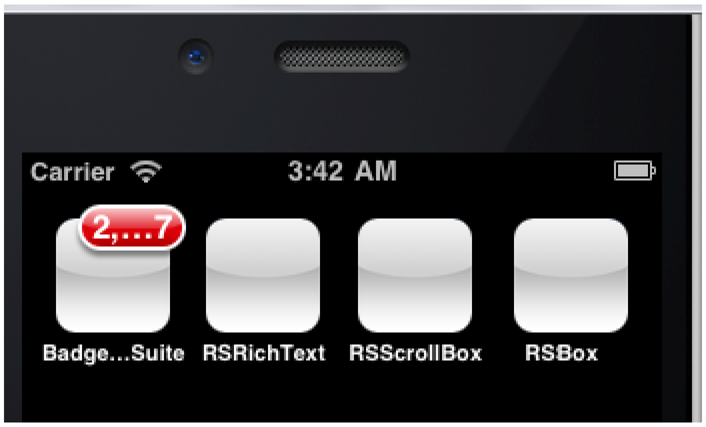
Parameter	Description
badgeValue [String] - Mandatory	Value of the badge. The value that you specify in the badgeValue parameter appears within the badge. Do not pass any other value except a numerical value. For example, to set a badge value for an appicon, specify the value as "2" instead of 2. If the length of the badge value is greater than 1 the badge is a rounded rectangle. For example, if you specify the value of the badge as 88, the number appears in a rounded rectangular badge. If the length of the badge value is 1, the badge is always a circle.
tileID [String] - Optional	The tile ID defined for the secondary tile using the setSecondaryTile API. The parameter is applicable only for Windows.

Example

```
function setApplicationBadgeValue(){
       /*Set the ApplicationBadgeValue to the  application icon on the
       mobile desktop at the top-right corner of the application icon.*/
       kony.application.setApplicationBadgeValue("234567");
}
//to set the badge value "10" to the secondary tile (Applicable only in the
Windows platform)
var tileId="myTileId";
kony.application.setSecondaryTile(tileId, "My App", "My Application",
"tile.png");
kony.application.setApplicationBadgeValue("10", tileId);
```

```
settingBadge: function() {
    this.view.btnBadge.setBadge("0", ""); //Set badge value on  button widget
    kony.application.setApplicationBadgeValue("" + 0);
},
BadgeIncrease: function() {
    var counter = kony.os.toNumber(this.view.btnBadge.getBadge()) + 1; // read
badge value from button and increment it with 1
    kony.print("this gets executed " + counter + "type is " + typeof
(counter));
    this.view.btnBadge.setBadge("" + counter, ""); // Set badge value on the
button widget
    kony.application.setApplicationBadgeValue("" + counter); //Set badge value
on app icon
```

The badge appears as follows when you execute the code given above:



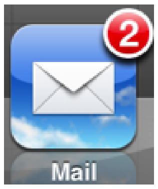
Return Values

None

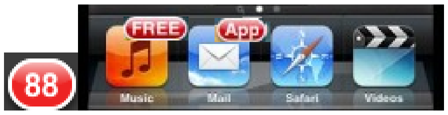
UI Behavior

The badge appears with white font on a red background. The shape of the badge varies with its value:

- If the badge value is a single digit, the badge shape is a circle.



- If the badge value contains multiple digits, the badge shape is a rectangle with rounded corners and borders.



Platform Availability

- iOS
- Windows

kony.application.setApplicationCallbacks

This API captures the callback events for various states of the application which include:

- *onactive* - state which indicates that the mobile device is active and application is running
- *oninactive* - state which indicates that the mobile device is inactive and application is running
- *onbackground* - state which indicates that the application is active and running in the background.

  For more information on iOS application behavior on background, click here
- *onforeground* - state which indicates that the application is active and running in the foreground
- *onappterminate* - state which indicates that the application is terminated and not running.
- *onkeyboardchange* - state which indicates whether a keyboard is deployed for an application.
- *onpowersourcechange* - state which indicates whether if a power source is attached to a device.
- *onmultiwindowmodechanged* - state that indicates whether the application changes from full-screen mode to multi-window mode, or vice-versa.
- *onnetworkchange* - this is specific to Mango. This state occurs whenever there is a change in the following:
  - *Status* - Indicates the status of the device. The different statuses are Connected, Disconnected, Roaming or Unknown.
  - *Network*- Indicates the network available on the device. The network statuses are Wireless80211, Ethernet, MobileBroadbandGSM, MobileBroadbandCDMA, or None.
  - *Date* - Indicates the date on which the event occurs.
- *onapplicationopenurl* - To support interprocess communication (IPC) message authentication, Kony provides an application callback `onapplicationopenurl`. The `onapplicationopenurl` callback is used if an application needs to authenticate another application. To use this callback, applications must register with it. The `onapplicationopenurl` callback takes `sourceApplication`, `url`, and `annotation (via UIDocumentInteractionController)` as input

parameters. If the developer chooses to process the commands/information, True must be returned. If the developer chooses to ignore the commands/information passed, then False must be returned.

Note: Android supports only the onbackground, onmultiwindowmodechanged, and onforeground states.

Syntax

```
kony.application.setApplicationCallbacks(callbacks)
```

Input Parameters

callbacks [Object] - Mandatory

Specifies an Object with key as appstate and value as the callback function for the corresponding appstate. The following are the different appstates possible:

- ***onappterminate***: This state which indicates that the application is terminated and not running.

Callbacks specific to Android

Callback	Description
onbackground	This state which indicates that the application is active and running in the background.
onforeground	This state which indicates that the application is active and running in the foreground.

Callback	Description

onmultiwindowmodechanged	This callback takes the function object that is called when the application changes from full-screen mode to multi-window mode, and vice-versa.
	
	• **Syntax**:
	`kony.application.setApplicationCallbacks( {onmultiwindowmodechanged:callbackfunction} );`
	
	• **Parameters**:Kony platform provides the following two parameters for this callback:
	  ◦ boolean: The value is true when the application changes from full-screen mode to multi-window mode, and the value is false if the application changes from multi-window mode to full-screen mode.
	  ◦ object: This contains height and width configurations. The object keys are height and width. If native Android does not provide the required configuration, the object remains undefined.
	Note: Native Android provides the config object from Android 8.0 onwards.
	
	• **Example**:
	`function onMultiWindowModeChangedCallback (isInMultiWindow, config)`
	`{`
	`kony.print("Is in multi window mode :" + isInMultiWindow);`
	`if(config != undefined)`
	`{ kony.print("Config values :height : " + config.height + " : width : " + config.width ); }`
	`}`

Callbacks specific to Windows

Callback	Description
ondpichange	This state occurs whenever the LogicalDpi property changes because the pixels per inch (PPI) of the display changes. This callback is called with a dictionary as the argument. The required keys are listed below. ***Note:*** The ondpichange callback is supported in Windows 10 only.
onkeyboardchange	This state indicates whether a keyboard is deployed for an application.
onpowersourcechange	This state indicates whether a power source is attached to a device.
onnetworkchange	This state occurs whenever there is a change in the following: ◦ *Status*: Indicates the status of the device. The different statuses are Connected, Disconnected, Roaming or Unknown. ◦ *Network*: Indicates the network available on the device. The network statuses are Wireless80211, Ethernet, MobileBroadbandGSM, MobileBroadbandCDMA, or None. ◦ *Date*: Indicates the date of the event.
onsizechange	This state occurs when the app window changes its rendering size. This callback is called with a dictionary as the argument. The required keys are listed below. ***Note:*** The onsizechange callback is supported in Windows 10 only.

Callback	Description
onvisibleboundschange	This state occurs when the value of VisibleBounds changes. This typically occurs when the status bar, app bar, or other chrome becomes hidden or visible. This callback is called with a dictionary as the argument. The required keys are listed below. ***Note:*** The onvisibleboundschange callback is supported in Windows 10 only.

Common keys for the dictionary argument

The callbacks **ondpichange**, **onsizechange**, and **onvisibleboundschange** are called with a **dictionary** as the argument with the following keys.

Key	Description
width [double]	The current window width
height [double]	The current window height
currentform [Form widget]	The current visible form
orientation [constant]	Gets the current orientation (landscape or portrait) of the window (app view) with respect to the display. The possible values are: • constants.FORM_DISPLAY_ORIENTATION_LANDSCAPE • constants.FORM_DISPLAY_ORIENTATION_PORTRAIT
isFullScreen [boolean]	Specifies whether the current window is full screen. This value is **true** if the window is full screen; **false** if it is not.

Key	Description
isFullScreenMode [boolean]	Specifies whether the current window is in full screen mode. This value is **true** if the app is running in full screen mode; **false** if it is not. The default value is **false**. ***Note:*** This key will have some value only after enterFullScreenMode() or exitFullScreenMode() is called.
visibleWidth [double]	Gets the visible width of the window (app view). The visible region is the region not obstructed by chrome such as the status bar or app bar.
visibleHeight [double]	Gets the visible height of the window (app view). The visible region is the region not obstructed by chrome such as the status bar or app bar.
dpi [double]	Gets the pixels per logical inch of the current environment.
nativeOrientation [constant]	Gets the native orientation of the display monitor. This is typically the orientation where the buttons on the device match the orientation of the monitor. The possible values are: • constants.FORM_DISPLAY_ORIENTATION_LANDSCAPE • constants.FORM_DISPLAY_ORIENTATION_PORTRAIT
resolutionScale [double]	Gets the scale factor of the immersive environment.
rawPixelsPerViewPixel [double]	Gets a value representing the number of raw (physical) pixels for each view (layout) pixel.

Key	Description
interactionMode [double]	Gets a value that indicates whether the user is interacting with the view using a mouse or touch. The possible values are: 0 : Mouse 1 : Touch
viewstate [constant]	Specifies the state of the current window (app view). The state indicates the orientation (landscape or portrait) and whether the app is snapped. The possible values are: • constants.VIEW_STATE_FULLSCREEN_LANDSCAPE • constants.VIEW_STATE_FILLED • constants.VIEW_STATE_SNAPPED • constants.VIEW_STATE_FULLSCREEN_PORTRAIT

Callbacks specific to iOS

onactive	This state occurs when there is an incoming SMS, phone call, or anything else that interrupts the automatic lock of the mobile device.
oninactive	This state occurs when the application is idle for a specific time interval

shouldAllowExtensions	You can use extensions to enable a specific task beyond your app functionality and make available to users while they are using other apps or the system. This callback provides a mechanism to enable or disable the extensions in iOS framework. Following are the input parameter and return values of this callback:
	• **string - [Mandatory** - Specifies the entensionID that contains the identifier to identify the extension.
	• **Return values** - Returns a boolean value. If true is returned, the extension is enabled. If false is returned, the extension is disabled. By default, all the extensions are enabled.
	Note: Extensions are available from iOS8 and above versions.
	`//Sample code`
	`function shouldAllowExtensions(extensionID)`
	`{`
	`    if (extensionID == "com.apple.keyboard -service")`
	`    {`
	`        return true; //will allow the keyboard extension to be accessed`
	`    }`
	`    return false;`
	`}`

onlowmemory	This state indicates that the app received a memory warning from the system.

onapplicationopenurl	To support interprocess communication (IPC) message authentication, Kony provides an application callback `onapplicationopenurl`. The `onapplicationopenurl` callback helps a developer do some authentication when an app is opened from an already running app. To use this callback, applications must register with it. The `onapplicationopenurl` callback takes `sourceApplication`, `url`, and `annotation (via UIDocumentInteractionController)` as input parameters. If the developer processes the commands/information, `True` must be returned. If the developer ignores the commands/information passed, then `False` must be returned. This callback accepts the following arguments: • **sourceApplication - String - Mandatory** - The application that requests to open a URL. • **url - String - Mandatory** - The URL that is to be opened. • **annotation - Dictionary - Mandatory** - If the URL refers to a file that is opened through a document interaction controller, then the source application can send information contained in this dictionary. Otherwise, this value is nil. The receiving app must know about the keys of the dictionary sent. • **Return Value** - True or False.

onwatchrequest	This method is used to handle the requests coming from Apple WatchKit Extension. The dictionary object is a dictionary of value pairs and represents the context for request to be processed.
	This callback accepts the following arguments:
	• **userInfo- [Dictionary]** This parameter is mandatory. A dictionary of key value pairs.
	• **replyObject- [JSObject]**- This parameter is mandatory. This object is used to send the response asynchronously. To send the response we should call a method `(executeWithReply)` on this JSObject. The method accepts a key value pairs that form the response to be sent back.
	`//Sample code for onwatchrequest callback method.`
	`function jsfunction(userInfo, replyObj)`
	`{`
	`   if (userInfo.requestName == "getData"){`
	`      var responseDict={`
	`          "key1":"value1",`
	`          "key2":"value2"`
	`      };`
	`      replyObj.executeWithReply(responseDict);`
	`   }`
	`}`
	Note: It is possible to that execution of this call back might happen while the application is in the background. It is suggested to check the application state and perform UI updates in the phone application only when the application is running in the foreground.You may use getapplicationstate APIs.

onactivityrequest	This method is used to handle the handoff activity request that are sent to the application.
	This callback accepts the following arguments:
	• **activityName - [String]** This parameter is mandatory. A string that represents the name of the activity.
	• **activityInfo - [Dictionary]** This parameter is mandatory. A dictionary containing custom user information.
	This callback method returns a boolean (true/false) value that indicates the request handle status.
	`//Sample code for onactivityrequest callback method.`
	`function jsfunction(activityName, activityInfo) {`
	`    if (activityName == "continueActivity") {`
	`        // use activityInfo dictionary to continue`
	`the past activity on the iPhone`
	`        return true;`
	`    }`
	`    return false;`
	`}`

onpreactivityrequest	This method is used to ask the receiving application, whether it wants to continue the handed off activity. If this callback method returns true, then the `onactivityrequest` callback method gets invoked where the receiving app gets a chance to continue the handed off activity.
	This callback accepts the following arguments:
	• **activityName - [String]** This parameter is mandatory. A string that represents the name of the activity.
	If this callback method returns *true*, then the you notify the user that the receiving app wants to continue the activity on the device. If the callback method returns *false*, then iOS notifies the user to continue activity on the device.
	`//Sample code for onpreactivityrequest callback method.`
	`function jsfunction(activityName)`
	`{`
	`    if (activityName == "continueActivity") {`
	`        return true;`
	`    }`
	`    return false;`
	`}`

Example

```
//onactive function
function test() {
    alert("===============Test executed=======");
}

//oninactive function
function test1() {
    alert("===============Test1 executed=======");
}
```

```
//onbackground function
function test2() {
    alert("===============Test2 executed=======");
}

//onforeground function
function test3() {
    alert("===============Test3 executed=======");
}

//onappterminate function
function test4() {
    alert("===============Test4 executed=======");
}

//setcallbackfunction
function setCallBacks() {
    alert("===============setApplicationCallbacks executed=======");
    var callbacksObj = {
        onactive: test,
        oninactive: test1,
        onbackground: test2,
        onforeground: test3,
        onappterminate: test4
    };
    kony.application.setApplicationCallbacks(callbacksObj);
}
```

```
//onapplicationopenurl
kony.application.setApplicationCallbacks({
    onapplicationopenurl: callbackFunction
});
Example 1.
function callbackFunction(sourceApplicaion, url, annotation) {
    var flag = false;
    if (sourceApplication == "AppOne" && url == "appTwo://….") {
        flag = true;
    }
    return flag;
}
Example 2.

function callbackFunction(sourceApplicaion, url, annotation) {
    var flag = false;
    if (annotation) //annotation can be nil as well… refer parameter
description… {
            if (annotation[key1] == 10) {
                flag = true;
            }
    return flag;
}
```

```
//onWatchRequest sample code
function jsfunction(requestObject) {
    return <returnValue > ;
}
kony.application.setApplicationCallbacks({
    onwatchrequest: jsfunction
})

//requestObject- is a dictionary of key value pairs.

//Returnvalue- Any JS Object consisting of numbers, strings and boolean data
types.
```

Return Values

None

Exceptions

An error is thrown if input is invalid or does not follow the expected structure.

- 100 - Invalid type pf parameters.
- 101 - Invalid number of arguments.
- 102 - Invalid input error

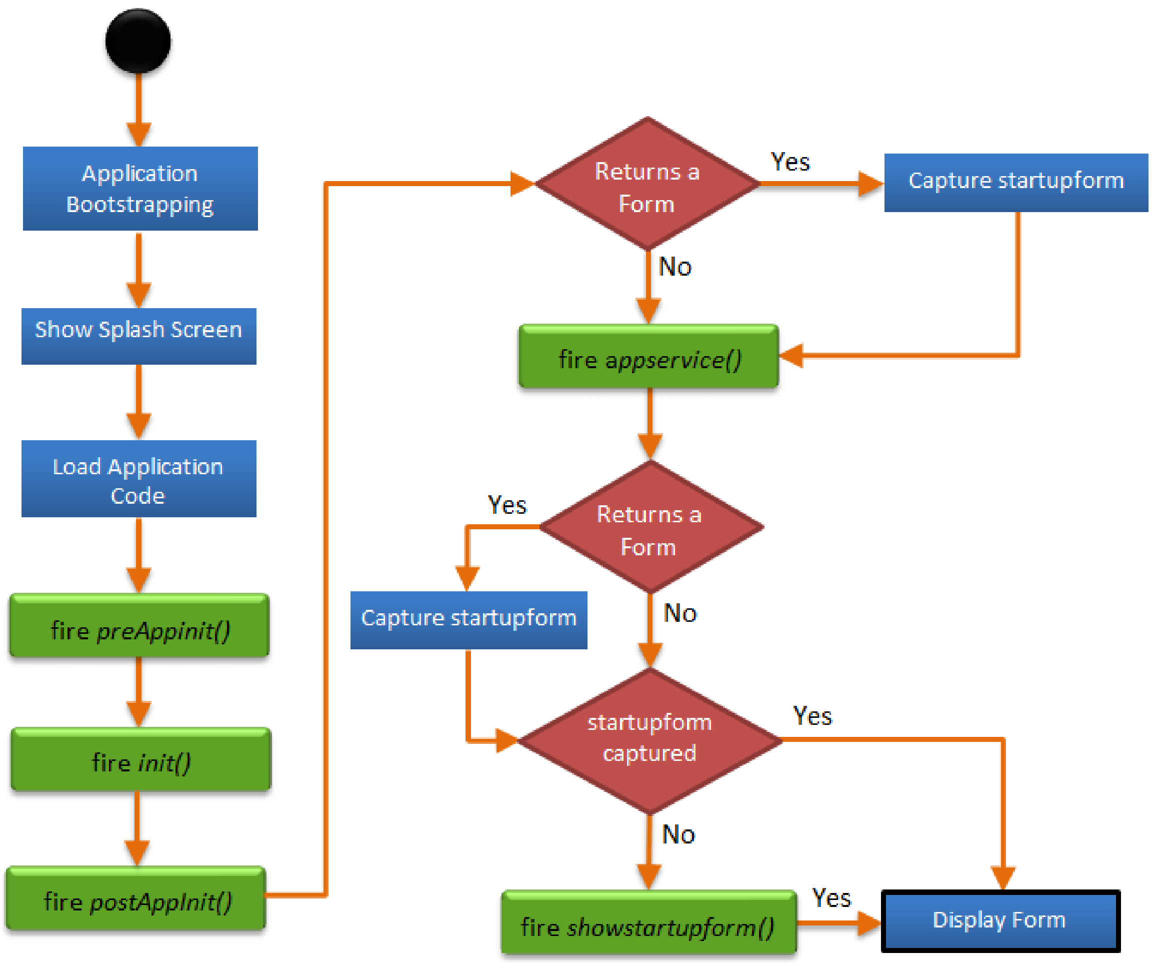
Platform Availability

Available on all platforms except Desktop Web, SPA, and Mobile Web.

kony.application.setApplicationInitializationEvents

This API is used to configure all initialization events such as, preappinit, postappinit, init, appservice, showstartupform and so on. The application lifecycle is as follows:

> ***Important:*** This API must be invoked only once during the application lifecycle. You should NOT invoke this API as part of the application lifecycle, because Kony Visualizer will generate the code that invokes the API automatically. Invoking this API multiple times will result in an undefined behavior.



Syntax

```
kony.application.setApplicationInitializationEvents(callbacks)
```

Input Parameters

callbacks [Object] - Mandatory

Is a Hash table which comprises of the following key value pairs:

Key	Description
preappinit[params]	This is an existing event that the developer registers in the IDE per application basis. This event consists of any logic that needs to be executed before the initialization of forms, skins, and any other application initialization activities. For e.g loading the i18n resource bundles dynamically etc.
init[params]	init is generated by the code and consists of the form and skin initialization data.

Key	Description
postappinit[params]	Is an existing event that the developer registers in the IDE per application basis. Developers use this function to define logic that needs to be executed before the first form is shown and after the application is initialized. ***Note:*** postappinit can return a form handle. If postappinit returns a form handle, then the platform shows the form returned from the postappinit callback, otherwise the platform invokes the showstartupform function.

Key	Description
appservice[params]	The closure provided against appservice will be invoked by the platform in a sequence. This function also returns the form handle. ***Note:*** appservice can return a form handle. If appservice returns a form handle, then the platform shows the form returned from the appservice callback. If appservice does not return a form handle, the platform invokes the showstartupform function. ***Note:*** If both postappinit and appservice return the

Key	Description
showstartupform[params]	This is a method invoked by the platform to show the startup form, as indicated in the IDE. ***Note:*** This event is invoked only when postappinit returns a nil. If appservice returns a form handle, showstartupform is not invoked.
deeplink[params]	This method is Thinclient/SPA specific and if appservice event is provided, then this event will be ignored.

Example

```
//onactive function
function test() {
    alert("===============Test excuted=======");
}

//oninactive function
```

```
function test1() {
    alert("===============Test1 excuted=======");
}

//onbackground function
function test2() {
    alert("===============Test2 excuted=======");
}

//onforeground function
function test3() {
    alert("===============Test3 excuted=======");
}

//onappterminate function
function test4() {
    alert("===============Test4 excuted=======");
}

//setAppInitialization function
function setAppInitializationEvents() {
    kony.application.setApplicationInitializationEvents({
        init: test,
        preappinit: test1,
        postappinit: test2,
        appservice: test3,
        showstartupform: test4,
        deeplink: test5
    });
    alert("===============setAppInitializationEvents excuted=======");
}
```

Return Values

None

Exceptions

- **102** - InvalidInputError: An error is thrown if input is invalid.
- **100**- TypeError: An error is thrown when the type is invalid.

Platform Availability

Available on all platforms.

---

kony.application.setApplicationLayout

This API specifies if the application must have a layout from "left to right" or "right to left".

UseCase

To support Arabic kind of languages, all the widgets on a form must be layout from right to left. This API impacts only horizontal layout.

Syntax

```
kony.applicationsetApplicationLayout(layoutDirection)
```

Input Parameters

Parameter	Description
layoutDirection [Constant] - Mandatory	Specifies the layout direction for the content and the widgets of an application. • constants.APPLICATION_ LAYOUT_RIGHT2LEFT • constants.APPLICATION_ LAYOUT_LEFT2RIGHT

Example

If an application supports English and Arabic languages, app developer can invoke below snippet anywhere in appservice or preappinit or postappinit.

```
function setApplicationLayoutBasedOnLocale() {
    if (kony.i18n.getCurrentLocale() == "en_US") {
        kony.application.setApplicationLayout(constants.APPLICATION_LAYOUT_
LEFT2RIGHT);
    } else {
        kony.application.setApplicationLayout(constants.APPLICATION_LAYOUT_
RIGHT2LEFT);
    }
}
```

Return Values

None

Implementation Details

There is no direct API available to change the application layout to support both "Left2Right" and "Right2Left" languages. App developers have to spend huge amount of time in redesigning the UI for both locales, setting the margins/paddings and in setting "contentAlignment" for all widgets on all forms. This API is simple solution to change the application layout based on locale.

This API can be called in appservice/preappinit/postappinit after checking the current locale or this should be called after the locale is changed successfully with "setCurrentLocaleAsync()".

Exceptions

An error is thrown if input is invalid or does not follow the expected structure.

- 102 - Invalid input error
- 101 - Insufficient arguments passed

Platform Availability

Available on Windows 8, Windows Phone 8, and Windows Phone 7.5 (Mango).

kony.application.setApplicationMode

This API enables you to set the application mode to Native, Hybrid, or Wrapper.

Syntax

```
kony.application.setApplicationMode(Integer constant)
```

Input Parameters

Parameter	Description
Integer Constant	• 1: constants.APPLICATION_ MODE_NATIVE • 2: constants.APPLICATION_ MODE_HYBRID • 3: constants.APPLICATION_ MODE_WRAPPER

Example

```
function setApplicationMode(){
       kony.application.setApplicationMode(APPLICATION_MODE_NATIVE);

}
```

Return Values

None

Special Considerations

This API should be called only in preappinit or before the execution of preappinit. If this API is invoked elsewhere in the program, it is invalid and leads to undefined behavior.

Exceptions

Error

Platform Availability

Available on all platforms.

kony.application.setApplicationProperties

The API enables you to set properties at the application level.

Syntax

```
kony.application.setApplicationProperties(key-value pairs)
```

Input Parameters

statusBarColor

Sets the color for a device's status bar.

statusBarForegroundColor

Sets the foreground color of a device's status bar; the color of each item (text, icon, and so on) displayed on the status bar. This parameter is not available on Android.

statusBarHidden

Hides or reveals the status bar.

You can achieve hidden status bar behavior by setting at Application level and at Form level as well. At the Application level, you can achieve the behavior by defining either statusBarHidden parameter, and also using the systemUiConfig parameter with value as IMMERSIVE_STICKY_HIDE_STATUS_BAR in the API.

When you define both statusBarHidden and systemUiConfig parameters in the API, the systemUiConfig parameter takes the precedence.

At the Form level, you can achieve the behavior using the statusBarHidden property and the systemUiConfig property. The following table provides the precedence levels when hidden status bar behavior is set using both properties at the Form level and both parameters at the application level (only when the systemUiConfig parameter's value is IMMERSIVE_STICKY_HIDE_STATUS_BAR) as well.

Note: The statusBarHidden flag is cleared when you move the application to the background and then bring it back to the foreground. So, the status bar is displayed by default when you move the application back to the foreground.

Property/Parameter	Precedence Level
systemUiConfig Property (Form Level)	1
systemUiConfig Parameter (Application Level)	2
statusBarHidden Property (Form Level)	3
statusBarHidden Parameter (Application Level)	4

Example

```
Form.statusBarHidden : true
```

statusBarStyle

Sets a style for the status bar. This parameter is not available on Android.

For more information on the input parameters, refer the status bar properties.

navigationBarColor

Sets the color for a device's navigation bar. Applicable only for Android. For more information, refer to the navigationBarColor property.

systemUiConfig

Controls the behavior of the status and navigation bars. Applicable only for Android. For more information, refer to the systemUiConfig property.

You can define any one of the following:

SYSTEM_UI_ DEFAULT	Resets any of the following modes set previously.
DIM_SYSTEM_BARS	Dims the status and navigation bars, and space occupied them are retained as is.
HIDE_STATUS_BAR	Hides the status bar and space occupied by it is not retained. User can reveal it by swiping downward from the top of the screen or by tapping anywhere on the screen
HIDE_NAVIGATION_ BAR	Hides the navigation bar and space occupied by it is not retained. User can reveal it by swiping upward from the bottom of the screen or by tapping anywhere on the screen.
HIDE_SYSTEM_ BARS	Hides both status and navigation bars and space occupied by them are not retained. User can reveal it by swiping downward from the top of the screen, by swiping upward from the bottom of the screen, or by tapping anywhere on the screen.

HIDE_STATUS_BAR_KEEP_LAYOUT_STABLE	Hides the status bar and space occupied by it is retained. Resetting this mode is same as HIDE_STATUS_BAR.
HIDE_NAVIGATION_BAR_KEEP_LAYOUT_STABLE	Hides the navigation bar and space occupied by it is retained. Resetting this mode is same as HIDE_NAVIGATION_BAR.
HIDE_SYSTEM_BARS_KEEP_LAYOUT_STABLE	Hides both status and navigation bars and space occupied by them are retained. Resetting this mode is same as HIDE_SYSTEM_BARS.
IMMERSIVE_HIDE_STATUS_BAR	Hides the status bar and space occupied by it is not retained. Unlike HIDE_STATUS_BAR, this mode can be reset only when user reveals the status bar by swiping downward from the top of the screen.
IMMERSIVE_HIDE_NAVIGATION_BAR	Hides the navigation bar and space occupied by it is not retained. Unlike HIDE_NAVIGATION_BAR, this mode can be reset only when user reveals the navigation bar by swiping upward from the bottom of the screen.
IMMERSIVE_HIDE_STATUS_BAR_KEEP_LAYOUT_STABLE	Hides the status bar and space occupied by it is retained. Resetting this mode is same as IMMERSIVE_HIDE_STATUS_BAR.
IMMERISIVE_HIDE_NAVIGATION_BAR_KEEP_LAYOUT_STABLE	Hides the navigation bar and space occupied by it is retained. Resetting this mode is same as IMMERSIVE_HIDE_NAVIGATION_BAR.
IMMERSIVE_HIDE_SYSTEM_BARS	Hides both status and navigation bars, and space occupied by them are not retained. Unlike HIDE_SYSTEM_BARS, user can reset this mode by revealing them by swiping down from the top of the screen or swiping upward from the bottom of the screen.
IMMERSIVE_HIDE_SYSTEM_BARS_KEEP_LAYOUT_STABLE	Hides both status and navigation bars, and space occupied by them are retained. Resetting this mode is same as IMMERSIVE_HIDE_SYSTEM_BARS.

IMMERSIVE_ STICKY_HIDE_ STATUS_BAR	Hides the status bar and space occupied by it is not retained. User cannot reset this mode. When user swipes down from the top of the screen, the status bar appears momentarily and disappears.
IMMERSIVE_ STICKY_HIDE_ STATUS_BAR_ KEEP_LAYOUT_ STABLE	Hides the status bar and space occupied by it is retained. User cannot reset this mode. When user swipes down from the top of the screen, the status bar appears momentarily and disappears.
IMMERSIVE_ STICKY_HIDE_ NAVIGATION_BAR	Hides the navigation bar and space occupied by it is not retained. User cannot reset this mode. When user swipes upward from the bottom of the screen, the navigation bar appears momentarily and disappears.
IMMERSIVE_ STICKY_HIDE_ NAVIGATION_BAR_ KEEP_LAYOUT_ STABLE	Hides the navigation bar and space occupied by it is retained. User cannot reset this mode. When user swipes upward from the bottom of the screen, the navigation bar appears momentarily and disappears.
IMMERSIVE_ STICKY_HIDE_ SYSTEM_BARS	Hides both status and navigation bars, and space occupied by it is not retained. User cannot reset this mode. System bars appears momentarily when user tries to reveal them by swiping downward from the top of the screen or by swiping upward from the bottom of the screen.
IMMERSIVE_ STICKY_HIDE_ SYSTEM_BARS_ KEEP_LAYOUT_ STABLE	Hides both status and navigation bars, and space occupied by it is retained. System bars appears momentarily when user tries to reveal them by swiping downward from the top of the screen or by swiping upward from the bottom of the screen.

Note: The systemUIConfig flag is cleared when you move the application to the background and then bring it back to the foreground. So, the status bar/navigation bar is displayed by default when you move the application back to the foreground.

Example

```
function setApplicationProperties(){
  kony.application.setApplicationProperties({
    "statusBarColor": "ffff0000",
    "statusBarForegroundColor": "ff0000",
    "statusBarHidden": true,
    "statusBarStyle": constants.STATUS_BAR_STYLE_LIGHT_CONTENT,
    "navigationBarColor" : ffff0000,
    "systemUiConfig" : constants.HIDE_SYSTEM_BARS
});
}
```

Return Values

None

Special Considerations

The setApplicationProperties API is an application-level API and accepts key-value pairs as an input. When you set properties using the API to an application, the properties get applied to all the artifacts available in the application.

The properties can be applied at the application-level and at the form-level as well. When you set the properties to a form (form-level), the properties get applied only to that particular form in the visible region. When a user navigates to another form, the status and navigation bars may change based on the properties set to that form. Otherwise, the status and navigation bar behaves based on the properties set at the application-level. If the properties are set at both form level and application level, the properties set at the form level overrides the property set at the application level. If the properties are not set at the form-level and at the application-level, the device’s default settings are applied to status and navigation bars.

Exceptions

Error

Platform Availability

Available on all platforms.

Note: The statusBarForegroundColor and statusBarStyle input parameters are not available on Android.

kony.application.setAppMenuBadgeValue

This API allows you to set a badge value to the specified app menu item on the top-right corner of the app menu item. If you pass an empty string as the parameter, the badge value of the app menu item is cleared.

Syntax

```
kony.application.setAppMenuBadgeValue(appmenuID, menuItemId,badgeValue)
```

Input Parameters

Parameter	Description
appMenuId [String] - Mandatory	If you are setting the badge for an app menu item that was created dynamically, use the same ID that was used to create the app menu item.If you are setting the badge for an app menu item that was created from the IDE, use the ID available in the generated script file.
menuItemId [String] - Mandatory	Id of the app menu item to which the badge value to be set.
badgeValue [String] - Mandatory	Value of the badge. The value you specify in the badge value appears within the badge. If the length of the badge value is greater than 1 the badge is a rounded rectangle. For example, if you specify the value of the badge as 88, the number appears in a rounded rectangular badge. If the length of the badge value is 1, the badge is always a circle. The maximum number of characters that can be specified in a badge value is 9. If the badge value id beyond 9 only the first 9 characters are displayed.

Example

```
function setAppMenuBadgeValue() {
    //Set the AppMenuBadgeValue as "3" for the menu item with
id:"appmenuitemid3"
    kony.application.setAppMenuBadgeValue("accountMenu", "appmenuitemid3",
        "3");
}
```

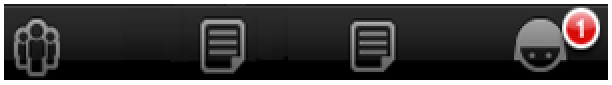
```
createAppMenu: function() {
    var appMenuItem1 = ["appmenuitemid1", "Accounts", "option1.png",
this.onClickMenuItem1];
    var appMenuItem2 = ["appmenuitemid2", "Examination", "option2.png",
this.onClickMenuItem2];
    var appMenu = [appMenuItem1, appMenuItem2];
    kony.application.createAppMenu("SampleAppMenu", appMenu, null, null);
    kony.application.setCurrentAppMenu("SampleAppMenu");
    kony.application.setAppMenuBadgeValue("SampleAppMenu", "appmenuitemid1",
"4");
    kony.application.setAppMenuBadgeValue("SampleAppMenu", "appmenuitemid2",
"6");

},
```

Return Values

None

UI Behavior

The following image depicts how a bade appears on an app menu item:



Platform Availability

Available only on iPhone and iPad.

kony.application.setAppTile

This API enables you to set the data for an application tile. If the user chooses to pin the application tile, the data set is visible. For more information on pinning a tile, refer http://www.microsoft.com/windowsphone/en-us/howto/wp7/start/move-or-delete-tile-on-start.aspx.

Syntax

```
//Mango
kony.application.seAppTitle(frontTileData, backTileData)

Windows 8
kony.application.setAppTitle(tileTemplateType, tileTemplateData)

Windows 8
setapptile(tileTemplateType, tileTemplateData)
```

Input Parameters for Mango

frontTileData [Object] - Mandatory, If you do not set the front tile data, a default image and values will be used

Specifies the data to be displayed at the front of a tile. This hash table has the following inputs:

Key	Description
title*[String]*	Specifies the title to be displayed at the front of a tile.The title appears at the bottom of the tile as white text.
backgroundImage*[String]*	Specifies the background image to be displayed at the front of a tile. ***Note:*** The backgroundImage will use the specified image to fill the entire space of the tile, regardless of its actual size.

Key	Description
count*[integer]*	specifies a number to be displayed at the top - right corner of the tile. The count value is displayed in a small black circle in white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

backTileData[Object] - Optional

Specifies the data to be displayed at the back of a tile. This table has the following key - value pairs:

Key	Description
title*[String]*	specifies the title to be displayed at the back of a tile.The title value appears at the bottom of the tile as white text.
backBackgroundImage*[String]*	Specifies the background image to be displayed at the back of a tile.
backcontent*[String]*	specifies the content to be displayed at the back of a tile. The back content appears at the top of the tile as white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

You can also set the backTileData to be nil. In this case, no information will be set on the back of the tile and the tile will not flip.

Input Parameters for Windows 8

Parameter	Description
tileTemplateType [String] - Mandatory	tileTemplateType is a string describing which tile template to use. Refer the tile template catalog for the list of supported tile templates
tileTemplateData [Array] - Optional	tileTemplateData is an array of data for the tile, according to the tileTemplateType being used. The tile template catalog. contains details of the data required for each tile template.

Example

Example1- Mango

```
frontTileData = ["Front Tile Title", "option1.png", 20];
backTileData = ["Front Tile Title", "calbtn.png", "This is the back tile
content"];
kony.application.setAppTile(frontTileData, backTileData);
```

Example2 - Windows 8

```
kony.application.setAppTile("TileSquareBlock", ["Hello", "World!"] );
```

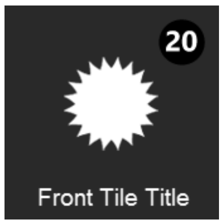
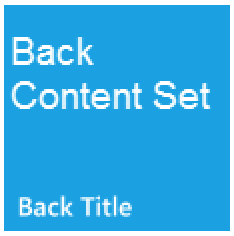
Return Values

None

UI Behavior - Mango

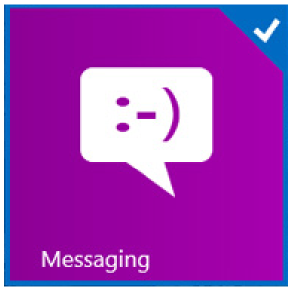
The Titles of both front and back tiles are displayed in white and this behavior cannot be changed even if the user sets a different theme or a background image. The count value for the front tile is displayed in a small black circle in white text at the top-right corner of the tile. The back content appears at the top-left corner of the tile as white text and is left justified.

Considering we have set values for both the frontTileData and the backTileData, the tiles might appear as follows:

Front Tile	Back Tile
	

UI Behavior - Windows 8

The title and the display name of the tile will be displayed in white or black depending on the setting in Kony Visualizer and this behavior cannot be changed even if the user sets a different theme or a background image. Unlike Mango. tiles cannot be flipped on Windows 8. The tile can be updated dynamically in Windows 8.



Platform Availability

Available on Windows Mango, Windows Phone 8, and Windows 8.

kony.application.setBMState

This API sets the bookmark state to the URL. This API accepts the *formID* and a *json* structure of key value pairs which will be added to the URL of the page.

Syntax

```
kony.application.setBMState(formID, State)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory	Identifier of the form to be bookmarked.
state [JSON Object] - Mandatory	A JSON object comprising key value pairs. The key value pairs are user defined. You cannot specify this as a nested structure. i.e the value part can not be another JSON object.

Example

```
//To set a bookmark for a URL, enter the following
var state = {
    Bookmark: "about",
    text: "About"
};
kony.application.setBMState("form1", state);
```

```
setState: function() {

    var state = {
        Bookmark: "about",
        text: "About"
    };
    kony.application.setBMState("Form1", state);
    alert("A new state is set to the URL ");
},
```

Return Values

None

Platform Availability

Supported for SPA and Desktop Web.

kony.application.setCheckBoxSelectionImageAlignment

This API is used to set the alignment of the checkBox selection image.

Syntax

```
kony.application.setCheckBoxSelectionImageAlignment(SelectionImageAlignment)
```

Input Parameters

Parameter	Description
SelectionImageAlignment [String] - Mandatory	One of the following string can be provided: • constants.CHECKBOX_SELECTION_IMAGE_ALIGNMENT_LEFT - Image will be aligned to left. • constants.CHECKBOX_SELECTION_IMAGE_ALIGNMENT_RIGHT - Image will be aligned to right.

Example

```
kony.application.setCheckBoxSelectionImageAlignment(constants.CHECKBOX_
SELECTION_IMAGE_ALIGNMENT_RIGHT);
```

Return Values

None

Exceptions

If input is invalid, an error is displayed.

Platform Availability

Available on iPhone and iPad.

kony.application.setCurrentSettingsMenu

This method uses the unique identifier which represents the Charm settings menu and sets it as current settings menu. There can be only one current settings menu that can be set any time. Calling this method multiple times, replaces the current Charm settings menu.

Syntax

```
kony.application.setCurrentSettingsMenu(id)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Identifier of the Charm setting menu to be set.

Example

```
//To create a Charm settings menu, enter the following
kony.application.setCurrentSettingsMenu("myMenu");
```

Return Values

None

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.setCurrentAppMenuFont

Sets the font name and font size of various app menu items in the current app menu.

Syntax

```
kony.application.setCurrentAppMenuFont(fontname,fontsize)
```

Input Parameters

Parameter	Description
fontname	A JavaScript string that corresponds to a particular font name. *Note:* If you want to set the system font to the required size, you must pass the fontname as “System,” or “System-Bold,” or “System-Italic.”
fontsize	A float value that corresponds to a specific font size.

Example

```
kony.application.setCurrentAppMenuFont("System-Italic", 12);
```

Return Values

None

Platform Availability

- iOS

kony.application.setDefaultListboxPadding

This API customizes the default paddings applied for a ListBox. Generally, default padding is appended to padding applied through layout configurations of the widget.

Syntax

```
kony.application.setDefaultListboxPadding(bool)
```

Input Parameters

Parameter	Description
bool [Boolean] - Mandatory	Indicates if the default padding is set. • True - Default padding is applied to ListBox widgets. • False - No default padding is applied to ListBox widgets.

Example

```
kony.application.setDefaultListboxPadding(false);
```

Return Values

None

Platform Availability

- iOS
- Android

kony.application.setDefaultTextboxPadding

This API customizes the default paddings applied for a Textbox. Generally, default padding is appended to padding applied through layout configurations of the widget.

Syntax

```
kony.application.setDefaultTextboxPadding(bool)
```

Input Parameters

Parameter	Description
bool [Boolean] - Mandatory	Indicates if the default padding is set. • True - Default padding is applied to textbox widgets. • False - No default padding is applied to textbox widgets.

Example

```
kony.application.setDefaultTextboxPadding(false);
```

Return Values

None

Platform Availability

- iOS

---

kony.application.setLibraryHeadlessModeCallback

This API is used to register a listener or a callback that receives request from a Native app to launch the Kony library without UI or in headless mode. The kony.application.setLibraryHeadlessModeCallback API is available from V8 SP3 onwards.

For more information about how to build an application in library mode, click here.

> ***Note:*** The kony.application.setLibraryHeadlessModeCallback API is only applicable when the app is launched in library mode.

Syntax

```
kony.application.sendLibraryResultHeadlessModeCallback(callback)
```

Input Parameters

Parameter	Description
callback	This parameter registers a function that receives the Native app's data as key-value pairs. The function registered in the callback parameter has the following syntax. `function callback(libraryArgs) { }` **libraryArgs** This parameter contains a JavaScript object with key-value pairs. This JavaScript object contains the data that is passed from the nNative app to the Kony library based depending on the contract of the Kony library.

Example

```
function callback(libraryArgs)
{
if (libraryArgs != null)
{
/* Use the data passed in libraryArgs and perform the required operation.
The result of the operation can be sent back to the native app using the
sendLibraryResultToNativeApp API.*/
kony.application.sendLibraryResultToNativeApp(resultData);
};
}
kony.application.setLibraryHeadlessModeCallback(callback);
```

Return Values

None.

Important Considerations

- The sendLibraryResultHeadlessModeCallback API must be set in either the pre-appinit or post-appinit event of the application.
- If you invoke this API more than once, the application only considers the callback that was set during the last instance.

Platform Availability

- Android
- iOS

kony.application.setRespectImageSizeForImageWidgetAlignment

This API sets the ImageWidget width to minimum or maximum according to available width or image width in absence of reference width. If ImageWidget is smaller than the available width, it is aligned using the widget alignment rules.

Syntax

```
kony.application.setRespectImageSizeForImageWidgetAlignment(bool)
```

Input Parameters

Parameter	Description
bool [Boolean] - Mandatory	Indicates if the default padding is set. • True - Default padding is applied to textbox widgets. • False - No default padding is applied to textbox widgets.

Example

```
kony.application.setRespectImageSizeForImageWidgetAlignment(true);
```

Return Values

None

Implementation Details

If this API is not used, the Image widget width is always the given width and image is center aligned.

Platform Availability

Available on iPhone and iPad.

kony.application.setSecondaryTile

This function enables you to create or update data for a secondary tile for an application. For more information about secondary tiles, refer Secondary tiles.

Syntax

```
Mango
kony.application.setSecondaryTile(id, frontTileData, backTileData)

Windows 8
kony.application.setSecondaryTile(id, shortname, displayname, imagename)

Windows 8
setSecondaryTile(id, shortname, displayname, imagename)
```

Input Parameters for Mango

id [String] - Mandatory

Unique identifier of the secondary tile.

frontTileData [Object] - Mandatory, If you do not set the front tile data, a default image and values will be used

Specifies the data to be displayed at the front of a tile. This hash table has the following inputs:

key	Description
title*[String]*	Specifies the title to be displayed at the front of a tile.The title appears at the bottom of the tile as white text.
backgroundImage*[String]*	Specifies the background image to be displayed at the front of a tile. ***Note:*** The backgroundImage will use the specified image to fill the entire space of the tile, regardless of its actual size.
count*[integer]*	specifies a number to be displayed at the top - right corner of the tile. The count value is displayed in a small black circle in white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

backTileData[Object] - Optional

Specifies the data to be displayed at the back of a tile. This table has the following key - value pairs:

key	Description
title*[String]*	specifies the title to be displayed at the back of a tile.The title value appears at the bottom of the tile as white text.
backBackgroundImage*[String]*	Specifies the background image to be displayed at the back of a tile.
backcontent*[String]*	specifies the content to be displayed at the back of a tile. The back content appears at the top of the tile as white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

You can also set the backTileData to be nil. In this case, no information will be set on the back of the tile and the tile will not flip.

Input Parameters for Windows 8

Parameter	Description
id [String] - Mandatory	Unique identifier of the secondary tile.
shortname [String] - Mandatory	A short name displayed directly on the tile. Anything over 40 characters will be truncated. The user has the option to change this value as part of the pinning process.
displayname[String] - Optional	The text specified here is displayed when you hover over the tile. There is no restriction on the display name length or the characters that it can contain.
imagename [String] - Optional	Name of the image to be displayed on the tile.

Example

The usage of this API on **Mango** is shown in the code snippet below:

```
//Setting the front and back tile data
frontTileData: {
    "Front Tile Title", "test.png", 20
}
backTileData: {
    "Front Tile Title", "test.png", "This is the back tile content"
}
//Set the secondary tile with the front and back tile data witht ile id 1234
kony.application.setSecondaryTile("1234", frontTileData, backTileData);
```

The usage of this API on **Windows 8** is shown in the code snippet below:

```
//Setting a secondary tile for an application with tile id myTile1.
kony.application.setSecondaryTile("myTile1", "title text", "display name",
"orange.png");
```

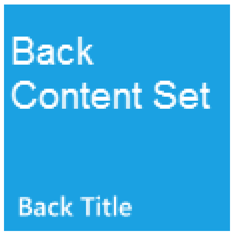
Return Values

None

UI Behavior Mango

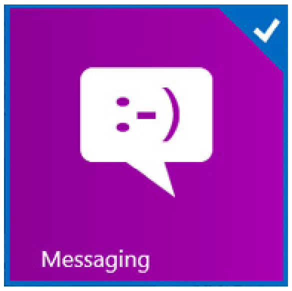
The Titles of both front and back tiles are displayed in white and this behavior cannot be changed even if the user sets a different theme or a background image. The count value for the front tile is displayed in a small black circle in white text at the top-right corner of the tile. The back content appears at the top-left corner of the tile as white text and is left justified.

Considering we have set values for both the frontTileData and the backTileData, the tiles are rendered as follows:



UI Behavior - Windows 8

The title and the display name of the tile will be displayed in white or black depending on the setting in Kony Visualizer and this behavior cannot be changed even if the user sets a different theme or a background image. Unlike Mango. tiles cannot be flipped on Windows 8. The tile can be updated dynamically in Windows 8.



Platform Availability

Available on Windows Mango, Windows Phone 8, and Windows 8.

kony.application.setSeoDataReadyFlag

Sets a flag indicating that the current form is ready to be cached for search engine optimization.

Syntax

```
kony.application.setSeoDataReadyFlag()
```

Example

```
// This function is added to the form.
function seoReady() {
kony.application.setSeoDataReadyFlag();
}
```

Input Parameters

None.

Return Values

None

Remarks

By default, most forms in Kony SPA web applications cannot be seen by web search spider robots when they crawl the web to index pages. For example, e Google robot crawls the web at regular intervals searching for web pages to index on servers all over the internet. When it encounters a Kony SPA web application, it will likely see only the first form. The content in other forms, which may be of significant value when users search the web, is not indexed.

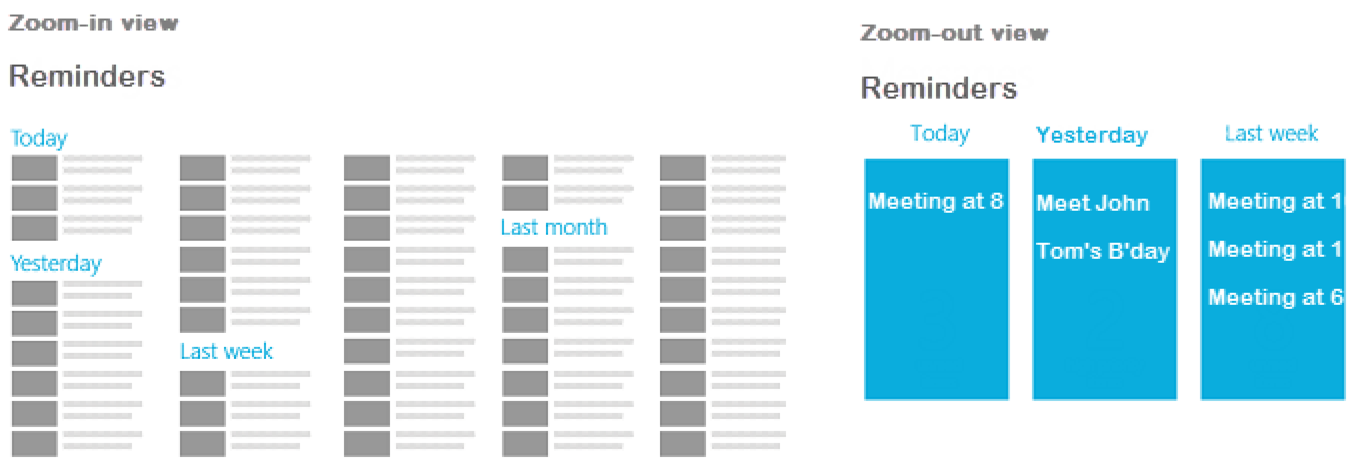
To make Kony SPA web application content visible to search engines so that they can perform optimal searches, you can enable search engine optimization (SEO) for their apps, as described in the Kony Visualizer User Guide. Calling `setSeoDataReadyFlag` for each form tells the Kony's SRO technology to cache the form for SEO.

You app should only call the `setSeoDataReadyFlag` function after all service call responses are processed, all required data is rendered, and the Kony form is ready to be cached. A good time for your app to invoke the `setSeoDataReadyFlag` function is when it is processing the form's `postShow` event.

kony.application.setZoomedOutView

You can design a user interface to be displayed to the end user when you zoom out of an application. This API enables you to set a form to be shown to the user when a zoom out gesture is performed.

This API is introduced to support the Semantic Zoom feature introduced with Windows 8. For more information about Semantic zoom,zoomed in, and zoomed out views, refer http://msdn.microsoft.com/en-us/library/windows/apps/hh465319.aspx.



Syntax

```
kony.application.setZoomedOutView(formid)
```

Input Parameters

Parameter	Description
formid [Widget Reference] - Mandatory	Specifies the id of the form to be displayed when the user zooms out of the application.

Example

```
//Zooming out of a an application and displaying a form with an id myForm1 on
zoomout

function zoomout()
{
    kony.application.setZoomedOutView(myForm1);
}
```

Return Values

None

Platform Availability

Windows 8

kony.application.showLoadingScreen

This API allows you to display a loading screen (following a certain color schema) to the user while another action is in progress. The loading screen can be defined in such a way it can either block the UI or does not block the UI. Typically, the loading screen is a semi-transparent screen over-laid on the current form.

Use Case

You can use this API to display a loading screen to the end user with a specific skin attached to it in the following scenarios:

- on invoking a service
- on executing a user-defined function that takes some time for execution.

Syntax

```
kony.application.showLoadingScreen(skin, text, position, isBlocked,
showProgressIndicator, properties)
```

Input Parameters

skin [skin identifier] - Mandatory

The skin to be applied to the loading screen. All the skin attributes supported by different platforms are applicable. This is a reference to an existing skin. If it is nil, the native platform skin is applied.

text [String] - Mandatory

The text to be displayed when displaying the loading screen. For example, "Searching flights...". If it is nil, no text is displayed.

position [String] - Mandatory

Indicates the position of the loading screen, whether it should occupy the entire screen or just the center of the screen. The possible values for position are :

- constants.LOADING_SCREEN_POSITION_FULL_SCREEN (default)
- constants.LOADING_SCREEN_POSITION_ONLY_CENTER.

> ***Note:*** Currently only center and fullscreen are supported.

isBlocked [Boolean] - Mandatory

Indicates if the UI should be blocked.

- true - no click and scroll is allowed on the UI. The Back button on the device also does not work. true is the default value.
- false - click and scroll is allowed(UI is not blocked) and back button of the device also works.

If nothing is specified, default value is used.

showProgressIndicator [Boolean] - Mandatory

Indicates if the progress indicator should be displayed.

- true - displays the progress indicator. true is the default value.
- false - does not display the progress indicator.

Note: The progress indicator is a set of animating images.

properties [Object] - Mandatory

Indicates the platform specific properties to be applied to the loading screen.

Note: You must pass this parameter as nil on all platforms except iPhone platform.

The following are the key-value pairs to be passed in the table for iPhone:

Key	Description
shouldShowLabelInBottom (boolean/string)	To display the text content at the bottom or the top side of the activity indicator. Default value is false. If string "true" is passed to this property then it will display the content at the bottom of the activity indicator.
separatorHeight(number)	Height between the activity indicator and text to be displayed. Default height is 30px. Should be greater than 0. *Important:* These key-value pairs are applicable only on iPhone platform.
progressIndicatorColor (String)	Color is applied for progress indicator when it is shown. (Optional) *Note:* Only "white" and "gray" colors are supported. Default is white. White and gray values are not case sensitive.

progressIndicatorType (Constant)	Set the type of progress indicator. The following types are provided: • constants.PROGRESS_INDICATOR_TYPE_SMALL • constants.PROGRESS_INDICATOR_TYPE_BIG • constants.PROGRESS__INDICATOR_TYPE_DEFAULT (Default value) ***Note:*** *progressIndicatorType:constants.PROGRESS_ INDICATOR_TYPE_BIG* with "progressIndicatorColor:GREY" is not supported in iOS platform.

The following are the key-value pairs to be passed in the table for all platforms except iPhone:

Key	Description
progressIndicatorColor (String)	Color is applied for progress indicator when it is shown. (Optional) ***Note:*** All colors with RGBA (Hex code) are supported, but for "white" and "gray" you can pass it as a string value. Color value should be similar to the value provided in the theme.

Example

```
//Invoking showLoadingScreen on iPhone platform.

kony.application.showLoadingScreen(null, "testf",
    constants.LOADING_SCREEN_POSITION_ONLY_CENTER, false, true, {
        shouldShowLabelInBottom: "true",
        separatorHeight: 200,
        progressIndicatorType: constants.PROGRESS_INDICATOR_TYPE_SMALL,
        progressIndicatorColor: "Gray"
    });
```

```
//Invoking showLoadingScreen on other platform.

kony.application.showLoadingScreen("loadskin", "LoadingScreen",
constants.LOADING_SCREEN_POSITION_ONLY_CENTER, false, true, {
    enableMenuKey: true,
    enableBackKey: true,
    progressIndicatorColor: "ffffff77"
});
```

Return Values

None

Platform Availability

Available on all platforms except Mobile Web.

kony.application.startForegroundService

When invoked, this API starts a foreground service that runs continuously even when an application is running in the background, i.e. when the user is not directly interacting with the app.

The foreground services display notifications to send updates to the users. The notifications can be customized using updateForegroundNotification API.

Foreground service can be started in scenarios which require your app to be actively running after being placed in the background or after turning the device's display off.

Some sample use cases include navigation while driving, tracing the path while running, and playing music. For more information, refer here.

Note: Ensure that you set the `locationListenerType` property to `always` in the **androidbuild.properties** file. Appropriate entries are added to the **AndroidManifest.xml** file depending on the type of the location listener.

Syntax

```
kony.application.startForegroundService
(foregroundServiceType,notificationConfig)
```

Input Parameters

Parameter	Description
foregroundServiceType [Constant] - Mandatory	Specifies the type of the foreground service. The available foreground service type constant is kony.service.LOCATION.

notificationConfig-Mandatory	An object with notification properties. Following are the properties: • title[String]- Mandatory: Specifies the title of the notification, i18n Strings can be used. • body[String]: Specifies the content body of the notification, i18n Strings can be used. • autoHideNotificationInForeground: Specifies whether the notification must be hidden when activity comes to the foreground. The property works until the foreground service is stopped. Notification is always shown when the activity goes to background irrespective of this property while service is running. • onNotificationClick: An event callback is invoked by the platform when the user performs a click action on the foreground service notification. • actions[JSObject[]]: Specifies a list of actions to be shown to the user in the notification. Each action object in the list contains the following keys: • actionType [Constant]- Specifies the type of the action, The available actions type constants are kony.application.FGSERVICE_ACTION_LAUNCH_ACTIVITY and kony.application.FGSERVICE_ACTION_CUSTOM • actionText [String]- A text describing the action. i18n Strings can be used. If this is set to null or an empty string, the action is not populated in notification. • actionCallback[Function]- An action callback is invoked by the platform when the user performs a click action on the button in notification.

Example

```
var notificationConfig= {
title : "Notification Title",
body : "Notification Content",
actions:[ {actionType: kony.application.LAUNCH_ACTIVITY,
actionText: "Launch Activity", actionCallback:  activityActionCallback}]
};

kony.application.startForegroundService
(kony.service.LOCATION,notificationConfig);

function activityActionCallback() {

}
```

Return Values

None

Platform Availability

Android

kony.application.stopForegroundService

When invoked, this API enables you to stop the foreground service for an application that is running in the background.

> ***Note:*** Ensure that you set the `locationListenerType` property to `always` in the **androidbuild.properties** file. Appropriate entries are added to the **AndroidManifest.xml** file depending on the type of the location listener.

Syntax

```
kony.application.stopForegroundService(foregroundServiceType)
```

Input Parameters

Parameter	Description
foregroundServiceType[Constant] - Mandatory	Specifies the type of the foreground service. The available foreground service type constant is kony.service.LOCATION.

Example

```
kony.application.stopForegroundService(kony.service.LOCATION);
```

Return Values

None

Platform Availability

Android

kony.application.unregisterForIdleTimeout

This API specifies that the application must *not* timeout after a defined period of inactivity (time difference between the current device time and the last time you clicked on any user interface component). This API unregisters the application from idletimeout.

Use Case

This API must be used in case you do not want to track the idle time, i.e., do not want to logout even if the application is inactive for a specific time interval.

Syntax

```
kony.application.unregisterForIdleTimeout()
```

Example

```
function unregisterForIdleTimeout() {
    kony.application.unregisterForIdleTimeout();
    alert("====unRegister======");
}
```

Input Parameters

None

Return Values

None

Rules and Restrictions

This API is applicable only when the Form property *Enable Time Out* is set to *true*. For more information about *Enable Time Out* property, see *Widget User Guide*.

Platform Availability

Available on all platforms.

kony.application.updateForegroundNotification

When invoked, this API enables you to customize and update the existing notifications shown by the foreground service.

Note: Ensure that you set the `locationListenerType` property to `always` in the **androidbuild.properties** file. Appropriate entries are added to the **AndroidManifest.xml** file depending on the type of the location listener.

Syntax

```
kony.application.updateForegroundNotification
([foregroundServiceType,notificationConfig)
```

Input Parameters

Parameter	Description
foregroundServiceType [Constant] - Mandatory	Specifies the type of the foreground service. The available foreground service type constant is kony.service.LOCATION.

notificationConfig- Mandatory	An object with notification properties. Following are the properties: • title[String]- Mandatory: Specifies the title of the notification, i18n Strings can be used. • body[String]: Specifies the content body of the notification, i18n Strings can be used. • autoHideNotificationInForeground: Specifies whether the notification must be hidden when activity comes to the foreground. The property works until the foreground service is stopped. Notification is always shown when the activity goes to background irrespective of this property while service is running. • onNotificationClick: An event callback is invoked by the platform when the user performs a click action on the foreground service notification. • actions[JSObject[]]: Specifies a list of actions to be shown to the user in the notification. Each action object in the list contains the following keys: ◦ actionType [Constant]- Specifies the type of the action, The available actions type constants are kony.application.FGSERVICE_ACTION_LAUNCH_ ACTIVITY and kony.application.FGSERVICE_ACTION_ CUSTOM ◦ actionText [String]- A text describing the action. i18n Strings can be used. If this is set to null or an empty string, the action is not populated in notification. ◦ actionCallback[Function]- An action callback is invoked by the platform when the user performs a click action on the button in notification.

Example

```
var notificationConfig= {
title : "Location Details",
content : "Longitude : 17.3850, Latitude:  78.4867",
actions:[ {actionType: kony.application.LAUNCH_ACTIVITY,
actionText: "Launch Activity", actionCallback:  activityActionCallback},
{actionType: kony.application.CUSTOM,
actionText: "Remove Foreground Service, actionCallback:
serviceActionCallback}]
};

kony.application.updateForegroundNotification
(kony.service.LOCATION,notificationConfig);


function activityActionCallback() {

}

function serviceActionCallback() {

}
```

Return Values

None

Platform Availability

Android

kony.application.zoomIn

This API enables you to zoom in on an application programmatically. If the application is already zoomed in, this API has no effect.

UseCase

Suppose your application is in a zoomed out state where the contacts associated with that letter is displayed. You can use this API to view all the contacts alphabeticallyand present the data using the letters of the alphabet.

Syntax

```
kony.application.zoomIn()
```

Example

```
//Enabling zoomin on an application

function zoomIn(){
    kony.application.zoomIn();
}
```

Input Parameters

None

Return Values

None

Platform Availability

Windows 8

8. Application Settings API

The Application Settings API enables you to retrieve and update certain settings of an application. Using the application settings functions, you can read the values that you have set for an application and write new values. For instance, you can choose to define different views for widgets and configure themes.

Important: The functions in this API are currently only applicable on iPhone platform.

The Application Settings API uses `kony.application.settings Namespace` and the following API elements.

Function	Description
`kony.application.settings.read`	Enables your app to read the configuration values associated with specified keys.
`kony.application.settings.write`	Enables your app to write the configuration values associated with specified keys.

To know the current settings of the application, use the `kony.application.settings.read` function. This function will provide information about the various app settings and their values. For example, you can read the settings such as themes and widget views. If you want to modify these settings, use the `kony.application.settings.write` function. This function will enable you to replace the existing values of the app settings with the new values.

To view the functionality of the Application Settings API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

8.1 kony.application.settings Namespace

The kony.application.settings Namespace provides the following API elements.

8.1.1 Functions

kony.application.settings.read

This function enables your app to read the configuration values associated with specified keys.

Syntax

```
kony.application.settings.read(
    key,
    onsuccesscallback,
    onfailureback)
```

Input Parameters

Parameters	Description
key	A string that specifies the keys for the settings to read.

Parameters	Description
onsuccesscallback	A function that is executed when the key-value pair specified in the *key* parameter is read successfully. For details, see the **Remarks** section below.
onfailurecallback	A callback function is executed when there is an error in fetching the app's settings. This callback function receives the error code and the error message. For details, see the **Remarks** section below.

Example

```
// Define the callback functions first.

function key1OnSuccessCallback(key, value) {
    //Write Your Logic here
}

function key1OnFailureCallback(errorcode, errormessage) {
    alert("The error is :" + errormessage);
}

function key2OnSuccessCallback(key, value) {
    //Write Your Logic here

}

function key2OnFailureCallback(errorcode, errormessage) {
    alert("The error :" + errormessage);
}

function readSettings() {
    //Read the settings
    kony.application.settings.read("key1", key1OnSuccessCallback,
key1OnFailureCallback);
    kony.application.settings.read("key2", key2OnSuccessCallback,
key2OnFailureCallback);
}
```

```
read: function() {

    kony.application.settings.read("key1", this.onsuccesscallbackR,
this.onfailureback);
    kony.application.settings.read("key2", this.onsuccesscallbackR,
this.onfailurebackR);
},

onfailureback: function(errorcode, errormessage) {
    alert("err is :" + errormessage);
},
onsuccesscallbackR: function(key, value) {
    var params = {};
    params.value = value;

    switch (value) {
        case "Table view":
            params.viewType = constants.SEGUI_VIEW_TYPE_TABLEVIEW;
            break;
        case "Page view":
            params.viewType = constants.SEGUI_VIEW_TYPE_PAGEVIEW;
            params.pageOffDotImage = "orngsld";
            params.pageOnDotImage = "whitesld";
            break;
        case "Coverflow view":
            params.viewType = constants.SEGUI_VIEW_TYPE_COVERFLOW;
            break;
        case "Cylinder view":
            params.viewType = constants.SEGUI_VIEW_TYPE_CYLINDER;
            break;
        case "Linear view":
            params.viewType = constants.SEGUI_VIEW_TYPE_LINEAR;
            break;
        case "Stack view":
            params.viewType = constants.SEGUI_VIEW_TYPE_STACK;
            break;
    }
    var ntf = new kony.mvc.Navigation("frmAppSetRead");
    ntf.navigate(params);
},

onfailurebackR: function(errorcode, errormessage) {
    alert("Err is :" + errormessage);
}
```

Return Values

None.

Remarks

This function reads an app's settings from the settings file that is bundled with the application's binary. Your app can use this function to read, among other things, the user-defined values such as the preferred locations.

This is an asynchronous call so it returns immediately. Upon successfully reading the app's settings, the Kony Visualizer API framework automatically invokes the callback your app passes in the *onsuccesscallback* parameter. The callback function has the following signature.

```
onsuccess(key, value);
```

When this callback function is invoked, the *key* parameter contains a string that specifies the name of the key whose value is being retrieved.

The *value* parameter contains the settings that were read. It can be `Boolean`, `string`, `double`, or an object. It contains an object when *Display option* is set to multiselect.

If the Kony Visualizer API framework cannot read the settings associated with the *key* parameter to the `kony.application.settings.read` function, then the framework automatically invokes the function in the *onfailurecallback* parameter. The callback function has the following signature.

```
onfailure(errorcode,errormessage);
```

When the framework calls the `onfailure` callback function *errorcode* parameter of contains a numeric error code and the *errormessage* parameter contains the corresponding error message as a `string`.

Platform Availability

Available only on iPhone

kony.application.settings.write

This function enables your app to write values associated with specified keys.

Syntax

```
kony.application.settings.write(
    key,
    value,
    onsuccesscallback,
    onfailurecallback);
```

Input Parameters

Parameter	Description
key	A string that specifies the unique key. The key that you specify must exist. You cannot introduce a new key.
value	A string that holds the value to be set for a key. If this function is called multiple times using the same key, existing value for the key is replaced with the new value. A key's value can be Boolean, string, double, or an object. It can contain an object only when *Display option* is set to multiselect.
onsuccesscallback	The callback function that is executed when the new value is successfully assigned to the specified key. For details, see the **Remarks** section below.
onfailurecallback	The callback function that is executed when there is an error in updating the configuration. This callback function receives an error code and an error message. For details, see the **Remarks** section below.

Example

```
//Callback functions.
function key1OnSuccessCallback(key, value) {
    //Write Your Logic here
}

function key1OnFailureBack(errorcode, errormessage) {
    alert("Error is :" + errormessage);
}

function key2OnSuccessCallback(key, value) {
    //Write Your Logic here
}

function key2OnFailureBack(errorcode, errormessage) {
    alert("Error is :" + errormessage);
}

function write() {
    //Write the settings
    kony.application.settings.write("key1", "Message1", key1OnSuccessCallback,
key1OnFailureBack);
    kony.application.settings.write("key2", "Message2", key2OnSuccessCallback,
key2OnFailureBack);
}
```

```
write: function() {

    var str1 = this.view.txtMsg.text + " ";
    var str2 = this.view.lstBxType.selectedKey;

    kony.application.settings.write("key1", str1, this.onsuccesscallbackw,
this.onfailurebackw);
    kony.application.settings.write("key2", str2, this.onsuccesscallbackWrite,
this.onfailurebackWrite);

},
onsuccesscallbackw: function(key, value) {
    kony.print("Value is :" + value);
},

onfailurebackw: function(errorcode, errormessage) {
    alert("Error is :" + errormessage);
},


onsuccesscallbackWrite: function(key, value) {

    this.view.lblAppAfterSet.text = "App settings are successful";


},
onfailurebackWrite: function(errorcode, errormessage) {
    alert("Error is :" + errormessage);
}
```

Return Values

None.

Remarks

This function sets the value of configuration options in the configuration file that is associated with the app. New keys cannot be introduced with this function. Your app can only write to existing keys. This is an asynchronous call, so it returns immediately and does not wait for the underlying hardware to write the value to the specified key. It uses callback functions to communicate success or failure. The callback functions for success and failure are set using this function's *onsuccesscallback* and *onfailurecallback*, respectively.

The callback function for the *onsuccesscallback* parameter of the `kony.application.settings.write` function has the following syntax.

```
onsuccesscallback(key,value);
```

where the `key` parameter to this callback is a string containing the name of the key that was set, and the callback function's `value` parameter contains the value that the key was set to. The callback function's `value` parameter can be `boolean`, `string`, `double`, or an object. An object is only passed in through the *value* parameter when "*Display option*" is multiselect.

If the `kony.application.settings.write` function cannot set the value of the specified key, in invokes another callback function. The callback function for the `kony.application.settings.write` function's *onfailurecallback* parameter has the following syntax.

```
onfailurecallback(errorcode,errormessage);
```

where the `errorcode` parameter contains a numeric error code and the `errormessage` parameter holds a string that specifies the error message.

Platform Availability

Available only on iPhone

9. Automation API

9.1 Kony Automation Framework

Kony Automation Framework provides you a more convenient way to test your application across various platforms. The framework follows the Write Once, Run Anywhere methodology, so that different automation efforts put for each platform for the same application can be reused. JavaScript is used for Kony Automation, and you do not need to have expertise on any other native language to use it. It is currently supported on iOS, Android, Windows 10 and DesktopWeb, SPA platforms.

9.1.1 JavaScript Testing Framework - Jasmine

Jasmine is a behavior-driven development framework for testing JavaScript code. The framework does not depend on any other JavaScript frameworks. Furthermore, the framework has a clear and obvious syntax, thus enabling you to easily write tests.

Sample Code

```
describe("Sample 1", function() {
    it("test case1", async
        function() {
            var username = kony.automation.widget.getWidgetProperty
(["LoginPage", "loginBox", "userName"], "text");

            expect(username).toEqual("Linda"); // assertion for
username value;
        });
});
```

9.2 Kony Automation APIs: Supported Items

Kony supports automation for both MVC and non-MVC projects. Automation API supports the following widgets:

- All views of all widgets
- Platform-specific widgets
- Masters and User widgets (components)
- Deprecated widgets - Box, Form, ScrollBox, etc.

In addition, Automation API supports the following items:

- Low-level actions on all supported widgets
- Touch
- Scroll
- Hardware buttons
- Back button
- Menu button
- Text entry
- Camera capture
- Map and browser interaction

Kony Automation API consists of the kony.automation Namespace and its following inherent API groups:

- Widget APIs
- Low-level Touch and Gesture APIs

- Miscellaneous Automation APIs
- Existing Kony APIs

The Kony Automation API uses the `kony.Automation` namespace and the following API elements:

Methods	Description
`kony.automation.alert.click`	Clicks the provided button, if visible, for the Alert. If multiple alerts are simultaneously shown, this API clicks the most recent alert displayed on the screen.
`kony.automation.appmenu.click`	Triggers the current app menu item click event, if it is visible and is enabled.
`kony.automation.box.click`	Triggers the Box click event on the specified widget, if it is visible and enabled.
`kony.automation.browser.click`	Triggers the Browser click event on the specified widget, if it is visible and enabled.
`kony.automation.button.click`	Triggers the Button click event on the specified widget, if it is visible and enabled.

Methods	Description
`kony.automation.calendar.selectDate`	Triggers the Calendar click event on the specified widget, if it is visible and enabled.
`kony.automation.camera.capture`	Triggers the Camera capture event on the specified widget, if it is visible and enabled.
`kony.automation.checkboxgroup.click`	Triggers the CheckBoxGroup click event on the specified widget, if it is visible and enabled.
`kony.automation.combobox.selectItem`	Selects the item in the ComboBox widget, if it is visible and enabled.
`kony.automation.cordovabrowser.click`	Triggers the CordovaBrowser click event on the specified widget, if it is visible and enabled.
`kony.automation.datagrid.click`	Triggers the DataGrid click event on the specified widget, if it is visible and enabled. Syntax

Methods	Description
`kony.automation.flexcontainer.click`	Triggers the FlexContainer click event on the specified widget, if it is visible and enabled.
`kony.automation.horizontalimagestrip.click`	Triggers the HorizontalImageStrip click event on the specified widget, if it is visible and enabled.
`kony.automation.imagegallery.click`	Triggers the ImageGallery click event on the specified widget, if it is visible and enabled.
`kony.automation.link.click`	Triggers the link click event on the specified widget, if it is visible and enabled.
`kony.automation.listbox.selectItem`	Triggers the ListBox click event on the specified widget, if it is visible and enabled.
`kony.automation.map.click`	Triggers the Map click event on the specified widget, if it is visible and enabled.
`kony.automation.map.clickOnPin`	Triggers the click event on the pin of the Map, if it is visible and enabled.

Methods	Description
kony.automation.map.clickOnPinCallout	Triggers the click event of the callout in Map widget, if it is visible and enabled.
kony.automation.pickerview.selectItem	Selects the item in PickerView widget, if it is visible and enabled.
kony.automation.radiobuttongroup.click	Triggers the RadioButtonGroup click event on the specified widget, if it is visible and enabled.
kony.automation.richtext.click	Triggers the RichText click event on the specified widget, if it is visible and enabled.
kony.automation.segmentedui.click	Triggers the SegmentedUI click event on the specified widget, if it is visible and enabled.
kony.automation.segmentedui.pull	Triggers the onPull event on the SegmentedUI, if it is set.
kony.automation.segmentedui.push	Triggers the onPush event on the SegmentedUI, if it is set.

Methods	Description
`kony.automation.segmentedui.scrollToBottom`	Makes the segment scroll to the last row and then triggers the onReachEnd event of SegmentedUI, if it is set.
`kony.automation.segmentedui.scrollToRow`	Triggers the segment to scroll to the row specified by the index of the specified widget, if it is visible and enabled.
`kony.automation.segmentedui.scrollToTop`	Makes the segment to scroll to the first row.
`kony.automation.segmentedui.scrollToWidget`	Scrolls to ensure that the widget appears in view. It is an awaitable API.
`kony.automation.slider.slide`	Triggers the Slider slide event on the specified widget, if it is visible and enabled.
`kony.automation.switch.toggle`	Toggles the Switch between ON/OFF on the specified widget, if it is visible and enabled.
`kony.automation.tabpane.click`	Clicks the tab with the specified tabID on the TabPane widget, if it is visible and enabled.

Methods	Description
`kony.automation.textbox.enterText`	Enters the specified text into the TextBox, if it is visible and enabled.
`kony.automation.textarea.enterText`	Enters the specified text into the TextArea, if it is visible and enabled.
`kony.automation.widget.touch`	Triggers the touch event on the specified widget, if it is visible and enabled.
`kony.automation.widget.scroll`	Triggers the scroll event on the specified widget, if it is visible and enabled.
`kony.automation.widget.canScroll`	Returns whether the scroll functionality is enabled for the specified widget.
`kony.automation.playback.wait`	Introduces a delay time in the playback as specified. It is an awaitable API.
`kony.automation.playback.waitFor`	Waits for the widget to load completely. It is an awaitable API.
`kony.automation.widget.getWidgetProperty`	Returns the particular Kony-defined property on the specified widget.

Methods	Description
`kony.automation.widget.getProperty`	Returns the particular native property on the specified widget.
`kony.automation.device.deviceBack`	Invokes the back action of the device. It is an awaitable API.
`kony.automation.capture`	The api takes a screenshot of the widget passed. If the widget does not pass, the screenshot captures the entire screen.
`kony.print`	Prints debugging output.
`kony.os.deviceInfo`	This API allows the developers to get information about the device in which the application is launched.

9.3 kony.automation Namespace

The kony.automation Namespace consists of the following API groups.

9.3.1 Widget APIs

Widget Path

Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array. It is applicable for almost all widget APIs.

> ***Note:*** Whenever a Segment row is a part of Widget path, it refers to the top level flex. This flex ID should not be provided in the subsequent path.

Syntax

```
<widgetpath> [array of strings];
```

Mandatory/Optional

Mandatory

Examples

- Each string denotes the widget ID in the hierarchy.

```
kony.automation.button.click(["frmHomeLogin", "btnLogin"]);

//Here, ["frmHomeLogin","btnLogin"]is the widget path.
```

- The widget path string can consist of an array indexer for SegmentedUI widget to denote the corresponding row.

```
kony.automation.button.click(["frmHomeLogin", "segUi1[0,2]",
"btnLogin"]);
kony.automation.button.click(["frmHomeLogin", "segUi1[2]",
"btnLogin"]);
```

- The widget path string can be a date for a Calendar cell template.

```
kony.automation.button.click(["frmHomeLogin", "calendar11[05,05,2016]
", "btnLogin"]);
```

- The widget path string can be the friendly name of a form.

```
kony.automation.button.click(["Login Page", "userwidget1",
"btnLogin"]);
```

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

The kony.automation Namespace comprises of the following Widget APIs.

kony.automation.alert.click

Clicks the provided button, if visible, for the Alert. If multiple alerts are simultaneously shown, this API clicks the most recent alert displayed on the screen.

Syntax

```
kony.automation.alert.click(<buttonIndex>);
```

Input Parameters

Parameters	Description
buttonIndex [0 or 1] [Optional]	Alert widget supports two buttons. This APIs triggers the click on the first button if the index is 0, and it triggers the click on the second button if the index is 1. If you do not pass any buttonIndex, the API triggers the click on the first button.

Example

```
kony.automation.alert.click(0);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10

kony.automation.appmenu.click

Triggers the current app menu item click event, if it is visible and is enabled.

Syntax

```
kony.automation.appmenu.click(<menuitem_id>);
```

Input Parameters

Parameters	Description
menuitem_id [string] [Mandatory]	Finds the menu item with the specified ID in the current app menu. If it is found, then it triggers a click on it.

Example

```
kony.automation.appmenu.click("menuLogout");
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.box.click

Triggers the Box click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.box.click (<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.box.click(["frmHomeLogin","boxId"]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.browser.click

Triggers the Browser click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.browser.click (<widgetpath>, <xyposition>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
xyposition [array of two numbers] [Mandatory]	Array of [x, y] co-ordinates.

Example

```
kony.automation.browser.click(["frmHomeLogin","browserId"], [12,50]);
```

Return Values

None

Platform Availability

- Android
- iOS

kony.automation.button.click

Triggers the Button click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.button.click (<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.button.click(["frmHomeLogin","btnLogin"]);
kony.automation.button.click(["frmHomeLogin","segUi1[0,2]", "btnLogin"]);
kony.automation.button.click(["frmHomeLogin","segUi1[2]", "btnLogin"]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.calendar.selectDate

Triggers the Calendar click event on the specified widget, if it is visible and enabled.

Note: Supported view type is pop-up grid

Syntax

```
kony.automation.calendar.selectDate (<widgetpath>, <newDate>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
newDate [array] [Mandatory]	Array representation of a date in mm/dd/yyyy format as [mm, dd, yyyy] co-ordinate. This format is irrespective of the calendar format.

Example

```
kony.automation.calendar.selectDate(["frmHomeLogin","calenderId"],
[12,15,2017]);
```

Return Values

None

> ***Note:*** Automation is supported only for the Calendar default view.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.camera.capture

Triggers the Camera capture event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.camera.capture(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.camera.capture(["frmHomeLogin","cameraId"]);
```

Return Values

None

> ***Note:*** Automation is supported only for the Overlay Camera.

Platform Availability

- Android
- iOS

kony.automation.checkboxgroup.click

Triggers the CheckBoxGroup click event on the specified widget, if it is visible and enabled.

> ***Note:*** Supported view type is onScreen Wheel

Syntax

```
kony.automation.checkboxgroup.click (<widgetpath>, <chkBoxKey>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
chkBoxKey [array] [Mandatory]	CheckBoxGroup item that is to be clicked.

Example

```
kony.automation.checkboxgroup.click(["frmHomeLogin","checkboxgroupId"],
"checkBoxKey");
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.combobox.selectItem

Selects the item in the ComboBox widget, if it is visible and enabled.

Syntax

```
kony.automation.combobox.selectItem(<widgetpath>, <valueIndex>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
key [string] [Mandatory]	Indicates the key in the ComboBox for which item is to be clicked.

Example

```
kony.automation.combobox.selectItem(["frmHomeLogin","comboboxId "], "key1");
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.cordovabrowser.click

Triggers the CordovaBrowser click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.cordovabrowser.click (<widgetpath>, <xyposition>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
xyposition [array of two numbers] [Mandatory]	Array of [x, y] co-ordinates.

Example

```
kony.automation.cordovabrowser.click(["frmHomeLogin","browserID"], [12,50]);
```

Return Values

None

Platform Availability

- Android
- iOS

kony.automation.datagrid.click

Triggers the DataGrid click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.datagrid.click (<widgetpath>, <row_col_position>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
row_col_position [object] [Mandatory]	Object representation of position as `{“row” : <rowIndex>, “col” : <colIndex>}`

Example

```
kony.automation.datagrid.click(["frmHomeLogin","datagridId"], {“row” : 1,
“col” : 2});
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.flexcontainer.click

Triggers the FlexContainer click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.flexcontainer.click (<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.flexcontainer.click(["frmHomeLogin","flexContainerId"]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.horizontalimagestrip.click

Triggers the HorizontalImageStrip click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.horizontalimagestrip.click (<widgetpath>, <imageItemIndex>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
imageItemIndex [number] [Mandatory]	The index of image items that are present in the list of image items

Example

```
kony.automation.horizontalimagestrip.click(["frmHomeLogin","hzImgStrip"], 2);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.imagegallery.click

Triggers the ImageGallery click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.imagegallery.click (<widgetpath>, <imageItemIndex>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
imageItemIndex [number] [Mandatory]	The index of image items that are present in the list of image items

Example

```
kony.automation.imagegallery.click(["frmHomeLogin","imageClickID"], 2);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.link.click

Triggers the link click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.link.click (<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.link.click(["frmHomeLogin","linkID"]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.listbox.selectItem

Triggers the ListBox click event on the specified widget, if it is visible and enabled.

Note: Supported view type is List.

Syntax

```
kony.automation.listbox.selectItem(<widgetpath>, <key>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
key [string] [Mandatory]	Indicates the key in the ListBox for which item is to be clicked.

Return Values

None

Example

```
kony.automation.listbox.selectItem(["frmHomeLogin","listbox"], "key1");
```

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.map.click

Triggers the Map click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.map.click(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.map.click(["frmHomeLogin","mapID"]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.map.clickOnPin

Triggers the click event on the pin of the Map, if it is visible and enabled.

Syntax

```
kony.automation.map.clickOnPin(<widgetpath>, <pinData>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
pinData [object] [Mandatory]	Represents the pin information as an object. For example, {lat:"17.445775", lon:"78.3731"}.

Example

```
kony.automation.map.clickOnPin(["frmHomeLogin", "btnLogin"], {
    "lat": "17.445775",
    "lon": "78.3731",
    "name": "Campus 1"
});
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.map.clickOnPinCallout

Triggers the click event of the callout in Map widget, if it is visible and enabled.

Syntax

```
kony.automation.map.clickOnPinCallout(<widgetpath>, <pinData>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
pinData [object] [Mandatory]	Represents the pin information as an object. For example, {lat:"17.445775", lon:"78.3731"}.

Example

```
kony.automation.map.clickOnPinCallout(["frmHomeLogin","btnLogin"], {
"lat": "17.445775",
"lon": "78.3731",
"name": "Campus 1"
});
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.pickerview.selectItem

Selects the item in PickerView widget, if it is visible and enabled.

Syntax

```
kony.automation.pickerview.selectItem(<widgetpath>, <newItem>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
newItem [array] [Mandatory]	Array of display values from component 1 to n, where 'n' is the number of components in the PickerView.

Example

```
kony.automation.pickerview.selectItem["frmHomeLogin","pickerView"], ["2009",
"May", "25"]);
```

Return Values

None

Platform Availability

- Android
- iOS

- Windows 10
- SPA and ResponsiveWeb

kony.automation.radiobuttongroup.click

Triggers the RadioButtonGroup click event on the specified widget, if it is visible and enabled.

> ***Note:*** Supported view type is onScreen Wheel.

Syntax

```
kony.automation.radiobuttongroup.click(<widgetpath>, <key>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
key [string] [Mandatory]	Radiobutton key that needs to be clicked.

Example

```
kony.automation.radiobuttongroup.click(["frmHomeLogin","rdBtnId"],
"rdBtnkey");
```

Return Values

None

Platform Availability

- Android
- iOS

- Windows 10
- SPA and ResponsiveWeb

kony.automation.richtext.click

Triggers the RichText click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.richtext.click(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.richtext.click(["frmHomeLogin","richTextId"]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.segmentedui.click

Triggers the SegmentedUI click event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.segmentedui.click(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.segmentedui.click(["frmHomeLogin","segmentedUIId[0,2]"]);
```

Return Values

None

Note: Automation is supported only for the Segment table view.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.segmentedui.pull

Triggers the onPull event on the SegmentedUI, if it is set.

Syntax

```
kony.automation.segmentedui.pull(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.segmentedui.pull(["frmHomeLogin","segmentedUIId"]);
```

Return Values

None

Note: Automation is supported only for the Segment table view.

Platform Availability

- Android
- iOS
- ResponsiveWeb

kony.automation.segmentedui.push

Triggers the onPush event on the SegmentedUI, if it is set.

Syntax

```
kony.automation.segmentedui.push(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.segmentedui.push(["frmHomeLogin"," segmentedUIId "]);
```

Return Values

None

Note: Automation is supported only for the Segment table view.

Platform Availability

- Android
- iOS
- ResponsiveWeb

kony.automation.segmentedui.scrollToBottom

Makes the segment scroll to the last row and then triggers the onReachEnd event of SegmentedUI, if it is set.

Syntax

```
kony.automation.segmentedui.scrollToBottom(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.segmentedui.scrollToBottom(["frmHomeLogin"," segmentedUIId"]);
```

Return Values

None

Note: Automation is supported only for the Segment table view.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.segmentedui.scrollToRow

Triggers the segment to scroll to the row specified by the index of the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.segmentedui.scrollToRow(<widgetpath>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.segmentedui.scrollToRow(["frmHomeLogin"," segmentedUIId[12]
"]);
kony.automation.segmentedui.scrollToRow(["frmHomeLogin"," segmentedUIId[1,3]
"]);
```

Return Values

None

Note: Automation is supported only for the Segment table view.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.segmentedui.scrollToTop

Makes the segment to scroll to the first row.

Syntax

```
kony.automation.segmentedui.scrollToTop(<widgetpath>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.segmentedui.scrollToTop(["frmHomeLogin","btnLogin"]);
```

Return Values

None

Note: Automation is supported only for the Segment table view.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.segmentedui.scrollToWidget

Scrolls to ensure that the widget appears in view. It is an awaitable API.

Note: Automation is supported only for the Segment table view.

Syntax

```
kony.automation.scrollToWidget(<widgetpath>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
await kony.automation.scrollToWidget(["frmHomeLogin","btnLogin"]);
```

Return Values

None

Note: If segment is a part of widgetpath, then it scrolls only to the segment.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.slider.slide

Triggers the Slider slide event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.slider.slide(<widgetpath>, <newValue>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
newValue [number] [Mandatory]	New slider value within a minimum and maximum range.

Example

```
kony.automation.slider.slide(["frmHomeLogin","sliderId"], 25);
```

Return Values

None

Platform Availability

- Android
- iOS

- Windows 10
- SPA and ResponsiveWeb

kony.automation.switch.toggle

Toggles the Switch between ON/OFF on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.switch.toggle(<widgetpath>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.

Example

```
kony.automation.switch.toggle(["frmHomeLogin","switchId"]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.tabpane.click

Clicks the tab with the specified tabID on the TabPane widget, if it is visible and enabled.

Syntax

```
kony.automation.tabpane.click(<widgetpath>, <tabID>);
```

Input Parameter

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
tabID [string] [Mandatory]	The tabID name.

Example

```
kony.automation.tabpane.click(["frmHomeLogin","tabpaneId"], ”tabId”);
```

Note: Automation is supported only for the TabPane default view.

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.textbox.enterText

Enters the specified text into the TextBox, if it is visible and enabled.

Syntax

```
kony.automation.textbox.enterText(<widgetpath>, <newText>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
newText [number] [Mandatory]	New text to be set to the TextBox. Specify null to clear the text.

Example

```
kony.automation.textbox.enterText(["frmHomeLogin","textbox"], “sampleText”);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.textarea.enterText

Enters the specified text into the TextArea, if it is visible and enabled.

Syntax

```
kony.automation.textarea.enterText(<widgetpath>, <newText>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
newText [number] [Mandatory]	New text to be set to the TextBox. Specify null to clear the text.

Example

```
kony.automation.textarea.enterText(["frmHomeLogin","textAreaId"], "sample");
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

9.3.2 Low-level Touch and Gesture APIs

The kony.automation Namespace comprises of the following Low-level Touch and Gesture APIs.

kony.automation.widget.touch

Triggers the touch event on the specified widget, if it is visible and enabled.

Syntax

```
kony.automation.widget.touch(<widgetpath>, <startPoint>, <movePoint>,
<endpoint>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
startPoint [array] [Mandatory]	Represents the start point as [x, y] co-ordinates. Alternatively, this can be assigned a null value.
movePoint [array] [Mandatory]	Represents an array of interim points such as [[x1, y1], [x2, y2]…[xn, yn]]. Alternatively, this can be assigned a null value.
endpoint [array] [Mandatory]	Represents the end point as [x, y] co-ordinates. Alternatively, this can be assigned a null value.

Example

```
kony.automation.widget.touch(["Home Page", "appMenuOption2"], [1, 1], [
    [30, 1],
    [50, 1],
    [60, 1]
]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.widget.scroll

Description 1: Triggers the scroll event on the specified widget, if it is visible and enabled.

Description 2: Triggers the scroll event on the specified widget with the specified direction, if it is visible and enabled.

Syntax

```
kony.automation.widget.scroll(<widgetpath>, <startPoint>, <movePoint>,
<endpoint>);
```

```
kony.automation.widget.scroll (<widgetpath>, <scrollDirection>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
startPoint [array] [Mandatory]	Represents the start point as [x, y] co-ordinates. Alternatively, this can be assigned a null value.
movePoint [array] [Mandatory]	Represents an array of interim points such as [[x1, y1], [x2, y2]...[xn, yn]]. Alternatively, this can be assigned a null value.
endpoint [array] [Mandatory]	Represents the end point as [x, y] co-ordinates. Alternatively, this can be assigned a null value.
scrollDirection [constant] [Mandatory]	The constants that are allowed are as follows: • kony.automation.scrollDirection.Top • kony.automation.scrollDirection.Bottom • kony.automation.scrollDirection.Left • kony.automation.scrollDirection.Right

Examples

```
kony.automation.widget.scroll(["Home Page", "masterScroll"],
kony.automation.scrollDirection.Top);
kony.automation.widget.scroll(["Home Page", "appMenuOption2"], [1, 1], [
    [30, 1],
    [50, 1],
    [60, 1]
]);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.widget.canScroll

Returns whether the scroll functionality is enabled for the specified widget.

Syntax

```
kony.automation.widget.canScroll (<widgetpath>, <scrollDirection>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
scrollDirection [constant] [Mandatory]	The constants that are allowed are as follows: • kony.automation.scrollDirection.Top • kony.automation.scrollDirection.Bottom • kony.automation.scrollDirection.Left • kony.automation.scrollDirection.Right

Example

```
var isScrollingEnabled = kony.automation.widget.canScroll(["Home Page",
"masterScroll"],
    kony.automation.scrollDirection.Top);
if (isScrollingEnabled) {
    kony.automation.widget.scroll(["Home Page", "masterScroll"],
kony.automation.scrollDirection.Top);
}
```

Return Values

true if scrolling is allowed; otherwise, false.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

9.3.3 Miscellaneous Automation APIs

The kony.automation Namespace comprises of the following miscellaneous Automation APIs.

kony.automation.playback.wait

Introduces a delay time in the playback as specified. It is an awaitable API.

Syntax

```
kony.automation.playback.wait(<delayTime>);
```

Input Parameters

Parameter	Description
delayTime [number] [Mandatory]	Time delay in millisecond.

Example

```
await kony.automation.playback.wait(2000);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.playback.waitFor

Waits for the widget to load completely. It is an awaitable API.

Syntax

```
kony.automation.playback.waitFor(<widgetpath> , <timeout in ms>);
```

Input Parameters

Parameters	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
timeout	It is an optional parameter. If the timeout is not specified, the API waits until the widget appears.

Example

```
await kony.automation.playback.waitFor(["Home Page", "amountSpentLabel"]);
```

Return Values

Boolean

Returns true if the widget is found within the timeout period.

Returns false if the wdget is not found within the timeout period.

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.widget.getWidgetProperty

Returns the particular Kony-defined property on the specified widget.

Syntax

```
kony.automation.widget.getWidgetProperty(<widgetpath>, <propertyName>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
propertyName [string] [Mandatory]	Widget property name.

Example

```
var labelText = kony.automation.widget.getWidgetProperty (["Home Page",
"amountSpentLabel"], "text");
```

Return Values

The value of the property specified for the widget

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.widget.getProperty

Returns the particular native property on the specified widget.

Syntax

```
kony.automation.widget.getProperty(<widgetpath>, <propertyName>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array.
propertyName [string] [Mandatory]	Widget property name.

Example

```
var isLabelEnabled = kony.automation.widget.getProperty (["Home Page",
"amountSpentLabel"], "isEnabled");
//Position, Size, isEnabled, isVisible
```

Return Values

The value of the native property specified for the widget

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.device.deviceBack

Invokes the back action of the device. It is an awaitable API.

Syntax

```
kony.automation.device.deviceBack();
```

Input Parameters

None

Example

```
await kony.automation.device.deviceBack();
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10
- SPA and ResponsiveWeb

kony.automation.capture

The api takes a screenshot of the widget passed. If the widget does not pass, the screenshot captures the entire screen.

Syntax

```
kony.automation.capture(<widgetpath>);
```

Input Parameters

Parameter	Description
widgetpath [array of strings] [Mandatory]	Widget ID from root element (form, and master, etc.). Comma-separated strings from root to widget represented in an array. If you do not specify this parameter, the screenshot of the entire current screen is taken.

Examples

```
kony.automation.capture(["frmHomeLogin","btnLogin"]);
kony.automation.capture();
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10

9.3.4 Existing Kony APIs

You can use the following existing Kony APIs for Automation. The respective syntax and usage remain the same.

- kony.print
- kony.os.deviceInfo

10. Background Agent API

After launching an application, and when the application is not in the foreground, none of the application’s code gets executed in the background. To overcome such issues, Kony introduced Background Agent functionality for Windows platform. The Background Agent API enables you to supply the code to be executed periodically by the Operating System (OS) in the background. Although, there is no user interface to the code supplied to the OS, the code keeps sharing information with the main application. The information shared by the code with the application includes isolated storage and application storage. For example, .xap file location.

Kony supports Background Agent of type periodic tasks. The periodic-task agents run for a less amount of time on a regular recurring interval. Typical scenarios for this type of task include uploading the device’s location, and performing small amount of data synchronization.

When a developer builds an application, a .xap file is submitted with all the code that the application requires to launch the user interface. But when a user launches multiple apps, none of the code will be executed for the apps that are not in the foreground.

Background agents enable a developer to supply some code from the background app that is executed periodically by the operating system. This code does not have any user interface but shares information with the active application. The information that is shared includes the isolated storage and application storage (for example, where the .xap file contents are located).

Background agents are of two types: Periodic and Resource intensive agents.

Periodic agents run for a small amount of time on a regular recurring interval. While the resource-intensive agents run for a relatively long period of time when the phone meets a set of requirements relating to processor activity, power source, and network connection. Kony supports periodic tasks only.

The Background agent API uses the `kony.backgroundtasks Namespace` and the following API elements.

Function	Description
`kony.backgroundtasks.startTask`	Registers a scheduled action with the Operating System.
`kony.backgroundtasks.stopTask`	Unregisters the already registered scheduled action with the Operating System.
`kony.backgrondtasks.getTaskDetails`	Returns the task details of already registered background task.

To begin a background task and set the success and failure callbacks, use the `kony.backgroundtasks.startTask` function. A success message appears when the appyou begins a task successfully. If the task fails to begin, an error message appears. After the background task begins, the app can retrieve the details of the task by using the `kony.backgrondtasks.getTaskDetails` function. To terminate a background task, use the `kony.backgroundtasks.stopTask` function.

10.1 kony.backgroundtasks Namespace

The kony.backgroundtasks namespace provides the functions to start and stop the background task, and also to get the task details. It contains the following API elements.

10.1.1 Functions

The kony.backgroundtasks namespace contains the following functions.

kony.backgroundtasks.startTask

Registers a scheduled action with the Operating System.

Syntax

```
kony.backgroundtasks.startTask(tasksettings, onsuccesscallback,
onfailurecallback)
```

Input Parameters

tasksettings [dictionary] - Mandatory

Specifies a task that must be set as a background agent of the application. You can define the task settings using the following arguments:

Windows Phone 8

Parameter	Description
Description	Sets a description for the task to be scheduled. The defined description is displayed to the user in the background task settings page.
ExpirationTime	Sets the time when the scheduled task should be expired. Maximum time is 14 days.
DateFormat	Sets the date format of the expiry time. The default format is dd/mm/yyyy.

Windows 8

Parameter	Description
isOneTime	Sets the task should be scheduled once or not. Set to true if you want the task to be triggered only once. Set to false if you want the task to be triggered each time refreshDuration elapses.
refreshDuration	Specifies the number of minutes to wait to schedule the background task. The system schedules the task within 15 minutes after refreshDuration elapses. The refreshDuration argument is considered when the isOneTime argument is set to false. If the refreshDuration is set to less than 15 minutes, an exception is thrown when attempting to register the background task.

onsuccesscallback [Function] - Mandatory	The platform calls the callback when an agent is set successfully. You can define your function in the callback that you want to be executed when the setting the agent is successful.
	`function successCallback(){`
	`  //code`
	`}`

onfailurecallback [Function] - Mandatory	The platform calls the callback when an agent is failed to set. You can define your function in the callback that you want to be executed when the setting the agent is failure.
	`function unsuccessfulCallback`
	`(){`
	`  //code`
	`}`

Example

```
function setbackgroundtask() {
    try {
        var taskSettings = {};
        kony.backgroundtasks.startTask(taskSettings onsuccesscallback,
            onfailurecallback);
    } catch (err) {
        alert("Exception in startTask: " + err);
    }
}
```

Return Values

None

Exception

1900 - miscellaneous error.

Platform Availability

Available on all Windows channels except Windows 7 desktop/Kiosk.

kony.backgroundtasks.stopTask

Unregisters the already registered scheduled action with the Operating System.

Syntax

```
kony.backgroundtasks.stopTask()
```

Input Parameters

None

Example

```
function removebackgroundtask() {
    try {
        kony.backgroundtasks.stopTask();
        alert("background task is stopped");
    } catch (err) {
        alert("Stopping the background task is failed with error: " + err);
    }
}
```

Return Values

None

Exception

1900 - miscellaneous error.

Platform Availability

Available on all Windows channels except Windows 7 desktop/Kiosk.

kony.backgroundtasks.getTaskDetails

Returns the task details of already registered background task.

Syntax

```
kony.backgroundtasks.getTaskDetails()
```

Input Parameters

None

Example

```
function getbackgroundtask() {
    try {
        var taskdetails = kony.backgroundtasks.getTaskDetails();
        alert(taskdetails);
    } catch (err) {
        alert("Stopping the background task is failed with error: " + err);
    }
}
```

Return Values

taskdetails [dictionary]

Contains the following task details:

Return Value	Description
ExpirationTime	Time at which the task expires.
isRegistered	The scheduled status of the action.
LastExitReason	Reason for the agent exited last time when the action executed.
LastScheduledTime	The time of the last scheduled action. The time is of the device's local time.

> ***Note:*** All the above task details are returned for the Windows Phone 8 channel.
> Only the isRegistered property is returned for the Windows 8 channel.

Exception

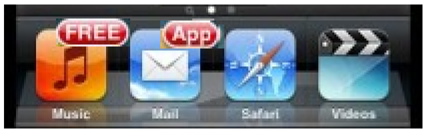
1900 - miscellaneous error.

Platform Availability

Available on all Windows channels except Windows 7 desktop/Kiosk.

11. Badge APIs

Badge is a small icon which displays some indicative information to the users. Badge can be used for an application, a specific widget, app menu, or an app menu icon.



A badge can be used in the following scenarios to:

- display a message on a specific widget
- emphasize the price
- advertise a promotion
- indicate the number of unread messages or notifications.

> ***Important:*** Badge APIs are applicable only on iPhone and iPad.

Badging is a concept used in both iOS and Android platforms to indicate to user, information about an app or an app menu. Badging works as a notification to the users. Based on the information provided, user's can take some action. However, badging does not explicitly ask the users to take any action.

The number of unread messages on your mobile device that is indicated is an example of the badge API.

Using the Badge API in Kony, you can configure a badge for the following:

- Apps
- App Menus

The Badge API uses the `kony.application Namespace` and the following API elements.

Function	Description
kony.application.setApplicationBadgeValue	Sets a badge value to an application icon on the mobile desktop at the top-right corner of the application icon
kony.application.getApplicationBadgevalue	Reads the badge value (if any) attached to the given application icon.
kony.application.setAppMenuBadgeValue	Sets a badge value to the specified app menu item on the top-right corner of the app menu item.

Function	Description
`kony.application.getAppMenuBadgeValue`	Reads the badge value (if any) attached to the specified app menu item.

Depending upon the type of badge you want to set, Badge API functions vary.

To configure the badge value for an App, you can use the `kony.application.setApplicationBadgeValue` function. To know about an existing badge value an app, you can use the `kony.application.getApplicationBadgeValue` function.

Similarly, to configure the badge value for an AppMenu, you can use the `kony.application.setAppMenuBadgeValue` function. To know about an existing badge value an app, you can use the `kony.application.getAppMenuBadgeValue` function.

In addition, each widget supports the following methods for working with badges.

- setBadge
- getBadge

To view the functionality of the Badge API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

11.1 Functions

The Badge API contains the following functions, which are part of the kony.application Namespace.

kony.application.setApplicationBadgeValue

This API allows you to set a badge value to an application icon on the mobile desktop at the top-right corner of the application icon. If you pass an empty string as a parameter, the badge applied on the application icon is removed.

Syntax

```
kony.application.setApplicationBadgeValue(badgeValue, tileID)
```

Input Parameters

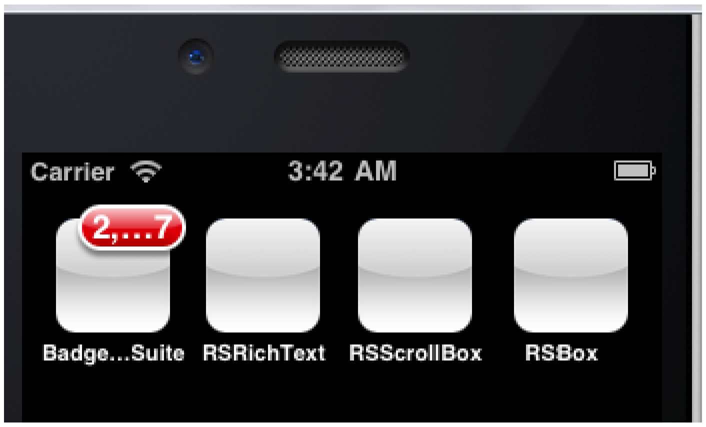
Parameter	Description
badgeValue [String] - Mandatory	Value of the badge. The value that you specify in the badgeValue parameter appears within the badge. Do not pass any other value except a numerical value. For example, to set a badge value for an appicon, specify the value as "2" instead of 2. If the length of the badge value is greater than 1 the badge is a rounded rectangle. For example, if you specify the value of the badge as 88, the number appears in a rounded rectangular badge. If the length of the badge value is 1, the badge is always a circle.
tileID [String] - Optional	The tile ID defined for the secondary tile using the setSecondaryTile API. The parameter is applicable only for Windows.

Example

```
function setApplicationBadgeValue(){
       /*Set the ApplicationBadgeValue to the  application icon on the
       mobile desktop at the top-right corner of the application icon.*/
       kony.application.setApplicationBadgeValue("234567");
}
//to set the badge value "10" to the secondary tile (Applicable only in the
Windows platform)
var tileId="myTileId";
kony.application.setSecondaryTile(tileId, "My App", "My Application",
"tile.png");
kony.application.setApplicationBadgeValue("10", tileId);
```

```
settingBadge: function() {
    this.view.btnBadge.setBadge("0", ""); //Set badge value on  button widget
    kony.application.setApplicationBadgeValue("" + 0);
},
BadgeIncrease: function() {
    var counter = kony.os.toNumber(this.view.btnBadge.getBadge()) + 1; // read
badge value from button and increment it with 1
    kony.print("this gets executed " + counter + "type is " + typeof
(counter));
    this.view.btnBadge.setBadge("" + counter, ""); // Set badge value on the
button widget
    kony.application.setApplicationBadgeValue("" + counter); //Set badge value
on app icon
```

The badge appears as follows when you execute the code given above:



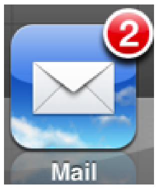
Return Values

None

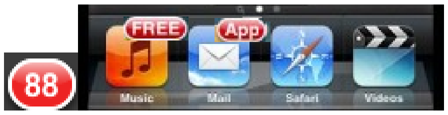
UI Behavior

The badge appears with white font on a red background. The shape of the badge varies with its value:

- If the badge value is a single digit, the badge shape is a circle.



- If the badge value contains multiple digits, the badge shape is a rectangle with rounded corners and borders.



Platform Availability

- iOS
- Windows

kony.application.getApplicationBadgeValue

This API allows you to read the badge value (if any) attached to the given application icon. If the applications icon does not have any badge value attached to it, this API returns an empty string.

Syntax

```
kony.application.getApplicationBadgeValue()
```

Input Parameters

None

Example

```
function getApplicationBadgeValue(){
       /*Get the ApplicationBadgeValue from the  application icon on the mobile
desktop at the top-right corner of the application icon.*/
       kony.application.getApplicationBadgeValue();
}
```

```
gettingBadge: function() {
    var badge = kony.application.getApplicationBadgeValue();
    alert("The badge value is " + badge);
},
```

Return Values

Return Value	Description
badgeValue [String]	Returns the badge value applied to the application icon If the application icon has no badge value attached to it, it returns null/nil.

Platform Availability

Available only on iPhone and iPad.

kony.application.setAppMenuBadgeValue

This API allows you to set a badge value to the specified app menu item on the top-right corner of the app menu item. If you pass an empty string as the parameter, the badge value of the app menu item is cleared.

Syntax

```
kony.application.setAppMenuBadgeValue(appmenuID, menuItemId,badgeValue)
```

Input Parameters

Parameter	Description
appMenuId [String] - Mandatory	If you are setting the badge for an app menu item that was created dynamically, use the same ID that was used to create the app menu item.If you are setting the badge for an app menu item that was created from the IDE, use the ID available in the generated script file.
menuItemId [String] - Mandatory	Id of the app menu item to which the badge value to be set.
badgeValue [String] - Mandatory	Value of the badge. The value you specify in the badge value appears within the badge. If the length of the badge value is greater than 1 the badge is a rounded rectangle. For example, if you specify the value of the badge as 88, the number appears in a rounded rectangular badge. If the length of the badge value is 1, the badge is always a circle. The maximum number of characters that can be specified in a badge value is 9. If the badge value id beyond 9 only the first 9 characters are displayed.

Example

```
function setAppMenuBadgeValue() {
    //Set the AppMenuBadgeValue as "3" for the menu item with
id:"appmenuitemid3"
    kony.application.setAppMenuBadgeValue("accountMenu", "appmenuitemid3",
        "3");
}
```

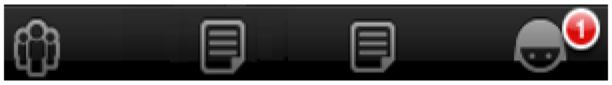
```
createAppMenu: function() {
    var appMenuItem1 = ["appmenuitemid1", "Accounts", "option1.png",
this.onClickMenuItem1];
    var appMenuItem2 = ["appmenuitemid2", "Examination", "option2.png",
this.onClickMenuItem2];
    var appMenu = [appMenuItem1, appMenuItem2];
    kony.application.createAppMenu("SampleAppMenu", appMenu, null, null);
    kony.application.setCurrentAppMenu("SampleAppMenu");
    kony.application.setAppMenuBadgeValue("SampleAppMenu", "appmenuitemid1",
"4");
    kony.application.setAppMenuBadgeValue("SampleAppMenu", "appmenuitemid2",
"6");

},
```

Return Values

None

UI Behavior

The following image depicts how a bade appears on an app menu item:



Platform Availability

Available only on iPhone and iPad.

kony.application.getAppMenuBadgeValue

This API enables you to read the badge value (if any) attached to the specified app menu item. If the specified app menu item does not have any badge value attached to it, the API returns an empty string.

Syntax

```
kony.application.getAppMenuBadgeValue(appmenuID, menuItemID)
```

Input Parameters

Parameter	Description
appMenuId [String] - Mandatory	If you are setting the badge for an app menu item that was created dynamically, use the same ID that was used to create the app menu item.If you are setting the badge for an app menu item that was created from the IDE, use the ID available in the generated script file.
menuItemId [String] - Mandatory	Identifier of the app menu item from which the badge value is to be read.

Example

```
function getAppMenuBadgeValue() {
    //Get the AppMenuBadgeValue for the menu item with id:"appmenuitemid3"
 kony.application.getAppMenuBadgeValue("accountMenu", "appmenuitemid3");
}
```

```
onClickMenuItem1: function() {
    alert("The Badge Value of Accounts App Menu Item is " +
kony.application.getAppMenuBadgeValue("SampleAppMenu", "appmenuitemid1"));

},
onClickMenuItem2: function() {
    alert("The Badge Value of Examination App Menu Item is " +
kony.application.getAppMenuBadgeValue("SampleAppMenu", "appmenuitemid2"));

},
```

Return Values

Return Value	Description
badgeValue [String]	Returns the badge value applied to the specified app menu. If the specified app menu has no badge value attached to it, it returns an empty string.

Platform Availability

Available only on iPhone and iPad.

---

12. Battery API

The Battery API provides a standard interface that can be used across multiple hardware platforms for checking the current state of a device's battery. It enables your app to register/ unregister for the battery monitoring service of the device operating system and check the battery level on the customer's device.

The Battery API uses `kony.os Namespace` and the following API elements.

Function	Description
`kony.os.getBatteryLevel`	Retrieves the current percentage charge level of the device battery, as an integer value.
`kony.os.getBatteryState`	Retrieves the current state of the battery.
`kony.os.registerBatteryService`	Registers for the battery monitoring service of the device operating system.
`kony.os.unregisterBatteryService`	Stops the monitoring process of the device battery.

To register for the battery monitoring service of the device operating system, invoke the `kony.os.registerBatteryService` function. The callback is delivered to the most recent registered battery service. If you want to stop the monitoring process of the device battery, use the `kony.os.unregisterBatteryService` function. You must call this API when the use of the battery monitoring service has been completed, to reduce the overhead.

To check the current battery level, use the `kony.os.getBatteryLevel` function. To see whether the battery is charging, discharging, and so forth, invoke the `kony.os.getBatteryState` function.

To view the functionality of the Battery API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

12.1 Constants

The Battery API, which belongs to the kony.os Namespace, contains the following types of constants.

Battery State Constants

These constants specify the current state of the device battery.

Constant	Description
BATTERY_STATE_CHARGING	Indicates that the state of the device battery as being charged.
BATTERY_STATE_DISCHARGING	Indicates that the state of the device battery as being discharged.

Constant	Description
BATTERY_STATE_FULL	Indicates that the state of the device battery charge is completely full.
BATTERY_STATE_UNKNOWN	Indicates that the state of the device battery charge as not known.

Example

When you query for the state of the device battery as shown in this example, any of the four available battery states is returned.

```
var batteryState = kony.os.getBatteryState();

if (kony.os.BATTERY_STATE_CHARGING == batteryState) {

    kony.print("Battery State: Charging");
}
```

Platform Availability

- iOS
- Android
- Windows

12.2 Functions

The Battery API contains the following functions, which belong to the kony.os Namespace.

kony.os.getBatteryLevel

Retrieves the current percentage charge level of the device battery, as an integer value.

Syntax

```
kony.os.getBatteryLevel()
```

Input Parameters

None

Example

```
kony.os.registerBatteryService();
var batteryLevel = kony.os.getBatteryLevel();
kony.os.unregisterBatteryService();
```

```
getBatteryLevel: function() {
    kony.os.registerBatteryService(this.batterySuccessCallback);
    var battery = kony.os.getBatteryLevel();
    kony.os.unregisterBatteryService();
    this.view.lblDisplay.text = battery + "%";
},
```

Return Values

Returns an integer that ranges from 0-100 (inclusive) that specifies the battery's current charge level in percentage. For example, a return value of 30 specifies that the current charge level of the battery is 30%.

Platform Availability

- iOS
- Android
- Windows

kony.os.getBatteryState

Retrieves the current state of the battery.

Syntax

```
kony.os.getBatteryState()
```

Input Parameters

None

Example

```
var batteryState = kony.os.getBatteryState();
if (kony.os.BATTERY_STATE_CHARGING == batteryState) {
    kony.print(“This is the battery state: charging”);
}
```

```
//This code is used to obtain your device battery state
getBatteryState: function() {
    kony.os.registerBatteryService(this.batterySuccessCallback);
    var batteryState = kony.os.getBatteryState();
    if (kony.os.BATTERY_STATE_CHARGING == batteryState) {
        alert("The Device is charging");
        kony.os.unregisterBatteryService();
    } else if (kony.os.BATTERY_STATE_DISCHARGING == batteryState) {
        alert("The Device is discharging");
        kony.os.unregisterBatteryService();
    } else if (kony.os.BATTERY_STATE_FULL == batteryState) {
        alert("The Device is completely charged");
        kony.os.unregisterBatteryService();
    } else if (kony.os.BATTERY_STATE_UNKNOWN == batteryState) {
        alert("The Device charging state is unkonwn");
        kony.os.unregisterBatteryService();
    }
},
```

Return Values

Returns a constant from the Battery State Constants.

Remarks

The battery state indicates whether it is charging, discharging, and so forth.

Platform Availability

- iOS
- Android
- Windows

kony.os.registerBatteryService

Registers for the battery monitoring service of the device operating system. The callback is delivered to the most recent registered battery service.

> ***Note:*** Whenever the battery state changes or for every 1% change in the battery level, a callback to the registerBatteryService function is triggered.

Syntax

```
kony.os.registerBatteryService(callbackMethod)
```

Input Parameters

Parameter	Description
callbackMethod	A JavaScript function that is automatically invoked when you register to the battery monitoring service of the device OS.

Example

```
kony.os.registerBatteryService(callBackservice);
kony.os.registerBatteryService(mybatterychangecallback);
var batterylevel = kony.os.getBatteryLevel();

function mybatterychangecallback(var batteryInfo) {
    var batterylevel = batteryinfo.batterylevel;

    var batteryState = batteryinfo.batterystate;

    if (batterylevel >= 30) {

        // User-defined logic

    }

    if (batterylevel >= 50) {

        kony.os.unregisterBatteryService();

        //You must unregister the battery monitoring service callback to
reduce the overhead.

    }

}
//Here, callbacks are only delivered to the mybatterychangecallback function.
```

```
//This code is used to register a battery service and deregister the service
based on your battery level
registerBatteryService: function() {
    kony.os.registerBatteryService(this.mybatterychangecallback);
    var batterylevel = kony.os.getBatteryLevel();
},
mybatterychangecallback: function(batteryInfo) {
    var batterylevel = batteryInfo.batterylevel;
    if (batterylevel <= 20) {
        alert("The Battery Level is below 20%, make sure that you charge your
device");
    } else {
        kony.os.unregisterBatteryService();
        alert("We are unregistering the Battery Service as it might cause an
overhead");
    }
},
```

Return Values

None

Limitations

- The callback for the registered battery service is delivered only when the application is running; this is because, you can only receive notifications when the application is in the foreground for the iOS, Windows, and Android platforms.
- The callback to the registered battery service is delivered after every one minute duration for iOS; whereas in case of in Android and Windows, the callback is delivered for every 1% change in the battery charge.

Platform Availability

- iOS
- Android
- Windows

kony.os.unregisterBatteryService

This API stops the monitoring process of the device battery. You must call this API when the use of the battery monitoring service has been completed, to reduce the overhead.

Note: After your app calls the kony.os.unregisterBatteryService API, the callback function registered by the kony.os.registerBatteryService API is no longer invoked.

Syntax

```
kony.os.unregisterBatteryService()
```

Input Parameters

None

Example

```
kony.os.registerBatteryService(callBackservice);
kony.os.registerBatteryService(mybatterychangecallback);
var batterylevel = kony.os.getBatteryLevel();

function mybatterychangecallback(var batteryInfo) {
    var batterylevel = batteryinfo.batterylevel;

    var batteryState = batteryinfo.batterystate;

    if (batterylevel >= 30) {

        // User-defined logic

    }

    if (batterylevel >= 50) {

        kony.os.unregisterBatteryService();

        //You must unregister the battery monitoring service callback to
reduce the overhead.

    }

}
```

```
//This code is used to register a battery service and deregister the service
based on your battery level
registerBatteryService: function() {
    kony.os.registerBatteryService(this.mybatterychangecallback);
    var batterylevel = kony.os.getBatteryLevel();
},

mybatterychangecallback: function(batteryInfo) {
    var batterylevel = batteryInfo.batterylevel;
    if (batterylevel <= 20) {
        alert("The Battery Level is below 20%, make sure that you charge your
device");
    } else {
        kony.os.unregisterBatteryService();
        alert("We are unregistering the Battery Service as it might cause an
overhead");
    }
},
```

Return Values

None

Platform Availability

- iOS
- Android
- Windows

13. Beacon API

The Beacon API helps you work with iBeacons. iBeacons are devices that transmit signals using Bluetooth low-energy wireless technology, and allow you to create and monitor beacons that advertise certain identifying information. Beacon Region is a region defined by a device's proximity to Bluetooth Beacons.

The Beacon API uses the `com.kony.Beacon` and the `com.kony.BeaconManager` Namespaces and the following API elements

13.0.1 com.kony.Beacon Methods

Method	Description
`getAccuracy`	The accuracy of the proximity value, measured in meters from the beacon.
`getMajor`	The most significant value in the beacon. A major value, which is a number that can be used to group related beacons that have the same proximity UUID.
`getMinor`	The least significant value in the beacon. A minor value, which is a number that differentiates beacons with the same proximity UUID and major value.
`getProximity`	The proximity value gives a general sense of the relative distance to the beacon. Use it to quickly identify beacons that are nearer to the user rather than farther away.

Method	Description
getProximityUUIDString	The proximity UUID (string representation) of the beacon.
getrssi	The received signal strength of the beacon, measured in decibels.

13.0.2 com.kony.BeaconManager Methods

Method	Description
authorizationStatus	Helps you know the authorization status of the location services for the application.
getMonitoredRegions	Gets the Beacon Regions that are currently being monitored.
getRangedRegions	Gets the the BeaconRegion objects that are currently being ranged.
isMonitoringAvailableForBeaconRegions	Determine whether monitoring is available for the beacon regions.
isRangingAvailableForBeaconRegions	Determine whether ranging is available for the beacon regions.
requestStateForRegion	Determine the state of the current device relative to the beacon region.

Method	Description
setAuthorizationStatusChangedCallback	Sets the callback function that retrieves the authorization status changes.
setMonitoringStartedForRegionCallback	Sets the monitoring started for region callback.
startMonitoringBeaconRegion	Start monitoring for the specified Beacon Region.
startRangingBeaconsInRegion	Starts ranging beacons in a specified beacon region.
stopMonitoringBeaconsRegion	Stops monitoring a specified beacon region.
stopRangingBeaconsInRegion	Stop ranging beacons in a specified Beacon Region.

13.1 Prerequisites

To use Beacon FFI APIs, you need iOS 7 or later, Bluetooth turned on, and a compatible iOS device:

- Xcode 5.0 or later
- iPhone 4s or later
- iPad (3rd generation) or later
- iPad mini or later
- iPod touch (5th generation) or later

The following classes are available in Beacon FFI:

- com.kony.Beacon
- com.kony.BeaconManager
- com.kony.BeaconRegion
- com.kony.PeripheralManager

13.2 How-to Sections

This overview provides the how-tos that demonstrate the use of the Beacon API in the following topics.

- Determining the Availability of Region Monitoring
- Monitoring Beacon Regions
- Handling Boundary-Crossing Events
- Determining the Proximity of a Beacon Using Ranging
- Turning an iOS Device Into an iBeacon

13.3 Determining the Availability of Region Monitoring

Before monitoring a Beacon Region on a device, the developer should check for the availability of the region monitoring and the authorization status.To determine the availability, follow these steps:

1. Check the availability of the Beacon Region Monitoring.

   The isMonitoringAvailableForBeaconRegions method helps determine whether a device supports Beacon Region Monitoring. If the method returns false, the application cannot use Beacon Region Monitoring. If the method returns true, the developer must check the authorization status of the Beacon Region Monitoring.

```
if (beaconManager.isMonitoringAvailableForBeaconRegions()) {
     kony.print("Monitoring is available");
     // Check for authorization status
}
```

2. Check the Beacon Region Monitoring Authorization Status.

   The authorizationStatus method of BeaconManager object determines whether the application is currently authorized to use iOS location services for monitoring the beacons. If the authorization status is `BeaconManagerAuthorizationStatusAuthorized`, the application will receive boundary-crossing notifications for any region it is monitoring. If the authorization status is any other value, the application does not receive those notifications.

   If the application is not authorized to use Beacon Region Monitoring, it can still register Beacon Regions for later use. If the user grants authorization to the application, monitoring for those regions will begin and will generate subsequent boundary-crossing notifications.

```
if (beaconManager.authorizationStatus() ==
"BeaconManagerAuthorizationStatusAuthorized") {
    kony.print("Authorized to use location services");
}
```

   You can use BeaconManager's `authorizationStatusChanged` callback to detect changes in authorization status to the application.

   Refer to the Apple's Location and Maps Programming Guide for more information.

13.4 Monitoring Beacon Regions

When a Beacon Region is monitored, respective callbacks are fired when the device crosses the boundary of the region. You can define a Beacon Region using the BeaconRegion class with proximityUUID, major, minor and identifier methods. The identifier is required and provides a way to refer to a particular beacon in your code. To register a Beacon Region for monitoring, call the startMonitoringBeaconRegion method of the BeaconManager object.

To monitoring Beacon Regions, follow these steps:

1. Create a Beacon Region object with beacon identifying information.

```
var aBeaconRegion = new com.kony.BeaconRegion(aProximityUUID,
aMajor, aMinor, anIdentifier);
```

2. Create a Beacon Manager object with event callback functions.

```
function monitoringCallback(beaconRegion, state) {}

function rangingCallback(beaconRegion, beacons) {}

function errorCallback(beaconManagerError, errorName,
errorDictionary, beaconRegion) {}
var aBeaconManager = new com.kony.BeaconManager
(monitoringCallback, rangingCallback, errorCallback);
```

3. Start monitoring the beacon region by calling the startMonitoringBeaconRegion method of the BeaconManager object.

```
aBeaconManager.startMonitoringBeaconRegion(aBeaconRegion);
```

13.5 Handling Boundary-Crossing Events

If the device enters or exits a Beacon Region, you will be notified through the `monitoringCallback` of BeaconManager object. A developer can postpone these notifications until the user turns on the device’s display by calling setNotifyEntryStateOnDisplay with true.

To handle boundary-crossing events, follow these steps:

1. Define a monitoring callback.

   A monitoring callback should accept two arguments, BeaconRegion object and Device State, of the Beacon Region. The monitoring callback is called on detection of any boundary-crossing event.

```
function monitoringCallback(beaconRegion, beaconRegionState) {
...
}
```

2. Handle events.

   If the device enters a Beacon Region, `beaconRegionState` will be `BeaconRegionStateInside`.

   If the device exits a Beacon Region, `beaconRegionState` will be `BeaconRegionStateOutside`.

```
function monitoringCallback(beaconRegion, beaconRegionState) {
    if (beaconRegionState == "BeaconRegionStateInside") {
        // Device is inside the beacon region -- start ranging
beacons
    }
}
```

13.6 Determining the Proximity of a Beacon Using Ranging

A Beacon Region can be ranged to determine the proximity of the beacon from the device using the `startRangingBeaconsRegion` method of the BeaconManager object.

You should call the isRangingAvailableForBeaconRegions method of the BeaconManager before attempting to range beacons.

Whenever the beacons come within range or go out of range, the BeaconManager object will notify you through rangingCallback with an array of Beacon objects. The beacon objects are detected in the order of closest to farthest. Use the getProximity property to determine relative distance of the beacon to the device. Determine the beacon identifying information using other properties of the beacon object.

To determine proximity, follow these steps:

1. Determining the availability of beacon ranging.

   The isRangingAvailableForBeaconRegions method determines whether the current device supports beacon region ranging. If the method returns false, the application cannot use beacon region ranging.

2. Define ranging callback for notifications.

   Ranging callback should accept two arguments, a beaconRegion and array beacons, which are in range.

   ```
   function rangingCallback(beaconRegion, beacons) {
       // Determine the proximity of beacons to the device.
   }
   ```

3. Ranging beacons in a region.

   To start ranging beacons in a beacon region, use `startRangingBeaconsInRegion` method to start ranging updates for beacons in that region.

   ```
   var aBeaconRegion = new com.kony.BeaconRegion(aProximityUUID,
   aMajor, aMinor, anIdentifier);
   var aBeaconManager = new com.kony.BeaconManager
   (monitoringCallback, rangingCallback, errorCallback);
   aBeaconManager.startRangingBeaconsInRegion(aBeaconRegion);
   ```

4. Determine proximity and other properties.

In the `rangingCallback`, the developer can determine the relative distance of the beacon from the device.

```
function rangingCallback(beaconRegion, beacons) {
    for (var beacon in beacons) {
        if (beacon.getProximity() == "BeaconProximityImmediate")
{
            // Immediate
        } else if (beacon.getProximity() ==
"BeaconProximityNear") {
            // Near
        } else {
          // beacon.getProximity() == "BeaconProximityFar"
            // Far
        }
    }
  }
```

13.7 Turning an iOS Device Into an iBeacon

Any iOS device that supports sharing data using Bluetooth low energy can be used as an iBeacon. Because the application you write must run in the foreground, iBeacon support on iOS devices is intended for testing purposes and for applications that always run in the foreground, such as point-of-sale apps. For other types of iBeacon implementations, you need to acquire dedicated beacon hardware from third-party manufacturers.

To turn an iOS device into an iBeacon, follow these steps:

1. Create a Beacon Region object.

   To use your iOS device as a beacon, you first generate a 128-bit UUID that will be your Beacon Region’s proximity UUID. Open Terminal(in Mac OS) and type uuidgen on the command line. You receive a unique 128-bit value in an ASCII string that is punctuated by hyphens, as in this example.

   ```
   $ uuidgen


   FBA1FFE5-7CD6-451B-8F1F-22B2AC70AA45
   ```

   Next, create a Beacon Region with the UUID you generated for the beacon’s proximity UUID, defining the major and minor values as needed. Be sure to also use a unique string identifier for the new region. This code shows how to create a new Beacon Region using the example UUID above.

   ```
   var proximityUUID = "FBA1FFE5-7CD6-451B-8F1F-22B2AC70AA45";
   var major = 10;
   var minor = 12;
   var identifier = "KonyBeaconSample";
   var beaconRegion = new com.kony.BeaconRegion(proximityUUID,
   major, minor, identifier);
   ```

2. Advertise the beacon information using the peripheral manager.

   Now that you have created a Beacon Region, you need to advertise your beacon’s proximity UUID (and any major or minor value you specified) using the com.kony.PeripheralManager object.

   ```
   var peripheralManager = new com.kony.PeripheralManager
   (stateUpdatedCallback, advertisingStatusCallback);
   ```

```
peripheralManager.startAdvertisingWithMeasuredPower(beaconRegion,
null);
```

13.8 com.kony.Beacon

You cannot create Beacon objects directly. Beacon objects are created by native platforms only.

The Beacon object provides the following API elements.

13.8.1 Methods

The Beacon class has the following methods.

getAccuracy

The accuracy of the proximity value, measured in meters from the beacon.

Syntax

```
getAccuracy()
```

Input Parameters

None.

Example

```
var beacon1 = new com.kony.Beacon();
var accuracy1 = beacon1.getAccuracy();
```

Return Values

Returns a number that specifies the accuracy of the proximity value.

Remarks

Indicates the one sigma horizontal accuracy in meters. Use this property to differentiate between beacons with the same proximity value. Do not use it to identify a precise location for the beacon. Accuracy values may fluctuate due to RF interference.

A negative value in this property signifies that the actual accuracy could not be determined. For more information, see Apple Documentation.

Platform Availability

Available only on iOS

getMajor

The most significant value in the beacon. A major value, which is a number that can be used to group related beacons that have the same proximity UUID.

Syntax

```
getMajor()
```

Input Parameters

None.

Example

```
var beacon1 = new com.kony.Beacon();
 ...
var major1 = beacon1.getMajor();
```

Return Values

Returns a number containing the most significant value in the beacon.

Platform Availability

Available only on iOS

getMinor

The least significant value in the beacon. A minor value, which is a number that differentiates beacons with the same proximity UUID and major value.

Syntax

```
getMinor()
```

Input Parameters

None.

Example

```
var beacon1 = new com.kony.Beacon();
...
var minor1 = beacon1.getMinor();
```

Return Values

Returns a number that specifies the least significant value in the beacon.

Platform Availability

Available only on iOS

getProximity

The proximity value gives a general sense of the relative distance to the beacon. Use it to quickly identify beacons that are nearer to the user rather than farther away.

Syntax

```
getProximity()
```

Input Parameters

None

Example

```
var beacon1 = new com.kony.Beacon();
 ...
```

```
var proximity1 = beacon1.getProximity();
```

Return Values

Returns a string that can be any of the following values.

- BeaconProximityUnknown - The proximity of the beacon could not be determined.
- BeaconProximityImmediate - The beacon is in the user’s immediate vicinity.
- BeaconProximityNear - The beacon is relatively close to the user.
- BeaconProximityFar - The beacon is far away.

Platform Availability

Available only on iOS

getProximityUUIDString

The proximity UUID (string representation) of the beacon.

Syntax

```
getProximityUUIDString()
```

Input Parameters

None.

Example

```
var beacon1 = new com.kony.Beacon();
 ...
var proximityUUIDString1 = beacon1.getProximityUUIDString();
```

Return Values

String

Returns a string that holds the proximity UUID of the beacon.

Remarks

A proximity UUID (universally unique identifier), which is a 128-bit value that uniquely identifies one or more beacons as a certain type or from a certain organization.

Platform Availability

Available only on iOS

---

getrssi

The received signal strength of the beacon, measured in decibels.

Syntax

```
getrssi()
```

Input Parameters

None.

Example

```
var beacon1 = new com.kony.Beacon();
 ...
var rssi1 = beacon1.getrssi();
```

Return Values

Returns a number containing the signal strength in decibels.

Remarks

The value returned by this method is the average RSSI value of the samples received since the range of the beacon was last reported to your app.

Platform Availability

Available only on iOS

13.9 com.kony.BeaconManager

BeaconManager is for managing iBeacons in iOS. Your app creates a BeaconManager object by calling the com.kony.BeaconManager constructor function.

The BeaconManager object contains the following methods.

13.9.1 Methods

The com.kony.BeaconManager class has the following methods.

authorizationStatus

This method helps you know the authorization status of the location services for the application.

Syntax

```
authorizationStatus();
```

Input Parameters

None.

Example

```
var authorizationStatus1 = aBeaconManager.authorizationStatus();
```

Return Values

Returns one of the following strings.

Constant	Description
BeaconManagerAuthorizationStatusNotDetermined	The user has not made a choice regarding whether this application can use location services.

Constant	Description
BeaconManagerAuthorizationStatusRestricted	This application is not authorized to use location services. The user cannot change this application's status, possibly due to active restrictions such as parental controls being in place
BeaconManagerAuthorizationStatusDenied	The user explicitly denied the use of location services for this application or location services are currently disabled in Settings.
BeaconManagerAuthorizationStatusAuthorized	This application is authorized to use location services.

Remarks

The authorization status of a given application is managed by the system and determined by several factors. Applications must be explicitly authorized to use location services by the user, and location services must currently be enabled for the system. A request for user authorization is displayed automatically when your application first attempts to use location services.

Availability

Available only on iOS.

getMonitoredRegions

This API gets the Beacon Regions that are currently being monitored. You cannot add regions to this property directly. Instead, you must register regions by calling the startMonitoringForRegion method.

Syntax

```
getMonitoredRegions();
```

Input Parameters

None.

Example

```
var monitoredRegions = aBeaconManager.getMonitoredRegions();
```

Return Values

Returns an array containing all of the BeaconRegion objects that are being monitored by the BeaconManager.

Availability

Available only on iOS.

getRangedRegions

Gets the the BeaconRegion objects that are currently being ranged.

Syntax

```
getRangedRegions();
```

Input Parameters

None

Example

```
var rangedRegions1 = aBeaconManager.getRangedRegions();
```

Return Values

Returns an array containing all of the BeaconRegion objects that are being ranged by the BeaconManager.

Availability

Available only on iOS.

isMonitoringAvailableForBeaconRegions

Determine whether monitoring is available for the beacon regions.

Syntax

```
isMonitoringAvailableForBeaconRegions();
```

Input Parameters

None.

Example

```

var isMonitoringAvailableForBeaconRegions1 =
aBeaconManager.isMonitoringAvailableForBeaconRegions();
```

Return Values

Returns `True` if monitoring is available for the beacon regions, or `false` if it is not.

Availability

Available only on iOS.

isRangingAvailableForBeaconRegions

Determine whether ranging is available for the beacon regions.

Syntax

```
isRangingAvailableForBeaconRegions();
```

Input Parameters

None.

Example

```
var isRangingAvailableForBeaconRegions1 =
aBeaconManager.isRangingAvailableForBeaconRegions();
```

Return Values

Returns `True` if ranging is available for the beacon regions, or `false` if it is not.

Availability

Available only on iOS.

requestStateForRegion

Determine the state of the current device relative to the beacon region.

Syntax

```
requestStateForRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	The beacon region whose state is queried.

Example

```
aBeaconManager.requestStateForRegion(beaconRegion);
```

```

```

Return Values

None.

Remarks

This method performs the request asynchronously and delivers the results through the *monitoringCallback* function that your app sets by calling setMonitoringStartedForRegionCallback.

Availability

Available only on iOS.

setAuthorizationStatusChangedCallback

Sets the callback function that retrieves the authorization status changes.

Syntax

```
setAuthorizationStatusChangedCallback(
    statusChangedCallbackFunction);
```

Input Parameters

Parameter	Description
statusChangedCallback	A callback function that retrieves changes in the authorization status. For details, see the **Remarks** section below.

Example

```

aBeaconManager.setAuthorizationStatusChangedCallback
(authorizationStatusChanged);
```

Return Values

None

Remarks

This method sets a callback function that is invoked whenever the authorization status changes. It enables your app to retrieve status change updates asynchronously whenever they occur. The callback must have the following signature.

```
function authorizationStatusChanged(Status);
```

where the callback's *Status* parameter is a string that contains one of the following values.

Constant	Description
BeaconManagerAuthorizationStatusAuthorized	This application is authorized to use location services.
BeaconManagerAuthorizationStatusDenied	The user explicitly denied the use of location services for this application or location services are currently disabled in Settings.
BeaconManagerAuthorizationStatusNotDetermined	The user has not made a choice regarding whether this application can use location services.

Constant	Description
BeaconManagerAuthorizationStatusRestricted	This application is not authorized to use location services. The user cannot change this application’s status, possibly due to active restrictions such as parental controls being in place.

Availability

Available only on iOS.

setMonitoringStartedForRegionCallback

Sets the monitoring started for region callback.

Syntax

```
setMonitoringStartedForRegionCallback(
    regionMonitoringCallback);
```

Input Parameters

Parameter	Description
regionMonitoringCallback	A callback function that is invoked when a monitoring starts in an new beacon region. For details, see **Remarks** below.

Example

```
aBeaconManager.setMonitoringStartedForRegionCallback
(monitoringStartedForRegionCallback);
```

Return Values

None

Remarks

This method sets a callback that informs the app that a new region is being monitored. The callback must have the following signature.

```
function monitoringStartedForRegionCallback(beaconRegion);
```

where the callback's *beaconRegion* parameter is a BeaconRegion object that contains the beacon region in which monitoring has started.

Availability

Available only on iOS.

startMonitoringBeaconRegion

Start monitoring for the specified Beacon Region.

Syntax

```
startMonitoringBeaconRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	A BeaconRegion object that contains the beacon region to monitor.

Example

```

aBeaconManager.startMonitoringBeaconRegion(beaconRegion);
```

Return Values

None

Remarks

Your app must call this method once for each region it needs to monitor. If an existing region with the same identifier is already being monitored by the application, the old region is replaced by the new one. Region events are delivered through the *monitoringCallback* function that your app sets by calling setMonitoringStartedForRegionCallback..

Availability

Available only on iOS.

---

startRangingBeaconsInRegion

Starts ranging beacons in a specified beacon region.

Syntax

```
startRangingBeaconsInRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	A BeaconRegion object to use for ranging.

Example

```

aBeaconManager.startRangingBeaconsInRegion(beaconRegion);
```

Return Values

None.

Remarks

Your app calls this function once the region monitored state is "`BeaconRegionStateInside`".

Availability

Available only on iOS.

stopMonitoringBeaconsRegion

Stops monitoring a specified beacon region.

Syntax

```
stopMonitoringBeaconRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	The BeaconRegion to stop monitoring.

Example

```

aBeaconManager.stopMonitoringBeaconsRegion(beaconRegion);
```

Return Values

None

Availability

Available only on iOS.

stopRangingBeaconsInRegion

Stop ranging beacons in a specified Beacon Region.

Syntax

```
stopRangingBeaconsInRegion();
```

Input Parameters

Parameter	Description
beaconRegion	The BeaconRegion to stop ranging.

Example

```
aBeaconManager.stopRangingBeaconsInRegion(beaconRegion);
```

Return Values

None.

Remarks

If the specified region object is not currently being monitored, this method has no effect. When you call this method, the beacon attributes should be the same object that you registered.

Availability

Available only on iOS.

13.9.2 com.kony.BeaconManager

BeaconManager is for managing iBeacons in iOS. Your app creates a BeaconManager object by calling the com.kony.BeaconManager constructor function.

The BeaconManager object contains the following methods.

13.9.2.1 Methods

The com.kony.BeaconManager class has the following methods.

authorizationStatus

This method helps you know the authorization status of the location services for the application.

Syntax

```
authorizationStatus();
```

Input Parameters

None.

Example

```
var authorizationStatus1 = aBeaconManager.authorizationStatus();
```

Return Values

Returns one of the following strings.

Constant	Description
BeaconManagerAuthorizationStatusNotDetermined	The user has not made a choice regarding whether this application can use location services.
BeaconManagerAuthorizationStatusRestricted	This application is not authorized to use location services. The user cannot change this application’s status, possibly due to active restrictions such as parental controls being in place

Constant	Description
BeaconManagerAuthorizationStatusDenied	The user explicitly denied the use of location services for this application or location services are currently disabled in Settings.
BeaconManagerAuthorizationStatusAuthorized	This application is authorized to use location services.

Remarks

The authorization status of a given application is managed by the system and determined by several factors. Applications must be explicitly authorized to use location services by the user, and location services must currently be enabled for the system. A request for user authorization is displayed automatically when your application first attempts to use location services.

Availability

Available only on iOS.

getMonitoredRegions

This API gets the Beacon Regions that are currently being monitored. You cannot add regions to this property directly. Instead, you must register regions by calling the startMonitoringForRegion method.

Syntax

```
getMonitoredRegions();
```

Input Parameters

None.

Example

```
var monitoredRegions = aBeaconManager.getMonitoredRegions();
```

Return Values

Returns an array containing all of the BeaconRegion objects that are being monitored by the BeaconManager.

Availability

Available only on iOS.

getRangedRegions

Gets the the BeaconRegion objects that are currently being ranged.

Syntax

```
getRangedRegions();
```

Input Parameters

None

Example

```
var rangedRegions1 = aBeaconManager.getRangedRegions();
```

Return Values

Returns an array containing all of the BeaconRegion objects that are being ranged by the BeaconManager.

Availability

Available only on iOS.

isMonitoringAvailableForBeaconRegions

Determine whether monitoring is available for the beacon regions.

Syntax

```
isMonitoringAvailableForBeaconRegions();
```

Input Parameters

None.

Example

```
var isMonitoringAvailableForBeaconRegions1 =
aBeaconManager.isMonitoringAvailableForBeaconRegions();
```

Return Values

Returns `True` if monitoring is available for the beacon regions, or `false` if it is not.

Availability

Available only on iOS.

isRangingAvailableForBeaconRegions

Determine whether ranging is available for the beacon regions.

Syntax

```
isRangingAvailableForBeaconRegions();
```

Input Parameters

None.

Example

```
var isRangingAvailableForBeaconRegions1 =
aBeaconManager.isRangingAvailableForBeaconRegions();
```

Return Values

Returns `True` if ranging is available for the beacon regions, or `false` if it is not.

Availability

Available only on iOS.

requestStateForRegion

Determine the state of the current device relative to the beacon region.

Syntax

```
requestStateForRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	The beacon region whose state is queried.

Example

```

aBeaconManager.requestStateForRegion(beaconRegion);
```

Return Values

None.

Remarks

This method performs the request asynchronously and delivers the results through the *monitoringCallback* function that your app sets by calling setMonitoringStartedForRegionCallback.

Availability

Available only on iOS.

setAuthorizationStatusChangedCallback

Sets the callback function that retrieves the authorization status changes.

Syntax

```
setAuthorizationStatusChangedCallback(
    statusChangedCallbackFunction);
```

Input Parameters

Parameter	Description
statusChangedCallback	A callback function that retrieves changes in the authorization status. For details, see the **Remarks** section below.

Example

```

aBeaconManager.setAuthorizationStatusChangedCallback
(authorizationStatusChanged);
```

Return Values

None

Remarks

This method sets a callback function that is invoked whenever the authorization status changes. It enables your app to retrieve status change updates asynchronously whenever they occur. The callback must have the following signature.

```
function authorizationStatusChanged(Status);
```

where the callback's *Status* parameter is a string that contains one of the following values.

Constant	Description
BeaconManagerAuthorizationStatusAuthorized	This application is authorized to use location services.
BeaconManagerAuthorizationStatusDenied	The user explicitly denied the use of location services for this application or location services are currently disabled in Settings.
BeaconManagerAuthorizationStatusNotDetermined	The user has not made a choice regarding whether this application can use location services.
BeaconManagerAuthorizationStatusRestricted	This application is not authorized to use location services. The user cannot change this application’s status, possibly due to active restrictions such as parental controls being in place.

Availability

Available only on iOS.

setMonitoringStartedForRegionCallback

Sets the monitoring started for region callback.

Syntax

```
setMonitoringStartedForRegionCallback(
    regionMonitoringCallback);
```

Input Parameters

Parameter	Description
regionMonitoringCallback	A callback function that is invoked when a monitoring starts in an new beacon region. For details, see **Remarks** below.

Example

```
aBeaconManager.setMonitoringStartedForRegionCallback
(monitoringStartedForRegionCallback);
```

Return Values

None

Remarks

This method sets a callback that informs the app that a new region is being monitored. The callback must have the following signature.

```
function monitoringStartedForRegionCallback(beaconRegion);
```

where the callback's *beaconRegion* parameter is a BeaconRegion object that contains the beacon region in which monitoring has started.

Availability

Available only on iOS.

startMonitoringBeaconRegion

Start monitoring for the specified Beacon Region.

Syntax

```
startMonitoringBeaconRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	A BeaconRegion object that contains the beacon region to monitor.

Example

```

aBeaconManager.startMonitoringBeaconRegion(beaconRegion);
```

Return Values

None

Remarks

Your app must call this method once for each region it needs to monitor. If an existing region with the same identifier is already being monitored by the application, the old region is replaced by the new one. Region events are delivered through the *monitoringCallback* function that your app sets by calling setMonitoringStartedForRegionCallback..

Availability

Available only on iOS.

startRangingBeaconsInRegion

Starts ranging beacons in a specified beacon region.

Syntax

```
startRangingBeaconsInRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	A BeaconRegion object to use for ranging.

Example

```
aBeaconManager.startRangingBeaconsInRegion(beaconRegion);
```

Return Values

None.

Remarks

Your app calls this function once the region monitored state is "`BeaconRegionStateInside`".

Availability

Available only on iOS.

stopMonitoringBeaconsRegion

Stops monitoring a specified beacon region.

Syntax

```
stopMonitoringBeaconRegion(
    beaconRegion);
```

Input Parameters

Parameter	Description
beaconRegion	The BeaconRegion to stop monitoring.

Example

```
aBeaconManager.stopMonitoringBeaconsRegion(beaconRegion);
```

Return Values

None

Availability

Available only on iOS.

stopRangingBeaconsInRegion

Stop ranging beacons in a specified Beacon Region.

Syntax

```
stopRangingBeaconsInRegion();
```

Input Parameters

Parameter	Description
beaconRegion	The BeaconRegion to stop ranging.

Example

```
aBeaconManager.stopRangingBeaconsInRegion(beaconRegion);
```

Return Values

None.

Remarks

If the specified region object is not currently being monitored, this method has no effect. When you call this method, the beacon attributes should be the same object that you registered.

Availability

Available only on iOS.

13.10 BeaconRegion Object

A `BeaconRegion` object defines the type of region based on the device's proximity to a Bluetooth beacon. A Beacon Region monitors for devices whose identifying information matches the information the user provides. If a device appears in range, the region triggers the delivery of an appropriate notification.

13.10.1 Example

```
var proximityUUID = "FBA1FFE5-7CD6-451B-8F1F-22B2AC70AA45";
var major = 10;
var minor = 12;
var identifier = "KonyBeaconSample";
var beaconRegion = new com.kony.BeaconRegion(proximityUUID, major,
minor, identifier);
```

13.11 com.kony.PeripheralManager

A Peripheral Manager object is used to manage published services within the local peripheral device's Generic Attribute Profile (GATT) database and to advertise these services to central devices. If a service is in the database, it is visible and can be accessed by any connected central. If your application does not have the specified Bluetooth-peripheral background mode, the contents of its services become disabled when it is in the background or in a suspended state. Any remote central trying to access the service's characteristic value or characteristic descriptors receives an error.

Before you call Peripheral Manager methods, the state of the peripheral manager object must be powered ON, as indicated by the *PeripheralManagerStatePoweredOn*. This state indicates that the peripheral device (for example, your iPhone or iPad) supports Bluetooth low energy, and Bluetooth is switched ON and available to use. Peripheral Manager is used for turning iOS device into iBeacon.

The com.kony.PeripheralManager class has the following APIs:

- authorizationStatus( )
- isAdvertising( )
- startAdvertisingWithMeasuredPower( )
- stopAdvertising( )

Here is an example to create a new Peripheral Manager object.

```
var BeaconRegion1= new com.kony.PeripheralManager();
```

Here is an example to create a PeripheralManager with callbacks:

```
var peripheralManager = new com.kony.PeripheralManager
(stateUpdatedCallback,advertisingStatusCallback);
```

Input Parameters

Parameter	Description
stateUpdatedCallback **[Function] - Optional**	To get updated callback state from the Peripheral Manager ◦ *peripheralManagerState* [String] - Can represent one of the following values: ▪ PeripheralManagerStatePoweredOff - Indicates Bluetooth is currently powered off. ▪ PeripheralManagerStatePoweredOn - Indicates Bluetooth is currently powered on. ▪ PeripheralManagerStateResetting - The connection with the system service was momentarily lost. An update will occur. ▪ PeripheralManagerStateUnauthorized - The app is not authorized to use the Bluetooth low energy peripheral or server role. ▪ PeripheralManagerStateUnknown - The current state of the peripheral manager is unknown. An update will occur. ▪ PeripheralManagerStateUnsupported - The platform does not support the Bluetooth low energy peripheral or server role. `function stateUpdatedCallback (peripheralManagerState){}`

Parameter	Description
advertisingStatusCallback **[Function] - Optional**	To get the status callback of advertising from the Peripheral Manager. ◦ *errorName* [String] - Name of the error if error occurrs, null otherwise. ◦ *errorInfo* [Object] - Information about the error if error occurs, null otherwise. `function advertisingStatusCallback (errorName, errorInfo){}`

Return Values

Object - com.kony.PeripheralManager

Platform Availability

Available only on iOS.

13.11.1 authorizationStatus

The authorization status of an application is managed by the system and determined by several factors. Applications must be clearly authorized to share data using Bluetooth services while in the background state. The system automatically displays a request for user authorization when your app first attempts to use Bluetooth services to share data.

Calling this method does not prompt the user for access. Instead, you can use this method to detect restricted access and simply hide any affected UI features from the user.

Syntax

```
<<PeripheralObject>>.authorizationStatus( )
```

Input Parameters

None

Example

```
var authorizationStatus1 = PeripheralManager1.authorizationStatus();
```

Return Values

String

Returns any one of the following values:

Return Value	Description
PeripheralManagerAuthorizationStatusDetermined	The user has made a choice regarding whether this application can share data using Bluetooth services while in the background state.
PeripheralManagerAuthorizationStatusRestricted	This application is not authorized to share data using Bluetooth services while in the background state. The user cannot change this application's status, possibly due to active restrictions such as parental controls.
PeripheralManagerAuthorizationStatusDenied	The user clearly denied this app from sharing data using Bluetooth services while in the background state.
PeripheralManagerAuthorizationStatusAuthorized	This application is authorized to share data using Bluetooth services while in the background state.

Platform Availability

Available only on iOS.

13.11.2 isAdvertising

To determine if the Peripheral Manager is currently advertising data.

Syntax

```
<<PeripheralObject>>.isAdvertising( )
```

Input Parameters

None

Example

```
var isAdvertising1 = PeripheralManager1.isAdvertising();
```

Return Values

Boolean

Platform Availability

Available only on iOS.

13.11.3 startAdvertisingWithMeasuredPower

Starts advertising beacon data with Measured Power.

Syntax

```
<<PeripheralObject>>.startAdvertisingWithMeasuredPower()
```

Input Parameters

Parameter	Description
beaconRegion [Object] - com.kony.BeaconRegion	A Beacon Region whose data is to be advertised.
measuredPower [Number] - Optional	The Received Signal Strength Indicator (RSSI) value (measured in decibels) for the device. This value represents the measured strength of the beacon from one meter away and is used during ranging. Specify null to use the default value for the device.

Example

```
PeripheralManager1.startAdvertisingWithMeasuredPower(beaconRegion,
measuredPower);
```

Return Values

None

Platform Availability

Available only on iOS.

13.11.4 stopAdvertising

Stops advertising Peripheral Manager data.

Syntax

```
<<PeripheralObject>>.stopAdvertising( )
```

Input Parameters

None

Example

```
var stopAdvertising1 = PeripheralManager1.stopAdvertising();
```

Return Values

None

Platform Availability

Available only on iOS.

14. Bookmark and Refresh API

In Kony's current desktop architecture, which is based on SPA, the URL of the page never changes throughout the navigation of the application. Consequently, if a user tries to bookmark a page, no matter where in the application the user is, the same URL is displayed. When the user tries to open the bookmarked page, the desktop application has no way of knowing the actual bookmarked page and as a result, the user is always taken to the home page of the application.

An example of the URL in the desktop web application built on Kony would be:

http://xyz.kony.com/kdw/#frm1

The same experience can be extended to other cases such as refreshing a page in the application, and sharing of a link of a specific page in the application. In these scenarios, as there is no change in the URL, the user is by default taken to the home page of the application.

In the traditional desktop web application development, the developers create URLs with some context information so that when they are used, the URL carries necessary context and the application can change based on the context information and take the user to the appropriate page of the application instead of always redirecting them to the home page. Similarly, Kony provides the following set of APIs which enable the developer to add context to the URL so that when the end user bookmarks it or shares it, the URL carries necessary parameters so that the application is rendered accordingly.

Generally, app users access certain pages of a Desktop Web application more frequently than other pages. As a result, app users would want to bookmark a page to access it easily.

Kony provides the Bookmark and Refresh API for Desktop Web apps, which enables the developer to add some context information to the URL. Using the context added to URL, the app user is taken to the appropriate page of the application instead of always redirecting to the home page. When the app user bookmarks or shares a page, the URL carries necessary parameters to render the application accordingly.

An example of the URL in the Desktop web application built on Kony would be:

http://xyz.kony.com/kdw/#frm1

The Bookmark and Refresh API uses `kony.application Namespace` and the following API elements.

Function	Description
`kony.application.addBMState`	Adds a specified key and value to the parameter list of the URL of the form.
`kony.application.getBMState`	Retrieves the list of parameters attached to a URL using the above add, set APIs.
`kony.application.removeBMState`	Removes a specified key from the parameter list of the URL of the form.
`kony.application.resetBMState`	Resets the state associated with the URL of a form.
`kony.application.setBMState`	Sets the bookmark state to the URL.

Add the specified key value pairs to the application URL by using the `kony.application.addBMState` function. To add a JSON object containing key value pairs, use the `kony.application.setBMState` function. Once the context information is added to the URL, you can store the information in the browser.

You can then retrieve the key value pairs by using the `kony.application.getBMState` function.

If you want to delete the key value pairs from the URL, use the `kony.application.removeBMState` function. To remove the JSON object containing the key value pairs, use the `kony.application.resetBMState` function.

To view the functionality of the Bookmark and Refresh API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

14.1 Functions

The Bookmark and Refresh API contains the following functions, which are part of the kony.application Namespace.

kony.application.addBMState

This API adds a specified key and value to the parameter list of the URL of the form.

Syntax

```
kony.application.addBMState(formID, key, value)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form to be bookmarked.
key [String] - Mandatory	Key string representing the LHS of the parameter.
value [String] - Mandatory	Value string representing the RHS of the key-value combination. The value can not be a nested structure.

Example

```
// To set a Bookmark, enter the following
kony.application.addBMState("form1", "About", "page2")
```

```
addbookmark: function() {


    kony.application.addBMState("Form1", "About", "page2");
    alert("A specified key and value are added to the parameter list of the
URL");


},
```

Return Values

None.

Platform Availability

Supported for SPA and Desktop Web.

kony.application.getBMState

This API retrieves the list of parameters attached to a URL using the above add, set APIs.

Syntax

```
kony.application.getBMState(formID)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form for which the parameters of the URL have to be fetched.

Example

```
// To fetch a list of parameters, enter the following
kony.application.getBMState("form1");
```

```
getbookmark: function() {

    var a = kony.application.getBMState("Form1");
    alert(" The list of parameters attached to the URL are " + JSON.stringify
(a));
},
```

Return Values

A JSON structure representing key-values of various parameters attached to the URL string of the given form.

Platform Availability

Supported for SPA and Desktop Web.

kony.application.removeBMState

This API removes a specified key from the parameter list of the URL of the form.

Syntax

```
kony.application.removeBMState(formID,key)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form for which the parameters of the URL have to be removed.
key [String] - Mandatory	Key string representing the key to be removed.

Example

To remove a bookmark for a URL, enter the following:

```
kony.application.removeBMState ("form1", "About");
```

```
removebookmark: function() {
    kony.application.removeBMState("Form1", "About");
    alert("The About key is removed from the parameter list");
},
```

Return Values

None

Platform Availability

Supported for SPA and Desktop Web.

kony.application.resetBMState

This API resets the state associated with the URL of a form. It removes all the parameters attached to the form URL

Syntax

```
kony.application.resetBMState(formID)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory.	Identifier of the form for which the parameters of the URL have to be removed.

Example

```
//To reset a bookmark, enter the following
kony.application.resetBMStat("form1");
```

```
resetBookmarkState: function() {
    kony.application.resetBMState("Form1");
    alert("The state is removed from the URL");
}
```

Return Values

None

Platform Availability

Supported for SPA and Desktop Web.

kony.application.setBMState

This API sets the bookmark state to the URL. This API accepts the *formID* and a *json* structure of key value pairs which will be added to the URL of the page.

Syntax

```
kony.application.setBMState(formID, State)
```

Input Parameters

Parameter	Description
formID [String] - Mandatory	Identifier of the form to be bookmarked.
state [JSON Object] - Mandatory	A JSON object comprising key value pairs. The key value pairs are user defined. You cannot specify this as a nested structure. i.e the value part can not be another JSON object.

Example

```
//To set a bookmark for a URL, enter the following
var state = {
    Bookmark: "about",
    text: "About"
};
kony.application.setBMState("form1", state);
```

```
setState: function() {

    var state = {
        Bookmark: "about",
        text: "About"
    };
    kony.application.setBMState("Form1", state);
    alert("A new state is set to the URL ");
},
```

Return Values

None

Platform Availability

Supported for SPA and Desktop Web.

15. Caching API

The Caching API provides a standard interface that can be used on iOS platform to manage network cache. It contains the following namespace and related API elements:

- kony.net.cache namespace

15.1 Overview

iOS provides a composite on-disk and in-memory cache allowing an app to reduce its dependence on a network connection and provide faster turnaround for previously cached responses.

iOS provides support for accessing resources using the protocols such as ftp://, http://, https://, file:///, data://. iOS does not support response caching for all the protocol implementations. Currently, only http and https requests are cached.

For default sessions, the default value is the shared URL cache object.

> ***Important:*** The response size is small enough to reasonably fit within the cache. (For example, if you provide a disk cache, the response must be no larger than about 5% of the disk cache size.

For more information, refer Apple Documentation.

The Caching API provides a standard interface that can be used on iOS platform to manage network cache. This interface enables you to store and retrieve network requests and the corresponding responses.

iOS provides a composite on-disk and in-memory cache allowing an app to reduce its dependence on a network connection and provide faster turnaround for previously cached responses. Further, iOS provides support for accessing resources using the protocols such as ftp://, http://, https://, file:///, data://. iOS does not support response caching for all the protocol implementations. Currently, only http and https requests are cached.

For default sessions, the default value is the shared URL cache object.

The Caching API uses `kony.cache Namespace` and the following API elements.

Function	Description
kony.net.cache.setMemoryAndDiskCapacity	Initializes a cache capacity memory and disk with the specified values, which can be invoked as part of the preApp or postApp init of the Kony app.
kony.net.cache.getMemoryCapacity	Returns memory capacity of the cache in bytes.
kony.net.cache.getDiskCapacity	Returns disk capacity of the cache, in bytes.
kony.net.cache.getCurrentDiskUsage	Returns current size of the on-disk cache in bytes.
kony.net.cache.getCurrentMemoryUsage	Returns current size of the in-memory cache, in bytes.
kony.net.cache.setCacheConfig	Initializes the cacheConfig is a dictionary which configures the cachePolicy and storagePolicy of the cache responses for the request at the app level.

Configure the memory capacity and the disk capacity of the cache by using the `kony.net.cache.setMemoryAndDiskCapacity` function. To set the cache policy and storage policy of the cache responses by using the `kony.net.cache.setMemoryAndDiskCapacity` function. The Cache policy deals with the URL load requests and the storage policy specifies the storage location of the response.

Retrieve the memory capacity of the cache using `kony.net.cache.getMemoryCapacity` and the disk capacity of the cache using the `kony.net.cache.getDiskCapacity`. To find the current size of the cache on disk, use the `kony.net.cache.getCurrentDiskUsage`, and to find the current size of the cache in memory, use the `kony.net.cache.getCurrentMemoryUsage`function.

To view the functionality of the Caching API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

15.2 kony.net.cache Namespace

The kony.net.cache Namespace contains the following API elements.

15.2.1 Functions

The kony.net.cache namespace implements the caching of responses by mapping HttpReq objects to HttpRes objects.

> ***Important:*** The response size is small enough to reasonably fit within the cache. (For example, if you provide a disk cache, the response must be no larger than about 5% of the disk cache size.

The kony.net.cache namespace contains the following functions.

kony.net.cache.setMemoryAndDiskCapacity

This API initializes a cache capacity memory and disk with the specified values, which can be invoked as part of the preApp or postApp init of the Kony app.

Syntax

```
kony.net.cache.setMemoryAndDiskCapacity(
    memoryCapacity, diskCapacity);
```

Input Parameters

Parameter	Description
memoryCapacity(JSNumber)	The memory capacity of the cache, in bytes.
diskCapacity(JSNumber)	The disk capacity of the cache, in bytes

Example

```
kony.net.cache.setMemoryAndDiskCapacity(1024*1024*5, 1024*1024*5)
Allocating the memory & disk capacity is 5 MB.
```

```
setMemoryandDiskCapacity: function() {
    var memory = this.view.tbxMemory.text;
    var disk = this.view.tbxDisk.text;
    kony.net.cache.setMemoryAndDiskCapacity(memory, disk);
    alert("The Memory and Disk Capacity is set");
},
```

Return Values

None.

Platform Availability

iOS

kony.net.cache.getMemoryCapacity

This API returns memory capacity of the cache in bytes.

Syntax

```
kony.net.cache.getMemoryCapacity();
```

Input Parameters

None

Example

```
var memoryCapacity = kony.net.cache.getMemoryCapacity();
kony.print("The memory capacity of the device is:"+memoryCapacity);
```

```
getMemoryCapacity: function() {
    var memoryCapacity = kony.net.cache.getMemoryCapacity();
    alert("The memory capacity of the device is: " + memoryCapacity + "B");
},
```

Return Values

Returns memory capacity in bytes of JSNumber.

Platform Availability

iOS

---

kony.net.cache.getDiskCapacity

This API returns disk capacity of the cache, in bytes.

Syntax

```
kony.net.cache.getDiskCapacity();
```

Input Parameters

None

Example

```
var diskCapacity = kony.net.cache.getDiskCapacity();
kony.print("The disk capacity of the device is:" + diskcapacity);
```

```
getDiskCapacity: function() {
    var diskCapacity = kony.net.cache.getDiskCapacity();
    alert("The disk capacity of the device is: " + diskCapacity + "B");
},
```

Return Values

Returns disk capacity in bytes of JSNumber.

Platform Availability

iOS

---

kony.net.cache.getCurrentDiskUsage

This API returns current size of the on-disk cache in bytes.

Syntax

```
kony.net.cache.getCurrentDiskUsage();
```

Input Parameters

None

Example

```
var diskUsage = kony.net.cache.getCurrentDiskUsage();
kony.print("The current disk usage is:" + diskUsage);
```

```
currentDiskUsage: function() {
    var diskUsage = kony.net.cache.getCurrentDiskUsage();
    alert("The current disk usage is: " + diskUsage);
},
```

Return Values

Returns current on-disk capacity in bytes of JSNumber.

Platform Availability

iOS

kony.net.cache.getCurrentMemoryUsage

This API returns current size of the in-memory cache, in bytes

Syntax

```
kony.net.cache.getCurrentMemoryUsage();
```

Input Parameters

None

Example

```
var memUsage = kony.net.cache.getCurrentMemoryUsage();
kony.print("The current memory usage is:" + memUsage);
```

```
currentMemoryUsage: function() {
    var memUsage = kony.net.cache.getCurrentMemoryUsage();
    alert("The current memory usage is: " + memUsage);
},
```

Return Values

Returns current in-memory capacity in bytes of JSNumber.

Platform Availability

iOS

kony.net.cache.setCacheConfig

This API initializes the cacheConfig dictionary which configures the cachePolicy and storagePolicy of the cache responses for the request at the app level.

Syntax

```
setCacheConfig(cacheConfig JSDictionary);
```

Input Parameters

Parameter	Description
cacheConfig (JSDictionary)	The cacheConfig is a dictionary which configures the **cachePolicy** and **storagePolicy** of the cache responses. **cacheConfig Constants** The cache config has the following constantsfor **cachePolicy**: • **kony.net.cache.USE_PROTOCOL_CACHE_POLICY**: Specifies that the caching logic defined in the protocol implementation, if any, is used for a particular URL load request. This is the default policy for URL load requests. • **kony.net.cache.RELOAD_IGNORING_LOCAL_CACHE_DATA**: Specifies that the data for the URL should be loaded from the originating source. No existing cache data should be used to satisfy a URL load request. • **kony.net.cache.RETURN_CACHE_DATA_ELSE_LOAD**: Specifies that the existing cached data should be used to satisfy the request, regardless of its age or expiration date. If there is no existing data in the cache corresponding the request, the data is loaded from the originating source. • **kony.net.cache.RETURN_CACHE_DATA_DONT_LOAD**: Specifies that the existing cached data should be used to satisfy the request, regardless of its age or expiration date. If there is no existing data in the cache corresponding the request, the data is not loaded from the originating source. The cache config has the following constantsfor **storagePolicy**: • **kony.net.cache.DISK_AND_MEMORY**: The response stored in disk and memory. • **kony.net.cache.MEMORY_ONLY**: The response stored in memory only. • **kony.net.cache.NOT_ALLOWED**: The response stored neither in the memory nor on the disk.

Example

```
 kony.net.cache.setCacheConfig(cacheConfig: {
cachePolicy: kony.net.cache.USE_PROTOCOL_CACHE_POLICY,
cacheStoragePolicy: kony.net.cache.DISK_AND_MEMORY
});
```

```
setCacheConfig: function() {
    var cacheConfig = {
        cachePolicy: kony.net.cache.USE_PROTOCOL_CACHE_POLICY,
        cacheStoragePolicy: kony.net.cache.DISK_AND_MEMORY
    };
    kony.net.cache.setCacheConfig(cacheConfig);
    alert("The Cache Config is set");
}
```

Return Values

None.

Remarks

This app level setting will be overridden by the per request level setting under widget and api level.

Platform Availability

iOS

16. Camera API

When you capture an image using the Camera widget, the device's camera captures the rawbytes of the image. The Camera API provides a function to delete rawbytes for the image.

The Camera API uses `kony.camera Namespace` and the following API elements.

Function	Description
`kony.camera.releaseRawBytes`	Enables your app to delete rawbytes for the image.

16.1 kony.camera Namespace

The kony.camera namespace contains the following API elements.

16.1.1 Functions

The kony.camera namespace contains the following functions.

kony.camera.releaseRawBytes

This function enables your app to delete rawbytes for the image.

Syntax

```
kony.camera.releaseRawBytes(
    rawBytes)
```

Input Parameters

Parameter	Description
rawBytes	A JavaScript objects that specifies the rawbytes of the image (captured from the camera) you want to release. This can be a kony.types.RawBytes object or an Image object.

Example

```
//To delete the rawbytes captured by the camera widget cam1 in form Frm1:
var imagetodel = frm1.cam1.rawBytes

// One way to release the rawbytes.
kony.camera.releaseRawBytes(imagetodel);

// Another way to release the rawbytes.
kony.camera.releaseRawBytes(frm1.cam1.rawBytes)

// Check the handle to safely release an image's rawbytes.
if (null != RawBytesHandle) {
    kony.camera.releaseRawBytes(RawBytesHandle)
    RawBytesHandle = null;
} else
//RawBytesHandle already released (or rawbytes not assigned to it)

// Safely reuse an existing handle
if (null == RawBytesHandle)
    RawBytesHandle = cameraWidget.rawBytes
else
// RawBytesHandle not empty, release it and then use.
```

```
var cameraBytes = this.view.camera3.rawBytes;
kony.camera.releaseRawBytes(cameraBytes);
if (this.view.camera3.rawBytes == null) {
    alert("The rawbytes are released");
} else {
    alert("The raw bytes could not be released");
}
```

Return Values

None.

Remarks

Apps typically call this function to ensure that the data from captured images does not overwhelm the device's memory capacity, causing the app to slow down or crash.

This function deletes the captured image from the disk or on-device memory. The image data cannot be used or released again by your app. If multiple handles to the same block of rawbytes exist and you release the rawbytes using any one of those handles, it is released for all of the handles. Therefore, none of the handles are usable by your app. If your app attempts to release rawbytes that have already been released, it will throw an exception.

Before your app releases an image's rawbytes, it should check to ensure that the handle associated with the block of rawbytes is not null. After your app releases an image's rawbytes, it should assign the value of null to the handle.

If you assign a handle to new rawbytes without releasing the rawbytes that the handle is currently pointing to, the access to the previous rawbytes is lost and will be released only at the time of application termination. To avoid such a scenario, your app should :

- Initialize the rawbytes handle to null before use.
- Before assigning the rawbytes, ensure that the rawbytes handle is null.

Platform Availability

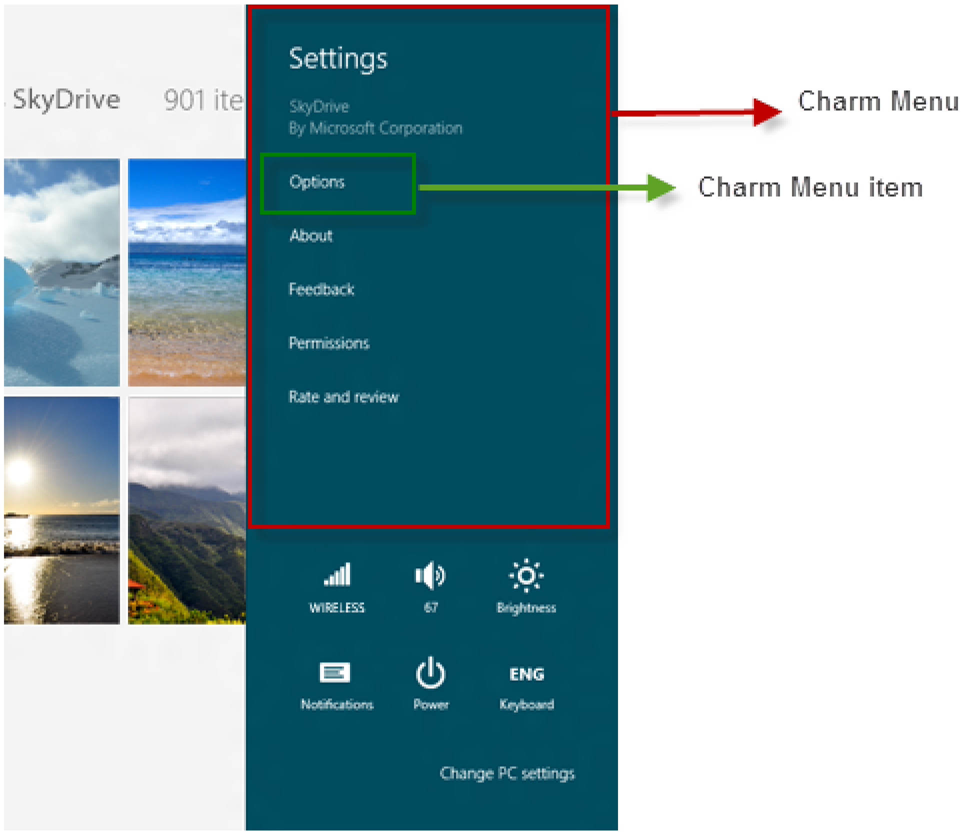
- Android
- iOS
- Windows 10
- DesktopWeb/ResponsiveWeb

17. Charm Setting API

The Charms is feature on Windows 8 which provides access to various application settings such as About, Feedback, and Permissions. The settings can be configured to control an application's access to system capabilities and some specific settings of the current application. To view the Charm settings pane, swipe from the right side of the screen from right to left. For more information about Charm settings, refer Charm Settings.

The Charms is a feature on Windows 8 which provides access to various application settings such as About, Feedback, and Permissions. The settings can be configured to control an application's access to system capabilities and some specific settings of the current application. To view the Charm settings pane, swipe from the right side of the screen from right to left. For more information about Charm settings, refer Charm Settings.

The Charm Setting API enables you to create a Charm menu, add and remove the Charm menu items.



Important: These functions are supported on Windows 8.0 and Windows 8.1 only.

The Charm Setting API uses `kony.application Namespace` and the following API elements.

Function	Description
`kony.application.createSettingsMenu`	Enables you to create a Charm settings menu for an application.
`kony.application.setCurrentSettingsMenu`	Uses the unique identifier which represents the Charm settings menu and sets it as current settings menu.
`kony.application.getCurrentSettingsMenu`	Returns the unique identifier of the current menu that is set through getCurrentSettingsMenu.
`kony.application.addSettingsMenuItemAt`	Enables you to add a menu item at a given index in the Charm settings menu.
`kony.application.removeSettingsMenuItemAt`	Enables you to removes the specified App Menu item based on the index.

To create a Charm settings menu for an application, use the `kony.application.createSettingsMenu` function. You can get the unique identifier of the current menu by using the `kony.application.getCurrentSettingsMenu` function. With the unique identifier of the Charm menu, you can set the menu as the current settings menu using the `kony.application.setCurrentSettingsMenu` function.

With this information, you can add a menu item at a given index in the Charm settings menu using the `kony.application.addSettingsMenuItemAt` function. If you want to delete a menu item from a given index, use the `kony.application.removeSettingsMenuItemAt` function.

17.1 Functions

The Charms API contains the following functions, which are part of the kony.application Namespace.

kony.application.createSettingsMenu

This API enables you to create a *Charm settings* menu for an application.

Syntax

```
kony.applicationcreateSettingsMenu (id, menuSettings)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Identifier of the Charm setting menu created.
menuSettings [Hash table] - Mandatory	The menuSettings hash table comprises the following key-value pairs: • **id**: ID of the Charm menu item. • **text**: Name of the menu item.

Example

```
//To create a Charm settings menu, enter the following
var settingsMenuItem1 = {
    id: "about",
    text: "About"
};
var settingsMenuItem2 = {
    id: "help",
    text: "Help"
};
var settingsMenu = [settingsMenuItem1, settingsMenuItem2];
kony.application.createSettingsMenu("mysettingsmenu", settingsMenu);
```

Return Values

None.

Special Considerations

If a Charm setting menu is already created with the identifier passed, a new Charm setting menu will be created and the old Charm setting menu will be replaced with the new one. The same holds true for menu items as well.

At least one menu item must be present in the Charm settings menu created. A Charm settings menu with no menu items is invalid.

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.setCurrentSettingsMenu

This method uses the unique identifier which represents the Charm settings menu and sets it as current settings menu. There can be only one current settings menu that can be set any time. Calling this method multiple times, replaces the current Charm settings menu.

Syntax

```
kony.application.setCurrentSettingsMenu(id)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Identifier of the Charm setting menu to be set.

Example

```
//To create a Charm settings menu, enter the following
kony.application.setCurrentSettingsMenu("myMenu");
```

Return Values

None

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.getCurrentSettingsMenu

This method returns the unique identifier of the current menu that is set through getCurrentSettingsMenu.

Syntax

```
kony.application.getCurrentSettingsMenu()
```

Input Parameters

None

Example

```
//To get the unique identifier a Charm settings menu, enter the following
kony.application.getCurrentSettingsMenu();

//Alert the Current Charm Settings menu
alert("Current charm menu id is: " + currCharmMenuId);
```

Return Values

Return value	Description
Unique Identifier	Identifier of the Charm setting menu to be set.

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.addSettingsMenuItemAt

This API enables you to add a menu item at a given index in the Charm settings menu.

Syntax

```
kony.application.addSettingsMenuItemAt (id, index, menuSettings)
```

Input Parameters

Parameter	Description
id [String] - Mandatory.	Identifier of the Charm setting menu created.
index [Number] - Mandatory.	The index at which the menu item must be added. The index value lies between 0 and n-1. If the index is beyond the current length of the Charm menu items then the item is added to the end.
menuSettings [Hash table] - Mandatory	The menuSettings hash table comprises the following key-value pairs: • **id**: ID of the Charm menu item. • **text**: Name of the menu item. • **onClick**: onclick event to be executed for the menu item.

Example

To add a menu item at a given index, enter the following:

```
//The below function is the callback function for onClickClosure event of app
menu item with id "appmenuitemid3".
function onClickClosure3() {
    //proceed with the logic
}

var settingsMenuItem1 = {
    id: "about",
    text: "About",
    onClick: onClickClosure3
};

//Adding the above app menu item at the index 3.
kony.application.addSettingsMenuItemAt("accountMenu", 3, settingsMenuItem1);
```

Return Values

None.

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

kony.application.removeSettingsMenuItemAt

This API enables you to removes the specified App Menu item based on the index.

Syntax

```
kony.application.removeSettingsMenuItemAt (id, index)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Identifier of the Charm setting menu created.
index [Number] - Mandatory	The index from which the menu item must be removed. The index value lies between 0 and n-1.

Example

To remove a menu item from a given index, enter the following:

```
//Removing a menu item from the index 3.
kony.application.removeSettingsMenuItemAt("charmmenu", 3);
```

Return Values

None

Platform Availability

Available on Windows 8.0 and Windows 8.1 only.

18. Client Authentication

The Client Authentication API provides your apps with the ability to authenticate clients that want to access an HTTPS servers. Client authentication is required in cases where HTTPS servers (optionally) request the client to authenticate itself using a Digital Certificate before establishing SSL connection.

To authentication client itself to HTTPS server, you need to load the required client certificate into the application. Once the certificate is loaded, the client can perform all HTTPS communications using the service invoker and HTTP request APIs till the application is alive.

The loaded client certificate is sent to a server for authentication in the SSL Handshake phase only when the server challenges the client to authenticate itself. If the server does not challenge the client to authenticate, the certificate is not shared with server.

The client certificate must be of PKCS12 certificate type (encoding format), which includes both private key and certificate of the client. The PKCS12 encoded client certificates must be loaded in the following data-type formats which can be obtained from various sources such as File System, HTTP Connection and so on .

- RawBytes [Userdata]
- base64String [String]

Optionally, you can also pre bundle the required client certificates within an application. In the Android platform, if a certificate is pre bundled in the `Assets` or `res\raw` folders, you may need to write an FFI to copy the certificate from the `Assets` or `res\raw` folders to application file system such as cache directory (`kony.io.FileSystem.getCacheDirectoryPath()`). In the iOS platform, you can place the certificate as an image file. In both the platforms, use the kony.io APIs to read the certificate in RawBytes format and load the certificate using the loadClientCertificate API.

The same certificate may not work to authenticate in multiple servers. In such cases, you can overwrite previously set certificate with new one before establishing communication with another server by calling loadClientCertificate API with new certificate.

In the Android platform, a single .pkcs12 file can contain multiple client certificates and helps to pick the appropriate client certificate automatically for a server during the SSL handshake to enable the communication.

19. Communication API for Native Library

Communication APIs for Native Library allow you to communicate from native apps to the Kony library. You can use these APIs to launch the Kony library from a Native app. This feature is available from V8 SP3 onwards.

Follow these steps to launch the Kony library from a Native app:

1. Initialize the library from the Native app by using Initialize API of Kony library class. This method loads the required resources for the Kony library in the background.
2. After successful initialization, use the Start API to launch the Kony library. To launch the Kony library without using a UI, use the invokeInHeadlessMode API.

The following APIs of `kony.application Namespace` help you in communicating from the Kony library to the Native app:

Function	Description
`kony.application.sendLibraryResultToNativeApp`	Enables you to send an acknowledgment to a Native app that launched this library.

Function	Description
`kony.application.setLibraryHeadlessModeCallback`	Registers a listener or a callback that receives request from a Native app to launch Kony library without UI or in headless mode.
`kony.application.exitLibrary`	Provides you the ability to exit the library.

After the Kony library is launched from a Native app, send an acknowledgement to the native app by using the `kony.application.sendLibraryResultToNativeApp` function. If you want to exit the Kony library, use the `kony.application.exitLibrary` function. After exiting the library, the control moves to the native app UI.

19.1 API to Initialize Library

Use the API for respective platforms to initialize the Kony library from the Native app. This feature is available from V8 SP3 onwards. The Initialize Library API is available for Android, iOS, and Windows platforms.

19.1.1 Android

Package

com.kony.library

Class

KonyLibrary

Syntax

```
public static void initialize(String libraryId,
     android.content.Context  applicationContext,
     KonyLibrary.OnLibraryInitializationListener
initializationListener)throws Exception
```

Input Parameters

Parameter	Description
libraryId	Specify the Library ID (Kony application AppID) that should be initialized. This parameter cannot be null.
applicationContext	Specify the application context of the app. This parameter cannot be null.

Parameter	Description
initializationListener	Register a callback to get the initialization status of the Kony library. This parameter cannot be null.

Note: The listener/callback methods are called from the main/UI thread.

Example

```
try {
    KonyLibrary.initialize( < libraryId > , < applicationContext > , <
listener > );
} catch (Exception ex)
```

19.1.1.1 OnLibraryInitializationListener Interface

Package

com.kony.library

Class

KonyLibrary

Syntax

```
public static interface OnLibraryInitializationListener
```

Methods

- **onLibraryInitSuccess**

This callback method is invoked when the initialization of the Kony library is successful.

```
abstract void onLibraryInitSuccess(String libraryId)
```

Parameter

Parameter	Description
libraryId	Specifies the ID of the Kony library that is successfully initialized.

- **onLibraryInitFailed**

This callback method is invoked when the initialization of the Kony library is not successful.

```
abstract void onLibraryInitFailed(String libraryId, Throwable
exception)
```

Parameters

Parameter	Description
libraryId	Specifies the ID of the Kony library that was not successfully initialized.
exception	Object that helps to find the cause of the library initialization failure.

19.1.2 iOS

Import

<CoreComponent/KonyLibrary.h>

Class

KonyLibrary

Method

+ initialize: delegate:

Declaration

```
+(void) initialize: (NSString * ) libraryId
delegate: (id < KonyLibraryInitializationDelegate > )
initializationDelegate
```

Parameters

Parameter	Description
libraryId	Specify the Kony application AppID that should be initialized.
initializationDelegate	Register a callback object to get the initialization status of the Kony library.

Note: The initializationDelegate methods are called from the main/UI thread.

Example

```
[KonyLibrary initialize: < libraryId > delegate: <
KonyLibraryInitializationDelegate >]
```

19.1.2.1 Protocol : KonyLibraryInitializationDelegate

Import

<CoreComponent/KonyLibrary.h>

Class

KonyLibrary

Methods

- **onLibraryInitSuccess:(NSString*)libraryId**

This callback method is invoked when the initialization of the library is successful.

Syntax

```
(@required): (void) onLibraryInitSuccess: (NSString * )libraryId
```

Parameter

Parameter	Description
libraryId	Specifies the ID of the Kony library that is successfully initialized.

- **onLibraryInitFailed:(NSString*)libraryId exception:(NSException*)**

This callback method is invoked if the initialization of the Kony library is not successful due to any error (s) in the library.

Syntax

```
(@required): (void )onLibraryInitFailed: (NSString * ) libraryId
exception: (NSException * )
```

Parameters

Parameter	Description
libraryId	Specifies the ID of the Kony library that is successfully initialized.
exception	Object that helps to find the cause of the library init failure.

19.1.3 Windows

Package

Framework

Class

KonyLibrary

Syntax

```
public static void Initialize(string libraryId,
Application app, ILibraryInitializationListener listener)
```

Parameters

Parameter	Description
libraryId	Specify the Kony application AppID that should be initialized. This parameter cannot be null.

Parameter	Description
app	Specify the application context of the app. This parameter cannot be null.
listener	Register a callback to get the library initialization status. This parameter cannot be null.

Note: The listener methods are called from the main/UI thread.

19.1.3.1 ILibraryInitializationListener Interface

Package

Framework

Syntax

```
public interface ILibraryInitializationListener
```

Methods

- **OnLibraryInitSuccess**

This callback method is invoked when the initialization of the library is successful.

```
void OnLibraryInitSuccess(string libraryId)
```

Parameter

Parameter	Description
libraryId	Specifies the ID of the library that is successfully initialized.

- **OnLibraryInitFailed(String libraryId)**

This callback method is invoked if the initialization of the Kony library is not successful due to error(s) in the library..

```
void OnLibraryInitFailed(String libraryId, Exception exception)
```

Parameter

Parameter	Description
libraryId	Specifies the ID of the library that is not successfully initialized.
exception	Object that helps to find the cause of the library initialization failure.

19.2 Invoke Library Without UI

Use the **invokeInHeadlessMode** API to open the Kony library without UI from your Native app. This API is applicable for Android and iOS platforms.

The feature to open the Kony library without UI from a Native app is available from Kony Visualizer V8 SP3 onwards.

> ***Note:*** To avoid build failure while generating a native library in Kony Visualizer, ensure that you have at least one form in the project. However, by using invokeInHeadlessMode API, you can launch the native library without UI.

19.2.1 Android

19.2.1.1 invokeInHeadlessMode Method

The **invokeInHeadlessMode** API is used to launch the Kony library without UI. You must call this API only after the library has been successfully initialized by using the initialize API.

The details of implementing the invokeInHeadlessMode method in Android are as follows:

Package

com.kony.library

Class

KonyLibrary

Syntax

```
public static void invokeInHeadlessMode(String libraryId,

       Hashtable<String,Object> libraryArgs,
       KonyLibrary.OnLibraryResultListener resultListener)throws
Exception
```

Input Parameters

Parameter	Data Type	Description
libraryId	String	This parameter specifies the name of the library (Kony application AppID) that is to be opened. This parameter must not be null.
libraryArgs	Hashtable consisting of <String,Object>	This parameter (key-value pair with supported values as primitive data types such as int, float, double, Boolean, and String and collection data types such as ArrayList and Hashtable) is used to pass the native app's data to the library (depending on the contract of the library).
resultListener	KonyLibrary.onLibraryResultListener. For more information about the onLibraryResultListener interface, click here.	This parameter registers a callback to send an acknowledgment (result) from Kony library to the Native app. The listener/callback methods must be called from the main/UI thread, and this parameter cannot be null.

Note: The *libraryArgs* values are passed to the callback that is registered by using the *kony.application.setLibraryHeadlessModeCallback* API.

Example

```
 try {
KonyLibrary.invokeInHeadLessMode( < libraryId > , < libraryArgs > , <
resultListener > );
} catch (Exception ex) {
}
```

19.2.2 iOS

19.2.2.1 invokeInHeadlessMode Method

The invokeInHeadlessMode API is used to launch the Kony library without UI. You must call this API only after the library has been successfully initialized by using the initialize API.

The details of implementing the invokeInHeadlessMode method in iOS are as follows:

Import

<CoreComponent/KonyLibrary.h>

Class

KonyLibrary

Method

+ invokeInHeadlessMode : libraryArgs : delegate:

Declaration

```
+(void) invokeInHeadlessMode : (NSString * )libraryId
libraryArgs: (NSDictionary*) libraryArgs
```

```
delegate: (id < KonyLibraryResultDelegate > ) resultDelegate
```

Input Parameters

Parameter	Data Type	Description
libraryId	String	This parameter specifies the name of the library (Kony application AppID) that is to be launched. This parameter must not be null.
libraryArgs	NSDictionary consisting of <NSString,NSObject>	This parameter (key-value pair with supported values as primitive data types such as int, float, double, Boolean, and NSString and collection data types such as NSArray and NSDictionary) is used to pass the native app's data to the library (depending on the contract of the library).
resultDelegate	KonyLibraryResultDelegate For more information about the KonyLibraryResultDelegate delegate, click here.	This parameter registers a callback to send an acknowledgment (result) from the Kony library to the Native app. You must call the resultDelegate method from the main/UI thread.

Note: The *libraryArgs* values are passed to the callback which is registered using *kony.application.setLibraryHeadlessModeCallback* API.

Example

```
[ KonyLibrary invokeInHeadLessMode:<libraryId> libraryArgs  :
<libraryArgs>  delegate:<resultDelegate> ] ;
```

19.3 API to Start Library UI

Use the API for respective platforms to launch the Kony library from the Native app. This feature is available from V8 SP3 onwards. The Start Library API is available for Android, iOS, and Windows platforms.

Note: Calling this API from the Native app will internally call the appService callback of the Kony application.

19.3.1 Android

You must use this API to launch the library UI. This API must be called after the library is successfully initialized.

Note: You must call the Initialize Library API before using the Start Library API; otherwise, this method throws an exception.

Note: Avoid passing null to mandatory parameters as that results in an exception.

Package

com.kony.library

Class

KonyLibrary

Syntax

```
public static void start(String libraryId,
       android.app.Activity activity,
       Hashtable<String,Object> libraryArgs,
       Hashtable<String,Object> libraryConfig,
       KonyLibrary.OnLibraryResultListener resultListener)throws
Exception
```

Input Parameters

Parameter	Description
libraryId	Specify the name of the library (Kony application AppID) that needs to be launched. This parameter cannot be null.
activity	Specify the Native app activity that is used to start the main activity of the Kony library. This parameter cannot be null.

Parameter	Description
libraryArgs	This parameter (key-value pair with supported values as primitive data types such as int, float, double, boolean, and String) is used to pass the native app's data to the library (based on the contract of the library). ***Note:*** libraryArgs are passed to the Kony application's appService callback in new launch mode, which is known as library mode (whose constant is kony.application. APP_LAUNCH_ MODE_ LIBRARY).

Parameter	Description
libraryConfig	The parameter that is used to configure the library. ***Note:*** The libraryConfig parameter is currently ignored. You can pass null for this parameter.
resultListener	Register a callback to send an acknowledgment (result) from Kony library to the Native app. This parameter cannot be null. ***Note:*** The resultListener methods are called from the main/UI thread.

Example

```
try {
    KonyLibrary.start( < libraryId > , < activity > , < libraryArgs >
, null, < resultListener > );
} catch (Exception ex) {}
```

19.3.1.1 OnLibraryResultListener Interface

Package

com.kony.library

Class

KonyLibrary

Syntax

```
public static interface OnLibraryResultListener
```

Methods

onLibraryResult

```
abstract void onLibraryResult(String libraryId,
Hashtable<String,Object> resultData)
```

Parameters

Parameter	Description
libraryId	Specifies the library for which the acknowledgment (result) is sent.

Parameter	Description
resultData	Specifies the result data in the form of key-value pairs. ***Note:*** The library should publish the keys of resultData to let the Native app handle the resultData.

19.3.2 iOS

You must use this API to launch the library UI. This API must be called after the library is successfully initialized.

Note: You must call the Initialize Library API before using the Start Library API; otherwise, this method throws an exception.

Note: Avoid passing null to mandatory parameters as that results in an exception.

Import

<CoreComponent/KonyLibrary.h>

Class

KonyLibrary

Method

+ start : viewController: libraryArgs : libraryConfig: delegate:

Declaration

```
+ (void) start:(NSString *)libraryId
         viewController:(UIViewController*) viewController
         libraryArgs:(NSDictionary*) libraryArgs
         libraryConfig:(NSDictionary*) libraryConfig
         delegate:(id<KonyLibraryResultDelegate>) resultDelegate
```

Input Parameters

Parameter	Description
libraryId	Specify the name of the library (Kony application AppID) that needs to be launched. This parameter cannot be null.
viewController	Specify the Native app's viewController reference to push the main UI controller into the library.

Parameter	Description
libraryArgs	This parameter (key-value pair with supported values as primitive data types such as int, float, double, boolean, and String) is used to pass the native app's data to the library (based on the contract of the library). ***Note:*** libraryArgs are passed to the Kony application's appService callback in new launch mode, which is known as library mode (whose constant is kony.application.APP_LAUNCH_MODE_LIBRARY).
libraryConfig	The parameter that is used to configure the launch of the library. ***Note:*** The libraryConfig parameter is currently ignored.

Parameter	Description
resultDelegate	Register a delegate (that conforms to KonyLibraryResultDelegate) to send an acknowledgment (result) from Kony library to the Native app. This parameter cannot be null. ***Note:*** The resultListener methods are called from the main/ UI thread.

```
[KonyLibrary start:<libraryId> viewController:<viewController>
libraryArgs:<libraryArgs>
libraryConfig:<libraryConfig> delegate:<resultDelegate>];
```

19.3.2.1 Protocol : KonyLibraryResultDelegate

Import

<CoreComponent/KonyLibrary.h>

Class

KonyLibrary

Method

(void)onLibraryResult: resultData:

Declaration

```
(void) onLibraryResult: (NSString * ) libraryId resultData:
(NSDictionary * ) resultData
```

Parameters

Parameter	Description
libraryId	Specifies the library for which the acknowledgment (result) is sent.
resultData	Specifies the result data in the form of key-value pairs. ***Note:*** The library should publish the keys of resultData to let the Native app handle the resultData.

19.3.3 Windows

You must use this API to launch the library UI. This API must be called after the library is successfully initialized.

Note: You must call the Initialize Library API before using the Start Library API; otherwise, this method throws an exception.

Note: Avoid passing null to mandatory parameters as that results in an exception.

Package

Framework

Class

KonyLibrary

Syntax

```
public static void Start(string libraryId,
       Dictionary<string, Object> libraryArgs,
       Dictionary<string, Object> libraryConfig,
       ILibraryResultListener resultListener)
```

Parameters

Parameter	Description
libraryId	Specify the name of the library (Kony application AppID) that needs to be launched. This parameter cannot be null.

Parameter	Description
libraryArgs	This parameter (key-value pair with supported values as primitive data types such as int, float, double, boolean, and String) is used to pass the native app's data to the library (based on the contract of the library). ***Note:*** libraryArgs are passed to the Kony application's appService callback in new launch mode, which is known as library mode (whose constant is kony.application. APP_LAUNCH_ MODE_ LIBRARY).

Parameter	Description
libraryConfig	The parameter that is used to configure the launch of the library. ***Note:*** The libraryConfig parameter is currently ignored.
resultListener	Register a callback to send the acknowledgment (result) from Kony library to the Native app. This parameter cannot be null.

Note: The resultListener methods are called from the main/ UI thread.

19.3.3.1 ILibraryResultListener Interface

Package

Framework

Syntax

```
public interface ILibraryResultListener
```

Methods

OnLibraryResult

```
void  OnLibraryResult(String libraryId, Dictionary <string,
Object> resultData)
```

Parameters

Parameter	Description
libraryId	Specifies the library for which the acknowledgment (result) is sent.
resultData	Specifies the result data in the form of key-value pairs. ***Note:*** The library should publish the keys of resultData to let the Native app handle the resultData.

20. Communication API for React Native App

Communication APIs for React Native allow you to communicate between the ReactNative app and a Kony app. This feature is available from V8 SP4 onwards.

When you embed a react native app into an app you are developing using Kony Visualizer, the Communication API for the React Native app helps in the interaction between the React Native app and app you are developing using Kony Visualizer. This feature is available from V8 SP4 onwards.

The following React Native APIs help you in communicating from the React Native app to the Kony app:

Function	Description
`reactNative.invokeKonyCallback`	Invokes the Kony callback, which is registered in the Kony app by using the kony.reactNative.setCallback API.
`reactNative.setCallback`	Registers a callback/listener to receive the result/response from the Kony app context.

The following APIs from the `kony.reactNative Namespace` help you in communicating from the Kony app to the React Native app:

Function	Description
`kony.reactNative.setCallback`	Registers a callback/listener for the incoming request from the React Native app context.
`kony.reactNative.sendResult`	Sends a response (for the request) to the React Native app, if the React Native app registers a callback.

To set a callback/listener in the React Native app, use the `kony.reactNative.setCallback` function. On receiving the request, invoke the required function in the React Native app by using the `reactNative.invokeKonyCallback` function.

Also, set a callback in the Kony app by using the `kony.reactNative.setCallback` function. The Kony app sends a data request to the React Native app by using the `kony.reactNative.sendResult` function.

For more information related to the React Native App Integration into Kony App feature, refer these sections:

- Integrate React Native App into a Kony App
- React Native Container

20.1 kony.reactNative Namespace

The kony.reactNative APIs namespace contains the following JavaScript APIs. The kony.reactNative APIs are available from V8 SP4 onwards.

20.1.1 Functions

The kony.reactNative namespace contains the following functions.

kony.reactNative.setCallback

This API registers a callback/listener for the incoming request from the React Native app context. The callback that is set by using the kony.reactNative.setCallback API will be invoked while using the reactNative.invokeKonyCallback API from the React Native app.

> ***Note:*** Calling the kony.reactNative.setCallback API two or more times with different callbacks will override the previously set callback with the recent one.

Syntax

```
kony.reactNative.setCallback(callback)
```

Input Parameters

Parameter	Description
callback	This parameter registers a function with the following signature. `function callback (id,args) { ................ }`

The callback function does not have any return value and consists of the following parameters:

- **id:** Unique identifier of the React Native app's request. This identifier is used to bind a request with the response.

  > ***Note:*** While using the kony.reactNative.sendResult API, you must pass the *id* parameter to send the result to a particular request instance.

- **args:** Object with key-value pairs that is passed from the React Native app to the Kony app based on the contract of the Kony app.

Example

```
function callback(id, args) {
        ........

    if (args != null & amp; & amp; args["operation"] == "fetchAccounts") {
        // fetchAccounts and store result
        account1 = {
            name: "kony1",
            accountNo: "0123456789",
            accountBalance: "xx".....
        };
        account2 = {
            name: "kony2",
            accountNo: "2345678901",
            accountBalance: "xx".....
        };
        resultData = {
            status: "success",
            data: [account1, account2]..
        };
        kony.reactNative.sendResult(id, resultData);

    };
}
kony.reactNative.setCallback(callback);
```

Return Value

None

Platform Availability

- iOS
- Android

kony.reactNative.sendResult

This API is used to send a response (for the request) to the React Native app, if the React Native app registers a callback.

Note: This API must be called only once per request to send the result or response. Calling this API more than once per request leads to no operation from the second instance onwards.

Syntax

```
kony.reactNative.sendResult(id, resultData)
```

Input Parameters

Parameter	Description
id	Unique identifier of the React Native app (this parameter is received in the callback function of kony.reactNative.setCallback API.)

Parameter	Description
resultData	Object with key-value pair elements with key as String and value as JavaScript native data types, such as Number, Boolean, String, Array, and Object. ***Note:*** The Kony app must publish the keys of the resultData parameter, in order to predict the resultData value by the React Native app.

Example

```
var resultData = {
    status: "success",
    ...
};
kony.reactNative.sendResult(id, resultData);
```

Return Value

None

Platform Availability

- iOS
- Android

20.2 React Native APIs

The following React Native APIs are utilized for the communication between the React Native app framework and the Kony framework. These React Native APIs are available from V8 SP4 onwards.

reactNative.invokeKonyCallback

This API invokes the Kony callback, which is registered in the Kony app by using the kony.reactNative.setCallback API.

Syntax

```
reactNative.invokeKonyCallback(id, args)
```

Note: You must prefix *NativeModules* while calling this API, if NativeModules.reactNative is not exported from the React Native module and if the same is not imported in the module file where this API is used.

Input Parameters

Parameter	Description
id	Request identifier of the ReactNative app. This parameter is used to bind the request with the response. ***Note:*** If a request has to listen for a response, you must register the listener/callback for the request by using the reactNative.setCallback API and leveraging the same *id* parameter as of the reactNative.invokeKonyCallback API. ***Note:*** The value of the *id* parameter must follow this format: ReactNativeAppName + unique string.
args	Object with key-value pair elements with key as String and value as Object, such as Number, Boolean, String, Array, and Object. This parameter is used to pass data from the React Native app to the Kony app context.

reactNativeCallback registers a React Native callback function. This callback will be invoked by using the kony.reactNative.sendResult API from the Kony app context.

Example

```
args = {
    operation:"fetchAccounts"
};
NativeModules.reactNative.invokeKonyCallback(<id>, args);

//You can omit the NativeModules prefix if NativeModules.reactNative is
exported from the React Native module
//and the same is imported in the module file where this API is used.
```

Return Value

None

Platform Availability

- iOS
- Android

reactNative.setCallback

This API is used to register a callback/listener to receive the result/response from the Kony app context. The registered callback will be invoked while using the kony.reactNative.sendResult API from the Kony app context.

Syntax

```
reactNative.setCallback(id, Callback)
```

Input Parameters

Parameter	Description
id	Request identifier for the React Native app. Since there can be multiple React Native apps integrated in to a Kony app, the *id* parameter is used to distinguish between the different React Native apps as well as to differentiate different invocations from the same app. ***Note:*** The value of the *id* parameter must follow this format: ReactNativeAppName + unique string.

Parameter	Description
Callback	JavaScript function. Callbacks are internally maintained as a dictionary of this API's *id* and *Callback* parameters. The callback will be removed from the dictionary once the result is sent (by using the kony.reactNative.sendResult API) to avoid an exception of the React Native framework. This exception occurs when the kony.reactNative.sendResult API is called more than once per request.

Example

```
function reactNativeCallback(resultData){
  ..........
};

NativeModules.reactNative.setCallback(<id>, reactNativeCallback);

//You can omit the NativeModules prefix if NativeModules.reactNative is
exported from the React Native module
//and the same is imported in the module file where this API is used.
```

Return Value

None

Platform Availability

- iOS
- Android

---

21. Cryptography API

Cryptography is the process of securing the information. It can be defined as the conversion of data into scrambled text to conceal its readability and meaning, and deciphering it using a key. This data can be sent across safely over public and private networks.

When your app has to transmit sensitive or critical information over the network (private or public), it needs to encrypt the information to ensure information security. A decryption mechanism is used to convert the encrypted text into a readable format.

There are two types of cryptography:

- **Symmetric Cryptography:** A single key is generated. The same key is used in encryption and decryption.
- **Asymmetric or Public Key Cryptography:** A public key and a private key are generated. The public key is used for encryption and the private key is used for decryption. The public key is available in a trusted certificate, whereas the private key is confidential and not shared. Encryption is done on the device and decryption on the server (that has access to the private key). As the private key is not exposed, we support only encryption using asymmetric cryptography.

The Cryptography API enables your app to provide information and services in a secure way. It uses the `kony.crypto Namespace` and the following API elements.

Function	Description
`kony.crypto.asymmetricEncrypt`	Encrypts the input string and returns the encrypted text. This API is available from V8 SP3 onwards.

Function	Description
`kony.crypto.asymmetricDecrypt`	Decrypts the input encrypted string. This API is available from V8 SP3 onwards.
`kony.crypto.createHash`	Provides your app with the ability to create a hash value in hexadecimal format for a given input string using a specified algorithm.
`kony.crypto.createHMacHash`	Generates a hash-based message authentication code (HMAC) that verifies the data integrity and authenticity of the data.

Function	Description
`kony.crypto.createPBKDF2Key`	Creates a Password-Based Key Derivation Function 2 (PBKDF2) key for protecting passwords and other similar tasks.
`kony.crypto.decrypt`	Provides the ability to decrypt the encrypted text with the specified key and algorithm. The API returns the decrypted text.
`kony.crypto.deleteKey`	Provides you the ability to delete a key from the device store.

Function	Description
`kony.crypto.encrypt`	Provides the ability to encrypt the input text with the specified key and algorithm. The rawbytes of the encrypted text are returned.
`kony.crypto.generateAsymmetricKeyPair`	Generates public and private keys for encryption and decryption processes.
`kony.crypto.generateSecureRandom`	Generates cryptographically secure random numbers. This API is available from V8 SP3 onwards.

Function	Description
`kony.crypto.newKey`	Creates a key for cryptography using the specified algorithm. The key created using this API is used for encrypting clear text and decrypting the encrypted data.
`kony.crypto.readKey`	Provides you the ability to read the key from the device store.
`kony.crypto.retrieveAsymmetricPublicKey`	Returns the public key for the alias that you provide. This API is available from V8 SP3 onwards.

Function	Description
`kony.crypto.retrievePublicKey`	Provides the ability to extract the public key from a base64 string of encoded X509 certificate or a locally packaged X509 certificate.
`kony.crypto.saveKey`	Enables your app to save a generated key on the device's storage.

The Cryptography API enables your app to provide information and services in a secure way.

For symmetric cryptography, create a new key by using the `kony.crypto.newKey` function, then save the key to the device’s local storage using the `kony.crypto.saveKey` function. To read the key, use the `kony.crypto.readKey` function. With the generated key, you can now encode the input data by using the kony.crypto.encrypt function. Then, to decrypt the encoded data, use the `kony.crypto.decrypt` function. If you want to delete any key from the device store, use the `kony.crypto.deleteKey` function.

For asymmetric cryptography, generate public and private keys using the `kony.crypto.generateAsymmetricKeyPair` function. Retrieve the generated keys by using the `kony.crypto.retrieveAsymmetricPublicKey` or `kony.crypto.retrievePublicKey` functions. With the generated public key, you can now encode the input data using the `kony.crypto.asymmetricEncrypt` function. Then, to decrypt the encoded data, use the `kony.crypto.asymmetricDecrypt` function.

Hashing is a one-way encryption process unlike the cryptography. The generated hash value cannot be decrypted. For Hashing, generate a Password-Based Key Derivation Function 2 (PBKDF2) key using the `kony.crypto.createPBKDF2Key`function. With the generated key, you can encrypt the input text using the `kony.crypto.createHash`function. Once the hash value is generated, you can check the authenticity of the data using the `kony.crypto.createHMacHash` function.

21.1 Concepts

To understand how the Kony Cryptography API works, you must be familiar with some basic concepts.

21.1.1 Ciphers

Apps encrypt and decrypt data using *ciphers*. A cipher is an algorithm for performing encryption and decryption. The Kony Cryptography API supports *block ciphers*.A block cipher is a symmetric key cipher that operates on fixed-length groups of bits called blocks. For example, a block cipher encryption algorithm takes a 128-bit block of plain text as input, and produces an output of a corresponding 128-bit block of cipher text. The process of transformation is governed by a secret key. Decryption is a similar process. The decryption algorithm takes a 128-bit block (in this example) of cipher text along with the secret key, and yields the original 128-bit block of plain text.

Block cipher algorithms like DES require their input to be an exact multiple of the block size. If the plain text to be encrypted is not an exact multiple, your app must add padding before encrypting the data.

The following table describes the supported padding for different cipher algorithms:

Algorithm	Platform	Possible Padding
AES/TripleDES	iPhone	pkcs7(default)
	Android/Android Tablet	none,pkcs1,pkcs5(default)
	BlackBerry	none,pkcs5(default)
	SPA	ISO10126, ISO97971, zeropaddding, nopadding, pkcs7
RSA	iPhone	none,pkcs1(default)
	Android/Android Tablet	none,pkcs5,pkcs1(default)
	BlackBerry	none,pkcs5,pkcs1(default)
	Windows Phone/Windows 8.1/Kiosk	pkcs1

RSA is an algorithm for public-key cryptography. It is the first algorithm known for both signing and encryption. This algorithm involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key.

21.1.2 Modes

Ciphers are controlled by specific encryption *modes*. The mode specifies how the data will be encrypted and decrypted . The Kony Cryptography API supports the following standard modes.

- Electronic Codebook (ECB) mode
- Cipher Block Chaining (CBC) mode
- Cipher Feedback (CFB) mode
- Output Feedback (OFB) mode
- ECB and CBC are valid modes for the Block ciphers

Because these are standard modes, you can find extensive information on the Internet about them if you are not familiar with them.

The following table describes the supported modes for various ciphers.

Algorithm	Platform	Possible Modes
AES/TripleDES	iPhone	ecb,cbc(default)
	Android	ecb,cbc(default)
	BlackBerry	ecb,cbc(default)
	SPA	ecb, cfb, ofb, cbc, ctr

Algorithm	Platform	Possible Modes
RSA	iPhone	ecb(default)
	Android/Android Tablet	ecb(default)
	BlackBerry	ecb(default)

21.1.3 Initialization Vectors

The initialization vector (IV) adds randomness to the cipher text. As a result, if the same message is encrypted twice with the same key, the resulting cipher texts will be different, as long as your app uses different initialization vectors.

The length in bytes of the IV should be the same as the block size of the algorithm (AES and TripleDES). For AES the IV has to be 16 bytes (128 bits). For TripleDES the IV has to be 8 bytes (64 bits).

21.2 Supported Algorithms

The Kony Cryptography API provides the following algorithms.

21.2.1 Symmetric Algorithms

For symmetric encryption and decryption, your app can use the following algorithms.

Algorithm	Supported Key Strengths (in bits)	Description
aes	128, 192, and 256	Advanced Encryption Standard (AES).

Algorithm	Supported Key Strengths (in bits)	Description

tripledes	Triple Data Encryption Standard	Triple Data Encryption Standard. This algorithm uses two keys for encryption and one key for decryption. The algorithm works as follows:
		
		`encryptedtext = E`$_{K3}$`(D`$_{K2}$`(E`$_{K1}$`(text to be encrypted)))`
		
		The text is encrypted with *key1*, decrypted with *key2*, and then again encrypted with *key3*. Decryption is the reverse:
		
		`decryptedtext = D`$_{K3}$`(E`$_{K2}$`(D`$_{K1}$`(text to be decrypted)))`
		
		tripledes applies the DES cipher algorithm three times to each data block.
		
		The following are the three different combinations of using this algorithm:
		
		◦ **Option 1**: K1, K2, and K3 are different (all three keys are independent)
		◦ **Option 2**: K1 and K2 are different; K3=K1 (two keys are independent)
		◦ **Option 3**: K1=K2=K3 (all three keys are identical)
		
		Note: Kony recommends using *Option 1* as it is more secure than the other two options.

21.2.2 Asymmetric Algorithms

For asymmetric encryption, the Kony Cryptography API supports the RSA algorithm, which is the industry standard. The key length for RSA must be 1024 bits or more.

21.2.3 Platform Limitations

SPA does not support the RSA algorithm.

To view the functionality of the Cryptography API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

21.3 kony.crypto Namespace

The kony.crypto namespace provides the following API elements.

21.3.1 Functions

The kony.crypto namespace contains the following functions.

kony.crypto.asymmetricEncrypt

This API encrypts the input string and returns the encrypted text. This API is available from V8 SP3 onwards.

Syntax

```
kony.crypto.asymmetricEncrypt(alias, inputstring, propertiesTable)
```

Input Parameters

Parameters	Description
alias [String] - Mandatory	You can generate the value of the alias parameter by using generateAsymmetricKeyPair API.
inputstring[String] - Mandatory	The input text to be encrypted.

Parameters	Description
propertiesTable [Object] - Mandatory	The applicable values for this parameter are as follows: • **transformation** (String): The cipher transformation to be used. Possible transformation values are as follows: • For iOS • RSA:raw • RSA:PKCS1 • RSA:OAEP:SHA1 • RSA:OAEP:SHA224 • RSA:OAEP:SHA256 • RSA:OAEP:SHA384 • RSA:OAEP:SHA512 • RSA:OAEP:SHA1:AESGCM • RSA:OAEP:SHA224:AESGCM • RSA:OAEP:SHA256:AESGCM • RSA:OAEP:SHA384:AESGCM • RSA:OAEP:SHA512:AESGCM • For Android and Windows • "RSA/ECB/PKCS1Padding" • "RSA/ECB/OAEPWithSHA-1AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-224AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-256AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-384AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-512AndMGF1Padding" • "RSA/ECB/OAEPPadding" • "RSA/NONE/NoPadding"

Example

```
kony.crypto.asymmetricEncrypt(alias, inputstring, propertiesTable);
```

```
asymmetricEncrypt: function() {
    var key = this.view.tbxasyencrypt.text;

    //#ifdef iphone
    encryptedobject = kony.crypto.asymmetricEncrypt("Kony", key, {
        "transformation": "RSA:OAEP:SHA1"
    });
    var encryptBase64foriOS = kony.convertToBase64(encryptedobject);
    alert("The Encrypted text is as follows " + encryptBase64foriOS);
    //#endif
    //#ifdef andorid
    encryptedobject = kony.crypto.asymmetricEncrypt("Kony", key, {
        "transformation": "RSA/ECB/PKCS1Padding"
    });
    var encryptBase64forAndroid = kony.convertToBase64(encryptedobject);
    alert("The Encrypted text is as follows " + encryptBase64forAndroid);
    //#endif
},
```

Return Value

rawbytes [Object] - The rawbytes for the encrypted version of the input text.

Limitations

- RSA can only encrypt data to a maximum amount of your keysize (256 bytes) - padding)/header data.
- keytype is not considered for Android.
- transformation is not considered for Windows.
- This API throws exceptions.
- This API does not work on Android devices with API level earlier than 18.
- On Android devices with API level 18 to 22 (both inclusive), only PKCS1Padding is supported

("RSA/ECB/PKCS1Padding" works on all devices with API level 18 and later).

- Both PKCS1Padding and OAEPPadding are supported on Android devices with API level 23 and later.
- OAEPPadding transformations are not supported on all Android devices, as there is no documentation from Android for this limitation.
- For iOS, this API works on devices with iOS 10 or later.

Platform Availability

- iOS
- Android
- Windows 10
- Windows Desktop

kony.crypto.asymmetricDecrypt

This API decrypts the input encrypted string. This API is available from V8 SP3 onwards.

Syntax

```
kony.crypto.asymmetricDecrypt(alias, encryptedString, propertiesTable)
```

Input Parameters

Parameters	Description
alias [String] - Mandatory	You can generate the value of the alias parameter by using generateAsymmetricKeyPair API.
encryptedString [Object] - Mandatory	An object that contains the encrypted text to be decrypted.

Parameters	Description
propertiesTable [Object] - Mandatory	The applicable values for this parameter are as follows: • **transformation (String)**: The cipher transformation to be used. Possible transformation values are as follows: • For iOS • RSA:raw • RSA:PKCS1 • RSA:OAEP:SHA1 • RSA:OAEP:SHA224 • RSA:OAEP:SHA256 • RSA:OAEP:SHA384 • RSA:OAEP:SHA512 • RSA:OAEP:SHA1:AESGCM • RSA:OAEP:SHA224:AESGCM • RSA:OAEP:SHA256:AESGCM • RSA:OAEP:SHA384:AESGCM • RSA:OAEP:SHA512:AESGCM • For Android and Windows • "RSA/ECB/PKCS1Padding" • "RSA/ECB/OAEPWithSHA-1AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-224AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-256AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-384AndMGF1Padding" • "RSA/ECB/OAEPWithSHA-512AndMGF1Padding" • "RSA/ECB/OAEPPadding" • "RSA/NONE/NoPadding"

Example

```
kony.crypto.asymmetricDecrypt(alias, encryptedString, propertiesTable);
```

```
asymmetricDecrypt: function() {
    if (kony.os.deviceInfo().name == "iPhone") {
        var decryptedForiOS = kony.crypto.asymmetricDecrypt("Kony",
encryptedobject, {
            "transformation": "RSA:OAEP:SHA1"
        });
        alert("The Decrypted Message is as follows " + decryptedForiOS);
    } else {
        var decryptedForAndroid = kony.crypto.asymmetricDecrypt("Kony",
encryptedobject, {
            "transformation": "RSA/ECB/PKCS1Padding"
        });
        alert("The Decrypted Message is as follows " + decryptedForAndroid);
    }
},
```

Return Value [String]

Returns the decrypted/cipher text.

Limitations

- transformation is not considered for Windows.
- keytype is not considered for Android
- This API does not work on Android devices with API level earlier than 18.
- On Android devices with API level 18 to 22 (both inclusive), only PKCS1Padding is supported ("RSA/ECB/PKCS1Padding" works on all devices with API level 18 and later).
- Both PKCS1Padding and OAEPPadding are supported on Android devices with API level 23 and later.
- OAEPPadding transformations are not supported on all Android devices, as there is no documentation from Android for this limitation.
- For iOS, this API works on devices with iOS 10 or later.

Platform Availability

- iOS
- Android
- Windows 10
- Windows Desktop

kony.crypto.createHash

This function provides your app with the ability to create a hash value in hexadecimal format for a given input string using a specified algorithm.

Syntax

```
kony.crypto.createHash(
    algo,
    inputstring)
```

Input Parameters

Parameters	Description
algo	A string containing the algorithm to be used for creating the hash value. For details, see the **Remarks** section below.
inputstring	A string that holds the data to be hashed.

Example

```
var algo = "md5";
var inputstr = "pleasecreatehash";
var myHashValue = kony.crypto.createHash(algo, inputstr);
```

```
createHashMD2: function() {
    try {
        var algo = "md2";
        var inputstr = this.view.txtMd2Hash.text;
        var myHashValue = kony.crypto.createHash(algo, inputstr);
        this.view.lblMD2Hash.text = myHashValue;
    } catch (err) {
        alert(typeof err);
        alert("Error in callbackCreateHashMD2 : " + err);
    }
},
```

Return Values

This function returns a string containing the hash value of the *inputstring* parameter created using the algorithm specified in the *algo* parameter. This string is in hexadecimal format. The length of the string in bytes is as follows.

Hashing Algorithm	Result Length (in bytes)	Result Length (in hexadecimal characters)
sha1	20	40
sha224	28	56
sha256	32	64
sha384	48	96
sha512	64	128
md2	16	32
md4	16	32
md5	16	32

© 2013 by Kony, Inc. All rights reserved

Exceptions

If an error occurs, this function throws on of the following errors.

Error Code	Description
2001	An unsupported algorithm was specified for the *algo* parameter.
2002	An invalid key strength was specified.
2003	A buffer of insufficient was provided for specified operation.
2004	A memory allocation failure occurred.
2005	The input data did not encode or encrypt properly.
2006	The specified name already exists.
2007	A key with the specified unique ID is not found.

Remarks

The `kony.crypto.createHash` function encrypts data by creating a hash of it. The first parameter to this function specifies the cipher. or the encryption algorithm, to use on the data. The *algo* parameter can be one of the following values.

Algorithm	Description
sha1	Secure Hash Algorithm 1 (SHA-1)
sha224	Secure Hash Algorithm 224 (SHA-224).
sha256	Secure Hash Algorithm 224 (SHA-256).
sha384	Secure Hash Algorithm 224 (SHA-384).
sha512	Secure Hash Algorithm 224 (SHA-512).
md2	Message-Digest Algorithm 2 (MD2).
md4	Message-Digest Algorithm 4 (MD4).
md5	Message-Digest Algorithm 5 (MD5).

- md5, sha1, sha256, hmacsha1, and hmacsha256 are supported for Windows phone 8.1.
- md5, sha1, sha256, sha384, and sha512 are supported for Windows 8.1 and Windows 10.
- md5, sha1, sha256, sha384, sha512, hmacsha1, and hmacsha256 are supported for Windows Desktop.

Platform Availability

Available on all platforms except J2ME and Windows Kiosk.

kony.crypto.createHMacHash

This function generates a hash-based message authentication code (HMAC) that verifies the data integrity and authenticity of the data.

Syntax

```
kony.crypto.createHMacHash(
    algo,
    key,
    message)
```

Input Parameters

Parameters	Description
algo	A string that contain the hashing algorithm. See the Remarks section below for more information.
key	A string that holds the input key for the algorithm.
message	A string that contains the plain text message for which the hash is generated.

Example

```
var myHash = kony.crypto.createHMacHash("SHA1", "key1", "test message to
generate hash");
kony.print("myHash :" + myHash);
```

```
createHMacHash: function() {
    var algo = "sha1";
    var message = this.view.txtHMacHash.text;
    var hMacHashKey = kony.crypto.newKey("passphrase", 128, {
        passphrasetext: ["inputstring1"],
        subalgo: "aes",
        passphrasehashalgo: "md2"
    });
    var myHashValue = kony.crypto.createHMacHash(algo, hMacHashKey, message);
    this.view.lblHMacHash.text = myHashValue;
},
```

Return Values

This function returns a string that holds the hash value created using the specified algorithm for the given input string. This string is in a hexadecimal format. The length of the string in bytes is as follows.

Hashing Algorithm	Result Length (in bytes)	Result Length (in hexadecimal characters)
sha1	20	40
sha224	28	56
sha256	32	64
sha384	48	96
sha512	64	128
md2	16	32
md4	16	32
md5	16	32

Exceptions

This function throws the following exceptions.

Error Code	Description
100	One or more input parameters are invalid.
101	An unsupported algorithm was specified for the *algo* parameter.
102	An unknown error occurred.
104	The key strength was invalid.
105	A mandatory algorithm parameter is missing.
109	The specified item could not be found.

Remarks

The following table lists algorithms supported for each platform.

> ***Note:*** From Kony Visualizer V8 release, the MD5 support is done through Java and not through the Bundle OpenSSL Library.

Platform Name	Supported Algorithms
Android Default Implementation	MD5, SHA1, SHA224, SHA256, SHA384, SHA512 (SHA224 supported only on API level 21 and above)
Android OpenSSL Implementation (Bundle OpenSSL Library option is selected in Kony Visualizer)	MD5, SHA1, SHA224, SHA256, SHA384, SHA512
iOS	MD5,SHA1,SHA224,SHA256,SHA384,SHA512

On Android, the *Bundle OpenSSL Library* option is available in the **Application Properties > Native > Android** section. If this option is selected, OpenSSL library is bundled along with the application and use by this function. If the *Bundle OpenSSL Library* option is not selected in Kony Visualizer, the default Java implementation offered by the underlying native Android platform is used.

If the device under testing does not support a the hashing algorithm your app selects, this function throws an exception.

Platform Availability

Available on iOS and Android.

kony.crypto.createPBKDF2Key

The `kony.crypto.createPBKDF2Key` function creates a Password-Based Key Derivation Function 2 (PBKDF2) key for protecting passwords and other similar tasks.

Syntax

```
kony.crypto.createPBKDF2Key(
    algo,
    password,
    salt,
    iteration,
    klen)
```

Input Parameters

Parameters	Description
algo	A string that specifies the hashing algorithm used for creating the key. For a list of supported algorithms by platform, see the **Remarks** section below.
password	A string that contains the master password from which a derived key is generated.
salt	A string that that holds a random salt input string from a programmer
iteration	A number that specifies the desired number of iterations. Should be at least 10,000, as per NIST standards.
klen	An optional numeric parameter that specifies the desired length of the derived key in bits. If the key length is not specified, this value defaults to 256-bits.

Example

```
var mySecureNewKey = kony.crypto.createPBKDF2Key("SHA1", "password", "salt",
1000, 256);
kony.print("mySecureNewKey :" + mySecureNewKey);
```

```
createPBKDF2KEY: function() {
    var algo = "SHA1";
    var password = this.view.txtPBKDF2Key.text;
    var PBKDF2Key = kony.crypto.createPBKDF2Key(algo, password, "salt", 10000,
256);
    this.view.lblPBKDF2Key.text = PBKDF2Key;
},
```

Return Values

Returns the key created using the PBKDF2 algorithm.

Exceptions

The following table shows the error codes for the exceptions that this function throws, as well as their descriptions .

Error Code	Description
100	Invalid Input parameters
101	Unsupported algorithm
102	Unknown error
104	Invalid key strength
105	Sub algorithm parameter is mandatory
109	The specified item could not be found.

Remarks

Note: kony.crypto.createPBKDF2Key API does not support md5 algorithm from Kony Visualizer V8 release.

The Password-Based Key Derivation Function 2 (PBKDF2) is a key derivation function that generates encryption keys of different lengths to protect passwords.

PBKDF2 applies a hash function (chosen by *algo* parameter) to the input password or passphrase (specified in the *password* parameter), along with a salt value and repeats the process as many times as is specified in the *iteration* parameter to produce a derived key that is of the length given in the *klen* parameter, if a value for *klen* is provided. The resultant key is used as a cryptographic key in subsequent operations. The added computational work caused by a high number of iterations, or key stretching, makes it more difficult to crack a password. So when you specify the number of iterations, you need to balance security against app performance.

The following table lists algorithms supported for a specific platform. When your app calls the `kony.crypto.createPBKDF2Key` function, it must select one of the algorithms given in the table for the value of the *algo* parameter.

Platform Name	Supported Algorithms
Android Default Implementation	SHA1
Android OpenSSL Implementation (Bundle Open SSL Library option is selected in Kony Visualizer)	SHA1, SHA224, SHA256, SHA384, SHA512
iOS	SHA1 , SHA224, SHA256, SHA384, SHA512
Windows	SHA1, SHA256, SHA384, SHA512, and MD5.

In Android, the *Bundle OpenSSL Library* option is available in **Application Properties > Native > Android** section. If this option is selected, the OpenSSL library is bundled along with the application.

- If the *Bundle OpenSSL Library* option is selected in Kony Visualizer, implementation in OpenSSL library is used.
- If the *Bundle OpenSSL Library* option is not selected in Kony Visualizer, default Java implementation offered by the native Android platform is used.

If the *klen* parameter is provided to this function, you must make sure that this key length is supported by a corresponding encryption or decryption algorithm. For aes ciphers, the supported key lengths are 128, 192, or 256 bits. For tripledes ciphers, the possible key length is 192.

Platform Availability

Available in iOS, Android , and Windows.

kony.crypto.decrypt

This function provides the ability to decrypt the encrypted text with the specified key and algorithm. The API returns the decrypted text.

Syntax

```
kony.crypto.decrypt(
    algo,
    generatedkey,
    encryptedRawbytes,
    propertiesTable)
```

Input Parameters

Parameters	Description
algo	A string that specifies the decryption algorithm. For possible values, see the **Remarks** section below.
generatedkey	An object that holds the key to be used for decryption.
encryptedRawbytes	An object that contains the rawbytes of the encrypted text to be decrypted.
propertiesTable [Table] - Mandatory	A JavaScript object that contains key-value pairs necessary for decryption. For details, see the **Remarks** section below.

Example

```
var algo = "aes";

var encryptDecryptKey = kony.crypto.newKey(
    "passphrase",
    128, {
        passphrasetext: ["inputstring1"],
        subalgo: "aes",
        passphrasehashlogo: "md5"
    });

var inputstr = "pleaseencryptme";

var prptobj = {
    padding: "pkcs5",
    mode: "cbc",
    initializationvector: "1234567890123456"
};

var myEncryptedText = kony.crypto.encrypt(
    algo,
    encryptDecryptKey,
    inputstr,
    prptobj);

var myClearText = kony.crypto.decrypt(
    algo,
    encryptDecryptKey,
    myEncryptedText,
    prptobj);
```

```
decrypt: function() {
    try {
        var algo = "aes";
        var myEncryptedTextRa = "";
        var encryptDecryptKey = kony.crypto.newKey("passphrase", 128, {
            passphrasetext: ["inputstring1"],
            subalgo: "aes",
            passphrasehashalgo: "md5"
        });

        var prptobj = {
            padding: "pkcs5",
            mode: "cbc",
            initializationvector: "1234567890123456"
        };

        if (this.view.lblEncrypt.text === "" || this.view.lblEncrypt.text ===
null || this.view.lblEncrypt.text === "Please enter the text to encrypt") {
            this.view.lblDecrypt.text = "There is no encrypted text";
            return;
        }
        var str = this.view.lblEncrypt.text;
        //convertToRawBytes is not supported in SPA
        //              if(kony.os.deviceInfo().name == "thinclient")
        //              {
        //                       myEncryptedTextRa = myEncryptedTextRaw;
        //              }
        //              else
        myEncryptedTextRa = kony.convertToRawBytes(str.substring(17));
        var myClearText = kony.crypto.decrypt(algo, encryptDecryptKey,
myEncryptedTextRa, prptobj);

        this.view.lblDecrypt.text = "Decrypted text = " + myClearText.toString
();

    } catch (err) {
        alert(typeof err);
        alert("Error in callbackDecryptAes : " + err);
    }
},
```

Return Values

Returns a string chat holds the clear text decrypted from the encrypted rawbytes.

Exceptions

CryptoError: Thrown by Crypto API.Various error conditions related to CryptoError will be covered through the following error codes.

- 2001 - unsupported algorithm.
- 2002 - invalid key strength specified.
- 2003 - insufficient buffer provided for specified operation.
- 2004 - memory allocation failure.
- 2005 - input data did not encode or encrypt properly.
- 2006 - specified name already exists.
- 2007 - key with the specified unique ID is not found.

Remarks

The values that your app can use for the *algo* parameter are as follows.

Constant	Description
aes	Selects AES encryption.
tripledes	Selects Triple DES encryption. Not available on Windows platforms.
rsa	Selects RSA encryption.

The JavaScript object in the *propertiesTable* parameter must have the following format.

Property	Description
padding	A string that specifies the padding characters to use. If no padding characters are given and the length of the encrypted text block is less than the block size, the underlying platform throws a Bad Padding error.
mode	A string that specifies the encryption mode.
initializationvector	A string that contains the initialization vector to use in performing the decryption. This property is applicable only if the algorithm is aes or tripledes.

The `padding` property of the object that is passed into this function through the *propertiesTable* parameter is used to pad the encrypted text so that the size of the encrypted text is the same as the block size used in the encryption/decryption algorithm selected in the *algo* parameter to this function. The block size for the available algorithms is as follows.

Property	Description
aes	128 bits
tripledes	64 bits
initializationvector	1024 or 2048 bits

For more information on padding, modes, and initialization vectors, see Concepts in the Cryptography API overviews.

Platform Availability

Available on all platforms except J2ME.

kony.crypto.deleteKey

This API provides you the ability to delete a key from the device store.

Use Cases

You can delete the key from the device store if you are sure that you do not need that key anymore in the application.

Syntax

```
kony.crypto.deleteKey(uniqueID)
```

Input Parameters

Parameters	
uniqueID [String] - Mandatory	Unique ID represents the key on the device store (this is the ID returned by kony.crypto.saveKey API).

Example

```
var encryptDecryptKey = kony.crypto.newKey("passphrase", 128, {
    passphrasetext: ["inputstring1"],
    subalgo: "aes",
    passphrasehashlogo: "md5"
});

var myUniqueIDKey = kony.crypto.saveKey("myKey", encryptDecryptKey);

kony.crypto.deleteKey(myUniqueIDKey);
```

```
deleteKey: function() {
    kony.crypto.deleteKey(saveKey);
    this.view.lblKey.text = "The key is deleted";
},
```

Return Values

None

API Usage

You can use this API only to delete the keys that you have saved earlier on the device store,which is keys that have a unique ID associated with it.

Exceptions

CryptoError: Thrown by Crypto API.Various error conditions related to CryptoError will be covered through the following error codes.

- 2001 - unsupported algorithm.
- 2002 - invalid key strength specified.
- 2003 - insufficient buffer provided for specified operation.
- 2004 - memory allocation failure.
- 2005 - input data did not encode or encrypt properly.
- 2006 - specified name already exists.
- 2007 - key with the specified unique ID is not found.

Platform Availability

Available on all platforms except J2ME and Windows Kiosk.

Note: You can use this API only to delete the keys that you have saved earlier on the device store, i.e., keys that have a unique ID associated with them.

kony.crypto.encrypt

Converting data into an encoded format using a key is known as *encryption*. Encryption of data is done through symmetric cryptography. We support both symmetric and asymmetric encryption.

This API provides the ability to encrypt the input text with the specified key and algorithm. The rawbytes of the encrypted text are returned.

Use Cases

You need to use encryption when you pass sensitive data like:

- passwords
- account numbers
- account information
- credit card information, and so on.

Syntax

```
kony.crypto.encrypt(algo, generatedkey, inputstring, propertiesTable)
```

Input Parameters

algo [String] - Mandatory	Specifies the algorithm using which the input string needs to be encrypted. Possible values are : • aes • tripledes • rsa tripledes algorithm is not supported in Windows Platforms.

generatedkey [Object] - Mandatory	The key to be used for encryption. • On Windows Phone and Kiosk, this parameter is the byte array that is returned in a call to kony.crypto.retrievePublicKey. • On Windows 8 this parameter is the name of the certificate that is included in the root directory of the Windows package, in "resources/common" in Kony Visualizer. • On all other platforms this API accepts the key generated using the kony.crypto.newKey and kony.crypto.createPBKDF2Key APIs. *Note:* The kony.crypto.createPBKDF2Key API is supported on iOS, Windows, and Android platforms.
inputstring [String] - Mandatory	Data which needs to be encrypted.
propertiesTable [Table] - Mandatory	*Note:* This parameter is ignored in Windows 8. This Object contains the following key-value pairs: • **padding** - a string that denotes the padding that needs to be applied. *Note:* Windows Phone and Kiosk support pkcs and oeap padding. • **mode** - a string that denotes the encryption mode. *Note:* This value is ignored for the rsa algorithm. • **initializationvector** - a string that denotes the Initialization Vector to be used. *Note:* This parameter is applicable only if the subalgo is aes or tripledes.

Examples

```
var algo = "aes";

var encryptDecryptKey = kony.crypto.newKey("passphrase", 128, {
    passphrasetext: ["inputstring1"],
    subalgo: "aes",
    passphrasehashlogo: "md5"
});

var inputstr = "pleaseencryptme";

var prptobj = {
    padding: "pkcs5",
    mode: "cbc",
    initializationvector: "1234567890123456"
};

var myEncryptedText = kony.crypto.encrypt(algo, encryptDecryptKey,
    inputstr, prptobj);
```

Example for Windows Phone and Kiosk

```
var algo = "rsa";
var encryptDecryptKey = kony.crypto.retrievePublicKey("rsa", "sample.cer",
true);
var inputstr = "pleaseencryptme";

var prptobj = {
    padding: "pkcs1",
    mode: "ecb"
};

var myEncryptedText = kony.crypto.encrypt(algo, encryptDecryptKey, inputstr,
prptobj);

kony.print("********************************myEncryptedText
is::****************************");
```

```
kony.print(myEncryptedText);
alert("Encrypted Text : " + myEncryptedText);
```

Example for Windows 8

```
var algo = "rsa";
var inputstr = "pleaseencryptme";

var prptobj = {
    padding: "",
    mode: ""
};

var myEncryptedText = kony.crypto.encrypt(algo, "sample.cer", inputstr,
prptobj);

kony.print("********************************myEncryptedText
is::****************************");
kony.print(myEncryptedText);
alert("Encrypted Text : " + myEncryptedText);
```

```
Encrypt: function() {
    try {
        var algo = "aes";
        var inputstr = "";
        var encryptDecryptKey = kony.crypto.newKey("passphrase", 128, {
            passphrasetext: ["inputstring1"],
            subalgo: "aes",
            passphrasehashalgo: "md5"
        });

        if (this.view.textEncrypt.text === "" || this.view.textEncrypt.text
=== null) {
            this.view.lblEncrypt.text = "Please enter the text to encrypt";
            return;
        } else {
            inputstr = this.view.textEncrypt.text;
        }
        var prptobj = {
            padding: "pkcs5",
            mode: "cbc",
            initializationvector: "1234567890123456"
        };
        myEncryptedTextRaw = kony.crypto.encrypt(algo, encryptDecryptKey,
inputstr, prptobj);
        var myEncryptedText = kony.convertToBase64(myEncryptedTextRaw);

        //              if(kony.os.deviceInfo().name == "Windows 8" || kony.os.deviceInfo
().name == "WindowsPhone")
        //              {
        //                      this.view.lblEncrypt.text = "Encrypted text =
"+myEncryptedTextRaw.toString();
        //              }
        //              else
        //              {
        this.view.lblEncrypt.text = "Encrypted text = " +
myEncryptedText.toString();
        //              }

    } catch (err) {
        alert(typeof err);
        alert("Error in callbackEncryptAes : " + err);
    }
},
```

Return Values

rawbytes [Object] - userdata

The rawbytes for the encrypted version of the input text.

Exceptions

CryptoError: Thrown by Crypto API.Various error conditions related to CryptoError will be covered through the following error codes.

- 2001 - unsupported algorithm.
- 2002 - invalid key strength specified.
- 2003 - insufficient buffer provided for specified operation.
- 2004 - memory allocation failure.
- 2005 - input data did not encode or encrypt properly .
- 2006 - specified name already exists.
- 2007 - key with the specified unique ID is not found.

Platform Availability

Available on all platforms except J2ME.

kony.crypto.generateAsymmetricKeyPair

This API is used to generate public and private keys for encryption and decryption processes. Typically, you can use the Public key to verify the digital signature and plain text data, whereas you can use the Private key to create a digital signature and to decrypt the text. This API is available from V8 SP3 onwards.

Syntax

```
kony.crypto.generateAsymmetricKeyPair(propertiesTable)
```

Input Parameters

Parameters	Description
propertiesTable [Object] - Mandatory	A key-value pair that you can use to generate asymmetric key pairs. The following input values are applicable for this parameter: • **alias** (String) [Mandatory]: UTF-8 string. • **keysize** (number): Size of the key that is to be generated by using this API. • **cipher** [String]: The cipher algorithm to be used. The applicable value is RSA. • **publicexponent** [odd integer]: The recommended value is 65337. • **padding** [bytes]: For RSA algorithm, the possible padding modes are PKCS1, OAEP, and None. The recommended value is OAEP. The maximum byte lengths for the padding input value are as follows: • PKCS1: < b - 11 • OAEP: < b - 41 • None: < b • **mode** [String]: Block mode. The possible values are ECB and you can also pass an empty string to use the platform default mode. • **digest** [String]: The hashing algorithm to be used. The possible values are SHA-1, SHA-256, SHA-224, SHA-384, and SHA-512.

Example

```
var status = kony.crypto.generateAsymmetricKeyPair(propertiesTable);
```

```
generateAsymmetricKeyPair: function() {
    var isGenerated = kony.crypto.generateAsymmetricKeyPair({
        "alias": "Kony",
        "algo": "RSA",
        "padding": "PKCS1Padding",
        "cipher": "RSA",
        "mode": "ECB",
        "digest": "",
        "keysize": "2048",
        "publicexponent": 3
    });
    alert("The Generated Key is " + isGenerated);
},
```

Return Value [Boolean]

Status of the key value generation.

Limitations

- For iOS
  - The `publicexponent`, `padding`, `digest`, and `mode` attributes are not considered for key generation.
  - This API is supported on devices with iOS 10.0 or later.
- For Android
  - This API does not work on devices with API level earlier than 18.
  - The following values are supported for `publicexponent` attribute: 3 and 65537.
  - The following values are supported for `digest` attribute: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512.
    If the `digest` attribute is not required, use an empty string as the digest value.
  - The following value is supported for the `mode`: ECB.
  - You cannot provide the value `None` to the `padding` attribute.
- For Windows
  - You cannot provide the value `None` to the `padding` attribute.
  - The `publicexponent` and `mode` are not considered while key generation.

- SHA-224 is not supported.
- The following bits values are supported for the `keysize` attribute: 512, 1024 , 2048, and 4096.

Platform Availability

- iOS
- Android
- Windows 10
- Windows Desktop

kony.crypto.generateSecureRandom

This API is used to generate cryptographically secure random numbers. This API is available from V8 SP3 onwards.

Syntax

```
kony.crypto.generateSecureRandom(propertiesTable)
```

Input Parameters

Parameters	Description
propertiesTable [Object] - Mandatory	A key-value pair that you can use to send the type and size of the key, in order to generate secure random cryptographic numbers.

- **type** [String]: The possible values for the type key are 'bytes' and 'Base64.'
- **size** [bytes]: The length of the random key to be generated.

Example

```
kony.crypto.generateSecureRandom({
    type:"bytes",
    size: < length >
});
```

Return Value [Object]

Secure random key of the bytes array or Base64 string of the specified length.

Limitations

- For Android
    - This API does not work on devices with API level earlier than 18.

Platform Availability

- Android
- iOS
- Windows 10
- Windows Desktop

---

kony.crypto.newKey

This API allows you to create a key for cryptography using the specified algorithm. The key created using this API is used for encrypting clear text and decrypting the encrypted data.

Use Case

You can use this API to generate cryptographic keys when you want to transmit information in a secured manner over the private or public networks.

Syntax

```
kony.crypto.newKey(algo, keystrength, propertiesTable)
```

Input Parameters

Parameters	Description
algo [String] - Mandatory	Scheme using which the key is to be created. Possible values are: • *securerandom* - uses a secure random number as the scheme to generate a key. This scheme always produces a unique key. • *random* - uses a random number as the scheme to generate a key. This scheme always produces a unique key. ***Note:*** *random* and *securerandom* are supported only on iPhone. ***Note:*** There is no differentiation between *securerandom* and *random* on Android. • *passphrase* - if this is the scheme, you need to pass the exact passphrase using which the key needs to be generated. The *passphrasetext* (an array of strings) is passed in the *properties* (JavaScript) parameter. The *passphrase* scheme always produces the same key for the same passphrase text. ***Note:*** Only Passphrase is supported on SPA.

Parameters	Description
keystrength [Number] - Mandatory	Number of bits that indicate the key strength. If the *subalgo* is: ***aes*** - possible value is 128, 192, 256. ***tripledes*** - possible value is 192. *Important:* • *tripledes* Algorithm is not supported in Windows Platforms. • *tripledes* - In Android and iOS, if the supplied key length is not equal to 192 an exception will be thrown with error message `Invalid Keystrength` and error code `104`. • On iPhone platform, keystrength of 192 is supported only if the algorithm is *random* or *securerandom*. You cannot apply a *passphrasehashlogo* to the key when the algorithm is *random* or *securerandom*. • As Android has deprecated support for various BouncyCastle implementations, the kony.crypto.newKey API that uses the tripledes algorithm with the keystrength of 192 is not supported on Android 12 devices. To use the kony.crypto.newKey API with the tripleDES algorithm and the keystrength of 192 on Android 12, enable the **useExternalBouncyCastleLibrary** property in the androidbuild.properties file.

Parameters	Description
propertiesTable [Table] - Mandatory	• *passphrasetext* [Array of Strings]- the exact passphrase using which the key needs to be generated if the scheme is *passphrase*. **Note:** This value in the table is mandatory only if the scheme is *passphrase*. If the subalgo is aes, it contains a single string, whereas if the subalgo is tripledes, it contains three strings. For example: ◦ for *aes*, passphrasetext = {"inputstring1"} ◦ for *tripledes*, passphrasetext = ["TestStr1","TestStr2","TestStr3"] **Note:** *passphrase* should contain at least 3 characters (24 bytes), else the API throws an *illegalargument* exception. **Note:** *tripledes* - in Android, if the passphrase length is less than 24 bytes or greater than 24 bytes an exception will be thrown with error message Invalid Keystrength and error code 104. • *subalgo* - represents the key algorithm that is used to create the key. This is a mandatory parameter (irrespective of the scheme). Possible values are: *aes* and *tripledes*. • *passphrasehashlogo* - hashing algorithm to be applied for the passphrase text. (applicable only on iPhone). **Note:** This value in the table is applicable only if the scheme is *passphrase*. Possible values for the hash algorithm are: ◦ *md2* (for key strength of 128) ◦ *md4* (for key strength of 128) ◦ *md5* (for key strength of 128) ◦ *sha2* (for key strength of 256)

Example

```
var myEncryptionKey = kony.crypto.newKey("passphrase", 192, {
    passphrasetext: ["TestStr1", "TestStr2", "TestStr3"],
    subalgo: "tripledes"
});
```

```
createNewKey: function() {
    newKey = kony.crypto.newKey("passphrase", 128, {
        passphrasetext: ["inputstring1"],
        subalgo: "aes",
        passphrasehashalgo: "md2"
    });
    this.view.lblKey.text = JSON.stringify(newKey);
},
```

Exceptions

CryptoError: Thrown by Crypto API.Various error conditions related to CryptoError will be covered through the following error codes.

- 2001 - unsupported algorithm.
- 2002 - invalid key strength specified.
- 2003 - insufficient buffer provided for specified operation.
- 2004 - memory allocation failure.
- 2005 - input data did not encode or encrypt properly .
- 2006 - specified name already exists.
- 2007 - key with the specified unique ID is not found.

Return Values

The following are the return values for this API:

key [userdata] - object

The key that is created using the specified algorithm.

API Usage

The recommended key strengths are as follows for this API:

- *aes* - 128
- *tripledes* - 192.

Platform Availability

Available on all platforms except J2ME.

kony.crypto.readKey

This API provides you the ability to read the key from the device store.

> ***Note:*** From V8 SP4 onwards, the readKey data for a Kony Visualizer App Viewer child app is stored in child app data and not under the parent app. This feature is applicable for iOS, Windows, and Android platforms.

> ***Note:*** Device store in case of iOS is Keychain. Keychain in iOS is the most secured place to store the crypto keys. saveKey and readKey APIs save and read from the Keychain. The Keychain can be shared between the applications provisioned and signed by the same certificate vendor.

> ***Important:*** To avoid accidental overwrite of one application content by the other application content, it is recommended to use the unique application specific identifier while saving and reading the crypto keys using saveKey and readKey APIs.

Use Cases

You can read the key from the device store if you want to use that key for encryption or decryption.

Syntax

```
kony.crypto.readKey(uniqueID)
```

Input Parameters

Parameters	Description
uniqueID [String] - Mandatory	Unique ID represents the key on the device store (this is the ID returned by kony.crypto.saveKey API).

Example

```
var myUniqueIDKey = kony.crypto.saveKey
("myKey",encryptDecryptKey,constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_
UNLOCKED);
var myCryptoKey = kony.crypto.readKey(myUniqueIDKey,constants.KONY_KEYCHAIN_
ITEM_ACCESSIBLE_WHEN_UNLOCKED);
```

```
readKey: function() {
    var read = kony.crypto.readKey(saveKey);
    this.view.lblKey.text = JSON.stringify(read);
}
```

The **constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED** parameter is an optional parameter. It indicates when a keychain item is accessible.

The following values are supported:

- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED : The data in the keychain item can be accessed when a device is unlocked by the user.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED_THIS_DEVICE_ONLY: The data in the keychain item can be accessed only when a specific device is unlocked by the user
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_ALWAYS_THIS_DEVICE_ONLY: The data in the keychain item can always be accessed regardless of whether a specific device is locked.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_PASSCODE_SET_THIS_DEVICE_ONLY: The data in the keychain can only be accessed when the device is unlocked. This is only available if a passcode is set on the device.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_ALWAYS: The data in the keychain item can always be accessed regardless of whether a device is locked.

- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_AFTER_FIRST_UNLOCK: The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_AFTER_FIRST_UNLOCK_THIS_DEVICE_ONLY: The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.

Return Values

The following are the return values for this API:

key [rawbytes - object]

This key is generated using aes, tripledes, or RSA algorithms and saved on the device store.

API Usage

You can use this API only to read the keys that you have saved earlier on the device store, i.e., keys that have a unique ID associated with them.

Exceptions

CryptoError: Thrown by Crypto API.Various error conditions related to CryptoError will be covered through the following error codes.

- 2001 - unsupported algorithm.
- 2002 - invalid key strength specified.
- 2003 - insufficient buffer provided for specified operation.
- 2004 - memory allocation failure.
- 2005 - input data did not encode or encrypt properly.
- 2006 - specified name already exists.
- 2007 - key with the specified unique ID is not found.

Platform Availability

Available on all platforms except J2ME and Windows Kiosk.

kony.crypto.retrieveAsymmetricPublicKey

This API returns the public key for the alias that you provide. This API is available from V8 SP3 onwards.

Syntax

```
kony.crypto.retrieveAsymmetricPublicKey(alias)
```

Input Parameters

Parameters	Description
alias [String]	The alias value generated by using generateAsymmetricKeyPair API.

Example

```
var alias = kony.crypto.retrieveAsymmetricPublicKey(alias);
```

```
retrieveAsymmetricKey: function() {
    var key = kony.crypto.retrieveAsymmetricPublicKey("Kony");
    alert("The Asymmetric key is " + key);
}
```

Return Value [String]

Returns the public part of the asymmetric key-pair for the provided alias.

Limitations

- For iOS
  - This API works on devices with iOS 10 or later.
- For Android
  - This API does not work on devices with API level earlier than 18.

Platform Availability

- iOS
- Android

- Windows 10
- Windows Desktop

kony.crypto.retrievePublicKey

Public Key Infrastructure (PKI) is the mechanism to secure the public networks (like Internet) to safely and securely transmit data with the use of keys. PKI assumes the use of public key cryptography (asymmetric cryptography). PKI is the most common method to authenticate the message sender or encrypt the message. PKI consists of a Certificate Authority (CA) that issues and verifies digital certificates (trusted certificates). A certificate includes the public key or information about the public key.

Note: Due to security reasons, Thin Client or Mobile Web applications cannot access public/private keys or certificates that are on the server.

This API provides the ability to extract the public key from a base64 string of encoded X509 certificate or a locally packaged X509 certificate.

Syntax

```
kony.crypto.retrievePublicKey(algo, inputsource, islocalresource)
```

Input Parameters

Parameters	Description
algo [String] - Mandatory	The algorithm used for the public key. Possible values are: • RSA • AES - Supported only on Windows platforms.

Parameters	Description
inputsource [String] - Mandatory	This parameter indicates the name of the input source certificate from which the key needs to be retrieved. ***Note:*** The certificate must be present in the resources folder. ***Note:*** In case of the Android platform, place the .cer file at the `../resources/mobile/native/android/assets/` location.
islocalresource [Boolean] - Mandatory	This flag defines how the inputsource string needs to be interpreted. • islocalresource is **false** - represents that the input source is base64 string of X509 certificate. • islocalresource is **true** - represents that the input source is name of the local resource for the certificate. For example, *public.cer*.

Example

```
var myKey = kony.crypto.retrievePublicKey("rsa", "public.cer", true);
```

Return Values

publickey - userdata [Object]

The public key extracted from the certificate.

Rules and Restrictions

- Self-signed certificates are not supported on Android.
- iOS supports only Distinguished Encoding Rules (DER) representation of an X.509 certificate, when input source is certificate.

Exceptions

CryptoError: Thrown by Crypto API.Various error conditions related to CryptoError will be covered through the following error codes.

- 2001 - unsupported algorithm.
- 2002 - invalid key strength specified.
- 2003 - insufficient buffer provided for specified operation.
- 2004 - memory allocation failure.
- 2005 - input data did not encode or encrypt properly.
- 2006 - specified name already exists.
- 2007 - key with the specified unique ID is not found.

Platform Availability

Available on all platforms except J2ME, Windows 8 Tablet, Windows Kiosk, Service Side Mobile Web, Desktop Web, and SPA.

kony.crypto.saveKey

This function allows your app to save a generated key on the device's storage.

Note: From V8 SP4 onwards, the saveKey data for a Kony Visualizer App Viewer child app is stored in child app data and not under the parent app. This feature is applicable for iOS, Windows, and Android platforms.

Syntax

```
kony.crypto.saveKey(
    name,
    key)
```

Input Parameters

Parameters	Description
name	A string that specifies a unique name with which you want to save the key on the device store.

Parameters	Description
key	An object that holds the key that you want to save on the device.

Example

```
var myUniqueKey = kony.crypto.saveKey("myKey",myEncryptionKey,constants.KONY_
KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED);
```

```
saveTheKey: function() {
    saveKey = kony.crypto.saveKey("SavedKey", newKey, constants.KONY_KEYCHAIN_
ITEM_ACCESSIBLE_WHEN_UNLOCKED);
    this.view.lblKey.text = "The Key is Saved";
},
```

The **constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED** parameter is an optional parameter. It indicates when a keychain item is accessible.

The following values are supported:

- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED : The data in the keychain item can be accessed when a device is unlocked by the user.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED_THIS_DEVICE_ONLY: The data in the keychain item can be accessed only when a specific device is unlocked by the user.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_ALWAYS_THIS_DEVICE_ONLY: The data in the keychain item can always be accessed regardless of whether a specific device is locked.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_PASSCODE_SET_THIS_DEVICE_ONLY: The data in the keychain can only be accessed when the device is unlocked. This is only available if a passcode is set on the device.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_ALWAYS: The data in the keychain item can always be accessed regardless of whether a device is locked.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_AFTER_FIRST_UNLOCK: The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.

- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_AFTER_FIRST_UNLOCK_THIS_DEVICE_ONLY: The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.

Return Values

Returns a string containing a unique ID that represents the saved key on the device's storage. Your app can access the key from the device's storage using this unique ID. The unique ID is determined by the system. On some platforms it might be the same as the name in the name parameter. However, that is not the case on all platforms.

Exceptions

CryptoError: Thrown by Crypto API.Various error conditions related to CryptoError will be covered through the following error codes.

Constant	Description
2001	The encryption algorithm is unsupported on the device.
2002	An invalid key length was specified.
2003	Insufficient buffer space was provided for operation.
2004	There was a memory allocation failure.
2005	The input data did not encode or encrypt properly.
2006	The specified name already exists

Constant	Description
2007	A key with the specified unique ID is not found.

Remarks

Your app can use this function to save the generated symmetric keys. If a key does not exist with the given name, this function creates a key. If a key exists with the given name, this function saves the key onto the device's storage.

The device store on iOS is the keychain. The keychain on iOS is the most secure place to store the cryptographic keys. saveKey and readKey APIs save to and read from the keychain. The keychain can be shared between applications that are provisioned and signed by the same certificate vendor.

In Android, the kony.crypto.saveKey saves the crypto key in the application's private file system. This crypto key is encrypted.

Important: To avoid accidentally overwriting one application's keys by another application, Kony recommends that your app use a unique application-specific identifier while saving and reading keys.

Platform Availability

Available on all platforms except J2ME and Windows Kiosk.

22. Drag and Drop API

Using the Drag and Drop API, you can enable the option to drag images, text, and files from one application to another. To drag and drop the files between two applications, ensure that both the applications are open using the app switcher or the Split-View multitasking window.

Important: The Drag and Drop API is applicable to ipad only. The feature is introduced in iOS 11.

Drag and Drop is a new feature introduced in iOS 11. You can use the Drag and Drop API to share data, such as, images, text (by using JSON), and files, between different apps by dragging items from an app and dropping it into another app.

Important: This API is applicable on ipad only.

The Drag and Drop API contains the `kony.dragdrop Namespace` and the following API elements.

Function	Description
`kony.dragdrop.DragInteraction`	Creates a new DragInteraction object and attaches the DragInteraction object to a widget.
`kony.dragdrop.DropInteraction`	Creates a new DropInteraction object and attaches it to a widget.
`kony.dragdrop.removeDragInteraction`	Detaches the DragInteraction object from the widget.
`kony.dragdrop.removeDropInteraction`	Detaches the DropInteraction object from the widget.

To enable the drag feature on a widget in an application, create a **dragInteraction** object and attach the object to the widget by using the `kony.dragdrop.DragInteraction` function. You can enable the drop feature foe a widget by using the `kony.dragdrop.DropInteraction` function. If you want to remove the Drag and Drop functionalities from the widget, use the `kony.dragdrop.removeDragInteraction` and `kony.dragdrop.removeDropInteraction` functions.

To view the functionality of the Drag and Drop API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

22.1 kony.dragDrop Namespace

The kony.dragdrop Namespace contains the following API elements.

22.1.1 Functions

The kony.dragdrop Namespace contains the following functions.

22.1.2 kony.dragdrop.DragInteraction

This function creates a new DragInteraction object and attaches the DragInteraction object to a widget.

Syntax

```
kony.dragdrop.DragInteraction()
```

Input Parameters

Parameters	Description
widget	The widget that has to be made draggable.
callbacks	This is a dictionary with the following keys: • **itemsForBeggining**(JavaScript function) [Mandatory] : In this JS function, you can return the data (dictionary of the supported format), which is to be sent (JSON/filepath). • **previewForLifting** (JavaScript function) [Mandatory]: This callback should return the preview that is to be shown while dragging on the screen. • **onLiftBeginAnimate** (JavaScript function) [Optional]: Any Kony-supported animations that can be done when the lift just started. • **onLiftBeginAnimationComplete** (JavaScript function) [Optional]: Animations that can be configured after the lift animations have been completed. • **previewForCancel** (JavaScript function) [Optional]: This callback returns the preview when the dragging action is canceled. • **onCancelAnimate** (JavaScript function) [Optional]: Any Kony-supported animations that can be done when the dragging is canceled. • **onCancelAnimationComplete** (JavaScript function) [Optional]: Any Kony-supported animations that can be done after the cancellation process has been completed.

Example

```
var callbacksDict = {
  "itemsForBeggining": itemsForBegginingCallback,
  "previewForLifting": fnPreviewForLifting,
  "onLiftBeginAnimate": fnLiftBeginAnimation,
  "onLiftBeginAnimationComplete": fnLiftBeginAnimationComplete,
  "onCancelAnimate": fnCancelAnimation,
  "onCancelAnimationComplete": fnCancelAnimationComplete,
  "previewForCancel": fnPreviewForCancel
};

var argsDict = {
 "widget": Form1.flx1.flx2.diamondImg,
 "callbacks": callbacksDict
};

var dragInteractionImg1 = new kony.dragdrop.DragInteraction(argsDict);
```

```
 //To create a drag interation object use the below code snippet
createcallbacksdictAndAddDragInteractionImg1: function() {
    var callbacksDict = {
        "itemsForBeggining": this.beginItemDragForm9,
        "previewForLifting": this.previewForLiftingForm9
    };
    var argsDict = {
        "widget": this.view.img1,
        "callbacks": callbacksDict
    };
    this.dragInteraction1 = new kony.dragdrop.DragInteraction
(argsDict);

},
/*   By specifying the below function, you can return the data, which
is to be sent
  (JSON/filepath).*/
beginItemDragForm9: function() {
    var applicationDirPath =
kony.io.FileSystem.getApplicationDirectoryPath();
    var filePath = applicationDirPath + "/puppy.png";
    var argsDict = {
        "data": filePath,
        "type": kony.dragdrop.ITEMDATA_FILE,
        "fileVisibility": kony.dragdrop.FILEVISIBILITY_ALL
    };
    return [argsDict];
},

/*By specifying the below function, you can return the preview that is
to be shown while
dragging on the screen.*/
previewForLiftingForm9: function() {
    var argsDict = {
        "preview": this.view.img1
    };
    return argsDict;
},
```

Return Values

- Success: A newly created DragInteraction object.
- Failure: null

Platform Availability

- iOS 11

22.1.3 kony.dragdrop.DropInteraction

This function creates a new DropInteraction object and attaches it to a widget.

Syntax

```
kony.dragdrop.DropInteraction()
```

Input Parameters

Parameters	Description
widget	The widget that has to be made droppable.
callbacks	This is a JSON dictionary with the following keys: • **performDrop**(JavaScript function) [Mandatory] : This JS function gets data (JSON/filepath) as an argument. • **previewForDrop** (JavaScript function) [Optional]: The drop preview that can be configured while dropping the data with the default preview. • **onBeginDropAnimate** (JavaScript function) [Optional]: Any Kony-supported animations that can be configured while dropping. • **onBeginDropAnimationComplete** (JavaScript function) [Optional]: Animations that can be configured after the drop animations have been completed. • **concludeDrop** (JavaScript function) [Optional]: This callback is invoked after the dropping action is successfully completed.

Example

```
var dict = {
  "performDrop": fnDropCallback,
  "onBeginDropAnimate": fnOnDropAnimate,
  "onBeginDropAnimationComplete": FnOnDropAnimateComplete,
  "previewForDrop": fnPreviewForDrop,
  "concludeDrop": fnConcludeDrop
};

var argsDict = {
 "widget": Form1.flx2,
 "callbacks": dict
};

var dropInteraction = new kony.dragdrop.DropInteraction(argsDict);
```

```
 //To create a drop interation object use the below code snippet
createcallbacksdictAndAddDropInteractionImg2: function() {
    var callbacksDict = {

        "performDrop": this.dropCallbackForm9
    };
    var argsDict = {
        "widget": this.view.img2,
        "callbacks": callbacksDict
    };
    this.drpInteraction1 = new kony.dragdrop.DropInteraction
(argsDict);
},

/*By specifying the below function, you can get the data
(JSON/filepath) as an argument*/
dropCallbackForm9: function(recievedData) {
    var file = kony.io.FileSystem.getFile(recievedData);
    if (file.exists()) {
        var fileBytes = file.read();
        this.view.img2.base64 = kony.convertToBase64(fileBytes);
    }
},
```

Return Values

- Success: A newly created DropInteraction object.
- Failure: null

Platform Availability

- iOS 11

22.1.4 kony.dragdrop.removeDragInteraction

This function detaches the DragInteraction object from the widget.

Syntax

```
kony.dragInteraction.removeDragInteraction()
```

Input Parameters

None

Example

```
var dragInteraction = new kony.dragdrop.DragInteraction(argsDict);
dragInteraction.removeDragInteraction();
```

```
 removeDragInteraction1: function() {
        this.dragInteraction1.removeDragInteraction();
    },
```

Return Values

None

Platform Availability

- iOS 11

22.1.5 kony.dragdrop.removeDropInteraction

This function detaches the DropInteraction object from the widget.

Syntax

```
kony.dragInteraction.removeDropInteraction()
```

Input Parameters

None

Example

```
var dropInteraction = new kony.dragdrop.DropInteraction(argsDict);
dropInteraction.removeDropInteraction();
```

```
 removeDropInteraction1: function() {
        this.drpInteraction1.removeDropInteraction();
    }
```

Return Values

None

Platform Availability

- iOS 11

---

23. ForceTouch API

Force Touch API enables your applications to support 3D Touch features on iOS and Android devices. Using the Force Touch API, a device can sense the amount of pressure you apply on the display, and trigger different actions. You need not explicitly open an app to perform an action. For iOS, you can add a list of quick actions and for Android, you can add app shortcuts that you want the app to display when you hold and swipe the app on a device's home screen.

The Force Touch API uses `kony.forceTouch Namespace` and the following API elements.

Function	Description
`kony.forcetouch.disableQuickActionItems`	Disables pinned shortcuts that were previously enabled.
`kony.forcetouch.enableQuickActionItems`	Enables pinned shortcuts that were previously disabled.
`kony.forcetouch.getPinnedQuickActionItems`	Returns an ID list of all the pinned Quick Action items.
`kony.forcetouch.getQuickActionItems`	Retrieves an array of dynamic Quick Action items that are set in the app.
`kony.forcetouch.getStaticQuickActionItems`	Returns an array of immutable or static Quick Action items that are set in the app.
`kony.forcetouch.removeQuickActionItems`	Removes all the dynamic Quick Action items that are set in the app.

Function	Description
`kony.forcetouch.removeQuickActionItems`	Removes the array of specified dynamic Quick Action items.
`kony.forcetouch.setQuickActionItems`	Sets dynamic Quick Action items in the app.

Quick actions can be static or dynamic. To configure dynamic quick actions that you want the app to display when you hold and swipe the app icon, use the `kony.forcetouch.setQuickActionItems` function. These quick actions are visible during run time, that is when you launch an app. Then you can find the list of all the quick actions that are created dynamically by using the `kony.forcetouch.getQuickActionItems` function. Futher, find the list of quick actions that are set at compile time in Kony Visualizer by using the `kony.forcetouch.getStaticQuickActionAItems` function. These quick actions are set at build time and are visible when you install or update an app.

The quick actions can be pinned to the app icon by using the `kony.forcetouch.enableQuickActionItems` function. After you enable the quick actions, you can view the list of all the pinned quick actions by using the `kony.forcetouch.getPinnedActionItems` function. In case you want to unpin any quick action item, you can disable the pinned shortcut by using the `kony.forcetouch.disableQuickActionItems`. To delete the quick action, use the `kony.forcetouch.removeQuickActionItems` function.

23.1 Overview

The ForceTouch API provides functions to support 3D Touch features. Devices that support 3D Touch can sense how much pressure the user applies to the display. The ForceTouch API supports the following features.

- Quick Actions
- App Shortcuts
- Peek and Pop
- Force Properties

23.1.1 Quick Actions

Quick Actions enables users to select actions from the home screen with a single touch. Quick Actions can display one line of a title, one line of subtitle, and an optional icon. For example, a user can create a Quick Action to start a message to a contact, or bring up the camera for a selfie.

There are two types of Quick Actions:

- **Static Quick Actions** - Static Quick Actions must be configured in Kony Visualizer.
- **Dynamic Quick Actions** - Dynamic Quick Actions are dynamically created using the APIs of the kony.forcetouch namespace.

A maximum of four Quick Actions, both dynamic and static, are displayed on the home screen. The static items are displayed first, then the dynamic items.

In addition, each widget supports the following methods for 3D Touch support.

- <widget>.registerForPeekAndPop
- <widget>.unregisterForPeekAndPop
- <widget>.setOnPeek
- <widget>.setOnPop
- <flexformwidget>.setPreviewActionItems

Here <widget> is the name of the widget, such as Button or Label, and <flexformwidget> is the name of the Flex Form widget, such as FlexForm or FlexScrollContainer. For more information about the methods, refer Kony Widget User Guide.

For more information about 3D Touch and Quick Actions, see the Apple documentation.

23.1.1.1 Handling Quick Actions

Quick actions are processed in a callback function. You can set this callback function in your app by invoking the kony.application.setApplicationInitializationEvents function. The `kony.application.setApplicationInitializationEvents` function takes one parameter that is a JavaScript object containing key-value pairs. Each of the keys specifies a different callback function, and each of the values are the callback function to set.

The callback function that your app uses to process force touch events is associated with the `appservice` key. The event handler for the `appservice` key is typically referred to as the appservice event handler. When the appservice event handler is invoked, its parameter list receives the launch mode and launch parameters as arguments. With these, your application can identify when a quick action has been invoked by the users and which action was triggered. The appservice event handler's parameter is a JavaScript object that holds key-value pairs. The keys for accessing the launch mode and launch parameters are "launchmode" and "launchparams", respectively.

The following sample code illustrates how an appservice event handler function that processes quick actions might look.

```
function onAppServiceCallback(params) {
    alert("launchoptions: " + JSON.stringify(params));

    // If launch mode = 3 and quickactionitem key present in
launchparams
    // denotes quick action item launch.
    if (params.launchmode == 3) {
        var quickActionItem = params.launchparams.quickactionitem;
        if (quickActionItem) {
            if (quickActionItem.id == "firstform") {
                return frmFirst;
            } else if (quickActionItem.id == "mapviewform") {
                return frmSecond;
            } else if (quickActionItem.id == "basket") {
                return frmThird;
            } else if (quickActionItem.id == "AccountDetails") {
                return frmFourth;
            }
        }
    }
}
```

As the sample code shows, your app must retrieve the launch mode to determine whether or not the user triggered a quick action. Once it has done so, it can determine which quick action was triggered and react accordingly.

> ***Note:*** The `appservice` function returns a form handle based on the Quick Action item it receives. Do not invoke the `form.show` method from the `appservice` callback.

23.1.1.2 Quick Actions and Internationalization

To support the internationalization of quick actions, your app must supply some code in the callback function for the `preappinit` event. Your app sets the callback function to the `preappinit` event by calling the kony.application.setApplicationInitializationEvents function. You app passes the `kony.application.setApplicationInitializationEvents` function a JavaScript object with key-value pairs. The `"preappinit"` key sets the callback function for the `preappinit` event. The following is an example of what your callback function might look like.

```
 preAppInt(){
    var previousLocale = kony.store.getItem("applinkLocale")
    var currentLocale = kony.i18n.getCurrentLocale();

    if (previousLocale != null && previousLocale != currentLocale) {
        // Update the dynamic shortcuts using api.
        var quicknotificationitems =
getApplicationQuickNotificationItems();
        if(quicknotificationitems!=null) {
            kony.forcetouch.setQuickActionItems
(quicknotificationitems);
        }
    } else {
        kony.store.setItem(applinkLocale)
    }
}
```

The example above shows that the callback function for the `preappinit` event can check the local of the user's device and update the locale strings for the quick action shortcuts. In this way, you can internationalize your app.

23.1.2 Peek and Pop

Apps that support the Peek ForceTouch action can indicate to the user that a preview of app content is available. Pressing on the app's icon then opens the preview, which is termed *peeking*. If the user continues to press, the app navigates to the view shown in the preview. The action of navigating to the view is called a *pop*.

Peek and Pop are implemented through FlexForm widgets. For more information, please refer to the Kony Widgets Programmer's Guide.

23.1.3 Force Properties

Apps can use the extended pressure-sensing capabilities of the iPhone 6s and iPhone 6s Plus display. The set of touch events supported on the widgets are onTouchStart, onTouchMove, onTouchEnd. These events include contextInfo table which includes force key-value pair on devices supporting force touch.

Example

```
contextInfo [“force”] : number

onTouchStart (source, x, y, contextInfo){
       if(contextInfo){
```

```
var force = contextInfo["force"];
                kony.print("value of force is " + force)
}
}
```

The predetermined value of force for an average touch is 1.0

23.1.4 Application Shortcuts

The Application shortcut feature is available on Android from API level 25 and later. Similar to quick actions on iOS, when you long press an app icon in Android, a list of shortcuts appear. Using the app shortcuts, you can directly navigate to a specific page in the app. You can drag a shortcut to the phone launcher.

Android supports two types of Application Shortcuts:

- Static Shortcuts
- Dynamic Shortcuts

23.1.4.1 Static Shortcuts

Static shortcut properties need to be declared in an XML file and the values of the properties or attributes for the shortcut XML must be declared in Android Resources. No direct strings are allowed or accepted by Android.

You can create static shortcuts manually by adding the XML files in their respective folders and then updating the manifest entries from IDE.

Creating Static Shortcuts Manually

1. Create an shortcut xml file as shown in the example below, and copy the file in the created directory in XML: \<application_resoruce_dir>\resources\mobile\native\android location.
2. Include a meta-data tag to the main launcher activity for the child tag of a manifest file in project settings of IDE.

Example

```
<?xml version="1.0" encoding="utf-8"?>
<shortcuts xmlns:android="http://schemas.android.com/apk/res/android">
<shortcut
android:shortcutId="shortcutid"
android:enabled="true/false"
android:icon="@drawable/icon"
android:shortcutShortLabel="@string/myFirstshortcut"
android:shortcutLongLabel="@string/MY_FIRST_SHORTCUT"
android:shortcutDisabledMessage="@string/shortcut_is_disabled">
<intent
android:action="applicationpackage.appshortcut.formid"
android:targetPackage="applicationpackage"
android:targetClass="applicationpackage.appname" >
<extra android:name="platform" android:value="android" />
<extra android:name="feature” android:value="appshortcut" />
<extra android:name="id” android:value="formidentifier" />
</intent>
</shortcut>
</shortcuts>
```

Note: Provide some value for the action tag as some constant followed by "applicationpackage.appshortcut" because this is used as an appshortcut identifier.

An extra attribute is used to get key-value pairs that are sent by the quicknotification Object on AppService callback. You must pass "id" and its corresponding Identifier so that the program logic becomes easy to understand in order for the correct form to be shown.

Properties Naming Example

```
<resources>
<string name="title_activity_shortcut_1">shortcut_1</string>
<string name="myFirstshortcut">myFirstshortcut</string>
```

```
<string name="MY_FIRST_SHORTCUT">shortcut1</string>
<string name="shortcut_is_disabled">shortcut is disabled</string>
</resources>
```

Manifest File Changes

```
<meta-data android:name="android.app.shortcuts"
android:resource="@xml/shortcutxml_file_name" />
```

Shortcut Properties

- shortcutId: Mandatory. Identifier for the shortcut.
- Enabled: Default value is true. Set to false if you want to disable for subsequent updates.
- Icon: Image placed in drawable. If not provided, the default Android icon is placed.
- shortcutShortcutLabel: Mandatory. Short message displayed on the tile of the shortcut.
- shortcutLongLabel: Mandatory. Long descriptive message about the shortcut, it is shown if the required space is available.
- shortcutDisabledMessage: Message that is displayed on the launch of a disabled shortcut, which is pinned.
- intent tag: Standard method of declaration. Whereas, extra tags can be anything user-defined and is delivered to the application developer via luatable values in the callback.
- Rank: Arranges items in the shortcut tray on long press of the application.

Note: The XML file name should be “shortcut” and the array of shortcut attributes to shortcuts on multiple shortcuts must be declared.

For the application icon, follow these rules: https://commondatastorage.googleapis.com/androiddevelopers/shareables/design/app-shortcuts-design-guidelines.pdf

23.1.4.2 Dynamic Shortcuts

Dynamic shortcuts allow you to dynamically add, update, and reorder the shortcuts to the application.

To learn more about handling an app shortcut, refer Handling Quick actions. To learn about internationalization support for app shortcuts, refer Quick actions and Internationalization.

To view the functionality of the ForceTouch API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

23.2 kony.forcetouch Namespace

The kony.forcetouch namespace provides the functions to add, remove, enable, disable, and access dynamic Quick Action items. It contains the following API elements.

23.2.1 Functions

The `kony.forcetouch` namespace contains the following functions.

kony.forcetouch.disableQuickActionItems

The `disableQuickActionItems`function disables pinned shortcuts that were previously enabled. If the target shortcuts are already disabled, this function does not take any action. It sets the mentioned disabled message for all Quick Action items. When you try to launch the disabled actionItem shortcut, a relevant toast message is displayed. You should use this function for dynamic shortcut Quick Action items, and you must not call the function with static quickAction IDs.

Syntax

```
disableQuickActionItems(Object quickActionList, String disableMessage)
```

Input Parameters

An object of quickAction ID and the disable message.

Example 1

```
var shortcuts = {
    "com.kony.first1": "disable1",
    "com.kony.first.imageName": "your shortcut disabled2"
};
kony.forcetouch.disableQuickActionItems(shortcuts, "Disabled all Shortcuts");
```

Note: The above example disables all shortcut IDs (com.kony.first1 and com.kony.first.imageName) with the common disable message as "Disabled all Shortcuts".

Example 2

```
kony.forcetouch.disableQuickActionItems(shortcuts);
```

Note: This example disables the shortcuts mentioned above with their respective disable messages.

Return Values

None.

Platform Availability

- Android 7.1 and later or API Level 25 or later

kony.forcetouch.enableQuickActionItems

The `enableQuickActionItems`function enables pinned shortcuts that were previously disabled. If the target shortcuts were already enabled, this function does not take any action.

Syntax

```
enableQuickActionItems(Array quickActionId)
```

Input Parameters

An array of quickAction IDs.

Example

```
 var quickActionItems = [{
    "id": "com.kony.first",
    "title": "firstPage",
    "subtitle": "takes to first page",
    "icon": "option1.png",
    "info": {
        "feed": "feed to first form"
    }
}, {
    "id": "com.kony.second",
    "title": "secondPage",
    "subtitle": "takes to second page",
    "icon": "option1.png",
    "info": {
        "feed": "feed to second form"
    }
}];
kony.forcetouch.enableQuickActionItems(quickActionItems);
```

Return Values

None.

Platform Availability

- Android 7.1 and later or API Level 25 or later

kony.forcetouch.getPinnedQuickActionItems

The `getPinnedQuickActionItems` function returns an ID list of all the pinned Quick Action items. This function helps you pin a shortcut of any of the functional options of an app on the home screen of a mobile device.

Syntax

```
getPinnedQuickActionItems();
```

Input Parameters

None.

Example

```
var pinnedQuickActionArray=kony.forcetouch.getPinnedQuickActionItems();
alert("The pinned quick action Items are:"+pinnedQuickActionArray);
```

Return Values

An array of ID strings.

Platform Availability

- Android 7.1 and later or API Level 25 or later

kony.forcetouch.getQuickActionItems

The `getQuickActionItems`function gets an array of dynamic Quick Action items that are set in the app.

Syntax

```
kony.forcetouch.getQuickActionItems()
```

Example

```
try {
    var quickActionItems = kony.forcetouch.getQuickActionItems();
    alert(JSON.stringify(quickActionItems));
} catch (args) {
    alert(args.toString());
}
```

Input Parameters

None.

Return Values

A string array of dynamic Quick Action items containing only IDs, if any are set in the app. Otherwise, this function returns an empty array.

Remarks

The returned array does not contain icon key-value pairs.

Platform Availability

- iOS 9.0 and later
- Android 7.1 and later or API Level 25 or later

kony.forcetouch.getStaticQuickActionItems

The `getQuickActionItems`function returns an array of immutable or static Quick Action items that are set in the app.

Syntax

```
kony.forcetouch.getStaticQuickActionItems()
```

Example

```
var staticQuickActionArray = kony.forcetouch.getStaticQuickActionItems();
alert("The static quick action Items are:" + staticQuickActionArray);
```

Input Parameters

None.

Return Values

An array of ID strings.

Platform Availability

- iOS 9.0 and later
- Android 7.1 and later or API Level 25 or later

kony.forcetouch.removeQuickActionItems

The `removeQuickActionItems`function removes all the dynamic Quick Action items that are set in the app.

Syntax

```
kony.forcetouch.removeQuickActionItems()
```

Example

```
kony.forcetouch.removeQuickActionItems(); //removes all quick action items
```

Input Parameters

None.

Return Values

None.

Platform Availability

- iOS 9.0 and later
- Android 7.1 and later or API Level 25 or later

kony.forcetouch.removeQuickActionItems

The `removeQuickActionItems`function removes the array of specified dynamic Quick Action items.

Syntax

```
kony.forcetouch.removeQuickActionItems(
    Array ID);
```

Input Parameters

Parameter	Description
ID	An array of quickNotification IDs.

Example

```
var quickActionArray = kony.forcetouch.getQuickActionItems();
kony.forcetouch.removeQuickActionItems(quickActionArray);
```

Return Values

None.

Platform Availability

- iOS 9.0 and later
- Android 7.1 and later or API Level 25 or later

kony.forcetouch.setQuickActionItems

The `setQuickActionItems`function sets dynamic Quick Action items in the app.

Syntax

```
kony.forcetouch.setQuickActionItems(quickActionItems)
```

Input Parameters

quickActionItems - Mandatory.

An array of key-value pairs that contains the following values.

Key	Description
id	A string containing an app-defined id the home screen quick action. For example, com.appName.formName.
title	A string that holds a user-visible title of the home screen quick action. This value can be localized.

Key	Description
subtitle	A user-visible string that holds the subtitle of the home screen quick action. This value can be localized.
icon	An optional value that selects the icon for the action. For system icons, this value can be set to one of the Force Touch System Icon Constants. For iOS, if your app uses a custom icon that is included in the app's bundle, set this value to a string containing the filename of the icon. Also for iOS, if your app uses a contact's picture as its quick action icon, set this value to a ReferenceTable object for the contact that your app obtains by calling the kony.contact.find function. For Android, you can set the value of icon as either an image bundled with the app or as a Kony image Object.
info	A dictionary with developer-provided key-value pairs that contain app-defined information about the home screen quick action. This information is used by the app to implement the action.

Example 1

You can create quick actions with system-defined icons, a custom image in your app's bundle, or with a picture from a contact in the device. To select a custom image as a Quick Action's icon, your app must specify the icon's file name as value for the **icon** key in the **quickActionItems** of the **setQuickActionItems** function.

The following example shows how to system-defined icon as the icon for the Quick Action.

```
var quickActionItems = [{
    "id": "com.kony.first",
    "title": "firstPage",
    "subtitle": "takes to first page",
    "icon": kony.forcetouch.QUICK_ACTION_ICON_TYPE_COMPOSE,
    "info": {
        "feed": "feed to first form"
    }
}, {
    "id": "com.kony.second",
    "title": "secondPage",
    "subtitle": "takes to second page",
    "icon": kony.forcetouch.QUICK_ACTION_ICON_TYPE_HOME,
    "info": {
        "feed": "feed to second form"
    }
}];
var actionsSet = kony.forcetouch.setQuickActionItems(quickActionItems);
```

Example 2

The following example shows how to specify a custom image as the icon for the Quick Action and how to use the image from a contact as the icon for a Quick Action.

```
var quickActionItems = [{
    //Icon using custom image
    "id": "com.kony.first",
    "title": "firstPage",
    "subtitle": "Takes you to the first page",
    "icon": "customImage1",
    "info": {
        "feed": "Feed to the first form"
    }
}, {
    //Icon using contact picture
    "id": "com.kony.second",
    "title": "secondPage",
    "subtitle": "Takes you to second page",
    "icon": {
        "firstName": "John",
        "lastName": "Doe",
        "company": "Kony"
    },
    "info": {
        "feed": "Feed to the second form"
    }
}];
var actionsSet = kony.forcetouch.setQuickActionItems(quickActionItems);
```

Example 3

Note: For this example, you may get an alert on Visualizer to write the code in dot notation. Please ignore that alert and execute the provided code as is.

```
function onAppServiceCallback(params) {

    alert("launchoptions: " + JSON.stringify(params));

    //if launch mode = 3 &amp;&amp; quickactionitem key present in
launchparams

    //denotes quick action item launch

    if (params.launchmode == 3) {

        var quickActionItem = params.launchparams.quickactionitem;

        if (quickActionItem) {

            if (quickActionItem.id == "firstform") {

                return frmFirst;

            } else if (quickActionItem.id == "mapviewform") {

                return frmSecond;

            } else if (quickActionItem.id == "basket") {

                return frmThird;
            } else if (quickActionItem.id == "AccountDetails") {

                return frmFourth;

            }

        }

    }

}
```

Return Values

A Boolean value that is `true` if at least one dynamic Quick Action item was set in the app; otherwise the value is `false`.

Platform Availability

- iOS 9.0 and later
- Android 7.1 and later or API Level 25 or later

24. Functional Modules API

The Functional Modules API is used to load a functional module (that is defined using `functionalModule.xml` file) through synchronous and asynchronous operations dynamically. The Functional Modules API contains the `kony.modules Namespace` and the following functions.

Function	Description
`kony.modules.loadFunctionalModule`	A synchronous API used to load functional module (views and jsModules) in scope of JavaScript.
`kony.modules.loadFunctionalModuleAsync`	Asynchronous API used to load functional module (views and jsModules) in scope of JavaScript. If the module is already loaded, then it will call successcallback without loading the same module.

Load the functional modules (views and js modules) in scope of JavaScript by using the `kony.modules.loadFunctionalModule` function. When you reload the modules again, the function returns a Boolean value true. You can also load the function modules by using the `kony.modules.loadFunctionalModuleAsync` function. Using this asynchronous function, you can call the success call back or failure call back when you reload the modules again.

24.1 kony.modules Namespace

The kony.modules namespace enables your apps to load functional modules through synchronous and asynchronous operations on demand. It provides the following API elements.

24.1.1 Functions

The kony.module namespace contains the following functions.

24.1.2 loadFunctionalModule

loadFunctionalModule is a synchronous API used to load functional module (views and jsModules) in scope of JavaScript. Until loading of module is complete, further executions will be stopped.

Note: If the module is already loaded, then it returns true without reloading the same module.

Syntax

```
kony.modules.loadFunctionalModule (modulename)
```

Input Parameters

Parameter	Description
modulename [String] - Mandatory	Unique module name that is defined in the `functionalModules.xml` file.

Example

```
//In module1 you can load module2 as below when required.
function buttonClick() {
    kony.modules.loadFunctionalModule("module2");
    //Now we cn access objects defined in module2
    frmLogin.show();
}
```

Return Values

Return Value	Description
Boolean	*true*: When module is loaded. *false*: When any issue found while loading.

Platform Availability

Available on iOS, Android, Windows, SPA and Desktop Web

24.1.3 loadFunctionalModuleAsync

The loadFunctionalModuleAsync is an asynchronous API used to load functional module (views and jsModules) in scope of JavaScript. A success callback is invoked after successful completions of load, else error callback will be invoked.

> ***Note:*** If the module is already loaded, then it will call *successcallback* without loading the same module.

Syntax

```
kony.modules.loadFunctionalModuleAsync (modulename,
successcallback, errorcallback)
```

Input Parameters

Parameter	Description
modulename [String] - Mandatory	Unique name of the module that is defined in the `functionalModules.xml` file
successcallback [Function] - Mandatory	The callback will be invoked after successful load of module files and after calling corresponding init function, if defined. • successcallback (modulename)
errorcallback [Function] - Mandatory	The callback which will be invoked if there are any issue while loading module files. • errorcallback (modulename, errorcode)

Example

```
//In module1 you can load module2 as below when required.
    var successcalback = function (modulename) {
        //Now we cn access objects defined in module2
        frmLogin.show();
 }
    var errorcalback = function (modulename, errorcode) {
        kony.print("error : " + errorcode + " in module - " +
modulename);
 }

    function buttonClick() {
        kony.modules.loadFunctionalModuleAsync("module2",
successcalback, errorcalback);
    }
```

Error Codes

1250 - Invalid module name is specified.

1251 - Unable to load module.

Platform Availability

Available on iOS, Android, Windows, SPA and Desktop Web.

25. GeoLocation API

The GeoLocation API defines a high-level interface to location information, such as latitude and longitude associated with the mobile device. The API uses the common sources of location information such as the Global Positioning System (GPS) to infer details such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs. This API does not always return the device's actual location.

The Geolocation API uses the `kony.location Namespace` and the following API elements.

Function	Description
`kony.location.getCurrentPosition`	Retrieves the current location of the device.
`kony.location.watchPosition`	Sets callbacks that report the device's position.
`kony.location.clearWatch`	Verifies the value of the given watchID argument. If the value corresponds to a previously started watch process, the process is stopped immediately.

The API is designed to enable both *one-shot* position (getCurrentPosition) requests and repeated position (watchPosition) updates, as well as the ability to explicitly query the cached positions. Location information is represented by latitude and longitude coordinates.

The GeoLocation APIs are modeled after W3C GeoLocation specification. For a more hands-on approach on the functionality of the various Geolocation APIs provided by Kony AppPlatform, import and preview the Geolocation Feature app by using Kony Visualizer.

To get the current position of the device, use the `kony.location.getCurrentPosition` function. Continue to watch the position of the device by using the `kony.location.watchPosition` function. To stop tracking the device movement, use the `kony.location.clearWatch` function.

Note: The `phone.mylocation` function is deprecated and should not be used in new software. However, there is still documentation for it.

To view the functionality of the GeoLocation API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

25.1 kony.location Namespace

The kony.location namespace defines a high-level interface to location information. It contains the following API elements.

25.1.1 Functions

The `kony.location` namespace provides the following functions.

kony.location.getCurrentPosition

Using the getCurrentPostion function, you can get the current location of the device.

Syntax

```
kony.location.getCurrentPosition(
    successcallback,
    errorcallback,
    positionoptions)
```

Parameters

successcallback [Function] - Mandatory

The successcallback function specifies the callback function that must be executed when the API call is successful. The signature of the callback function is `successcallback(position)` where, **position** contains the coordinates of the geo-location that are created and returned by the API. It is an object that contains certain key-value pairs.

coords [Object] - Coordinates that has the following key-value pairs:

key	Description
latitude [Number]	Latitude in decimal degrees.
longitude [Number]	Longitude in decimal degrees.

key	Description
altitude [Number]	Height of the location in meters above the ellipsoid.
accuracy [Number]	Accuracy level of the latitude and longitude coordinates in meters.
altitudeaccuracy [Number]	Accuracy level of the altitude coordinate in meters.
heading [Number]	Direction of travel, specified in degrees counting clockwise relative to the true north.
speeding [Number]	Current ground speed of the device, specified in meters per second.
timestamp [Number]	Represents the time when the Position object was acquired.

errorcallback [Function] - Optional

The errorcallback function specifies the callback function that must be executed when the API call fails. The callback function has the following signature:

errorcallback(positionerror)- positionerror is an object that has the following key-value pairs:

key	Description
code [Number]	error code.
message [String]	error message.

For more information on the Error codes and messages, refer error code.

positionoptions [Object] - Optional

Using the positionoptions parameter, the user can customize the retrieval of the geolocation. It is an object that has the following key-value pairs:

key	Description
enableHighAccuracy [Boolean]	Provides a hint to the implementation in order to receive the best possible result.
timeout [Number]	Denotes the maximum length of time in milliseconds that is allowed to pass from the call.
maximumAge [Number]	Indicates the application to accept a cached position whose age is no greater than the specified time in milliseconds.
minimumTime [Number]	Indicates the desired interval for active location updates in milliseconds. *Note:* This property is only available on the Android platform.
requireBackgroundAccess [Boolean]	Set to `true` to fetch location updates even when the app is running in the background on Android devices. In apps that use TargetSDK version 29 (and later), you must add the ACCESS_BACKGROUND_LOCATION permission in the Android Manifest file to get location updates in the background. This property is only available on the Android platform. *Important:* This parameter has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

key	Description
useBestProvider	Set to `true` to improve the reliability of calls to this function on Android devices. Omitting this option on Android could cause calls to this function to have intermittent timeouts.

Example

```
/*****************************************************************
 *     Name    : getCurrentPosition function
 *     Author  : Kony
 *     Purpose : This function helps to get the current location
 *****************************************************************/
function getPosition() {
    var positionoptions = {
        timeout: 15000
    }; // 15 secs
    kony.location.getCurrentPosition(successcallback, errorcallback,
positionoptions);
}

// Callback that is executed on success of getCurrentPosition function.
function successcallback(position) {
    var geoPosition = "Latitude: " + position.coords.latitude;
    geoPosition = geoPosition + " Longitude: " + position.coords.longitude;
    geoPosition = geoPosition + " Altitude: " + position.coords.altitude;
    geoPosition = geoPosition + " Accuracy: " + position.coords.accuracy;
    geoPosition = geoPosition + " Altitude Accuracy: " +
position.coords.altitudeAccuracy;
    geoPosition = geoPosition + " Heading: " + position.coords.heading;
    geoPosition = geoPosition + " Speeding: " + position.coords.speeding;
    geoPosition = geoPosition + " Timestamp: " + position.timestamp;
    alert(geoPosition);
}

// Callback that is executed on error of getCurrentPosition function.
```

```
function errorcallback(positionerror) {
    var errorMesg = "Error code: " + positionerror.code;
    errorMesg = errorMesg + " message: " + positionerror.message;
    alert(errorMesg);
}
```

MVC Example

```
currentPositionSuccessCallback: function(position) {
    /*
  //  position object will have the below properties .
         latitude = position.coords.latitude
         longitude = position.coords.longitude
         altitude = position.coords.altitude
         atitudeAccuracy = position.coords.altitudeAccuracy
         heading = position.coords.heading
         speeding = position.coords.speeding
         timestamp = position.timestamp
    */
    alert(JSON.stringify(position));
    /* use the position depending on the use case ,some of the use cases are
listed below .
       1. Get the nearby events(like ATMs, Restaurants …etc.)  based on the user
current location
       2. In a tracking-based scenario ,use as an initial position of the user .
       */


},
currentPositionFailureCallback: function(error) {
    alert(JSON.stringify(error));
},
getCurrentPositionOfDevice: function() {
    var options = {};
    options.enableHighAccuracy = true; //  uses provider that gets accurate
location
    options.timeout = 10000; // timeout in milliseconds
    options.requireBackgroundAccess  = true; // gets the location updates in
the background as well
    kony.location.getCurrentPosition(this.currentPositionSuccessCallback,
this.currentPositionFailureCallback, options);
}

/* Please see example of getCurrentPosition() in "frmTrackingUserLocation"
Form of sample app*/
```

Free form Example

```
function currentPositionSuccessCallback(position) {
  /*
       // position object will have the below properties .
       latitute = position.coords.latitude
    longitude = position.coords.longitude
    altitude = position.coords.altitude
    atitudeAccuracy = position.coords.altitudeAccuracy
    heading = position.coords.heading
    speeding = position.coords.speeding
    timestamp = position.timestamp
       */
  alert(JSON.stringify(position));
  /* use the position depending on the use case ,some of the use cases are
listed below .
       1. Get the nearby events(like ATMs, Restaurants …etc.)  based on the user
current location
       2. In a tracking-based scenario ,use as an initial position of the user .
       */

}

function currentPositionFailureCallback(error) {
  alert(JSON.stringify(error));
}

function getCurrentPositionOfDevice () {
  var options = {};
  options.enableHighAccuracy = true;
  options.timeout = 10000; // timeout in milli seconds
  options.requireBackgroundAccess  = true; // gets the location updates in the
background as well
  kony.location.getCurrentPosition(currentPositionSuccessCallback,
currentPositionFailureCallback, options);
}
```

Return Values

None

Exceptions

LocationError

Error Code

Error Code	Description
1	PERMISSION_DENIED
2	POSITION_UNAVAILABLE
3	TIMEOUT

Android-specific Error Codes

Error Code	Error Message	Description
4	Missing android.permission.ACCESS_ BACKGROUND_LOCATION permission in AndroidManifest.xml	The developer has missed adding the android.permission.ACCESS_ BACKGROUND_LOCATION permission in the AndroidManifest.xml
5	BACKGROUND_PERMISSION_ DENIED	The end-user has selected ”Allow only while the app is in use" instead of “Allow all the time” option on devices that run on Android 9 (and later).
6	Permission Denied for <PermissionName> with Don't Ask Again	The user has denied permission with the Don't ask again or Never ask again option.

Remarks

This API takes up to three arguments. When invoked, the API returns and asynchronously attempts to obtain the current location of the device. The app on which this API is used must contain runtime permission from the end-user to obtain the current location of the device. If the API is invoked without obtaining the permission, device native platforms automatically display a system permission dialog box with **Allow** and **Deny** options. The end-user can grant permission to get the current location.

In Android apps that use TargetSDK version 29 (and later), and the **requireBackgroundAccess** property is enabled, you must manually add the `ACCESS_BACKGROUND_LOCATION` permission in the Android Manifest file to get location updates in the background.

If the end-user taps the **Allow** option, the attempt is successful, the successCallback is invoked (i.e. the handleEvent operation must be called on the callback object) with a new Position object, reflecting the current location of the device. If the attempt fails, the errorCallback is invoked with a new PositionError object, reflecting the reason for the failure. This is applicable only for Android and iOS platforms.

If the end-user taps the **Deny** option, the **errorcallback** is invoked with the **PERMISSON_DENIED** error code and error message.

> ***Note:*** Runtime permissions are applicable only on iOS and Android platforms

Platform Availability

Available on all platforms except prior to IE8 releases.

Remarks

This API takes up to three arguments. When invoked, the API returns and asynchronously attempts to obtain the current location of the device. The app on which this API is used must contain runtime permission from the end-user to obtain the current location of the device. If the API is invoked without obtaining the permission, device native platforms automatically display a system permission dialog box with **Allow** and **Deny** options. The end-user can grant permission to get the current location.

In Android apps that use TargetSDK version 29 (and later), and the **requireBackgroundAccess** property is enabled, you must manually add the `ACCESS_BACKGROUND_LOCATION` permission in the Android Manifest file to get location updates in the background.

If the end-user taps the **Allow** option, the attempt is successful, the successCallback is invoked (i.e. the handleEvent operation must be called on the callback object) with a new Position object, reflecting the current location of the device. If the attempt fails, the errorCallback is invoked with a new PositionError object, reflecting the reason for the failure. This is applicable only for Android and iOS platforms.

If the end-user taps the **Deny** option, the **errorcallback** is invoked with the **PERMISSON_DENIED** error code and error message.

> *Note:* Runtime permissions are applicable only on iOS and Android platforms

Platform Availability

Available on all platforms except Windows 7 Kiosk and prior to IE8 releases.

kony.location.watchPosition

Using the watchPosition function, you can set callbacks that report the device's position.

Syntax

```
kony.location.watchPosition(
    successcallback,
    errorcallback,
    positionoptions)
```

Parameters

successcallback [Function] - Mandatory

The successcallback function specifies the callback function that must be executed when the API call is successful. The signature of the callback function is `successcallback(position)` where, **position** contains the coordinates of the geo-location that are created and returned by the API. It is an object that contains certain key-value pairs.

coords [Object] - Coordinates that has the following key-value pairs:

key	Description
latitude [Number]	Latitude in decimal degrees.
longitude [Number]	Longitude in decimal degrees.
altitude [Number]	Height of the location in meters above the ellipsoid.
accuracy [Number]	Accuracy level of the latitude and longitude coordinates in meters.
altitudeaccuracy [Number]	Accuracy level of the altitude coordinate in meters.
heading [Number]	Direction of travel, specified in degrees counting clockwise relative to the true north.
speeding [Number]	Current ground speed of the device, specified in meters per second.
timestamp [Number]	Represents the time when the Position object was acquired.

errorcallback [Function] - Optional

The errorcallback function specifies the callback function that must be executed when the API call fails. The callback function has the following signature:

errorcallback(positionerror)- positionerror is an object that has the following key-value pairs:

key	Description
code [Number]	error code.
message [String]	error message.

For more information on the Error codes and messages, refer error code.

positionoptions [Object] - Optional

Using the positionoptions parameter, the user can customize the retrieval of the geolocation. It is an object that has the following key-value pairs:

key	Description
enableHighAccuracy [Boolean]	Provides a hint to the implementation in order to receive the best possible result.
fastestInterval [Number]	Sets the fastest interval for location updates in milliseconds. The fastestInterval key controls the rate at which your application will receive location updates, which might be faster than minimumTime in some cases. This happens when other apps fetch location updates and the current app receives them passively to save power. ***Note:*** This property is only available on the Android platform. The rate at which the app receives the fastest update will not be less than the value specified for the fastestInterval property. The value for the fastestInterval must be more than zero and less than the value of minimumTime (i.e, 0 < fastestInterval <= minimumTime). If you do not set the value for fastestInterval, the value of minimumTime is set, by default. ***Note:*** Ensure that you have enabled the **Use Google Play Location Services** checkbox in the **Project Settings** > **Native** > **Android** section of Kony Visualizer.
timeout [Number]	Denotes the maximum length of time in milliseconds that is allowed to pass from the call.

key	Description
maximumAge [Number]	Indicates the application to accept a cached position whose age is no greater than the specified time in milliseconds.
minimumDistance [Number]	Minimum distance in meters between location updates.
minimumTime [Number]	Minimum time in milliseconds between location updates.
requireBackgroundAccess [Boolean]	Set to `true` to fetch location updates even when the app is running in the background on Android devices. In apps that use TargetSDK version 29 (and later), you must add the ACCESS_BACKGROUND_LOCATION permission in the Android Manifest file to get location updates in the background. This property is only available on the Android platform. ***Important:*** This parameter has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Return Values

Return Value	Description
watchID [Number]	Returns a number that denotes the unique ID of the watch operation.

Example

```
function successcallback1(position) {
    lblTest.text = "working with watchPosition success full call back";
    var geoPosition = "Latitude: " + position.coords.latitude;
    geoPosition = geoPosition + " Longitude: " + position.coords.longitude;
    geoPosition = geoPosition + " Altitude: " + position.coords.altitude;
    geoPosition = geoPosition + " Accuracy: " + position.coords.accuracy;
    geoPosition = geoPosition + " Altitude Accuracy: " +
position.coords.altitudeAccuracy;
    geoPosition = geoPosition + " Heading: " + position.coords.heading;
    geoPosition = geoPosition + " Speeding: " + position.coords.speeding;
    geoPosition = geoPosition + " Timestamp: " + position.timestamp;
    alert(geoPosition);
}

function errorcallback1(positionerror) {
    lblTest.text = "working with watchPosition err call back";
    var errorMesg = "Error code: " + positionerror.code;
    errorMesg = errorMesg + " message: " + positionerror.message;
    alert(errorMesg);
}
var positionoptions = {
    maximumAge: 3000,
    minimumDistance: 5,
    minimumTime: 5000
};
watchID = kony.location.watchPosition(successcallback1, errorcallback1,
positionoptions);
```

MVC Example

```
watchID: null,
  watchPositionSuccessCallback: function(position) {
    /*
    // position object will have the below properties .
```

```
       latitute = position.coords.latitude
       longitude = position.coords.longitude
       altitude = position.coords.altitude
       atitudeAccuracy = position.coords.altitudeAccuracy
       heading = position.coords.heading
       speeding = position.coords.speeding
       timestamp = position.timestamp
     */
    alert(JSON.stringify(position));
  },
  watchPositionFailureCallback: function(error) {
    alert(JSON.stringify(error));
  },
  watchPositionOfDevice: function() {
    var self = this;
    var options = {};
    options.maximumAge = 1000;
    options.minimumTime = 2000; // time in milli seconds
    options.minimumDistance = 2; // distance in meters
    options.requireBackgroundAccess  = true; // gets the location updates in
the background as well
    this.watchID = kony.location.watchPosition
(this.watchPositionSuccessCallback,this.watchPositionFailureCallback,
options);
    /* Use-Cases:
       In tracking-based scenarios, the watchPosition() API can be used to
    monitor a position
       */

       /*Please see example of watchPosition() in "frmTrackingUserLocation" Form of
sample app*/
  }
```

Free Form Example

```
watchID = null;

function watchPositionSuccessCallback(position) {
    /*
        // position object will have the below properties .
       latitute = position.coords.latitude
       longitude = position.coords.longitude
       altitude = position.coords.altitude
       atitudeAccuracy = position.coords.altitudeAccuracy
       heading = position.coords.heading
       speeding = position.coords.speeding
       timestamp = position.timestamp
       */

    alert(JSON.stringify(position));
}

function watchPositionFailureCallback(error) {
    alert(JSON.stringify(error));
}

function watchPositionOfDevice() {
    var self = this;
    var options = {};
    options.maximumAge = 1000; // use cached position if exists in mentioned
time(in milliseconds)
    options.minimumTime = 2000; // time criteria for location updates
    options.minimumDistance = 2; // distance criteria for location updates
    options.requireBackgroundAccess  = true; // gets the location updates in
the background as well
    watchID = kony.location.watchPosition(watchPositionSuccessCallback,
watchPositionFailureCallback, options);
}
```

Exceptions

LocationError

Error Code

Error Code	Description
1	PERMISSION_DENIED
2	POSITION_UNAVAILABLE
3	TIMEOUT

Remarks

Android-specific Error Codes

Error Code	Error Message	Description
4	Missing android.permission.ACCESS_ BACKGROUND_LOCATION permission in AndroidManifest.xml	The developer has missed adding the android.permission.ACCESS_ BACKGROUND_LOCATION permission in the AndroidManifest.xml
5	BACKGROUND_PERMISSION_ DENIED	The end-user has selected ”Allow only while the app is in use" instead of “Allow all the time” option on devices that run on Android 9 (and later).
6	Permission Denied for <PermissionName> with Don't Ask Again	The user has denied permission with the Don't ask again or Never ask again option.

Remarks

The behavior of this function depends on the underlying hardware platform. For example, if your app is running on Android and you set minimumTime and minimumDistance to their minimum possible values, the callback function in the *successcallback* parameter will not be called, as per the Android documentation.

This API takes one, two, or three arguments. When invoked, it must immediately return a number that uniquely identifies a watch operation and then asynchronously start the watch operation. This operation first attempts to obtain the current location of the device. Your app needs runtime permission from the end-user to obtain the current location of the device. If you call the API without obtaining the permission, platforms automatically pops up a system permission dialog box with **Allow** and **Deny** options, asking the end-user to grant permission to get the current location.

In Android apps that use TargetSDK version 29 (and later), and the **requireBackgroundAccess** property is enabled, you must manually add the `ACCESS_BACKGROUND_LOCATION` permission in the Android Manifest file to get location updates in the background.

If the end-user taps the **Allow** option, the attempt is successful, the succesCallback is invoked (i.e. the handleEvent operation must be called on the callback object) with a new Position object, reflecting the current location of the device. If the attempt fails, the errorCallback is invoked with a new PositionError object, reflecting the reason for the failure.

If the end-user taps the **Deny** option, the errorcallback in invoked with the **PERMISSON_DENIED** error code and error message.

> ***Note:*** The runtime permissions are applicable only in the iOS and Android platforms.

The watch operation continues to monitor the position of the device and invokes the appropriate callback every time this position changes. The watch operation continues until the clearwatch method is called with the corresponding identifier.

Platform Availability

Available on all platforms except Desktop Web, IE8 and prior to IE8 releases.

The behavior of this function depends on the underlying hardware platform. For example, if your app is running on Android and you set minimumTime and minimumDistance to their minimum possible values, the callback function in the *successcallback* parameter will not be called, as per the Android documentation.

This API takes one, two, or three arguments. When invoked, it must immediately return a number that uniquely identifies a watch operation and then asynchronously start the watch operation. This operation first attempts to obtain the current location of the device. Your app needs runtime permission from the end-user to obtain the current location of the device. If you call the API without obtaining the permission, platforms automatically pops up a system permission dialog box with **Allow** and **Deny** options, asking the end-user to grant permission to get the current location.

If the end-user taps the **Allow** option, the attempt is successful, the succesCallback is invoked (i.e. the handleEvent operation must be called on the callback object) with a new Position object, reflecting the current location of the device. If the attempt fails, the errorCallback is invoked with a new PositionError object, reflecting the reason for the failure.

If the end-user taps the **Deny** option, the errorcallback in invoked with the **PERMISSON_DENIED** error code and error message.

> ***Note:*** The runtime permissions are applicable only in the iOS and Android platforms.

The watch operation continues to monitor the position of the device and invokes the appropriate callback every time this position changes. The watch operation continues until the clearwatch method is called with the corresponding identifier.

Platform Availability

Available on all platforms except Desktop Web, Windows 7 Kiosk, IE8 and prior to IE8 releases.

kony.location.clearWatch

When invoked, the clearWatch first checks the value of the given watchID argument. If this value does not correspond to any previously started watch process, then the method returns immediately without performing any further action. Otherwise, the watch process identified by the watchID argument is stopped immediately and no further callbacks are invoked.

Syntax

```
kony.location.clearWatch(
    watchID);
```

Parameters

Function	Description
watchID [Number] - Mandatory	Specifies the number that uniquely identifies the watch.

Example

```
kony.location.clearWatch(watchID);
```

MVC

```
stopWatchingPosition: function() {
    try {
        kony.location.clearWatch(this.watchID); // clears/stops watching
position of the user which was being monitored using watchPosition API
        alert("Cleared !");
    } catch (exception) {
        alert(exception);
    }

    /* Please see example of clearWatch() in "frmTrackingUserLocation" Form of
sample app */
}
```

Free Form

```
function stopWatchingPosition () {
  try{
    kony.location.clearWatch(watchID); // clears/stops watching position of
the user which was being monitored using watchPosition API
    alert("Cleared !");
  }catch(exception){
    alert(exception);
  }
}
```

Return Values

None.

Exceptions

LocationError

Error

Platform Availability

Available on all platforms except Desktop Web and Windows 7 Kiosk.

---

26. Gesture API

A *gesture* is a user's action associated with touch screens. Using Gesture API, you can determine how a gesture must be interpreted and the function that must be executed when a gesture is recognized. Gestures are supported for any widget in a flex layout. Currently, Kony supports only Tap, Swipe, Longpress, Pan, Rotation, Pinch, and 3D Touch gesture types.

The Gestures API uses `kony.application Namespace` and provides support for the following functions.

Function	Description
`kony.gesture.addGestureRecognizer`	Sets a gesture recognizer for a specified widget.
`kony.gesture.addGestureRecognizerForAllForms`	Sets a gesture recognizer for all the forms.
`kony.gesture.removeGestureRecognizerForAllForms`	Enables you to remove a specified gesture recognizer for all Forms.

Configure a specified gesture recognizer for a widget using the `kony.gesture.addGestureRecognizer` or the `kony.gesture.addGestureRecognizerForAllForms` function. In the function, define the type of the gesture to be recognized by using gestureType key-value pair. Define the configuration parameters needed to setup a gesture recognizer by using gestureConfigParams object. Further, define a function that must be executed once the gesture is recognized by using the OnGestureClosure function.

The configuration parameters used to set up a gesture recognizer are, gesture duration, number of taps, and the gesture type number. If you want to remove a specified gesture recognizer for a widget, use the `kony.gesture.removeGestureRecognizerForAllForms` function.

> ***Note:*** Gestures are applicable only on mobile or tablet devices that have touch support. In tablet, longpress is only triggered when it is a touch event and not a mouse event.

26.1 Important Considerations

- Single tap matches with the *onClick* functionality. If a box has an *onClick* event defined, both gesture event and *onClick* are triggered.
- Number of taps is limited to one or two, only single tap or double tap are recognized.
- Number of fingers is limited to one, only single finger is recognized.

To view the functionality of the Gesture API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

26.2 Functions

The Gesture API contains the following functions, which are part of the kony.application Namespace.

kony.application.addGestureRecognizer

Using the addGestureRecognizerForAllForms function, you can set a gesture recognizer for a specified widget.

Syntax

```
kony.application.addGestureRecognizer (gestureType,
gestureConfigParams,onGestureClosure)
```

Input Parameters

gestureType [Number] - Mandatory

Indicates the type of gesture that must be detected on the widget. Following are the possible gestureType values:

- 1 - constants.GESTURE_TYPE_TAP
- 2 - constants.GESTURE_TYPE_SWIPE
- 3 - constants.GESTURE_TYPE_LONGPRESS
- 4 - constants.GESTURE_TYPE_PAN
- 5 - constants.GESTURE_TYPE_ROTATION
- 6 - constants.GESTURE_TYPE_PINCH
- 7 - constants.GESTURE_TYPE_RIGHTTAP

Note:

- RIGHTTAP is applicable only to Windows 8.1 and Windows Desktop/Kiosk platforms.
- ROTATION is not supported on Android.

gestureConfigParams [object] - Mandatory

Specifies a table that has the configuration parameters that are required to setup a gesture recognizer. The configuration parameters vary based on the type of the gesture.

This parameter has the following key-value pairs:

Gesture Type	Configuration Parameter
TAP	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. Default value is 1. • taps [Number] - specifies the maximum number of taps that are allowed for a gesture. The possible values are 1, 2. Default value is 1.

Gesture Type	Configuration Parameter
SWIPE	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. The Default value is 1. • recognizeSimultaneously [Boolean] - Indicates whether the swipe gesture of a child widget must be recognized simultaneously along with the actual gesture recognizer of the form or a parent Flex Container. The Default value is true. For example, consider a form that contains a Segment widget, and a Swipe gesture is added to the form. Use the recognizeSimultaneously parameter to enable the swipe gesture for the Segment widget along with the form. If the parameter is set as true, the Segment widget triggers the Swipe gesture callback, when swiped. If the parameter is set as false, the Segment widget will not trigger the swipe gesture callback, when swiped.

Gesture Type	Configuration Parameter
LONGPRESS	• pressDuration [Number] - specifies the minimum time interval (in seconds) after which the gesture is recognized. The default value is 1. This is not applicable to Windows. `For example, {pressDuration:1}`
PAN	• fingers [number] - specifies the minimum number of fingers that are required to recognize this gesture. Default value is 1. • continuousEvents [Boolean] - indicates if callback should be called continuously for every change beginning from the time the gesture is recognized to the time it ends.

Gesture Type	Configuration Parameter
ROTATION	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] - indicates if callback must be called continuously for every change beginning from the time the gesture is recognized to the time it ends.
PINCH	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] indicates if callback should be called continuously every change beginning from the time the gesture is recognized to the time it ends.

onGestureClosure [function] - Mandatory

Specifies the function that needs to be executed when a gesture is recognized. This function will be raised asynchronously and has the following signature:

```
onGestureClosure(widgetRef, gestureInfo, context)
```

Parameter	Description
widgetRef	specifies the handle to the widget on which the gesture was recognized.
gestureInfo	Table with information about the gesture. The contents of this table vary based on the gesture type.
context	Table with SegmentedUI row details.

gestureInfo table has the following key-value pairs:

Key	Description
gestureType [number]	Indicates the gesture type
gesturesetUpParams [object]	Specifies the set up parameters passed while adding the gesture recognizer
gesturePosition [number]	Indicates the position where the gesture is recognized. Possible values are: • 1 for TOPLEFT • 2 for TOPCENTER • 3 for TOPRIGHT • 4 for MIDDLELEFT • 5 for MIDDLECENTER • 6 for MIDDLERIGHT • 7 for BOTTOMLEFT • 8 for BOTTOMCENTER • 9 for BOTTOMRIGHT • 10 for CENTER

Key	Description
swipeDirection [number]	Indicates the direction of swipe. Direction is w.r.t the view and not device orientation. This parameter is applicable only if the gesture type is SWIPE. Possible values are: • 1 for LEFT • 2 for RIGHT • 3 for TOP • 4 for BOTTOM
gestureX [number]	specifies the X coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
gestureY [number]	specifies the Y coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
widgetWidth [number]	specifies the width of the widget (in pixels).
widgetHeight [number]	specifies the height of the widget (in pixels).

Key	Description
gestureState[number]	Indicates the gesture state. The gestureState is applicable only for continuous gestures like PAN, ROTATION, and PINCH. • 1 - gesture state begin • 2 - gesture state changed • 3 - gesture state ended
rotation [number]	Rotation of the gesture in degrees since its last change.(Applicable only when gesture type is ROTATION)
velocityX and velocityY	horizontal and vertical component of velocity expressed in points per second. (Applicable only for PAN gesture type)
velocity [number]	velocity of pinch in scale per second. (Applicable only for Pinch gesture)
scale [number]	scale factor relative to the points of the two touches in screen coordinates.

Key	Description
touchType[number]	(Applicable to windows platform only) • 0 - constants.TOUCHTYPE_FINGER • 1 - constants.TOUCHTYPE_PEN • 2 - constants.TOUCHTYPE_MOUSE
translationX and translationY [number]	Cumulative distance as number. (Applicable only for PAN gesture type)

context table has the following key-value pairs:

Key	Description
rowIndex [number]	Row index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)
sectionIndex [number]	Section index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)

Example

```
 addLongPressGesture: function() {
    var getTime = parseInt(this.view.lstbx.selectedKey);

    var longConfig = {
        pressDuration: getTime
    };
    gesturehandle = this.view.flxLongpress.addGestureRecognizer
(constants.GESTURE_TYPE_LONGPRESS, longConfig, this.onLongpressClosure);

}
```

```
onLongpressClosure: function(widgetRef, gestureInfo) {

    this.view.lblGesture.text = "A longpress gesture was performed for " +
gestureInfo.gesturesetUpParams.pressDuration + " Seconds";

},
addLongPressGesture: function() {
    var getTime = parseInt(this.view.lstbx.selectedKey);

    var longConfig = {
        pressDuration: getTime
    };
    gesturehandle = this.view.flxLongpress.addGestureRecognizer
(constants.GESTURE_TYPE_LONGPRESS, longConfig, this.onLongpressClosure);

}
```

Return Values

String - Reference to the gesture is returned.

Platform Availability

Available on all platforms except Server Side Mobile Web, Windows 7/Kiosk, and Desktop Web.

kony.application.addGestureRecognizerForAllForms

Using the addGestureRecognizerForAllForms function, you can set a gesture recognizer for all the forms.

Syntax

```
kony.application.addGestureRecognizerForAllForms (gestureType,
gestureConfigParams,onGestureClosure)
```

Input Parameters

gestureType [Number] - Mandatory

Indicates the type of gesture that must be detected on the widget. Following are the possible gestureType values:

- 1 - constants.GESTURE_TYPE_TAP
- 2 - constants.GESTURE_TYPE_SWIPE
- 3 - constants.GESTURE_TYPE_LONGPRESS
- 4 - constants.GESTURE_TYPE_PAN
- 5 - constants.GESTURE_TYPE_ROTATION
- 6 - constants.GESTURE_TYPE_PINCH
- 7 - constants.GESTURE_TYPE_RIGHTTAP

Note:

- RIGHTTAP is applicable only to Windows 8.1 and Windows Desktop/Kiosk platforms.
- ROTATION is not supported on Android.

gestureConfigParams [object] - Mandatory

Specifies a table that has the configuration parameters that are required to setup a gesture recognizer. The configuration parameters vary based on the type of the gesture.

Gesture Type	Configuration Parameter
TAP	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. Default value is 1. • taps [Number] - specifies the maximum number of taps that are allowed for a gesture. The possible values are 1, 2. Default value is 1.
SWIPE	• fingers [Number] - specifies the maximum number of fingers that are allowed for a gesture. The possible values are 1, 2. Default value is 1.

Gesture Type	Configuration Parameter
LONGPRESS	• pressDuration [Number] - specifies the minimum time interval (in seconds) after which the gesture is recognized. The default value is 1. This is not applicable to Windows. `For example, {pressDuration:1}`
PAN	• fingers [number] - specifies the minimum number of fingers that are required to recognize this gesture. Default value is 1. • continuousEvents [Boolean] - indicates if callback should be called continuously for every change beginning from the time the gesture is recognized to the time it ends.

Gesture Type	Configuration Parameter
ROTATION	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] - indicates if callback must be called continuously for every change beginning from the time the gesture is recognized to the time it ends.
PINCH	• fingers [Number] - The number of fingers that are required to recognize the gesture. The Default value is 2. • continuousEvents [Boolean] indicates if callback should be called continuously every change beginning from the time the gesture is recognized to the time it ends.

onGestureClosure [function] - Mandatory

Specifies the function that needs to be executed when a gesture is recognized. This function will be raised asynchronously and has the following signature:

```
onGestureClosure(widgetRef, gestureInfo, context)
```

Parameter	Description
widgetRef	specifies the handle to the widget on which the gesture was recognized.
gestureInfo	Table with information about the gesture. The contents of this table vary based on the gesture type.
context	Table with SegmentedUI row details.

gestureInfo table has the following key-value pairs:

Key	Description
gestureType [number]	Indicates the gesture type
gesturesetUpParams [object]	Specifies the set up parameters passed while adding the gesture recognizer
gesturePosition [number]	Indicates the position where the gesture is recognized. Possible values are: • 1 for TOPLEFT • 2 for TOPCENTER • 3 for TOPRIGHT • 4 for MIDDLELEFT • 5 for MIDDLECENTER • 6 for MIDDLERIGHT • 7 for BOTTOMLEFT • 8 for BOTTOMCENTER • 9 for BOTTOMRIGHT • 10 for CENTER

Key	Description
swipeDirection [number]	Indicates the direction of swipe. Direction is w.r.t the view and not device orientation. This parameter is applicable only if the gesture type is SWIPE. Possible values are: • 1 for LEFT • 2 for RIGHT • 3 for TOP • 4 for BOTTOM
gestureX [number]	specifies the X coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
gestureY [number]	specifies the Y coordinate of the point (in pixels) where the gesture has occurred. The coordinate is relative to the widget coordinate system.
widgetWidth [number]	specifies the width of the widget (in pixels).
widgetHeight [number]	specifies the height of the widget (in pixels).

Key	Description
gestureState[number]	Indicates the gesture state. The gestureState is applicable only for continuous gestures like PAN, ROTATION, and PINCH. • 1 - gesture state begin • 2 - gesture state changed • 3 - gesture state ended
rotation [number]	Rotation of the gesture in degrees since its last change.(Applicable only when gesture type is ROTATION)
velocityX and velocityY	horizontal and vertical component of velocity expressed in points per second. (Applicable only for PAN gesture type)
velocity [number]	velocity of pinch in scale per second. (Applicable only for Pinch gesture)
scale [number]	scale factor relative to the points of the two touches in screen coordinates.

Key	Description
touchType[number]	(Applicable to windows platform only) • 0 - constants.TOUCHTYPE_FINGER • 1 - constants.TOUCHTYPE_PEN • 2 - constants.TOUCHTYPE_MOUSE
translationX and translationY [number]	Cumulative distance as number. (Applicable only for PAN gesture type)

context table has the following key-value pairs:

Key	Description
rowIndex [number]	Row index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)

Key	Description
sectionIndex [number]	Section index of the segment UI where gesture is recognized. (Applicable to gestures added to segUI rows)

Example

```
//Defining a function
function formGesture(widgetID, gestureInfo) {
    var y = kony.type(gestureInfo); //expected value of y = table
    var z = kony.type(gestureInfo.gesturesetUpParams); //expected values of z
= table
    var a = gestureInfo.gestureType;
    var b = gestureInfo.gesturesetUpParams;
    var c = gestureInfo.gesturePosition;
    var d = gestureInfo.gestureX;
    var e = gestureInfo.gestureY;
    var f = gestureInfo.widgetWidth;
    var g = gestureInfo.widgetHeight;
    kony.print("*******************************************");
    if (kony.os.toNumber(gestureInfo.gestureType) == 2) {
        h = gestureInfo.swipeDirection;
        kony.print("swipe direction is: " + h);
    } else {
        h = "";
    }
    if (kony.os.toNumber(a) == 1) {
        b1 = "fingers: " + gestureInfo.gesturesetUpParams.fingers;
        b2 = "taps: " + gestureInfo.gesturesetUpParams.taps;
        kony.print("" + b1 + "" + b2);
    } else if (kony.os.toNumber(a) == 2) {
        b1 = "fingers :" + gestureInfo.gesturesetUpParams.fingers;
        b2 = "";
        kony.print("" + b1 + "" + b2);
    } else if (kony.os.toNumber(a) == 3) {
        b1 = "pressduration:" + gestureInfo.gesturesetUpParams pressDuration;
        b2 = "";
        kony.print("" + b1 + "" + b2);
    }

    kony.print("widget id is: " + widgetID[id]); //will print the widgetID.
    //To print widgetID use widgetID.id
    kony.print("type of gestureInfo is: " + y);
    kony.print("type of gesturesetUpParams is: " + z);
    kony.print("gestureType is: " + a); //gestureType=1 or 2 or 3
    kony.print("gesturesetUpParams is: " + b.fingers);
  /*gesturesetUpParams
    = {
        fingers = 1, taps = 1
    }
    or {
        fingers = 1, taps = 2
    }
    or {
        fingers = 1
    }
    or {
        pressDuration = 1
    }*/
    kony.print("gesturePosition is: " + c); //gesturePosition=1 or 2 or 3 or
.....9
    kony.print("gestureX is: " + d); //ex: gestureX=30
    kony.print("gestureY is: " + e); //ex: gestureY=100
    kony.print("widgetWidth is: " + f); //ex: widgetWidth=320
    kony.print("widgetHeight is: " + g); //ex: widgetHeight=28
    //gesturePosition, gestureX, gestureY, widgetWidth, widgetHeight params
are not applicable in android
    kony.print("*******************************************");
}

function callbackSingleTapGesture() {
    var x = {
        fingers: 1,
        taps: 1
    };
    try {
        kony.application.setGestureRecognizerForAllForms(1, x,
            formGesture);
    } catch (err) {
        alert(typeof err);
        alert("error in function callbackSingleTapGesture: " + err.message);
    }
}
```

Return Values

String - Reference to the gesture is returned.

Platform Availability

Available on all platforms except Server Side Mobile Web, Windows 7/Kiosk, and Desktop Web.

kony.application.removeGestureRecognizerForAllForms

This method allows you to remove a specified gesture recognizer for all Forms.

Syntax

```
kony.application.removeGestureRecognizerForAllForms(uniqueIdentifier)
```

Input Parameters

Parameter	Description
uniqueIdentifier - Mandatory	Reference to the gesture. The reference to the gesture is returned by the setGestureRecognizerForAllForms.

Example

```
function callbackClearLongPressGesture() {
    try {
        kony.application.removeGestureRecognizerForAllForms(uniqueidentifier);
    } catch (err) {
        alert(typeof err);
        alert("error in function callbackClearLongPressGesture: " +
err.message);
    }
}
```

Platform Availability

Available on all platforms except Server Side Mobile Web, Windows 7/Kiosk, and Desktop Web.

27. Image API

The Image API gives your app image processing tools. The Image API consists of the functions and constants in both the kony.image and the kony.filter namespaces. In addition, it makes use of the image Object and the filter Object. With the Image API, you can create JavaScript Image objects, scale them, compress them, apply filters to them, and so forth. Your app can use Image objects on Image widgets and button widgets.

27.1 Overview

The Image API enables you to create Image objects that store bitmap data from GIF, animated GIF, JPEG, and PNG files. Image objects can be applied to widgets and used for button background, form backgrounds, and more.

Your app can also use Filter objects to apply visual filters, such as brightening an image or giving it an embossed look, to bitmap images that your app uses. The following example demonstrates how an app might apply a filter to an image and apply the image to a form.

The Image API provides your app with image processing tools. By using the Image API, you can scale them, compress them, apply filters to them, and so forth. Your app can use Image objects on Image and Button widgets.

The Image API enables you to create Image objects that store bitmap data from GIF, animated GIF, JPEG, and PNG files. Image objects can then be applied to widgets and used for button background, form backgrounds, and more.

Your app can also create Filter objects that are used to apply visual filters, such as brightening an image or giving an embossed look to bitmap images that your app uses. The following example demonstrates how an app might apply a filter to an image and apply the image to a form.

```
var filter = kony.filter.createFilter();
var img = kony.image.createImage("catpng.png");

filterData = {
    "filterName": kony.filter.HUE_ADJUST,
    "inputImage": img,
    "inputAngle": angle
};
filter.applyFilter(filterData);

var imageObj = filter.getOutputImage();
frm.image2.rawBytes = imageObj.getImageAsRawBytes();
```

As of version 8.0 of Kony Visualizer, the Image API supports SVG vector images. Vector images enable your app to zoom in on images without losing image quality. They also enable your app to use the same image files across multiple devices without a degradation of the image's appearance.

To supply a SVG file to your app, you must first embed it inside a PDF. You can then include the PDF in your app's bundle.

It is important to note that iOS does not provide native SVG support. iOS requires that you supply PNG bitmaps with multiple resolutions of all of your icons. If you use an SVG file for an icon, the system generates different PNG bitmaps at the required resolutions automatically.

All other hardware platforms support SVG files natively, so bitmaps are not generated for icon files on those platforms.

The Image API provides you with the following namespaces and API elements:

- image Object

Method	Description
`compress`	Compresses an image by the specified compression ratio.

Method	Description
cropToRect	Crops the bitmap contained by the Image object to the size of the input rectangle.
findImageInGallery	Searches for and retrieves and image in the device's gallery of pictures.
getImageAsRawBytes	Retrieves the image height as an integer.
getImageHeight	Retrieves the image height as an integer.
getImageWidth	Retrieves the image width as an integer.

Method	Description
releaseImage	Removes the internal image from the image object.
rotate	Rotates an imageObject either in a clockwise or counter-clockwise manner, depending on the specified rotation degree.
scale	Scales the bitmap in the current Image object to a larger or smaller size.
writeToMediaGallery	Writes an image to device's media gallery.

- filter Object

Method	Description
`applyFilter`	Applies a filter to an Image object.
`clearFilterData`	Clears all of the data stored in a filter.
`getOutputImage`	Gets the image that results from applying the filter.

- kony.filter Namespace

Function	Description
`kony.filter.createFilter`	Creates a new filter object for use with the Image widget.

- kony.image Namespace

Function	Description
`kony.image.createImage`	Creates an Image. This function has three overloads.
`kony.image.createImageFromSnapShot`	Creates an Image by taking a snapshot of a widget.
`kony.image.cropImageInTiles`	Crops the bitmap in an Image object and returns it as an array of tiles.

Function	Description
kony.image.cropImageInTilesForRects	Crops portions of an Image widget's bitmap to a set of rectangles and returns an array of Image widgets containg the cropped bitmaps.

To generate an image from raw bytes, bundled files, or image widget; use the kony.image.createImage function. After you create an image, you can crop the image using the kony.image.cropImageInTiles and the kony.image.cropImageInTilesForRects functions.

To apply visual filters such as brightening an image or giving it an embossed look, create a filter object by using the kony.filter.createFilter function. You can then apply a filter to an image object by using the applyFilter method. To view the filtered image, use the getOutputImage method. If you want to remove the filter, use the clearFilterData method.

To view the functionality of the Image API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

27.2 image Object

The image Object, not to be confused with the Image Widget, is a JavaScript object that retrieves images as raw bytes. It provides the following API element:

27.2.1 Methods

The image Object contains the following methods.

compress Method

Compresses an image by the specified compression ratio.

Syntax

```
<<imageObject>>.compress(compressionRatio)
```

Input Parameters

Parameter	Description
compressionRatio	A floating point value that specifies the amount of compression to use.

Example

```
var imgObj = kony.image.createImage(rawBytes);
imgObj.compress(0.8);
```

Return Values

None.

Remarks

This method compresses the Image object's bitmap using the JPEG compression algorithm. The floating point value in the *compressionRatio* parameter must be in the range of 0.0<=*compressionRatio*<=1.0. If *compressionRatio* is 0.0, this method uses the minimum amount of compression. A value of 1.0 specifies the maximum amount of compression. Values outside the allowed range will be clamped to the nearest valid value. The compression data size that this method produces will vary depending on the hardware platform.

Availability

Available on iOS and Android.

cropToRect Method

This method crops the bitmap contained by the Image object to the size of the input rectangle.

Syntax

```
cropToRect(
    array)
```

Input Parameters

Parameter	Description
array	An array of integers specifying the cropping rectangle in the order (x,y,width,height).

Example

```
var imgObj = kony.image.createImage(rawBytes);
imgObj.cropToRect([0, 0, 720, 720]);
```

Return Values

None.

Remarks

This method crops the current Image object's bitmap to the size of the rectangle specified in the *array* parameter, altering the bitmap in the process.

If there is no intersection between the Image object's bitmap and the rectangle in the *array* parameter, then no cropping is performed.

Availability

Available on iOS and Android.

findImageInGallery Method

Searches for and retrieves and image in the device's gallery of pictures.

Syntax

```
findImageInGallery(
    config)
```

Input Parameters

Parameter	Description
config	A JavaScript object containing the information needed to search for the image. This object holds the following key-value pairs. • albumName: An optional string that specifies the album to search. Not used on iOS. • imageName: A string that holds the file name (including the extension) of the image file to search for.

Example

```
var imgObj = kony.image.createImage("src.png");
imgObj.writeToGallery();

var uniqueImgIdentifier;

function onSuccess(uniqueIdentifier) {
    uniqueImgIdentifier = uniqueIdentifier;
}

config = {
    ImageName: uniqueImgIdentifier
};

var rawBytesObj = findImageInGallery(config);
```

Return Values

Returns an object of type kony.types.RawBytes that contains the RawBytes image data if the file exists, or `null` if the file is not found.

Exceptions

Value	Description
100	Either `albumName` or `imageName` was not of type String.

getImageAsRawBytes Method

Retrieves the image height as an integer.

Syntax

```
getImageAsRawBytes(
    encodingFormat)
```

Input Parameters

Parameter	Description
encodingFormat	A constant from the Image Format Constants in the kony.image namespace that specifies the format of the bitmap image.

Example

```
var imgobj = kony.image.createimage(rawbytes);
var imgobj = kony.image.createimage(form1.camera1.rawbytes);
imgobj.writetomediagallery();
var uniqueimgidentifier;

function onsuccess(uniqueidentifier) {
    uniqueimgidentifier = uniqueidentifier;
}
config = {
    imagename: uniqueimgidentifier
};

var rawbytesobj = kony.image.findimageingallery(config);
form1.img1.rawbytes = rawbytesobj;
```

Return Values

The Image object's bitmap in RawBytes format if an image format is specified. If not, this method returns the RawBytes data in a platform-specific formats.

Availability

Available on iOS and Android.

getImageHeight Method

Retrieves the image height as an integer.

Syntax

```
getImageHeight();
```

Example

```
var imgObj = kony.image.createImage(rawBytes);
var imgHeight = imgObj.getImageHeight();
```

```
kony.print("Image height is:" + imgHeight);
form1.img1.rawbytes = rawbytesobj;
```

Input Parameters

None.

Return Values

An integer that specifies the height of the Image.

Availability

Available on iOS and Android.

getImageWidth Method

Retrieves the image width as an integer.

Syntax

```
getImageWidth();
```

Example

```
var imgObj = kony.image.createImage(rawBytes);
var imgWidth = imgObj.getImageWidth();
kony.print("Image width is:" + imgWidth);
```

Input Parameters

None.

Return Values

An integer that specifies the object of the Image.

Availability

Available on iOS and Android.

releaseImage Method

Removes the internal image from the image object.

Syntax

```
<<imageObject>>.releaseImage()
```

Example

```
var imgObj = kony.image.createImage(rawB);
imgObj.releaseImage();
```

Input Parameters

None.

Return Values

None.

Availability

Available on iOS.

rotate Method

Rotates an imageObject either in a clockwise or counter-clockwise manner, depending on the specified rotation degree. In addition, you can use this API on Windows 10 platform to crop the edges of the rotated image based on the provided cropImage value.

Syntax

```
<<imageObject>>.rotate(degree, cropImage)
```

Input Parameters

Parameter	Description
degree [Number] - Mandatory	The degree by which the imageObject is to be rotated. You can specify any number for the degree parameter: positive or negative. • For positive number: rotation occurs in a clockwise manner. • For negative number: rotation occurs in a counter-clockwise manner. For example, rotate(90) ,rotate(-90) , rotate(355.5), and rotate (367.5).
cropImage [Boolean] - Optional	If cropImage is true, the rotated imageObject is cropped at the edges; otherwise, the imageObject is not cropped. The default value for cropImage is false. For example, rotate(45, true) and rotate(-145, false).

Example

```
//Rotate image without crop filter applied
var imageObject = kony.image.createImage("Image.png");
imageObject.rotate(45);

//Rotate image without crop filter applied
var imageObject = kony.image.createImage("Image.png");
imageObject.rotate(45, true);
```

Return Values

None.

Remarks

- The rotate API does not return a new rotated image, instead it rotates the received image.

Limitations

- The cropImage parameter is applicable only for the Windows 10 platform.

Platform Availability

- iOS
- Android
- Windows

scale Method

Scales the bitmap in the current Image object to a larger or smaller size.

Syntax

```
scale(scaleFactor)
```

Input Parameters

Parameter	Description
scaleFactor	A floating point number that is used to scale the bitmap to a larger or smaller size.

Example

```
var imgObj = kony.image.createImage(rawB);
imgObj.scale(0.4);
```

Return Values

None.

Remarks

The floating point number in the *scaleFactor* parameter cannot be less than zero. If it is in the range 0.0<=*scaleFactor*<1.0, the bitmap size will be reduced. Depending on the hardware and the size of the bitmap, distortion or blurring of the image can occur when it is reduced. If *scaleFactor* equals 1, this method does nothing.

When your app sets *scaleFactor* to a value greater than 1.0, the size of the bitmap increases. Values greater than 2.0 may result into memory warnings on some platforms. The resultant image quality may differ on platforms due to interpolation algorithms used.

Availability

Available on iOS and Android.

writeToMediaGallery Method

Writes an image to device's media gallery.

Syntax

```
writeToMediaGallery(
    config)
```

Input Parameters

config

Optional. A dictionary with configurable properties. If you do not specify the config parameter as an argument, the images will be written to the default public location based on the device's OS. You can pass the following properties in the config parameter.

Key	Description
albumName	Optional. A string that specifies a sub-folder name under the media gallery folder to save images into. You can make use of the property in the following cases: • You can directly define a name to the album. In this case, when you call the API, a folder with the name that you defined is created under the media gallery, and then the image is saved to the folder. • You can prompt end-user with a dialog box asking to give a name to the album or folder to save the image into it. Write a code to fetch the name defined by the end-user, and pass it as the albumName property. This creates a folder with the name defined by the end-user, and then the image gets saved to it. ***Note:*** If you do not specify the albumName property in the config parameter, the images will be written to the default public location based on the device's OS. On iOS devices, the images are saved to the `Camera Roll` folder. On Android and Windows devices, the images are saved to the `Pictures` folder. ***Note:*** If the value of the albumName key is not of String type, an exception is thrown with error code as '100' with the message "Invalid argument."

Key	Description
imageName	Optional. A string that specifies a name to an image with which the image should be written to the gallery. The image will be saved to the gallery with the given name without any extension. If any extension is given along with the image name, an exception is thrown with error code '100' with the message "Invalid argument." The cases defined for the albumName property is also applies to the imageName property. If no name is specified to the image, the SDK will give a name to the image, and then write to the gallery. The property is respected only in Windows and Android platforms. ***Note:*** If the value of the imageName key is not of String type, an exception is thrown with error code as '100' with the message "Invalid argument."
extensionType	Optional. A constant that specifies the file format type of the image in which the image should be saved to the gallery. The following are the file format constants that you can specify: • kony.image.ENCODE_JPEG: the image will be saved in JPEG format. • kony.image.ENCODE_PNG: the image will be saved in PNG format. The default value of the property is kony.image.ENCODE_JPEG. This parameter is available on all platforms.

Key	Description
handleRecoverableException [Optional]	A Boolean value that handles the **RecoverableSecurityException** that occurs when the overwrite parameter is used to overwrite an image that is owned by another app. ***Important:*** This parameter has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements. • `true`: Displays the system permission dialog box that requests confirmation to overwrite the image. If the user grants permission to overwrite, the image is overwritten. If the user denies permission to overwrite, the same RecoverableSecurityException will be passed to the error callback. • `false`: The RecoverableSecurityException is not handled and the default error message `(Failed to insert/update the image)` is passed to the error callback. The default value of the property is `false`. ***Note:*** This is an Android-specific parameter and is only applicable on Android 10 (and later) devices.

Key	Description
overwrite [Boolean]	A Boolean value that specifies whether or not to overwrite existing images. • `true`: overwrites the image if already exists with the name specified for a new image. • `false`: Appends time stamp to the specified image name if already an image exists with same name; then the image is saved to the gallery. The default value of the property is `false`. This parameter is available only for Android and Windows platforms. ***Note:*** If the value of the overwrite key is not of Boolean type, an exception is thrown with error code as '100' with the message "Invalid argument."
onSuccess	Optional. A callback function that is invoked when writing the image to the media gallery is successful. You can define your own logic in the callback function. For example, you can define an alert message stating "your photo saved successfully." This parameter is available on all platforms. On IOS, a local device-specific unique identifier (910E7DBE-1DB0-455F-93B3-4500AA93042F/L0/001) is a string of the written image from the media gallery. In case of Android and Windows platform, the image name is returned.

Key	Description

onFailure	Optional. A callback function is invoked when this function has failed to write an image to the media gallery. When it is invoked, the callback is passed a failure status and an error message.
	
	This parameter is available on all platforms.
	
	The failure status values can be one of the following.
	
	• kony.application.PERMISSION_DENIED: The app does not have required permissions to access the media gallery.
	• kony.image.SAVE_FAILED: The app failed to save an image to the media gallery.
	• kony.image.INSUFFICIENT_STORAGE: There is no enough space in the media gallery.
	• kony.image.SAVE_FAILED_RECOVERABLE: The app failed to overwrite an image that is owned by another app. This error occurs only on Android devices when the handleRecoverableException key is set to true.
	
	Example (onFailure)
	
	`if (statusOfFailure == kony.application.PERMISSION_DENIED) {`
	`    {`
	
	`    } else if (statusOfFailure == kony.image.SAVE_FAILED) {`
	
	`    } else if (statusOfFailure == kony.image.INSUFFICIENT_STORAGE) {`
	
	`    }`
	`    kony.print("reason for the failure" + errorMessage);`
	
	`}`

Example

```
var config =

{
    albumName: "MyAlbum",
    extensionType: kony.image.ENCODE_PNG,
    onSuccess: successCallback2,
    onFailure: failureCallback2
};
var imgName = "sample.png";
var img = kony.image.createImage(imgName);
img.writeToMediaGallery(config);
```

Return Values

None.

Remarks

You can make use of the `writeToImageGallery` function when end user wants to save an image from your app to device's gallery. The image gets saved to media gallery based on the properties you pass in the `config` parameter. If you do not pass the `config` parameter to the function, the image gets saved to the device's public location.

On iOS, there is no loss of image quality when you call the `writeToImageGallery` function.

Your app may require runtime permissions to access the device's media gallery. For more information on checking, requesting, and obtaining runtime permissions, please see Runtime Permissions API.

Limitations

iOS

To use the `writeToImageGallery` function for the iOS platform, open the the app's Info.plist and define the key NSPhotoLibraryUsageDescription. Add reason for accessing the media gallery as a string value to the key. Otherwise, the app crashes.

Android

The directory path of the primary external storage is dependent on the device. When your app calls the `writeToImageGallery` function , the function accesses the device's external storage to save the image. Generally, the external storage is an SD card inserted into the device that can store relatively large amount of data. There is also the possibility that the devices uses built-in storage that is distinct from the protected internal storage. The `writeToImageGallery` function uses the directory path of the external storage provided by the device's OS for saving images.

Platform Availability

Android

iOS 8 and later versions

Windows

27.3 Filter Object

The filter object enables your app to apply filters to the bitmaps contained in Image widgets. This section discusses the filter object for the following item:

27.3.1 Methods

The filter object contains the following methods.

applyFilter Method

This method applies a filter to an Image object.

Syntax

```
filter.applyFilter(
    dictionary)
```

Input Parameters

Parameter	Description
dictionary	A dictionary of attributes that contains the name of the filter and the filter's supporting data.

Example

```
var imgBlurBg = kony.image.createImage(rawB);
var filterObj = kony.filter.createFilter();
var filterData = {
    "filterName": kony.filter.GAUSSIAN_BLUR,
    "inputImage": imgBlurBg,
    "inputRadius": 10,
    "inputRotation": 90
};
filterObj.applyFilter(filterData);
```

```
//By using the below function, you can add a cartoon filter over your image.
  createFilterForiOS: function(){

  var imgBright =  kony.image.createImage(this.imageBytes);
  var filterobj =  kony.filter.createFilter();
  var filterData = {
    "filterName": kony.filter.COMIC_EFFECT,
    "inputImage": imgBright,
  };
  filterobj.applyFilter(filterData);
  var imageObj = filterobj.getOutputImage();
  this.view.imgBrite.rawBytes = imageObj.getImageAsRawBytes();
  this.view.forceLayout();
  }
```

Return Values

None.

Remarks

For information about the available filters and their required supporting data, please see the constants in the kony.filter namespace.

The following filters change the dimensions of the Image object.

- GaussianBlur
- BoxBlur
- DiskBlur
- MotionBlur
- ZoomBlur
- BumpDistortion
- CircularWrap
- AffineTransform

Platform Availability

Available on all platforms.

clearFilterData Method

Clears all of the data stored in a filter.

Syntax

```
filter.clearFilterData()
```

Example

```
var filterObj = kony.filter.createFilter();
filterObj.clearFilterData();
```

Input Parameters

None.

Return Values

None.

Platform Availability

Available on all platforms.

getOutputImage Method

Gets the image that results from applying the filter.

Syntax

```
filter.getOutputImage()
```

Example

```
var imgBlurBg = kony.image.createImage(rawB);
var filterObj = kony.filter.createFilter();
var filterData = {
    "filterName": kony.filter.GAUSSIAN_BLUR,
    "inputImage": imgBlurBg,
    "inputRadius": 10,
    "inputRotation": 90
};
filterObj.applyFilter(filterData);
var imageObj = filterObj.getOutputImage();
```

```
 var imgBright =  kony.image.createImage(this.imageBytes);
  var filterobj =  kony.filter.createFilter();
  var filterData = {
    "filterName": kony.filter.COMIC_EFFECT,
    "inputImage": imgBright,
  };
  filterobj.applyFilter(filterData);
  var imageObj = filterobj.getOutputImage();//displays
```

Input Parameters

None.

Return Values

The image that results from applying the filter.

Remarks

IF the output image is not yet available, this method throws an exception.

Platform Availability

Available on all platforms.

27.4 kony.filter Namespace

The kony.filter namespace provides your apps with the ability to create and manage filters for use with the Image widget. Information about the kony.filter namespace is presented in the following topics:

- Constants
- Functions

27.4.1 Functions

The kony.filter namespace provides the following function:

kony.filter.createFilter Function

Creates a new filter object for use with the Image widget.

Syntax

```
kony.filter.createFilter()
```

Example

```
var filterObj = kony.filter.createFilter();
```

Input Parameters

None.

Return Values

A new filter object.

Platform Availability

All platforms.

27.4.2 Constants

The kony.filter namespace contains the following constants:

Android Filters

The following filters are available on Android devices.

Filter Name	Filter Type	Description
AdditionCompositing	kony.filter.ADDITION_ COMPOSITING	This filter is available from Android 3.0 and above.
BoxBlur	kony.filter.BOX_BLUR	Increase or decreases the blurriness of image.
Brightness	kony.filter.BRIGHTNESS	Increases or decreases the brightness of an image.
ClearCompositing	kony.filter.CLEAR_COMPOSITING	This filter is available from Android 3.0 and above.
ColorClamp	kony.filter.COLOR_CLAMP	This filter clamps the colors of input image within the given range.

ColorInversion	kony.filter.COLOR_INVERSION	Inverts the colors of given image. When this filter is applied, for each pixel having [R,G,B] components in given image, this filter inverts the color components of those pixels to [255-R, 255-G, 255-B]. Transparency or Alpha component is untouched
ColorMatrix	kony.filter.COLOR_MATRIX	This filter transforms the colors of given image as per the input parameters
DarkenCompositing	kony.filter.DARKEN_ COMPOSITING	This filter is available from Android 3.0 and above.
EdgeDetection	kony.filter.EDGE_DETECTION	Detects or Highlight the edges of given image.
Emboss	kony.filter.EMBOSS	Applies an embossing effect for a given image.
GaussianBlur	kony.filter.GAUSSIAN_BLUR	Spreads source pixels by an amount specified by a Gaussian distribution to increase the blurriness. Blurriness of this filter is smooth compared to BoxBlur.
Greyscale	kony.filter.GREYSCALE	Converts given image to black & white or monochrome image.

MultiplyCompositing	kony.filter.MULTIPLY_ COMPOSITING	This filter is available from Android 3.0 and above.
LightenCompositing	kony.filter.LIGHTEN_ COMPOSITING	This filter is available from Android 3.0 and above.
OverlayCompositing	kony.filter.OVERLAY_ COMPOSITING	This filter is available from Android 3.0 and above.
SourceAtopCompositing	kony.filter.SOURCE_ATOP_ COMPOSITING	This filter is available from Android 3.0 and above.
ScreenCompositing	kony.filter.SCREEN_ COMPOSITING	This filter is available from Android 3.0 and above.
Sharpen	kony.filter.SHARPEN	Increases the sharpness of image.
SourceInCompositing	kony.filter.SOURCE_IN_ COMPOSITING	This filter is available from Android 3.0 and above.
SourceOutCompositing	kony.filter.SOURCE_OUT_ COMPOSITING	This filter is available from Android 3.0 and above.
SourceOverCompositing	kony.filter.SOURCE_OVER_ COMPOSITING	This filter is available from Android 3.0 and above.
XorCompositing	kony.filter.XOR_COMPOSITING	This filter is available from Android 3.0 and above.

Filter Input Parameters

The filter object uses the following input parameters.

Parameter	Description
inputImage	Image object
inputRadius	Floating point number
inputMask	Masking image
inputAngle	Floating point number
inputNoiseLevel	Floating point number
inputCenter	Array of two values defining the (x,y) location of the center
inputScale	Floating point number
inputAmount	Floating point number
inputSharpness	Floating point number
inputMinComponents	Array of four floating values
inputMaxComponents	Array of four floating values
inputSaturation	Floating point number
inputBrightness	Floating point number
inputIntensity	Floating point number
inputContrast	Floating point number
inputRVector	Array of four floats
inputGVector	Array of four floats

inputBVector	Array of four floats
inputAVector	Array of four floats
inputBiasVector	Array of four floats
inputRedCoefficients	Array of four floats
inputGreenCoefficients	Array of four floats
inputBlueCoefficients	Array of four floats
inputAlphaCoefficients	Array of four floats
inputEV	Floating point number
inputPower	Floating point number
inputNeutral	Array of two floating point numbers
inputTargetNeutral	Array of two floating point numbers
inputPoint0	Array of two floating point numbers
inputPoint1	Array of two floating point numbers
inputPoint2	Array of two floating point numbers
inputPoint3	Array of two floating point numbers
inputPoint4	Array of two floating point numbers
inputColor	Array of four numbers representing the color
inputRedCoefficients	Array of ten floating point numbers

inputGreenCoefficients	Array of ten floating point numbers
inputBlueCoefficients	Array of ten floating point numbers
inputGradientImage	Gradient Image
inputBackgroundImage	Background Image
inputLevels	Floating point number
inputColor0	Array of four hex numbers representing the color
inputColor1	Array of four hex numbers representing the color
inputInsetPoint0	Array of two floating point numbers
inputInsetPoint1	Array of two floating point numbers
inputStrands	Floating point number
inputPeriodicity	Floating point number
inputRotation	Floating point number
inputZoom	Floating point number
inputDisplacementImage	Image object
inputTexture	Image object
inputRefraction	Floating point number
inputCropAmount	Floating point number

inputCenterStretchAmount	Floating point number
inputWidth	Floating point number
inputTransform	Array of nine floats representing the transform
inputRectangle	Array of four floating point numbers
inputAspectRatio	Floating point number
inputTopLeft	Array of two floating point numbers
inputTopRight	Array of two floating point numbers
inputBottomRight	Array of two floating point numbers
inputBottomLeft	Array of two floating point numbers
inputExtent	Array of four floating point numbers
inputRadius0	Floating point number
inputRadius1	Floating point number
inputGCR	Floating point number
inputUCR	Floating point number
inputCount	Integer
inputHeight	Floating point number
inputHighLimit	Floating point number
inputLowLimit	Floating point number

inputMaskImage	Masking image
inputWeights	Array of floating point numbers
inputBias	Floating point number
inputUnsharpMaskRadius	Floating point number
inputUnsharpMaskIntensity	Floating point number
inputHighlightAmount	Floating point number
inputShadowAmount	Floating point number
inputNRNoiseLevel	Floating point number
inputNRSharpness	Floating point number
inputEdgeIntensity	Floating point number
inputThreshold	Floating point number
inputContrast	Floating point number
inputShadingImage	Image object
inputLightPosition	Array of three floating point numbers
inputLightPointsAt	Array of three floating point numbers
inputConcentration	Floating point number

Filter Type Constants

Filter Type	Parameters	Description
kony.filter.ADDITION_ COMPOSITING	filterName:- kony.filter.ADDITION_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	Adds color components to achieve a brightening effect. This filter is available from Android 3.0 and above. This filter is available in iOS 6 and above.
kony.filter.AFFINE_ TRANSFORM	filterName :- kony.filter.AFFINE_ TRANSFORM inputImage :- Image widget inputTransform:- transform object	This filter is available in iOS 5 and above.
kony.filter.BLEND_WITH_ MASK	filterName :- kony.filter.BLEND_WITH_ MASK inputImage :-Image widget inputBackgroundImage :- Image widget inputMaskImage :- Image widget	This filter is available in iOS 6 and above.

kony.filter.BLEND_WITH_ ALPHA_MASK	filterName :- kony.filter.BLEND_WITH_ ALPHA_MASK inputImage :- Image widget, inputBackgroundImage :- Image widget inputMaskImage :- Image widget	This filter is available in iOS 7 and above.
kony.filter.BLOOM	filterName :- kony.filter.BLOOM inputImage :- Image widget inputRadius :- Number inputIntensity: Number	This filter is available in iOS 6 and above.
kony.filter.BOX_BLUR	filterName:- kony.filter.BOX_ BLUR inputImage:- Image widget inputRadius:- number(1 to 25)	Increase or decreases the blurriness of image. Available on Android. This filter is available in iOS 9 and above.
kony.filter.BRIGHTNESS	inputImage :- Image widget inputBrightness :- number (0.0 to infinite)	Increases or decreases the brightness of an image. Available on Android.

kony.filter.COLOR_BLEND_ MODE	filterName :- kony.filter.COLOR_BLEND_ MODE inputImage :- Image widget inputBackgroundImage :- Image widget	Uses the luminance values of the background with the hue and saturation values of the source image. This filter is available in iOS 6 and above.
kony.filter.CLEAR_ COMPOSITING	filterName:- kony.filter.CLEAR_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above.

kony.filter.COLOR_CLAMP	filterName:- kony.filter.COLOR_CLAMP inputImage:- Image widget inputMinComponents:- array of four floats representing [R,G,B,A] in that order where each float ranges between 0.0 to 1.0. inputMaxComponents:- array of four floats representing [R,G,B,A] in that order where each float ranges between 0.0 to 1.00	This filter clamps the colors of the input image within the given range specified by inputMinComponents and inputMaxComponents. At each pixel, color component values less than those in inputMinComponents will be increased to match those in inputMinComponents, and color component values greater than those in inputMaxComponents will be decreased to match those in inputMaxComponents. Available on Android. This filter is available in iOS 7 and above.
kony.filter.COLOR_ CONTROLS	filterName :- kony.filter.COLOR_ CONTROLS inputImage :- Image widget inputSaturation:- number (0.0 - 1.0) inputBrightness:- number (0.0 - 1.0) inputContrast:- number (0.0 - 1.0)	Adjusts saturation, brightness, and contrast values. This filter is available in iOS 5 and above.

kony.filter.COLOR_CROSS_ POLYNOMIAL	filterName :- kony.filter.COLOR_CROSS_ POLYNOMIAL inputImage :- Image widget inputRedCoefficients:- Array of 10 numbers [x,y,z,w,...] (0.0 - 1.0) inputGreenCoefficients:- Array of 10 numbers [x,y,z,w,....] (0.0 - 1.0) inputBlueCoefficients:- Array of 10 numbers [x,y,z,w,.....] (0.0 - 1.0)	Modifies the pixel values in an image by applying a set of polynomial cross-products. This filter is available in iOS 6 and above.
kony.filter.COLOR_DODGE_ BLEND_MODE	filterName :- kony.filter.COLOR_DODGE_ BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	Brightens the background image samples to reflect the source image samples. This filter is available in iOS 6 and above.
kony.filter.COLOR_ INVERSION	filterName:- kony.filter.COLOR_ INVERSION inputImage:- Image widget	Inverts the colors of given image. Available on Android. Available on iOS 6 and above.

kony.filter.COLOR_MAP	filterName :- kony.filter.COLOR_MAP inputImage :- Image widget inputGradientImage :- Image widget	Performs a nonlinear transformation of source color values using mapping values provided in a table. inputGradientImage is an nx1 or nxn gradient image used to describe a color ramp. This filter is available in iOS 6 and above.
kony.filter.COLOR_MATRIX	filterName :- kony.filter.COLOR_MATRIX inputImage :- Image widget inputRVector:- Array of four numbers [x,y,z,w] (0.0 - 1.0) inputGVector:- Array of four numbers [x,y,z,w] (0.0 - 1.0) inputBVector:- Array of four numbers [x,y,z,w] (0.0 - 1.0) inputAVector:- Array of four numbers [x,y,z,w] (0.0 - 1.0) inputBiasVector:- Array of four numbers [x,y,z,w] (0.0 - 1.0)	Multiplies source color values and adds a bias factor to each color component. Available on iOS 5 and above.

kony.filter.COLOR_MATRIX	filterName:- kony.filter.COLOR_MATRIX
	
	inputImage:- Image widget
	
	inputRVector:- array of four numbers representing [R,G,B,A] in that order.
	
	inputGVector:- array of four numbers representing [R,G,B,A] in that order.
	
	inputBVector:- array of four numbers representing [R,G,B,A] in that order.
	
	inputAVector:- array of four numbers representing [R,G,B,A] in that order.
	
	inputBiasVector:- array of four values where each value should be added to each channel of RGBA in that order.
		This filter transforms the colors of given image as per the input parameters. For a given color [R,G,B,A] of a pixel in the input image, the resulting color [rR, rG, rB, rA] is computed as follows:
		
		rR = R*Rv[0] + G*Rv[1] + B*Rv[2] + A*Rv[3] + bias [0]
		
		rG = R*Gv[0] + G*Gv[1] + B*Gv[2] + A*Gv[3] + bias [1]
		
		rB = R*Bv[0] + G*Bv[1] + B*Bv[2] + A*Bv[3] + bias [2]
		
		rA = R*Av[0] + G*Av[1] + B*Av[2] + A*Av[3] + bias [3]
		
		where Rv is inputRVector, Gv is inputGVector, Bv is inputBVector, Av is inputAVector and bias is inputBiasVector. The resulting color values [rR, rG, rB, rA] are clamped to 255 if exceeded beyond 255. Available on Android.

kony.filter.COLOR_ MONOCHROME	filterName :- kony.filter.COLOR_ MONOCHROME inputImage :- Image widget inputColor :- Array of three numbers [r,g,b] (0.0 - 1.0) inputIntensity:- Number	Remaps colors so they fall within shades of a single color. As inputColor only opaque color needs to be passed, ie, only red, blue, green components and no alpha. inputColor values should be in the range of 0.0 - 1.0. inputIntensity accepts the values in range 0.0 - 1.0. This filter is available on iOS 6 and above.
kony.filter.COLOR_ POLYNOMIAL	filterName :- kony.filter.COLOR_ POLYNOMIAL inputImage :- Image widget inputRedCoefficients:- Array of four numbers [x,y,z,w] (0.0 - 1.0) inputGreenCoefficients:- Array of four numbers [x,y,z,w] (0.0 - 1.0) inputBlueCoefficients:- Array of four numbers [x,y,z,w] (0.0 - 1.0) inputAlphaCoefficients:- Array of four numbers [x,y,z,w] (0.0 - 1.0)	Modifies the pixel values in an image by applying a set of cubic polynomials. This filter is available in iOS 7 and above.

kony.filter.COLOR_ POSTERIZE	filterName :- kony.filter.COLOR_ POSTERIZE inputImage :- Image widget inputLevels :- Number	Remaps red, green, and blue color components to the number of brightness values you specify for each color component. This filter is available on iOS 6 and above.
kony.filter.COMIC_EFFECT	filterName :- kony.filter.COMIC_EFFECT inputImage :- Image widget	This filter is available on iOS 9 and above.
kony.filter.CONVOLUTION_ 3X3	filterName :- kony.filter.CONVOLUTION_ 3X3 inputImage :- Image widget inputWeights :- Array of 9 numbers (0.0 - 1.0) inputBias :- number (0.0 - 1.0)	This filter is available on iOS 7 and above.
kony.filter.CONVOLUTION_ 5X5	filterName :- kony.filter.CONVOLUTION_ 5X5 inputImage :- Image widget inputWeights :- Array of 25 numbers (0.0 - 1.0) inputBias :- number (0.0 - 1.0)	This filter is available on iOS 7 and above.

kony.filter.CONVOLUTION_7X7	filterName :- kony.filter.CONVOLUTION_7X7 inputImage :- Image widget, inputWeights :- Array of 7x7 = 49 numbers (0.0 - 1.0) inputBias :- number (0.0 - 1.0)	This filter is available on iOS 9 and above.
kony.filter.CONVOLUTION_9_HORIZONTAL	filterName :- kony.filter.CONVOLUTION_9_HORIZONTAL inputImage :- Image widget, inputWeights :- Array of 9 numbers (0.0 - 1.0) inputBias :- number (0.0 - 1.0)	This filter is available on iOS 7 and above.
kony.filter.CONVOLUTION_9_VERTICAL	filterName :- kony.filter.CONVOLUTION_9_VERTICAL inputImage :- Image widget, inputWeights :- Array of 9 numbers (0.0 - 1.0) inputBias :- number (0.0 - 1.0)	This filter is available on iOS 7 and above.

© 2013 by Kony, Inc. All rights reserved

kony.filter.CRYSTALLIZE	filterName :- kony.filter.CRYSTALLIZE inputImage :- Image widget, inputRadius:- number inputCenter :- Array of two numbers [x,y]	This filter is available on iOS 9 and above.
kony.filter.DARKEN_BLEND_MODE	filterName :- kony.filter.DARKEN_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	Creates composite image samples by choosing the darker samples. This filter is available on iOS 6 and above.
kony.filter.DARKEN_COMPOSITING	filterName:- kony.filter.DARKEN_COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above.
kony.filter.DIFFERENCE_BLEND_MODE	filterName :- kony.filter.DIFFERENCE_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	Subtracts the greater brightness values from lower. This filter is available on iOS 6 and above.

kony.filter.DISC_BLUR	filterName :- kony.filter.DISC_BLUR inputImage :- Image widget inputRadius:- number	This filter is available on iOS 9 and above.
kony.filter.DIVIDE_BLEND_MODE	filterName :- kony.filter.DIVIDE_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	Divides the backgroundImage color from source image color. This filter is available on iOS 8 and above.
kony.filter.EDGES	filterName :- kony.filter.EDGES inputImage :- Image widget, inputIntensity :- number	This filter is available on iOS 9 and above.
kony.filter.EDGE_WORK	filterName :- kony.filter.EDGE_WORK inputImage :- Image widget, inputRadius :- number	This filter is available on iOS 9 and above.
kony.filter.EXCLUSION_BLEND_MODE	filterName :- kony.filter.EXCLUSION_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	Producess the similar effect as CIDifferenceBlend mode but has lower contrast.This filter is available on iOS 6 and above.

kony.filter.EXPOSURE_ ADJUST	filterName :- kony.filter.EXPOSURE_ ADJUST inputImage :- Image widget inputEV:- number (0.0 - 1.0)	Adjusts the exposure setting for an image similar to the way you control exposure for a camera when you change the F-stop. This filter is available on iOS 5 and above.
kony.filter.EDGE_ DETECTION	filterName:- kony.filter.EDGE_ DETECTION inputImage:- Image widget inputRadius:- number(1 or 2)	Detects or Highlight the edges of given image. Available on Android.
kony.filter.EMBOSS	filterName:- kony.filter.EMBOSS inputImage:- Image widget inputRadius:- number(1 or 2)	Applies an embossing effect for a given image. Available on Android.
kony.filter.FALSE_COLOR	filterName :- kony.filter.FALSE_COLOR inputImage :- Image widget inputColor0 :- Array of four numbers of 0.0 - 1.0 [r,g,b,a] inputColor1 :- Array of four numbers of 0.0 - 1.0 [r,g,b,a]	Maps luminance to a color ramp of two colors. This filter is available in iOS 6 and above.

kony.filter.GAMMA_ADJUST	filterName :- kony.filter.GAMMA_ADJUST inputImage :- Image widget inputPower:- number (0.0 - 1.0)	This filter is available in iOS 5 and above.
kony.filter.GAUSSIAN_BLUR	filterName:- kony.filter.GAUSSIAN_BLUR inputImage:- Image widget inputRadius:- number(1 to 25)	Spreads source pixels by an amount specified by a Gaussian distribution to increase the blurriness. The blurriness of this filter is smooth compared to BoxBlur. Available on Android. Available on iOS 6 and above.
kony.filter.GLOOM	filterName :- kony.filter.GLOOM inputImage :- Image widget inputRadius :- Number inputIntensity: Number	This filter is available in iOS 6 and above.
kony.filter.GREYSCALE	filterName:- kony.filter.GREYSCALE inputImage:- Image widget	Converts given image to black & white or monochrome image. Available on Android.

kony.filter.HARD_LIGHT_BLEND_MODE	filterName :- kony.filter.HARD_LIGHT_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	If input image sample is lighter than 50% grey then backgroundImage sample is lightened. If input image sample is greater that 50% grey then backgroundImage sample is darkened. This filter is available in iOS 6 and above.
kony.filter.HUE_ADJUST	filterName :- kony.filter.HUE_ADJUST inputImage :- Image widget inputAngle:- number (angle in Radians)	Changes the overall hue, or tint, of the source pixels. This filter is available in iOS 5 and above.
kony.filter.HUE_BLEND_MODE	filterName :- kony.filter.HUE_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	Uses luminance and saturation values from backgroundImage with the hue values on input image. This filter is available in iOS 6 and above.
kony.filter.LIGHTEN_BLEND_MODE	filterName :- kony.filter.LIGHTEN_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	The lighter samples are taken from both the images and will be used in resultant image. This filter is available in iOS 6 and above.

kony.filter.LIGHTEN_COMPOSITING	filterName:- kony.filter.LIGHTEN_COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above.
kony.filter.LINEAR_BURN_BLEND_MODE	filterName :- kony.filter.LINEAR_BURN_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 8 and above.
kony.filter.LINEAR_DODGE_BLEND_MODE	filterName :- kony.filter.LINEAR_DODGE_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 8 and above.
kony.filter.LINEAR_TO_SRGB_TONE_CURVE	filterName :- kony.filter.LINEAR_TO_SRGB_TONE_CURVE inputImage :- Image widget	This filter is available in iOS 7 and above.

kony.filter.LUMINOSITY_ BLEND_MODE	filterName :- kony.filter.LUMINOSITY_ BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.
kony.filter.MASK_TO_ALPHA	filterName :- kony.filter.MASK_ TO_ALPHA inputImage :- Image widget	Converts a grayscale image to a white image that is masked by alpha. This filter is available in iOS 6 and above.
kony.filter.MAXIMUM_ COMPONENT	filterName :- kony.filter.MAXIMUM_ COMPONENT inputImage :- Image widget	Returns a grayscale image from max(r,g,b). This filter is available in iOS 6 and above.
kony.filter.MAXIMUM_ COMPOSITING	filterName :- kony.filter.MAXIMUM_ COMPOSITING inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.
kony.filter.MEDIAN_FILTER	filterName :- kony.filter.MEDIAN_FILTER inputImage :- Image widget	This filter is available in iOS 9 and above.

kony.filter.MINIMUM_ COMPONENT	filterName :- kony.filter.MINIMUM_ COMPONENT inputImage :- Image widget	This filter is available in iOS 5 and above.
kony.filter.MINIMUM_ COMPOSITING	filterName :- kony.filter.MINIMUM_ COMPOSITING inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.
kony.filter.MOTION_BLUR	filterName :- kony.filter.MOTION_BLUR inputImage :- Image widget inputRadius:- number inputAngle:- number (angle in radians)	This filter is available in iOS 9 and above.
kony.filter.MULTIPLY_ BLEND_MODE	filterName :- kony.filter.MULTIPLY_ BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.

kony.filter.MULTIPLY_ COMPOSITING	filterName:- kony.filter.MULTIPLY_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above. This filter is available in iOS 6 and above.
kony.filter.NOISE_ REDUCTION	filterName :- kony.filter.NOISE_ REDUCTION inputImage :- Image widget inputNoiseLevel:- number (0.0 - 1.0) inputSharpness:- number (0.0 - 1.0)	This filter is available in iOS 9 and above.
kony.filter.OVERLAY_ BLEND_MODE	filterName :- kony.filter.OVERLAY_ BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.
kony.filter.OVERLAY_ COMPOSITING	filterName:- kony.filter.OVERLAY_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above.

kony.filter.PHOTO_EFFECT_CHROME	filterName :- kony.filter.PHOTO_EFFECT_CHROME inputImage :- Image widget	Applies a preconfigured set of effects that imitate vintage photography film with exaggerated color.This filter is available in iOS 6 and above.
kony.filter.PHOTO_EFFECT_FADE	filterName :- kony.filter.PHOTO_EFFECT_FADE inputImage :- Image widget	Applies a preconfigured set of effects that imitate vintage photography film with diminished color. This filter is available in iOS 7 and above.
kony.filter.PHOTO_EFFECT_INSTANT	filterName :- kony.filter.PHOTO_EFFECT_INSTANT inputImage :- Image widget	Applies a preconfigured set of effects that imitate vintage photography film with distorted colors. This filter is available in iOS 7 and above.
kony.filter.PHOTO_EFFECT_MONO	filterName :- kony.filter.PHOTO_EFFECT_MONO inputImage :- Image widget	Applies a preconfigured set of effects that imitate black-and-white photography film with low contrast. This filter is available in iOS 7 and above.
kony.filter.PHOTO_EFFECT_PROCESS	filterName :- kony.filter.PHOTO_EFFECT_PROCESS inputImage :- Image widget	Applies a preconfigured set of effects that imitate vintage photography film with emphasized cool colors. This filter is available in iOS 7 and above.

kony.filter.PHOTO_EFFECT_ TONAL	filterName :- kony.filter.PHOTO_EFFECT_ TONAL inputImage :- Image widget	Applies a preconfigured set of effects that imitate black-and-white photography film without significantly altering contrast.This filter is available in iOS 7 and above.
kony.filter.PHOTO_EFFECT_ TRANSFER	filterName :- kony.filter.PHOTO_EFFECT_ TRANSFER inputImage :- Image widget	Applies a preconfigured set of effects that imitate vintage photography film with emphasized warm colors. This filter is available in iOS 7 and above.
kony.filter.PIN_LIGHT_ BLEND_MODE	filterName :- kony.filter.PIN_ LIGHT_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 8 and above.
kony.filter.SATURATION_ BLEND_MODE	filterName :- kony.filter.SATURATION_ BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.

kony.filter.SCREEN_BLEND_ MODE	filterName :- kony.filter.SCREEN_BLEND_ MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.
kony.filter.SCREEN_ COMPOSITING	filterName:- kony.filter.SCREEN_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above.
kony.filter.SEPIA_TONE	filterName :- kony.filter.SEPIA_TONE inputImage :- Image widget inputInternsity:- number	Maps the colors of an image to various shades of brown. This filter is available in iOS 5 and above.
kony.filter.SHARPEN	filterName:- kony.filter.SHARPEN inputImage:- Image widget inputRadius:- number(1 or 2)	Increases the sharpness of image. Available on Android.
kony.filter.SOFT_LIGHT_ BLEND_MODE	filterName :- kony.filter.SOFT_ LIGHT_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 6 and above.

kony.filter.SOURCE_ATOP_ COMPOSITING	filterName:- kony.filter.SOURCE_ATOP_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above. This filter is available in iOS 6 and above.
kony.filter.SOURCE_IN_ COMPOSITING	filterName:- kony.filter.SOURCE_IN_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above. This filter is available in iOS 6 and above.
kony.filter.SOURCE_OUT_ COMPOSITING	filterName:- kony.filter.SOURCE_OUT_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above. This filter is available in iOS 6 and above.
kony.filter.SOURCE_OVER_ COMPOSITING	filterName:- kony.filter.SOURCE_OVER_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above. This filter is available in iOS 5 and above.

kony.filter.SRGB_TONE_CURVE_TO_LINEAR	filterName :- kony.filter.SRGB_TONE_CURVE_TO_LINEAR inputImage :- Image widget	This filter is available in iOS 7 and above.
kony.filter.SUBTRACT_BLEND_MODE	filterName :- kony.filter.SUBTRACT_BLEND_MODE inputImage :- Image widget inputBackgroundImage :- Image widget	This filter is available in iOS 8 and above.
kony.filter.TEMPERATURE_AND_TINT	filterName :- kony.filter.TEMPERATURE_AND_TINT inputImage :- Image widget inputNeutral:- Array of two numbers [x,y] inputTargetNeutral:- Array of two numbers [x,y]	Adapts the reference white point for an image. This filter is available in iOS 5 and above.

kony.filter.TONE_CURVE	filterName :- kony.filter.TONE_ CURVE inputImage :- Image widget inputPoint0:- Array of two numbers [x,y](0.0 - 1.0) inputPoint1:- Array of two numbers [x,y](0.0 - 1.0) inputPoint2:- Array of two numbers [x,y](0.0 - 1.0) inputPoint3:- Array of two numbers [x,y](0.0 - 1.0) inputPoint4:- Array of two numbers [x,y](0.0 - 1.0)	Adjusts tone response of the R, G, and B channels of an image. This filter is available in iOS 5 and above.
kony.filter.VIGNETTE	filterName :- kony.filter.VIGNETTE inputImage :- Image widget inputRadius :- number inputInternsity:- number (0.0 - 1.0)	Reduces the brightness of an image at the periphery. This filter is available in iOS 5 and above.

kony.filter.VIGNETTE_ EFFECT	filterName :- kony.filter.VIGNETTE_ EFFECT inputImage :- Image widget inputCenter :- Array of two numbers [x,y] inputIntensity :- number (0.0 - 1.0) inputRadius:- number	Modifies the brightness of an image around the periphery of a specified region. This filter is available in iOS 7 and above.
kony.filter.WHITE_POINT_ ADJUST	filterName :- kony.filter.WHITE_POINT_ ADJUST inputImage :- Image widget inputColor:- Array of four numbers [0.0 - 1.0]	Adjusts the reference white point for an image and maps all colors in the source using the new reference. This filter is available in iOS 5 and above.
kony.filter.XOR_ COMPOSITING	filterName:- kony.filter.XOR_ COMPOSITING inputImage:- Image widget inputBackgroundImage:- Image widget	This filter is available from Android 3.0 and above.

kony.filter.ZOOM_BLUR	filterName :- kony.filter.ZOOM_BLUR inputImage :- Image widget inputCenter:- Array of two numbers [x,y] inputAmount:- number (default : 20.0)	This filter is available in iOS 9 and above.

iOS Filters

The following filters are available on iOS devices.

Filter Name	Filter Type	Description
CIAdditionCompositing	kony.filter.ADDITION_ COMPOSITING	Adds color components to achieve a brightening effect. This filter is available in iOS 6 and above.
CIAffineTransform	kony.filter.AFFINE_TRANSFORM	This filter is available in iOS 5 and above.
CIBlendWithMask	kony.filter.BLEND_WITH_MASK	This filter is available in iOS 6 and above.
CIBlendWithAlphaMask	kony.filter.BLEND_WITH_ ALPHA_MASK	This filter is available in iOS 7 and above.
CIBloom	kony.filter.BLOOM	This filter is available in iOS 6 and above.

CIBoxBlur	kony.filter.BOX_BLUR	This filter is available in iOS 9 and above.
CIColorBlendMode	kony.filter.COLOR_BLEND_MODE	Uses the luminance values of the background with the hue and saturation values of the source image. This filter is available in iOS 6 and above.
CIColorClamp	kony.filter.COLOR_CLAMP	This filter is available in iOS 7 and above.
CIColorControls	kony.filter.COLOR_CONTROLS	Adjusts saturation, brightness, and contrast values. This filter is available in iOS 5 and above.
CIColorCrossPolynomial	kony.filter.COLOR_CROSS_POLYNOMIAL	Modifies the pixel values in an image by applying a set of polynomial cross-products. This filter is available in iOS 7 and above.
CIColorDodgeBlendMode	kony.filter.COLOR_DODGE_BLEND_MODE	Brightens the background image samples to reflect the source image samples. This filter is available in iOS 6 and above.

CIColorInvert	kony.filter.COLOR_INVERSION	Inverts the colors in an image. This filter is available in iOS 6 and above.
CIColorMap	kony.filter.COLOR_MAP	Performs a nonlinear transformation of source color values using mapping values provided in a table. inputGradientImage is an nx1 or nxn gradient image used to describe a color ramp. This filter is available in iOS 6 and above.
CIColorMatrix	kony.filter.COLOR_MATRIX	Multiplies source color values and adds a bias factor to each color component. This filter is available in iOS 5 and above.
CIColorMonochrome	kony.filter.COLOR_ MONOCHROME	Remaps colors so they fall within shades of a single color. This filter is available in iOS 6 and above.
CIColorPolynomial	kony.filter.COLOR_POLYNOMIAL	Modifies the pixel values in an image by applying a set of cubic polynomials. This filter is available in iOS 6 and above.

CIColorPosterize	kony.filter.COLOR_POSTERIZE	Remaps red, green, and blue color components to the number of brightness values you specify for each color component. This filter is available on iOS 6 and above.
CIComicEffect	kony.filter.COMIC_EFFECT	This filter is available on iOS 9 and above.
CIConvolution3X3	kony.filter.CONVOLUTION_3X3	This filter is available on iOS 7 and above.
CIConvolution5X5	kony.filter.CONVOLUTION_5X5	This filter is available on iOS 7 and above.
CIConvolution7X7	kony.filter.CONVOLUTION_7X7	This filter is available on iOS 9 and above.
CIConvolution9Horizontal	kony.filter.CONVOLUTION_9_ HORIZONTAL	This filter is available on iOS 7 and above.
CIConvolution9Vertical	kony.filter.CONVOLUTION_9_ VERTICAL	This filter is available on iOS 7 and above.
CICrystallize	kony.filter.CRYSTALLIZE	This filter is available on iOS 9 and above.
CIDarkenBlendMode	kony.filter.DARKEN_BLEND_ MODE	Creates composite image samples by choosing the darker samples. This filter is available on iOS 6 and above.

CIDifferenceBlendMode	kony.filter.DIFFERENCE_ BLEND_MODE	Subtracts the greater brightness values from lower. This filter is available on iOS 6 and above.
CIDiscBlur	kony.filter.DISC_BLUR	This filter is available on iOS 9 and above.
CIDivideBlendMode	kony.filter.DIVIDE_BLEND_ MODE	Divides the backgroundImage color from source image color. This filter is available on iOS 8 and above.
CIEdges	kony.filter.EDGES	This filter is available on iOS 9 and above.
CIEdgeWork	kony.filter.EDGE_WORK	This filter is available on iOS 9 and above.
CIExclusionBlendMode	kony.filter.EXCLUSION_BLEND_ MODE	Produces an effect similar to CIDifferenceBlend mode but has lower contrast. This filter is available on iOS 6 and above.
CIExposureAdjust	kony.filter.EXPOSURE_ADJUST	Adjusts the exposure setting for an image similar to the way you control exposure for a camera when you change the F-stop. This filter is available in iOS 5 and above.

CIFalseColor	kony.filter.FALSE_COLOR	Maps luminance to a color ramp of two colors. This filter is available in iOS 6 and above.
CIGammaAdjust	kony.filter.GAMMA_ADJUST	This filter is available in iOS 5 and above.
CIGaussianBlur	kony.filter.GAUSSIAN_BLUR	Spreads source pixels by an amount specified by a Gaussian distribution to increase the blurriness. This filter is available in iOS 6 and above.
CIGloom	kony.filter.GLOOM	This filter is available in iOS 6 and above.
CIHardLightBlendMode	kony.filter.HARD_LIGHT_ BLEND_MODE	If input image sample is lighter than 50% grey then backgroundImage sample is lightened. If input image sample is greater that 50% grey then backgroundImage sample is darkened. This filter is available in iOS 6 and above.
CIHueAdjust	kony.filter.HUE_ADJUST	Changes the overall hue, or tint, of the source pixels. This filter is available in iOS 5 and above.

CIHueBlendMode	kony.filter.HUE_BLEND_MODE	Uses luminance and saturation values from backgroundImage with the hue values on input image. This filter is available in iOS 6 and above.
CILightenBlendMode	kony.filter.LIGHTEN_BLEND_MODE	The lighter samples are taken from both the images and will be used in resultant image. This filter is available in iOS 7 and above.
CILinearBurnBlendMode	kony.filter.LINEAR_BURN_BLEND_MODE	This filter is available in iOS 8 and above.
CILinearDodgeBlendMode	kony.filter.LINEAR_DODGE_BLEND_MODE	This filter is available in iOS 8 and above.
CILinearToSRGBToneCurve	kony.filter.LINEAR_TO_SRGB_TONE_CURVE	This filter is available in iOS 6 and above.
CILuminosityBlendMode	kony.filter.LUMINOSITY_BLEND_MODE	This filter is available in iOS 6 and above.
CIMaskToAlpha	kony.filter.MASK_TO_ALPHA	Converts a grayscale image to a white image that is masked by alpha. This filter is available in iOS 6 and above.

CIMaximumComponent	kony.filter.MAXIMUM_ COMPONENT	Returns a grayscale image from max(r,g,b). This filter is available in iOS 6 and above.
CIMaximumCompositing	kony.filter.MAXIMUM_ COMPOSITING	This filter is available in iOS 6 and above.
CIMedianFilter	kony.filter.MEDIAN_FILTER	This filter is available in iOS 9 and above.
CIMinimumComponent	kony.filter.MINIMUM_ COMPONENT	Returns a grayscale image from min(r,g,b). This filter is available in iOS 6 and above.
CIMinimumCompositing	kony.filter.MINIMUM_ COMPOSITING	This filter is available in iOS 6 and above.
CIMotionBlur	kony.filter.MOTION_BLUR	This filter is available in iOS 9 and above.
CIMultiplyBlendMode	kony.filter.MULTIPLY_BLEND_ MODE	This filter is available in iOS 6 and above.
CIMultiplyCompositing	kony.filter.MULTIPLY_ COMPOSITING	This filter is available in iOS 6 and above.
CINoiseReduction	kony.filter.NOISE_REDUCTION	This filter is available in iOS 9 and above.
CIOverlayBlendMode	kony.filter.OVERLAY_BLEND_ MODE	This filter is available in iOS 6 and above.

CIPhotoEffectChrome	kony.filter.PHOTO_EFFECT_CHROME	Applies a preconfigured set of effects that imitate vintage photography film with exaggerated color.This filter is available in iOS 6 and above.
CIPhotoEffectFade	kony.filter.PHOTO_EFFECT_FADE	Applies a preconfigured set of effects that imitate vintage photography film with diminished color. This filter is available in iOS 7 and above.
CIPhotoEffectInstant	kony.filter.PHOTO_EFFECT_INSTANT	Applies a preconfigured set of effects that imitate vintage photography film with distorted colors. This filter is available in iOS 7 and above.
CIPhotoEffectMono	kony.filter.PHOTO_EFFECT_MONO	Applies a preconfigured set of effects that imitate black-and-white photography film with low contrast. This filter is available in iOS 7 and above.

CIPhotoEffectNoir	kony.filter.PHOTO_EFFECT_NOIR	Applies a preconfigured set of effects that imitate black-and-white photography film with exaggerated contrast. This filter is available in iOS 7 and above.
CIPhotoEffectProcess	kony.filter.PHOTO_EFFECT_PROCESS	Applies a preconfigured set of effects that imitate vintage photography film with emphasized cool colors. This filter is available in iOS 7 and above.
CIPhotoEffectTonal	kony.filter.PHOTO_EFFECT_TONAL	Applies a preconfigured set of effects that imitate black-and-white photography film without significantly altering contrast. This filter is available in iOS 7 and above.
CIPhotoEffectTransfer	kony.filter.PHOTO_EFFECT_TRANSFER	Applies a preconfigured set of effects that imitate vintage photography film with emphasized warm colors. This filter is available in iOS 7 and above.
CIPinLightBlendMode	kony.filter.PIN_LIGHT_BLEND_MODE	This filter is available in iOS 8 and above.

CISaturationBlendMode	kony.filter.SATURATION_BLEND_MODE	This filter is available in iOS 6 and above.
CIScreenBlendMode	kony.filter.SCREEN_BLEND_MODE	This filter is available in iOS 6 and above.
CISoftLightBlendMode	kony.filter.SOFT_LIGHT_BLEND_MODE	This filter is available in iOS 6 and above.
CISourceAtopCompositing	kony.filter.SOURCE_ATOP_COMPOSITING	This filter is available in iOS 6 and above.
CISourceInCompositing	kony.filter.SOURCE_IN_COMPOSITING	This filter is available in iOS 6 and above.
CISourceOutCompositing	kony.filter.SOURCE_OUT_COMPOSITING	This filter is available in iOS 6 and above.
CISRGBToneCurveToLinear	kony.filter.SRGB_TONE_CURVE_TO_LINEAR	Maps color intensity from the sRGB color space to a linear gamma curve. This filter is available in iOS 7 and above.
CISubtractBlendMode	kony.filter.SUBTRACT_BLEND_MODE	This filter is available in iOS 8 and above.
CITemperatureAndTint	kony.filter.TEMPERATURE_AND_TINT	Adapts the reference white point for an image. This filter is available in iOS 5 and above.

CIToneCurve	kony.filter.TONE_CURVE	Adjusts tone response of the R, G, and B channels of an image. This filter is available in iOS 5 and above.
CIVibrance	kony.filter.VIBRANCE	Adjusts the saturation of an image while keeping pleasing skin tones. This filter is available in iOS 5 and above.
CIVignette	kony.filter.VIGNETTE	Reduces the brightness of an image at the periphery. This filter is available in iOS 5 and above.
CIVignetteEffect	kony.filter.VIGNETTE_EFFECT	Modifies the brightness of an image around the periphery of a specified region.
CIZoomBlur	kony.filter.ZOOM_BLUR	This filter is available in iOS 9 and above.

Windows Filter Constants

The following filters are available for Windows.

Filter Type	Parameters	Description

kony.filter.ALPHATOGRAYSCALEFILTER	"filterName": kony.filter.ALPHATOGRAYSCALEFILTER, "inputImage": Image Object	Copies the alpha channel in the image to the color channels, resulting in a grayscale representation of the alpha channel.
kony.filter.AFFINE_TRANSFORM	"filterName" :kony.filter.ANTIQUEFILTER, "inputImage" : Image Object	Produces the look of an old color photo.
kony.filter.AUTOENHANCEFILTER	"filterName": kony.filter.AUTOENHANCEFILTER, "inputImage": Image Object, "isContrastAndBrightnessEnhancementEnabled": true, "isLocalBoostEnhancementEnabled": false	Automatically enhances white balance, brightness and contrast, and/or applies local boost to the image. isContrastAndBrightnessEnhancementEnabled [Boolean] - Enables or disables contrast and brightness enhancement. isLocalBoostEnhancementEnabled [Boolean] - Enables or disables local boost enhancement.
kony.filter.AUTOLEVELSFILTER	"filterName": kony.filter.AUTOLEVELSFILTER, "inputImage": Image Object	Balances the intensity level of the image, for example making dark images lighter and vice versa.

kony.filter.BLENDFILTER	"filterName": kony.filter.BLENDFILTER, "inputImage": Image Object, "blendImage": blendimg, "blendFunction": 5	Blends an image onto the current image, with optional masking, positioning, scaling, and rotation. blendImage [Image] - The image to blend with source image. blendFunction [int] - The blend function to use. Possible values are: • 0 - Normal "alpha" blending. • 1 - Multiplicative blending. • 2 Additive blending (alias of Add). • 3 - Color replacement. • 4 - "Colorburn" blending. • 5 - "Colordodge" blending. • 8 - "Screen" blending. • 6 - "Overlay" blending. • 7 - "Softlight" blending. • 9 - "Hardlight" blending. • 10 - Darken by taking the minimum. • 11 - Lightness by taking the maximum. • 12 - Hue replacement. • 13 - "Exclusion" blending.

kony.filter.BLURFILTER	"filterName": kony.filter.BLURFILTER, "inputImage": Image Object, "kernelSize": 25	Blurs the image with a specific kernel size. kernelSize [int] - The kernel size in pixels. Range [1, 256]. Larger kernel size results in more blur.
kony.filter.BRIGHTNESSFILTER	"filterName": kony.filter.BRIGHTNESSFILTER, "inputImage": Image Object, "level": 0.4	Adjusts the brightness in the image with a brightness level. level [double] - The brightness level. Range [-1.0, 1.0], where 0.0 results in no adjustment.
kony.filter.CARTOONFILTER	“filterName” : kony.filter.CARTOONFILTER, “inputImage” : Image Object, “distinctEdges”: false (Default value is true)	Transforms the image into a cartoon graphic style.

kony.filter.CHROMAKEYFILTER	"filterName": kony.filter.CHROMAKEYFILTER, "inputImage": Image Object, "chromacolor": "FF123400", "colorDistance": 0.7, "noiseSuppression": 0.4, "invertAlpha": false	Adds transparency to the pixels of a specific color. color [String] - The chroma key color that will be made transparent. The alpha component will be ignored. colorDistance [double] - Specifies how much variation around the chroma key color will be considered for transparency. If the color distance is small, only a small variation around the chroma key color will be considered for transparency. Range [0.0, 1.0]. noiseSuppression [double] - Specifies the noise suppression level. If the noise suppression level is low, only a small amount of noise will be suppressed. Range [0.0, 1.0]. invertAlpha [Boolean] - Controls how the alpha channel is generated. If false, full transparency in the resulting alpha channel is the value 0. If true, full transparency in the resulting alpha channel is the value 255.

kony.filter.COLORADJUSTFILTER	"filterName": kony.filter.COLORADJUSTFILTER, "inputImage": Image Object, "red": 0.4, "green": 0.7, "blue": -0.4	Adjusts the RGB color composition of the image with specified channel adjustment values. Red [double] - Red channel adjustment. Range [-1.0, 1.0]. Green [double] - Green channel adjustment. Range [-1.0, 1.0]. Blue [double] - Blue channel adjustment. Range [-1.0, 1.0].
kony.filter.COLORBOOSTFILTER	"filterName": kony.filter.COLORBOOSTFILTER, "inputImage": Image Object, "gain": 12.0	Amplifies the colors of the image with the specified gain level. Gain [double] - The color boost gain level. Range [-1.0, 20.0]. Negative values reduce the colors. Recommended values are in the range [-1.0, 1.0].

kony.filter.COLORSWAPFILTER	"filterName": kony.filter.COLORSWAPFILTER, "inputImage": Image Object, "sourceColor": "00FF0000", " swapColor": "FF00FF00", "colorDistance": 0.35, "isMonoColor": false, "swapLuminance": true	Creates ColorSwapEffect with specified settings and applies on the provided image. sourceColor [String] - Specifies the color that should be converted. The alpha-component is ignored. swapColor [String] - Specifies the color to which the source Color should be converted. The alpha-component is ignored. colorDistance [double] - Specifies the tolerance level of the effect. Range [0.0, 1.0]. isMonoColor [Boolean] - Specifies if the other colors should be converted to grayscale or be preserved. swapLuminance [Boolean] - Controls if the luminance should be included in the color swap.

kony.filter.COLORIZATIONFILTER	"filterName": kony.filter.COLORIZATIONFILTER, "inputImage": Image Object, "color": "FF123400", "luminance": 1.5, "chrominance": -0.2	Adjusts color tone and luminance/chrominance levels, adds a single color to an image. Color [string] - The reference color. The alpha-component is ignored. luminance [double] - The luminance adjustment. Range: [-1.0, 2.0]. chrominance [double] - The chrominance adjustment. Range: [-1.0, 2.0].
kony.filter.CONTRASTFILTER	"filterName": kony.filter.CONTRASTFILTER, "inputImage": Image Object, "level": 0.4	Adjusts the contrast in the image with a contrast level. level [double] - The contrast level. Range [-1.0, 1.0], where values below zero decrease contrast and above zero increase it.
kony.filter.CURVESFILTER	"filterName": kony.filter.CURVESFILTER, "inputImage": Image Object, "RedCurveIndex": 3, "GreenCurveIndex": 2, "BlueCurveIndex": 5	Transforms colors in the image using a set of color curve index values. RedCurveIndex [int] - Color curve for the red channel. Range [0, 7] GreenCurveIndex [int] - Color curve for the green channel. Range [0, 7] BlueCurveIndex [int] - Color curve for the blue channel. Range [0, 7]

kony.filter.DESPECKLEFILTER	"filterName": kony.filter.DESPECKLEFILTER, "inputImage": Image Object, "despeckleLevel": 3	Applies a despeckle effect, removing noise from the image with a specified level. despeckleLevel [int] - The despeckle level. (0-Minimum, 1-Low, 2-High, 3-Maximum)
kony.filter.EMBOSSFILTER	"filterName": kony.filter.EMBOSSFILTER, "inputImage": Image Object, "level": 0.35	Creates EmbossEffect with specified level and applies on the provided image. Level [double] - The level of the emboss effect. Range: [0.0, 1.0].
kony.filter.EXPOSUREFILTER	"filterName": kony.filter.EXPOSUREFILTER, "inputImage": Image Object, "exposureMode": 1, "gain": -0.7	Adjust the brightness of an image with specified exposure mode and gain. exposureMode [int] - The exposure mode. (0 -Natural, 1-Gamma) gain [double] - The desired exposure level. Positive values will make the image brighter, negative values will make it darker. Range is [-1.0, 1.5], but for gamma mode the effect saturates at 1.0, which means that everything between [1.0, 1.5] will be interpreted as 1.0. Recommended range for both gamma and natural mode for normal usage is [-1.0, 1.0]. 0.0 equals no change of exposure.

kony.filter.FLIPFILTER	"filterName": kony.filter.FLIPFILTER, "inputImage": Image Object, "flipMode": 2	Flips the image horizontally and/or vertically with a specified flip mode. flipMode [int] - The flip mode, around horizontal, vertical or both axes. (0 -None, 1-Vertical, 2-Horizontal, 3-Both)
kony.filter.FOGFILTER	"filterName": kony.filter.FOGFILTER, "inputImage": Image Object	Applies fog effect on the image.
kony.filter.FOUNDATIONFILTER	"filterName": kony.filter.FOUNDATIONFILTER, "inputImage": Image Object	Applies foundation effect on the image.
kony.filter.GAUSSIANNOISEFILTER	"filterName": kony.filter.GAUSSIANNOISEFILTER, "inputImage": Image Object, "level": 1.4	Applies Gaussian noise to the image. level [double] - The amount of noise, given as the standard deviation of the Gaussian distribution. Must be >= 1.0.

kony.filter.GRAYSCALEFILTER	"filterName": kony.filter.GRAYSCALEFILTER, "inputImage": Image Object, "redWeight": 127.9, "greenWeight": -10.9, "blueWeight": 27.9, "constant": 86	Converts the image to grayscale using a mixture of the color components. The gray shade is calculated by multiplying each color component of every pixel with the matching Weight and adding a constant: resultShade = redValue * redWeight + greenValue * greenWeight + blueValue * blueWeight + constant. The resulting shade is clamped to the range [0,1]. redWeight [double] - The red component of the equation for mixing weights. Range [-255.0, 255.0]. greenWeight [double] - The green component of the equation for mixing weights. Range [-255.0, 255.0]. blueWeight [double] - The blue component of the equation for mixing weights. Range [-255.0, 255.0]. Constant [double] The constant component of the equation for mixing weights. Range [-255.0, 255.0].

kony.filter.GRAYSCALENEGATIVEFILTER	"filterName": kony.filter.GRAYSCALENEGATIVEFILTER, "inputImage": Image Object	Converts the image to a grayscale negative.

kony.filter.HDRFILTER	"filterName": kony.filter.HDRFILTER, "inputImage": Image Object, "gamma": 1.9, "noiseSuppression": 0.9, "saturation": 2.9, "strength": 0.2	Applies local tone mapping to a single image to achieve an HDR-like effect. gamma [double] - Controls global contrast enhancement using a power law. Values lower than 1.0 makes the image more white and values above 1.0 makes it more black. Recommended range is between 0.6 and 1.0. gamma must be > 0. noiseSuppression [double] - Controls supression of noise amplification, value should rise with the noisiness of the input image combined with the strength applied. Recommended value is between 0.1 and 0.3. Range [0.0, 1.0]. saturation [double] - Controls color saturation using a power law. Values lower than 1.0 will decrease the saturation, and values above 1.0 will increase the saturation of the image. Recommended range is between 0.4 and 0.8. saturation must be > 0. Strength [double] - Controls strength of local contrast enhancement. The higher the value, the stronger effect. Recommended range is between 0.1 and 0.30. Having a high value may result in dark noisy images. Range [0.0, 1.0].

kony.filter.HUESATURATIONFILTER	"filterName": kony.filter.HUESATURATIONFILTER, "inputImage": Image Object, "hue": -0.5, "saturation": 0.2	Adjusts the hue and saturation of the image with specified hue and saturation. hue [double] - Hue adjustment level. Range [-1.0, 1.0] where 0 implies no adjustment. saturation [double] - Saturation adjustment level. Range [-1.0, 1.0] where 0 implies no adjustment.
kony.filter.LEVELSFILTER	"filterName": kony.filter.LEVELSFILTER, "inputImage": Image Object, "white": 0.7, "gray": 0.2, "black": 0.5	Adjusts levels in the current image. white The position of the white saturating point. Range [0.0, 1.0]. gray - The relative position of the middle gray point. Range [0.0, 1.0], where 0.0 is at the bright saturating point and 1.0 is at the dark saturating point. black - The position of the black saturating point. Range [0.0, 1.0]. The value of black should be smaller than, or equal to the value of white.

kony.filter.LOCALBOOSTFILTER	"filterName": kony.filter.LOCALBOOSTFILTER, "inputImage": Image Object, "gamma": 3, "darkContrast": 0.9, "lightContrast": 0.7, "sensitivityLevel": 0.2	Manually boost and enhance images with unequal illumination. gamma [double] - Gamma correction for the dark part of the image. Range [0.1, 5.0]. darkContrast [double] - Contrast amplification of the dark part. Range [0.5, 1.0]. lightContrast [double] - Contrast amplification of the light part. Range [0.5, 1.0]. sensitivityLevel [double] - Specifies how much of the processed image will be blended into the original image. Range: [0.0, 1.0].
kony.filter.LOCALBOOSTAUTOMATICFILTER	"filterName": kony.filter.LOCALBOOSTAUTOMATICFILTER, "inputImage": Image Object, "level": 10	Manually boost and enhance images with unequal illumination with the specified level. level [int] - Intensity of the effect. Range [0, 14].

kony.filter.LOMOFILTER	"filterName": kony.filter.LOMOFILTER, "inputImage":- Image Object, "brightness": 0.7, "saturation": 0.4, "vignetting": 1, "style": 4	Creates LomoEffect with specified settings and applies on the provided image. Brightness [double] - The brightness adjustment. Range [0.0, 1.0], where 0.0 gives a bright image and 1.0 gives a dark image. Saturation [double] - The color saturation adjustment. Range [0.0, 1.0], where 0.0 implies no saturation. Vignetting [int] - The vignetting level. Possible values are: • 0 - Low • 1 - High • 2 - Medium Style [int] - The color style. Possible values are: • 0 - Neutral • 1 - Red • 2 - Green • 3 - Blue • 4 - Yellow
kony.filter.MAGICPENFILTER	"filterName": kony.filter.MAGICPENFILTER, "inputImage": Image Object	Applies a mix of edge distinction and color manipulation to the image.

kony.filter.MILKYFILTER	"filterName": kony.filter.MILKYFILTER, "inputImage": Image Object	Applies a milky surface to the image.
kony.filter.MIRRORFILTER	"filterName": kony.filter.MIRRORFILTER, "inputImage": Image Object	Mirrors the left half of the image onto the right half.
kony.filter.MONOCOLORFILTER	"filterName": kony.filter.MONOCOLORFILTER, "inputImage": Image Object, "preserveColor": "FF00FF00", "colorDistance":0.7	Creates and initializes a new MonoColorEffect with specified settings for the color to be preserved. preserveColor [String] - The color that will be preserved. colorDistance [double] - Specifies the tolerance level of the effect. Range [0.0, 1.0].
kony.filter.MOONLIGHTFILTER	"filterName": kony.filter.MOONLIGHTFILTER, "inputImage": Image Object, "clock": 22	Creates a MoonlightEffect specifying the time of day and applies to the image. Clock [int] - Time of night in clock time [0, 23]. The filter has effect only at night time, between 17 and 7. The default value is 0.
kony.filter.NEGATIVEFILTER	"filterName": kony.filter.NEGATIVEFILTER, "inputImage": Image Object	Converts the image to a negative.

kony.filter.NOISEFILTER	"filterName": kony.filter.NOISEFILTER, "inputImage": Image Object, "level": 2	Applies noise to the image with specified noise level. level [int] - The amount of noise. (0 -Minimum, 1-Medium, 2-Maximum)
kony.filter.OILYFILTER	"filterName": kony.filter.OILYFILTER, "inputImage": Image Object, "oilBrushSize ": 2	Creates OilyEffect with the specified OilinessLevel and applies to the image. oilBrushSize [int] - Specifies the strength of the effect, where higher oiliness results in higher distortion of the source image.
kony.filter.PAINTFILTER	"filterName": kony.filter.PAINTFILTER, "inputImage": Image Object, "level ": 4	Creates PaintEffect with the specified level and applies to the image. Level [int] - Paint effect level. Range [1, 4].
kony.filter.POSTERIZEFILTER	"filterName": kony.filter.POSTERIZEFILTER, "inputImage": Image Object, "colorComponentValueCount": 12	Applies a painting-like effect to the image. colorComponentValueCount [int] - The number of allowed values for each color component. Range [2, 16].
kony.filter.ROTATIONFILTER	"filterName": kony.filter.ROTATIONFILTER, "inputImage": Image Object, "rotationAngle": 135.0	Rotates the image around its center in a clock-wise direction with a specified angle. rotationAngle [double] - The rotation angle in degrees.

kony.filter.SEPIAFILTER	"filterName": kony.filter.SEPIAFILTER, "inputImage": Image Object	Applies a sepia tone to the image.
kony.filter.SHARPNESSFILTER	"filterName": kony.filter.SHARPNESSFILTER, "inputImage": Image Object, "level": 0.7	Enhances the sharpness of the image, allowing for precise settings. level [double] - Image sharpness level. Level must be greater than 0. Recommended range is [0, 1] however values greater than 1.0 are supported.
kony.filter.SKETCHFILTER	"filterName": kony.filter.SKETCHFILTER, "inputImage": Image Object, "sketchMode": 0	Applies a sketch effect to the image with the specified SketchMode. sketchMode [int] - The sketch mode. Possible values are: • 0 - Gray-Sketch in grayscale • 1 - Color-Sketch in color
kony.filter.SOLARIZEFILTER	"filterName": kony.filter.SOLARIZEFILTER, "inputImage": Image Object, " threshold": 0.7	Applies a solarize effect to the image with the specified threshold. Threshold [double] - The threshold level of the solarize effect. Range [0.0, 1.0].

kony.filter.STAMPFILTER	"filterName": kony.filter.STAMPFILTER, "inputImage": Image Object, "smoothness": 2, "threshold": 0.6	Applies a stamp-like effect, resulting in a black and white image with specified smoothness and threshold values. Smoothness [int] - The smoothness level. Range [0, 6]. Threshold [double] - The threshold level. Range [0.0, 1.0].
kony.filter.TEMPERATUREANDTINTFILTER	"filterName": kony.filter.TEMPERATUREANDTINTFILTER, "inputImage": Image Object, "temperature": -0.4, "tint": 0.7	Adjusts the color temperature and tint of the image using provided temperature and tint. temperature [double] - Color temperature adjustment. Range [-1.0, 1.0] where 0 implies an unmodified color temperature. tint [double] - Color tint adjustment. Range [-1.0, 1.0] where 0 implies an unmodified color tint.
kony.filter.VIGNETTINGFILTER	"filterName": kony.filter.VIGNETTINGFILTER, "inputImage": Image Object, "vignettingColor": "00FFFF00", "transitionSize": 12	Applies vignetting effect to the image with specified transition size and vignetting color. transitionSize [double] - The size of the transition region as a fraction of the radius. Range [0.0, 15.0]. Color [String] - The color to use for the vignetting effect.

kony.filter.WATERCOLORFILTER	"filterName": kony.filter.WATERCOLORFILTER, "inputImage": Image Object, "lightIntensity": 0.2, "colorIntensity": 0.7	Applies a watercolor effect to the image with specified intensities. lightIntensity [double] - The light intensity. Range [0.0, 1.0]. colorIntensity [double] - The color intensity. Range [0.0, 1.0].

kony.filter.WARPFILTER	"filterName": kony.filter.WARPFILTER, "inputImage": Image Object, "wrapmode": 12, "wraplevel": 0.7	Applies a warp effect to an image with specified warp effect and level. wrapmode [int] - The warp mode to apply. Possible values are: • 0 - Upnose • 1-Twister • 2 - SmallNose • 3 - WideSmile • 4 - Grit • 5 - BigFace • 6 - Professor • 7 - Alien • 8 - BigNose • 9 - AlienHybrid • 10 - Gobbler • 11 - Square • 12 - Sharpchin • 13 - LongFaced • 14 - HappyFool • 15 - Insect. wraplevel [double] - Amount of effect applied. Range [0.0, 1.0], where 0.0 means no effect and 1.0 means full effect. Default is 0.5.

kony.filter.WHITEBALANCEFILTER	"filterName": kony.filter.WHITEBALANCEFILTER, "inputImage": Image Object, "whitePointCalculationMode": 2	Adjusts the white balance in the image with specified white point calculation mode. whitePointCalculationMode [int] - Possible values are: • 0 - Uses the mean value of the 256-bin distribution. • 1 - Uses the estimated mean gray for the color correction. • 2 - Uses the estimated maximum intensity color. • 3 - Uses a specified white reference color.
kony.filter.WHITEBOARDENHANCEMENTFILTER	"filterName": kony.filter.WHITEBOARDENHANCEMENTFILTER, "inputImage": Image Object, "whiteboardEnhancementMode": 0	Enhances text and drawings in an image of a whiteboard with a specified mode. whiteboardEnhancementMode [int] - Possible values are: • 0 - Enhances the contrast. • 1 - Preserve colors better.

27.5 kony.image Namespace

The kony.image namespace provides the following API elements:

- Constants
- Functions

27.5.1 Constants

The kony.image namespace provides the following constant:

Image Format Constants

The following constants are used to specify the format of an Image widget's bitmap.

Constant	Description
kony.image.ENCODE_JPEG	The bitmap is in JPEG format.
kony.image.ENCODE_PNG	The bitmap is in PNG format.

27.5.2 Functions

The kony.image namespace provides the following functions:

kony.image.createImage Function

Creates an Image. This function has three overloads.

Syntax

```
kony.image.createImage(rawBytes);
```

```
kony.image.createImage(
    bundledImageFileName);
```

```
kony.image.createImage(
    Image);
```

Input Parameters

Any one of the following parameters can be passed in the function.

Parameter	Description
rawBytes	A RawBytes object containing the image's bitmap.
bundledImageFileName	A string containing the file name of the bitmap to use for the image.
Image	An Image widget containing the bitmap to use for the Image being created.

Example

```
var imgObj = kony.image.createImage(rawB);
```

```
var imgBright = kony.image.createImage(this.imageBytes);
```

Return Values

This function returns an image object with its associated bitmap.

Remarks

If your app creates an Image from a RawBytes object, it should not make copies of the Image by creating it again from the initial RawBytes object. Creating multiple Image objects from the same RawBytes object results in undefined behavior. Rather than copying the RawBytes object into multiple Image objects, your app can make further copies by calling this function and passing it the Image object created in the first call to this function.

If your app creates an image from a bundled file, the file specified by the *bundledImageFileName* parameter can be in the GIF, an animated GIF, a JPEG, or PNG file format.

Platform Availability

Available on iOS and Android.

kony.image.createImageFromSnapShot Function

Creates an Image by taking a snapshot of a widget.

Syntax

```
kony.image.createImageFromSnapShot(
    widget);
```

Input Parameters

Parameter	Description
widget	The widget that this function takes a snapshot of. The snapshot is used for the bitmap of the Image created by the function.

Example

```
var imgBlurBg = kony.image.createImageFromSnapShot(frmHome.widget1);
```

Return Values

This function returns an Image that contains a snapshot of the widget passed in through the *widget* parameter.

Remarks

When the image source is snapshot, the source is device screen, which is having twice the density of actual image, so the scale factor of image will be twice the image size because of the retina display of device.

Platform Availability

Available on iOS and Android.

cropImageInTiles Function

Crops the bitmap in an Image object and returns it as an array of tiles.

Syntax

```
kony.image.cropImageInTiles(
    image,
    xTiles,
    yTiles)
```

Input Parameters

Parameter	Description
image	The Image object containing the bitmap to be cropped.
xTiles	The number of equally-sized tiles that can be created in the x direction.
yTiles	The number of equally-sized tiles that can be created in the x direction.

Example

```
var img = kony.image.createImage(rawB); // Here rawB is the rawBytes of the
image
var imageArray = kony.image.cropImageInTiles(img, 10, 20);
```

Return Values

This function returns an array of Image widgets that were created from tiles of the bitmap in the Image object in the *image* parameter.

Platform Availability

Available on iOS and Android.

cropImageInTilesForRects Function

Crops portions of an Image widget's bitmap to a set of rectangles and returns an array of Image widgets containg the cropped bitmaps.

Syntax

```
kony.image.cropImageInTilesForRects(
    image,
    [ [x,y,w,h],[x1,y1,w1,h1],... ])
```

Parameters

Parameter	Description
image	An Image widget containing the bitmap to be cropped.

Parameter	Description
[[x,y,w,h],[x1,y1,w1,h1],...]	An array of rectangles specified as the (x,y) coordinates of the upper left corner and the height and width of the rectangle. Each rectangle in this array must be an array of four integers.

Example

```
function getImageFromLocalStorage(imageName) {
    var img = kony.image.createImage(imageName);
    return img;
}

/*var clippingRects = [
        [10, 12, 50, 100],
        [30, 45, 10, 200],
        [300, 100, 200, 10]];*/
function cropImageToTilesFromRects(clippingRects, localImage, FormToaddImage)
{
    try {
        var img = getImageFromLocalStorage(localImage);
        var imageArray = kony.image.cropImageInTilesForRects(img,
clippingRects);
        for (var j = 0; j < imageArray.length; j++) {
            var imgW = createImageWidget(imageArray[j]);
            FormToaddImage.add(imgW);
        }
        FormToaddImage.forceLayout();
    } catch (err) {
        alert(err);
    }
}
```

Return Values

This function returns an array of Image widgets that contain the bitmap information from the bitmap in the Image widget passed through the *image* parameter.

Remarks

This method iterates through an array of rectangles and uses each rectangle to obtain a cropped version of the bitmap associated with the Image widget in the *image* parameter. The original bitmap is not changed. It then creates an Image widget from each cropped bitmap and collects then into an array of Image widgets. It then returns the array of Image widgets.

Platform Availability

Available on iOS and Android.

28. Input and Output API

The Input and Output API enables your app to perform common tasks such as copy, rename, and delete files. This API enables you to browse through file systems and then select multiple files to upload on the server. In addition, the I/O API enables you to manage directories, determine if external storage is available, and so forth.

The Input and Output API, often abbreviated as the I/O API, is composed of the following Namespaces.

kony.io.File Namespace

Function	Description
`kony.io.File.copyTo`	Copies a file to the given destination path.
`kony.io.File.createDirectory`	Creates a directory on the file system represented by this file object.
`kony.io.File.createFile`	Creates a file on the file system represented by this file object.
`kony.io.File.exists`	Checks if the file or directory exists on the file system represented by this file object.
`kony.io.File.getFilesList`	Returns kony.io.FileList object representing the files and directories available under this file object directory.
`kony.io.File.isDirectory`	Checks if this object represents a directory file on the file system.
`kony.io.File.isFile`	Checks if this object represents a typical file on the file system but not a directory.

Function	Description
`kony.io.File.moveTo`	Moves a file to the given destination path.
`kony.io.File.read`	Returns the kony.types.RawBytes of this file.
`kony.io.File.readAsText`	Returns the data as text represented by this rawBytes .
`kony.io.File.remove`	Deletes a file or a directory.
`kony.io.File.rename`	Renames a file or a directory.
`kony.io.File.write`	Writes the given content into the file.

kony.io.FileList Namespace

Function	Description
`kony.io.File.item`	Returns the File object at the specified index in the FileList.

kony.io.FileSystem Namespace

Function	Description
kony.io.FileSystem.browse	Provides the ability to browse and select files from your local system.
kony.io.FileSystem.copyBundledRawFileTo	Copies a bundled file from the application binary to the specified destination path on the device.
kony.io.FileSystem.getAppGroupDirectoryPath	Retrieves the root path of an app group.
kony.io.FileSystem.getApplicationDirectoryPath	Returns the application directory path.

Function	Description
`kony.io.FileSystem.getCacheDirectoryPath`	Returns the application's cache directory path.
`kony.io.FileSystem.getDatabaseDirectoryPath`	Returns the application's database directory path (from application's private file system) where kony.db APIs access the database files.
`kony.io.FileSystem.getDataDirectoryPath`	Returns the application's data directory path.
`kony.io.FileSystem.getExternalStorageDirectoryPath`	Returns the path to the external storage, typically sdcard.

Function	Description
kony.io.FileSystem.getFile	Returns a kony.io.File object representing the file for the given path.
kony.io.FileSystem.getRawDirectoryPath	Returns the application’s “raw” directory path.
kony.io.FileSystem.getSupportDirectoryPath	Returns the application’s support directory path.
kony.io.FileSystem.isExternalStorageAvailable	Checks if the external storage is available and accessible on the device.

Function	Description
`kony.io.FileSystem.uploadFiles`	Internally iterates through the fileList and makes the network call to the URL mentioned and uploads the files.

kony.types Namespace

Function	Description
`kony.types.RawBytes.getTempPath`	Returns the path where the source files of all RawBytes are stored.

File Handling

The `kony.io.File` Namespace consists of functions that enable you to perform common file handling tasks. Your app can create a File object to represent files in the device's file system and perform common operations on them. Now create a file using the `kony.io.File.createFile` function. To verify if a file exists in the file system using the `kony.io.File.exists` function. You can also check if the object represents a file or a directory using the `kony.io.File.isFile` and the `kony.io.File.isDirectory` functions; and return all the files in the filesystem using the `kony.io.File.getFilesList` function.

To move a file, use the `kony.io.File.moveTo` function. To copy a selected file use the `kony.io.File.copyTo` function. You can read the data in the file using the `kony.io.File.read` and the `kony.io.File.readAsText` functions. You can also write in a file using `kony.io.File.write` function. If you want to rename a file, use the `kony.io.File.rename` function; and to delete a file, use the `kony.io.File.remove` function. Using the `kony.io.File.item`, you can return a file object at a specified index in the File list.

File System Handling

The `kony.io.FileSystem` Namespace consists of the functions that handles a file system. Use the `kony.io.FileSystem.getFile` function to get an instance of a File object. You can browse and select files from the file system using the `kony.io.FileSystem.browse` function. Then get the root path of an application using the `kony.io.FileSystem.getApplicationDirectoryPath` and the app group directory path using the `kony.io.FileSystem.getAppGroupDirectoryPath` function. Use the `kony.io.FileSystem.getCacheDirectoryPath` function to get the application cache’s directory path, the `kony.io.FileSystem.getDataDirectoryPath` to get the application’s directory path, the `kony.io.FileSystem.getDatabaseDirectoryPath` to get the application’s database directory path and external storage directory path using the `kony.io.FileSystem.getExternalStorageDirectoryPath` function. For the application’s raw directory path, use the `kony.io.FileSystem.getRawDirectoryPath` function and for the application's support path, use the `kony.io.FileSystem.getSupportDirectoryPath` function. You can also check if the external storage is available and accessible on the device using the `kony.io.FileSystem.isExternalStorageAvailable` function, and upload files using the `kony.io.FileSystem.uploadFiles` function.

28.1 Overview

With the I/O API, your app can perform common tasks on files such as copying, renaming, and deleting them. In addition, the I/O API enables you to manage directories, determine if external storage is available, and so forth.

28.1.1 Pre bundling Files

Kony gives an ability to bundle files such as SQLite database (pre-configured), video, image, and so on as part of an application binary. After bundling, the files can be accessed from the app at the run time on the device. You need to follow certain guidelines to bundle the files.

Here are the guidelines:

1. Place the files to be bundled in the application workspace at the following locations. When you place the files in these locations, the files will be bundled with the application built for all the platforms.

   Mobile:

   <WorkSpace>\<App>\resources\mobile\common\raw

   Tablet:

   <WorkSpace>\<App>\resources\tablet\common\raw

2. You can also place the files in the platform-specific folders. When you place the files in platform-specific folders, the files will be bundled only with the application built for that particular platform. For example, if you place the files in the android\raw folder, the files will be bundled only for the application built for the Android platform.

Android Mobile:

<WorkSpace>\<App>\resources\mobile\native\android\raw

Android Tablet:

<WorkSpace>\<App>\resources\tablet\native\androidtab\raw

IOS Mobile:

<WorkSpace>\<App>\resources\mobile\native\iphone\raw

IOS Tablet:

<WorkSpace>\<App>\resources\tablet\native\ipad\raw

Windows Mobile:

<WorkSpace>\<App>\resources\mobile\native\winphone8\raw

Windows Tablet:

<WorkSpace>\<App>\resources\tablet\native\windows8\raw

After placing the files either in common or platform-specific folder and build the respective application, the files are packaged in the application binary file. Example, .apk, .ipa, and so on.

In case same file is located at both common and platform-specific folders, the file in the platform-specific folder will override the file in the common folder.

3. In case of Android, the file name should contain only the following characters:

    - a - z (lower case)
    - 0 - 9
    - _ (underscore)
    - . (period)

    This guideline is applicable only for the Android platform.

4. Application binary size limitation: After bundling the files, the size of the application binary should not exceed the size defined for each platform in the table below. These size limits are specified by the platform-specific stores where the apps will be uploaded.

> ***Note:*** In future these values may vary based on the application store decisions for respective platforms.

OS	Limit	Binary Type
Android	100 MB	.apk
IOS	60 MB	.ipa
Windows 10	25 GB	.appxbundle
Windows 8.1	8 GB	.appx packages
Windows 8	2 GB	.appx packages
Windows Phone 8.1	4 GB	.appx packages
Silver light applications	1 GB	.xap packages

Kony provides the following APIs to support accessing the pre-bundled files programmatically.

- kony.io.FileSystem.copyBundledRawFileTo
- kony.io.FileSystem.getDatabaseDirectoryPath

28.2 kony.io.File Namespace

The kony,io.File namespace provides functions and properties for doing various operations related on files, such as copying them, renaming them , deleting them, and so on.

28.2.1 Overview

Your app can create a `File` object to represent files in the device's file system and perform common operations on them. To create a new `File` object , your app calls the `kony.io.File` function. Alternatively, it can use the kony.io.FileSystem.getFile function to get an instance of a `File` object.

```
var myfile = new kony.io.File("myfile.txt");
or
var myfile = kony.io.FileSystem.getFile("myfile.txt")
```

File Properties:Name	Description
name: [type: String, ReadOnly]	Name of the file or directory
fullPath: [type: String, ReadOnly]	Full absolute path of the file or directory
parent: [type:kony.io.File, ReadOnly]	Returns the parent directory of this file. This property may return nil or a File object which can neither Readable or Writable, especially in case of Folders which are not accessible by the given Application context
readable: [type: Boolean, ReadOnly]	Returns true if this file is readable
writable: [type: Boolean, ReadOnly]	Returns true if this file is writable
modificationTime: [type:Number, ReadOnly]	Returns the last modification time of the file in UTC

File Properties:Name	Description
size: [type:Number, ReadOnly	Returns the size of a file in number of bytes.

The following function is often used in conjunction with the RawBytes object to read data of type RawBytes.

- readAsText

28.2.2 Functions

The kony.io.File namespace contains these functions: To use the copyTo, moveTo, remove, rename, createFile, createDirectory, read, and write File APIs, your app needs runtime permission from the end-user (to perform any of the action correspondent to a file). If you call any API without obtaining the permission, platforms automatically pops up a system permission dialog box with "Allow" and "Deny" options, asking the end-user to grant permission. This is applicable only for the Android platform.

If the end-user taps the "Allow" option, platform proceeds to access the underlying OS API. If the end-user taps the "Deny" option, the PermissionError exception is thrown with the 2300 error code, that means permission is denied.

28.2.3 kony.io.File.copyTo

copyTo API copies a file to the given destination path.

Syntax

```
kony.io.File.copyTo(String path, String newName)
```

Input Parameters

Parameter	Description
Path	path to the destination directory.
newName (optional)	New name of the file/directory. Defaults to current name if unspecified.

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var copyToLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDir1";
var newFile = new kony.io.File(origFileLoc).copyTo(copyToLoc,
"NewNameForCopy.txt");
```

Return Values

Kony.io.File returns a handle to the File object pointing to the destination file, if successful. If failure, then returns null.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.4 kony.io.File.createDirectory

The createDirectory API creates a directory on the file system represented by this file object.

Syntax

```
kony.io.File.createDirectory()
```

Input Parameters

None

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var dirLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDir1";
var myDir = new kony.io.File(dirLoc).createDirectory();
```

Return Values

Boolean - true if the creation of directory is successful. False if directory already exists or could not be created.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.5 kony.io.File.createFile

The createFile API creates a file on the file system represented by this file object.

Syntax

```
kony.io.File.createFile()
```

Input Parameters

None

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var fileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFileToCopy.txt";
var myFile = new kony.io.File(fileLoc).createFile();
```

Return Values

Boolean - true if the creation of file is successful. False if file already exists or could not be created.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.6 kony.io.File.exists

The exist API checks, if the file or directory exists on the file system represented by this file object.

Syntax

```
kony.io.File.exists()
```

Input Parameters

None

Example

```
var copiedFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
    "myDir1" + constants.FILE_PATH_SEPARATOR +
    "NewNameForCopy.txt";
if (new kony.io.File(copiedFileLoc).exists()) {
    kony.print("copy of file was successful");
} else {
    kony.print("copy of file failed");
}
```

Return Values

Boolean - true if the file or directory exists on file system.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.7 kony.io.File.getFilesList

The getFilesList API returns kony.io.FileList object representing the files and directories available under this file object directory.

Syntax

```
kony.io.File.getFilesList()
```

Input Parameters

None

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myDirLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDir416";
var myDirName = new kony.io.File(myDirLoc);
var createDir = myDirName.createDirectory();
var fileListLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myDir416";
var filesList = new kony.io.File(fileListLoc).getFilesList();
if (filesList.length === 0) {
    kony.print("getFilesList successful for zero files");
} else {
    kony.print("getFilesList failed for zero files");
}
```

Return Values

kony.io.FileList - FileList object or null if this File is not identified as a directory.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.8 kony.io.File.isDirectory

The isDirectory API checks, if this object represents a directory file on the file system.

Syntax

```
kony.io.File.isDirectory()
```

Input Parameters

None

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var dirLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDir765";
var myDir = new kony.io.File(dirLoc);
try {
    var isDirec = new kony.io.File(dirLoc).isDirectory();
    if (isDirec) {
        kony.print("isDirectory True for nonExistent Directory");
    }
} catch (err) {
    kony.print("isDirec doesn't work over nonExistent directory");
}
```

Return Values

Boolean - true, if this file object represents a directory, false otherwise.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.9 kony.io.File.isFile

The isFile API checks, if this object represents a typical file on the file system but not a directory.

Syntax

```
kony.io.File.isFile()
```

Input Parameters

None

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFile244.txt";
var myFileName = new kony.io.File(myFileLoc);
try {
    var isFileThere = new kony.io.File(myFileLoc).isFile();
    if (isFileThere) {
        kony.print("isFile true for nonExistent File");
    } else {
        kony.print("isFile false for nonExistent File");
    }
} catch (err) {
    kony.print("isFile doesn't work on non-existent files");
}
```

Return Values

Boolean - true if this file object represents a file, false otherwise.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.10 kony.io.File.moveTo

The moveTo API moves a file to the given destination path.

Syntax

```
kony.io.File.moveTo(String path, String newname)
```

Input Parameters

Parameter	Description
Path	path to the destination directory.
newName (optional)	New name of the file/directory. Defaults to current name, if unspecified.

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var dirLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDir25";
var myDir = new kony.io.File(dirLoc).createDirectory();
var fileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFileToMove25.txt";
var myFile = new kony.io.File(fileLoc).createFile();
var newFile = new kony.io.File(fileLoc).moveTo(mainLoc);
if (newFile !== null) {
    kony.print("moving to same loc with same name was successful");
} else {
    kony.print(" can't move to same loc with same name");
}
```

Return Values

Kony.io.File - returns a handle to File object pointing to destination file on success. Returns null on failure.

Exceptions

None

Platform Availability

Available to iOS, Android, and Windows platforms.

28.2.11 kony.io.File.read

The read API returns the kony.types.RawBytes of this file.

Syntax

```
kony.io.File.read()
```

Input Parameters

None

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFile313.txt";
var myFileName = new kony.io.File(myFileLoc);
try {
    var reading = new kony.io.File(myFileLoc).read();
    kony.print(reading);
    if (reading === null) {
        kony.print("null is coming from reading i.e can't be done");
    } else {
        kony.print("reading can be done on NonExistentFile");
    }
} catch (err) {
    kony.print("can't try read on nonExistent File causes Error");
}
```

Return Values

kony.types.RawBytes - rawbytes representing the content of the file.Returns null in case of non existent file.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

> ***Note:*** RawBytes will hold a handle of File object that it represents. The file content is not actually loaded into memory.

28.2.12 kony.io.File.remove

The remove API deletes a file or a directory.

Syntax

```
remove(boolean, deleteRecursive)
```

Input Parameters

Parameter	Description
boolean	By default, this is false. • True - deletes the folder and all its content recursively. • False - if the directory is empty it shall be removed.
deleteRecursive (optional)	Ignored in case of a file.

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFileToMove12.txt";
var myFile = new kony.io.File(myFileLoc);
myFile.createFile();
myFile.remove(true);
if (new kony.io.File(myFileLoc).exists()) {
    kony.print("removing file failed");
} else {
    kony.print("removing file was successful");
}
```

Return Values

None

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.13 kony.io.File.rename

The rename API renames a file or a directory.

Syntax

```
kony.io.File.rename(String newname)
```

Input Parameters

Parameter	Description
newname	new name for a file or a directory.

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFileToReName7578.txt";
var myFile = new kony.io.File(myFileLoc).createFile();
var newFile = new kony.io.File(myFileLoc).rename
("myFileToReName7577");
if (newFile) {
    kony.print("renaming file with name successfull with extension");
} else {
    kony.print("renaming file failed for name without Extension");
}
```

Return Values

Boolean - If successful, then boolean value is true,. Boolean value is false, if invalid file name or if the destination is a different directory than the current file.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.2.14 kony.io.File.write

The write API writes the given content into the file.

Syntax

```
kony.io.File.write(rawbytes/string, append)
```

Input Parameters

Parameter	Description
Rawbytes/string	data to write of type text string or kony.types.RawBytes
Append (optional)	true to append the data. Default is false, that means overrides the content.

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFile376.txt";
var myFileName = new kony.io.File(myFileLoc).createFile();
try {
    var writing = new kony.io.File(myFileLoc).write("How are you?");
    if (writing !== null) {
        kony.print("writing can be done on Non Existing Files");
    } else {
        kony.print("writing on nonExisting file returns null");
    }
} catch (err) {
    kony.print("can't try write on NonExistingFile, causes Error");
}
```

Return Values

Boolean - true if success, false otherwise.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.3 kony.io.FileList Namespace

FileList namespace represents the list of file objects.

FileList Properties:Name	Description
length: [type: Number, ReadOnly]	Returns the number of files in the FileList object.

The FileList library comprises of the following API.

28.3.1 Functions

The kony.io.FileList namespace contains the following functions.

28.3.1.1 item

The item API returns the File object at the specified index in the FileList.

Syntax

```
kony.io.File.item(index)
```

Input Parameters

Parameter	Description
Index	Index in the FileList.

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var newDirLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDirItem";

var myNewDir = new kony.io.File(newDirLoc).createDirectory();
var myFile1Loc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDirItem"
+ constants.FILE_PATH_SEPARATOR + "myFileItem1.txt";

var myFile1 = new kony.io.File(myFile1Loc).createFile();
var myFile2Loc = mainLoc + constants.FILE_PATH_SEPARATOR + "myDirItem"
+ constants.FILE_PATH_SEPARATOR + "myFileItem2.txt";

var myFile2 = new kony.io.File(myFile2Loc).createFile();
var myFileTemp = new kony.io.File(newDirLoc);

var filesListTemp = myFileTemp.getFilesList(); //getFilesList using
getFilesList API
alert(filesListTemp.length); //File list length must be greater than
0.
var myFile = filesListTemp.item(1); //Provide index of file
alert(JSON.stringify(myFile.name));
```

Return Values

Return Value	Description
File	File object at specified index. Null if there is no File at specified index or if index is not in FileList range.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

28.4 kony.io.FileSystem Namespace

The kony.io.FileSystem namespace includes APIs to create, read, and navigate through the device file system.

Information about the kony.io.FileSystem namespace is presented in the following topic.

28.4.1 Functions

The kony.io.FileSystem namespace contains the following functions.

28.4.1.1 kony.io.FileSystem.browse

This API provides the ability to browse and select files from your local system.

Syntax

```
kony.io.FileSystem.browse(browseConfig,browseCallBack);
```

Input Parameters

Parameter	Description
browseConfig [String] - Mandatory	Configuration params which is a simple JSON object of key value pairs, that drive the possible options during the browse. Currently the following keys are supported in the configuration: • **selectMultipleFiles**: (Boolean) - Default (true) • **filter**: An array indicates what file Filters should be applied to the file selection dialog. The file Filters need to follow the conventions specified in the IANAMediaTypes specification. For more information, see http://www.iana.org/assignments/media-types/media-types.xhtml.
browseCallBack [Object] - Mandatory	The callback is executed once the user closes the browse dialog. As part of call back, the platform provides the list of files that have been selected as an array of kony.io.FileSystem objects.

Example

```
function selectBtn_onClick_seq0() { // Event on button click
    var config = {
        selectMultipleFiles: true,
        filter: ["image/png", "image/jpeg"]
    };
    kony.io.FileSystem.browse(config, selectedFileCallback);
}
```

Return Values

None.

Platform Availability

- Desktop Web only

28.4.1.2 kony.io.FileSystem.copyBundledRawFileTo

Copies a bundled file from the application binary to the specified destination path on the device. While copying the pre-bundled file, the destination file name can be different from the source.

Syntax

```
kony.io.FileSystem.copyBundledRawFileTo (String rawFileName, String
destFilePath)
```

Input Parameters

Parameter	Description
rawFileName [String]	Specify the file name with the extension of a pre-bundled raw file that you want to copy.

Parameter	Description
destFilePath [String]	Specify the absolute file path along with the file name and extension (if any) where you want to copy the pre-bundled file. In this parameter, the destination file name can be different from the source.

Example

```
//The destination file name can be different from the source.
var destFilePath = kony.io.FileSystem.getDataDirectoryPath() +
"/Destinationfile.pdf";
var fileObj = null;
try {
    var file = new kony.io.File(destFilePath);
    //copyBundledRawFileTo API overrides the destination file with new
one.
    //Hence check before copying
    if (!file.exists()) {
        var fileObj = kony.io.FileSystem.copyBundledRawFileTo
(“Sourcefile.pdf”, destFilePath);
    } else {
        fileObj = file;
        alert("File is already available");
        return;
    }
} catch (e) {
    kony.print(“Exception‟ + e);
}
```

Remarks

- When the app functionality is tested in functional preview application, the kony.io.FileSystem.copyBundledRawFileTo API looks for the file in the child app (Downloaded test application). If not found, searches for the same file in the parent app (Functional Preview application).

- The kony.io.FileSystem.copyBundledRawFileTo API overrides the destination file with new one if any file exists with the same name at the destination path.

  For more information about bundling the files in the application binary, refer to the Pre bundling the Files topic.

Return Values

Kony.io.File returns a handle to the File object pointing to the destination file, if successful. If failure, then throws appropriate exception.

Exceptions

Error Code	Error Message
100	Source file not found.
100	Invalid Destination.
101	Invalid number of arguments.
106	Unknown error.

Platform Availability

- iOS
- Android
- Windows

28.4.1.3 kony.io.FileSystem.getAppGroupDirectoryPath

Retrieves the root path of an app group.

Syntax

```
kony.io.FileSystem.getAppGroupDirectoryPath(
    appGroupID)
```

Parameters

Parameter	Description
appGroupID	Holds a string containing the unique ID of the app group.

Example

```
var appGroupPath = kony.io.FileSystem.getAppGroupDirectoryPath
("String");
```

Return Values

A string containing the fully-qualified path to the root directory of the app group.

Platform Availability

- iOS only

28.4.1.4 kony.io.FileSystem.getApplicationDirectoryPath

getApplicationDirectoryPath API returns the application directory path.

Syntax

```
kony.io.FileSystem.getApplicationDirectoryPath()
```

Input Parameters

None

Example

```
var appDirectoryPath = kony.io.FileSystem.getApplicationDirectoryPath
();
```

Return Values

String

Exceptions

None

Platform Availability

- iOS only

28.4.1.5 kony.io.FileSystem.getCacheDirectoryPath

getCacheDirectoryPath API returns the application's cache directory path.

Syntax

```
kony.io.FileSystem.getCacheDirectoryPath()
```

Input Parameters

None

Example

```
var cacheDirectoryPath = kony.io.FileSystem.getCacheDirectoryPath();
```

Return Values

String

Exceptions

None

Platform Availability

- All native platforms
- Desktop Web

28.4.1.6 kony.io.FileSystem.getDatabaseDirectoryPath

Returns the application's database directory path (from application's private file system) where kony.db APIs access the database files.

This API is useful in accessing the pre-bundled SQLite database file. You need to copy the pre-bundled database file to the directory path returned by the kony.io.FileSystem.getDatabaseDirectoryPath API, using the kony.io.FileSystem.copyBundledRawFileTo API. You can then use the kony.db.openDatabase API to open the SQLite database files after copying.

Syntax

```
kony.io.FileSystem.getDatabaseDirectoryPath()
```

Input Parameters

None

Example

```
//Example for opening the pre-bundled database
//The destination file name can be different from the source.
var destFilePath = kony.io.FileSystem.getDatabaseDirectoryPath() +
”test.db”;
var fileObj = null;
try {
    var file = new kony.io.File(destFilePath);
    //copyBundledRawFileTo API overrides the destination file with new
one.
    //Hence check before copying
    if (!file.exists()) {
        fileObj = kony.io.FileSystem.copyBundledRawFileTo(dbName,
destFilePath);
    } else {
        fileObj = file;
        alert("File is already available");
        return;
    }
} catch (e) {
    kony.print(“Exception‟ + e);
}
if (fileObj == null) {
    kony.print(“Copy failed”);
} else {
    kony.print(“Copy Success”);
}
dbObject = kony.db.openDatabase("test.db", "1.0", "Prebundled SQL
Database", 5 * 1024 * 1024);
```

Return Values

Application's database directory path in the form of String.

Remarks

For Functional Preview, the path returned by the getDatabaseDirectoryPath API points to the child application on your device. This feature is available on iOS, Android, and Windows devices.

Exceptions

None

Platform Availability

- iOS
- Android
- Windows

28.4.1.7 kony.io.FileSystem.getDataDirectoryPath

Returns the application’s data directory path.

Syntax

```
kony.io.FileSystem.getDataDirectoryPath()
```

Input Parameters

None

Example

```
var dataDirectoryPath = kony.io.FileSystem.getDataDirectoryPath();
```

Return Values

String

Remarks

For Functional Preview, the path returned by the getDataDirectoryPath API points to the child application on your device. This feature is available on iOS, Android, and Windows devices.

Exceptions

None

Platform Availability

- Applicable for all native platforms, except Desktop Web

28.4.1.8 kony.io.FileSystem.getExternalCacheDir

The getExternalCacheDir API returns the absolute path to the directory on the primary shared or external storage device where the application can store its cached files.

> *Important:* This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.getExternalCacheDir();
```

Example

```
var dirPath = kony.io.FileSystem.getExternalCacheDir();
```

Return Values

String

Note: If the shared storage is not available, this API returns a `null` value.

Exceptions

None

Platform Availability

- Android

28.4.1.9 kony.io.FileSystem.getExternalCacheDirs

The getExternalCacheDirs API returns the absolute paths to the application-specific directories on all the shared or external storage devices where the application can store its cached files.

Important: This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.getExternalCacheDirs();
```

Example

```
var dirPath = kony.io.FileSystem.getExternalCacheDirs();
```

Return Values

Array of Strings

Note: If the shared storage is not available, this API returns a `null` value.

Exceptions

None

Platform Availability

- Android

> ***Note:*** If your device runs on Android versions before Android KitKat, the kony.io.FileSystem.getExternalFilesDirs API returns the same result as the kony.io.FileSystem.getExternalFilesDir API.

28.4.1.10 kony.io.FileSystem.getExternalFilesDir

The getExternalFilesDir API returns the absolute path to the directory on the primary shared or external storage device where the application can store its persistent files.

> ***Important:*** This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

> ***Note:***
>
> - From API level 29, apps do not require additional permissions to read and write files to this directory.
> - The files in this directory are internal to the application, and are deleted when you uninstall the app.

Syntax

```
kony.io.FileSystem.getExternalFilesDir(type);
```

Input Parameters

Parameters	Description

type	The type of file directory to return.
	
	The **type** can have one of the following values:
	
	`kony.io.FileSystem.DIRECTORY_MUSIC`
	
	`kony.io.FileSystem.DIRECTORY_PODCASTS`
	
	`kony.io.FileSystem.DIRECTORY_ALARMS`
	
	`kony.io.FileSystem.DIRECTORY_RINGTONES`
	
	`kony.io.FileSystem.DIRECTORY_NOTIFICATIONS`
	
	`kony.io.FileSystem.DIRECTORY_PICTURES`
	
	`kony.io.FileSystem.DIRECTORY_MOVIES`
	
	Note: If you provide null or invalid values for the type, the API returns the main directory path of the primary shared or external storage.

Example

```
var dirPath = kony.io.FileSystem.getExternalFilesDir
(kony.io.FileSystem.DIRECTORY_PICTURES);
```

Return Values

String

> ***Note:*** If the shared storage is not available, this API returns a `null` value.

Exceptions

None

Platform Availability

- Android

28.4.1.11 kony.io.FileSystem.getExternalFilesDirs

The getExternalFilesDirs API returns the absolute paths to the application-specific directories on all the shared or external storage devices where the application can store its persistent files.

> ***Important:*** This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

> ***Note:***
>
> - From API level 29, apps do not require additional permissions to read and write files to this directory.
> - The files in this directory are internal to the application, and are deleted when you uninstall the app.

Syntax

```
kony.io.FileSystem.getExternalFilesDirs(type);
```

Input Parameters

Parameters	Description

type	The type of file directory to return.
	
	The **type** can have one of the following values:
	
	`kony.io.FileSystem.DIRECTORY_MUSIC`
	
	`kony.io.FileSystem.DIRECTORY_PODCASTS`
	
	`kony.io.FileSystem.DIRECTORY_ALARMS`
	
	`kony.io.FileSystem.DIRECTORY_RINGTONES`
	
	`kony.io.FileSystem.DIRECTORY_NOTIFICATIONS`
	
	`kony.io.FileSystem.DIRECTORY_PICTURES`
	
	`kony.io.FileSystem.DIRECTORY_MOVIES`
	
	Note: If you provide null or invalid values for the type, the API returns the main directory path of the primary shared or external storage.

Example

```
var dirPath = kony.io.FileSystem.getExternalFilesDirs
(kony.io.FileSystem.DIRECTORY_PICTURES);
```

Return Values

Array of Strings

> ***Note:*** If the shared storage is not available, this API returns a `null` value.

Exceptions

None

Platform Availability

- Android

> ***Note:*** If your device runs on Android versions before Android KitKat, the kony.io.FileSystem.getExternalFilesDirs API returns the same result as the kony.io.FileSystem.getExternalFilesDir API.

28.4.1.12 kony.io.FileSystem.getExternalStorageDirectoryPath

getExternalStorageDirectoryPath API returns the path to the external storage, typically sdcard.

Syntax

```
kony.io.FileSystem.getExternalStorageDirectoryPath()
```

Input Parameters

None

Example

```
var mainLoc = kony.io.FileSystem.getExternalStorageDirectoryPath();
```

Return Values

String

Exceptions

None

Platform Availability

- Android only

Note: If the targetSdk version used is above version 28, scoped storage is enabled by default, and the legacy Storage APIs (getExternalStorageDirectoryPath) will not work. You must use scoped Storage APIs such as getExternalFilesDir and getExternalFilesDirs on devices that run on Android version 11, and above.

28.4.1.13 kony.io.FileSystem.getExternalStorageState

The getExternalStorageState API returns the current state of the shared or external storage media at the specified path.

Important: This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.getExternalStorageState(path);
```

Input Parameters

Parameters	Description
Path	Path to the file or directory.

Example

```
var path = kony.io.FileSystem.getExternalFilesDir();
var stats = kony.io.FileSystem.getExternalStorageState(path);
```

Return Values

File System Constant

The **File System Constant** can have one of the following values:

`kony.io.FileSystem.MEDIA_UNKNOWN`: Unknown storage state, such as when a path isn't backed by known storage media.

`kony.io.FileSystem.MEDIA_REMOVED`: Storage state if the media is not present.

`kony.io.FileSystem.MEDIA_UNMOUNTED`: Storage state if the media is present but not mounted.

`kony.io.FileSystem.MEDIA_CHECKING`: Storage state if the media is present and the disk is being checked.

`kony.io.FileSystem.MEDIA_NOFS`: Storage state if the media is present but is blank or is using an unsupported filesystem.

`kony.io.FileSystem.MEDIA_MOUNTED`: Storage state if the media is present and mounted at its mount point with read/write access.

`kony.io.FileSystem.MEDIA_MOUNTED_READ_ONLY`: Storage state if the media is present and mounted at its mount point with read/write access.

`kony.io.FileSystem.MEDIA_SHARED`: Storage state if the media is present not mounted, and shared via USB mass storage.

`kony.io.FileSystem.MEDIA_BAD_REMOVAL`: Storage state if the media was removed before it was unmounted.

`kony.io.FileSystem.MEDIA_UNMOUNTABLE`: Storage state if the media is present but cannot be mounted. Typically, this happens if the file system on the media is corrupted.

`kony.io.FileSystem.MEDIA_EJECTING`: Storage state if the media is in the process of being ejected.

Exceptions

None

Platform Availability

- Android

28.4.1.14 kony.io.FileSystem.getFile

getFile API returns a kony.io.File object representing the file for the given path. It returns a kony.io.File object for the specified file path. getFile API does not create a File.

Syntax

```
kony.io.FileSystem.getFile(string path)
```

Input Parameters

Parameters	Description
Path	path to the file or directory.

Example

```
var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR +
"myFileToGet.txt";
var myFile = new kony.io.File(myFileLoc).createFile();
var getMyFile = kony.io.FileSystem.getFile(myFileLoc);
if (getMyFile === null) {
    alert("getting File failed with null");
} else {
    alert(JSON.stringify(getMyFile.name));
}
```

Return Values

Kony.io.File

Platform Availability

- All native platforms
- Desktop Web

28.4.1.15 kony.io.FileSystem.getFileSystemStats

The getFileSystemStats API retrieves the detailed information about the space on a file system.

> ***Important:*** This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.getFileSystemStats(path);
```

Input Parameters

Parameters	Description
Path	Path to the file or directory.

Example

```
var path = kony.io.FileSystem.getExternalFilesDir();
var stats = kony.io.FileSystem.getFileSystemStats(path);
```

Return Values

A **JSON Object** that contains the values for the following parameters

`freeBytes`: Number of bytes that are free, including reserved blocks, on the file system.

`availableBytes`: Number of bytes that are free and available to applications on the file system.

`totalBytes`: Total number of bytes supported by the file system.

> ***Note:*** If the file path does not exist, this API returns a `null` value.
> On devices that run Android version 18 (or earlier), the value for totalBytes parameter is not returned.

Exceptions

None

Platform Availability

- Android

28.4.1.16 kony.io.FileSystem.getNoBackupFilesDir

The getNoBackupFilesDirAPI returns the absolute path to the directory on the file system that is similar to the kony.io.FileSystem.getDataDirectoryPath API. However, the files present in the directory returned by this API are excluded during the automatic backup process to the remote storage.

> ***Important:*** This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.getNoBackupFilesDir();
```

Input Parameters

None

Example

```
var dirPath = kony.io.FileSystem.getNoBackupFilesDir();
```

Return Values

String

Exceptions

None

Platform Availability

- Android

28.4.1.17 kony.io.FileSystem.getRawDirectoryPath

getRawDirectoryPath API returns the application’s “raw” directory path. getRawDirectoryPath API API is applicable only on Android.

Syntax

```
kony.io.FileSystem.getRawDirectoryPath()
```

Input Parameters

None

Example

```
var rawDirectoryPath =  kony.io.FileSystem.getRawDirectoryPath();
```

Return Values

Returns the directory path in the form of String.

Exceptions

None

Remarks

> ***Note:*** In the Android platform, you should not perform read and write operations to the raw directory.

Platform Availability

- Android
- iOS

28.4.1.18 kony.io.FileSystem.getSupportDirectoryPath

Returns the application's support directory path. This is only applicable for iOS.

Syntax

```
kony.io.FileSystem.getSupportDirectoryPath()
```

Input Parameters

None

Example

```
var supportDirectoryPath = kony.io.FileSystem.getSupportDirectoryPath
();
```

Return Values

String

Remarks

For Functional Preview, the path returned by the getSupportDirectoryPath API points to the child folder present inside the Application Support folder. This feature is available only on iOS devices.

Exceptions

None

Platform Availability

- iOS

28.4.1.19 kony.io.FileSystem.isExternalStorageAvailable

isExternalStorageAvailable API is used to check, if the external storage is available and accessible on the device. isExternalStorageAvailable API returns boolean value as true/false accordingly on Android platform.

Syntax

```
kony.io.FileSystem.isExternalStorageAvailable()
```

Input Parameters

None

Example

```
var isExternalStorageAvialable =
kony.io.FileSystem.isExternalStorageAvailable();
```

Return Values

Boolean

Exceptions

None

Platform Availability

- iOS
- Android
- Windows

28.4.1.20 kony.io.FileSystem.isExternalStorageEmulated

The isExternalStorageEmulated API is used to check if the primary shared or external storage media is emulated. The isExternalStorageAvailable API returns a boolean value as true/false accordingly.

Important: This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.isExternalStorageEmulated(File)
```

Input Parameters

Parameters	Description
File	File Object or path to the file.

Example

```
var dirPath = kony.io.FileSystem.getExternalFilesDirs
(kony.io.FileSystem.DIRECTORY_PICTURES);
var isEmulated = kony.io.FileSystem.isExternalStorageEmulated
(dirPath);
```

Return Values

Boolean

Exceptions

None

Platform Availability

- Android

Note: If this API is invoked on a device that runs on Android version KitKat, it always returns a `false` value.

28.4.1.21 kony.io.FileSystem.isExternalStorageLegacy

The isExternalStorageLegacy API is used to check if the primary shared or external storage media at the specified path is from the legacy view and includes files that are not specific to the app. The isExternalStorageLegacy API returns a boolean value as true/false accordingly.

Important: This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.isExternalStorageLegacy(File)
```

Input Parameters

Parameters	Description
File	File Object or path to the file.

Example

```
var dirPath = kony.io.FileSystem.getExternalFilesDirs
(kony.io.FileSystem.DIRECTORY_PICTURES);
var isLegacy = kony.io.FileSystem.isExternalStorageLegacy(dirPath);
```

Return Values

Boolean

Exceptions

None

Platform Availability

- Android

Note: If this API is invoked on a device that runs on Android versions prior to Android Q, it always returns a `true` value.

28.4.1.22 kony.io.FileSystem.isExternalStorageRemovable

The isExternalStorageRemovable API is used to check if the primary shared or external storage media at the specified path is physically removable. The isExternalStorageRemovable API returns a boolean value as true/false accordingly.

Important: This API has been introduced in the Kony Visualizer V8 Service Pack 4 Fixpack 98 release, to support scoped storage that has been added as part of the Android TargetSDK Level 29 enhancements.

Syntax

```
kony.io.FileSystem.isExternalStorageRemovable(File)
```

Input Parameters

Parameters	Description
File	File Object or path to the file.

Example

```
var dirPath = kony.io.FileSystem.getExternalFilesDirs
(kony.io.FileSystem.DIRECTORY_PICTURES);
var isRemovable = kony.io.FileSystem.isExternalStorageRemovable
(dirPath);
```

Return Values

Boolean

Exceptions

None

Platform Availability

- Android

> ***Note:*** If this API is invoked on a device that runs on Android version KitKat, it always returns a `false` value.

28.4.1.23 kony.io.FileSystem.uploadFiles

When the developer invokes this API with the list of files, the API internally iterates through the fileList and makes the network call to the URL mentioned and uploads the files. The API assumes that the application using this API has the necessary server side infrastructure to handle the upload.

Internally, the API uses AJAX-based multi-part upload of the files to the URL mentioned in all browsers except IE 8 and 9. In the case of IE 8 and 9, we provide frame-based submit as the multi-part through AJAX (as it is not handled by the browser).

Note: The API `kony.io.FileSystem.uploadFiles` cannot be tested in local jetty server included in the Kony Visualizer. Use Tomcat to test the API. The uploadURL must have code to receive and store the files sent via kony app. The sample war file UploadServlet30.war, contains code to handle uploading or storing files at server end. Both kony app and UploadURL needs to be on the same domain.

Syntax

```
kony.io.FileSystem.uploadFiles(URL, fileList, upLoadCallBack,
upLoadConfig)
```

Input Parameters

Parameter	Description
URL [String] - Mandatory	Provide target location to upload the file.
fileList [Object] - Mandatory	Specifies the list of files to be uploaded. This is as an array of kony.io.FileSystem.file objects.

Parameter	Description
uploadCallBack [Object] - Mandatory	The callback is invoked each time when the file is getting uploaded. The signature of the call back is expected to be as follows: *uploadCallBack(URL, upLoadStates).* • **upLoadState [Object] - Mandatory** This is an array of key value map states of the individual files that are uploaded. The key value map currently consists of the following keys: ◦ **file**: File (kony.io.FileSystem.file) object ◦ **status**: One of the following four status are possible : ▪ FILE_UPLOAD_START_STATE = 0 ▪ FILE_UPLOAD_PROGRESS_STATE = 1 ▪ FILE_UPLOAD_COMPLETE_STATE = 2 ▪ FILE_UPLOAD_ERROR_STATE = 3 ◦ **uploadBytes**: Number of bytes that are uploaded.
uploadConfig [Object]	Configuration parameters, which is a simple JSON object of key value pairs, that provides the possible options during the upload. Currently there are no keys.

Example

```
function UploadBtn_onClick_seq0() { // Event on upload button click
    var uploadURL = "http://10.10.4.17:8080/fileupload/upload";
    var filesQueue = [{
        File object
    }, {
        File Object
    }]; //Array of File objects got in browseCallBack()
    kony.io.FileSystem.uploadFiles(uploadURL, filesQueue,
upLoadCallBack);
};
```

Limitations for File Browse and Upload APIs

1. API cannot be used in pre-show, post-show, appService, pre-appinit, or post-app init.
2. For details on filter types for Browse API, follow the specification provided in the IANAMediaTypes. The Kony Visualizer depends on the browsers respecting the given filter type and enable appropriate file selection for the end user.
3. File Upload API does not work in the cross origin request scenarios. That is, the file can be uploaded only to the site from which the original app has been accessed. For more information, check CORS documentation.
4. For security reasons all browsers just give name or size of the file and not parent or full path. So, file parent and full path are not available in File object in browse or upload APIs. For more information, see http://stackoverflow.com/questions/15201071/how-to-get-full-path-of-selected-file-on-change-of-input-type-file-using-jav.
5. **IE 8 and 9 Limitations**:
    a. Multiple file selection will not work. As *multiple* attribute is not supported for input type= file, it allows the user to select only one file from the browse window even when ‘selectMultipleFiles’ is passed as true. For more information, see http://www.w3schools.com/tags/att_input_multiple.asp.
    b. Filters in the Browse API do not work. As *accept* attribute is not supported for input type= file, even through the filter is passed, it does not have any effect on browse window. User can select any file of any type. Hence, server side file validation is recommended. For more information, see http://www.w3schools.com/tags/att_input_accept.asp.
    c. The file object in Browse API callback method will not have file size in IE 8. IE 8 browsers do provide file path, but it is inconsistent and depends on IE 8 security configuration (IE 9

provide fakepath). We recommend to avoid accessing file parent, path and size in IE 8 and 9. File size is provided in IE 10, Firefox, and Chrome.

d. IE 8 and 9 uses iframe for file upload. The uploadprogress event is not triggered and bytes uploaded information is not provided in Upload API callback. So, callback is fired only twice.

   i. When you submit form to iframe with FILE_UPLOAD_PROGRESS_STATE.

   ii. When you get error or success onload event of iframe with respective state constants.

e. **Multiple Upload**. If a widget onclick action invokes the Browse API and user selects the file multiple times from the browse window, in browse API callback, you must handle all the files (selected by the user one by one). When Upload API is called, all the files user selected is uploaded to the specified URL. If you want to upload different files on different servers or URLs, then they need to call Browse API and Upload API subsequently one after the other. That is, you need to upload the selected file before letting the user select an other file (in case user wants to upload files on different URLs).

f. Empty files cannot be uploaded in case of IE browsers.

6. **IE 8 and 9 Limitations**:

   a. Multiple file selection will not work. As *multiple* attribute is not supported for input type= file, it allows the user to select only one file from the browse window even when 'selectMultipleFiles' is passed as true. For more information, see http://www.w3schools.com/tags/att_input_multiple.asp.

   b. Filters in the Browse API do not work. As *accept* attribute is not supported for input type= file, even through the filter is passed, it does not have any effect on browse window. User can select any file of any type. Hence, server side file validation is recommended. For more information, see http://www.w3schools.com/tags/att_input_accept.asp.

c. The file object in Browse API callback method will not have file size in IE 8. IE 8 browsers do provide file path, but it is inconsistent and depends on IE 8 security configuration (IE 9 provide fakepath). We recommend to avoid accessing file parent, path and size in IE 8 and 9. File size is provided in IE 10, Firefox, and Chrome.

d. IE 8 and 9 uses iframe for file upload. The uploadprogress event is not triggered and bytes uploaded information is not provided in Upload API callback. So, callback is fired only twice.

   i. When you submit form to iframe with FILE_UPLOAD_PROGRESS_STATE.

   ii. When you get error or success onload event of iframe with respective state constants.

e. **Multiple Upload**. If a widget onclick action invokes the Browse API and user selects the file multiple times from the browse window, in browse API callback, you must handle all the files (selected by the user one by one). When Upload API is called, all the files user selected is uploaded to the specified URL. If you want to upload different files on different servers or URLs, then they need to call Browse API and Upload API subsequently one after the other. That is, you need to upload the selected file before letting the user select an other file (in case user wants to upload files on different URLs).

f. Empty files cannot be uploaded in case of IE browsers.

7. When the user uploads a file that has special characters in its file name, the special characters in the file name are replaced with other special characters on the server.

8. Check if the user is selecting the same file again and create appropriate conditions for browseCallBack. The check will help identify same files if selected earlier and eliminate duplicate files from getting uploaded.

9. **Firefox Limitation**. Filters with specific Mimetype might not work properly. It is a known bug in Firefox, for more information, see:

   - http://techblog.procurios.nl/k/618/news/view/15872/14863/mimetype-corruption-in-firefox.html

- http://support.mozilla.org/en-US/questions/953914
- https://bugzilla.mozilla.org/show_bug.cgi?id=826176
- https://bugzilla.mozilla.org/show_bug.cgi?id=373621

> ***Note:*** Constants: Constants.FILE_PATH_SEPARATOR: Platform specific File path separator.

Return Values

None.

Platform Availability

- Desktop Web only

28.5 kony.types Namespace

The kony.types namespace offers the following API elements.

- RawBytes Object

28.5.1 RawBytes Object

The **RawBytes** Object represents the file content.

RawBytes is an opaque object that is returned from the following sources:

- camera object: <object>.rawBytes
- HttpRequest object(kony.net.HttpRequest): <object>.response
- File object(kony.io.File): <object>.read()
- Image object: <object>.getImageAsRawBytes() and <object>.findImageInGallery()
- kony.convertToRawBytes()

The RawBytes Object provides the following API elements:

- Properties
- Functions

28.5.1.1 Properties

The RawBytes object has the following property:

text

This API returns the UTF-8 representation of data in the RawBytes. Returns null if the RawBytes object represents a binary data.

Syntax

```
<RawBytes object>.text
```

Input Parameters

None

Example

```
 var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myFile.txt";
var rawBytesObj = new kony.io.File(myFileLoc).read();
if (rawBytesObj === null) {
    kony.print("rawBytes object is null");
} else {
    kony.print(rawBytesObj.text);
}
```

Return Value

String

Read/ Write

Read only.

Platform Availability

Android

28.5.1.2 Functions

The kony.types namespace provides the following functions:

kony.types.RawBytes.getResourcePath

This API returns the location of rawbytes which can be Android content URI or a file path.

Syntax

```
kony.types.RawBytes.getResourcePath()
```

Input Parameters

None

Example

```
 var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myFile.txt";
var rawBytesObj = new kony.io.File(myFileLoc).read();
if (rawBytesObj === null) {
    kony.print("The resource path is not available");
} else {
    kony.print("The resource path for file is" + rawBytesObj.getResourcePath
())
}
```

Return Value

Return Value	Description
String	Returned when file path is found.
Null	Returned when file path is not available.

Exceptions

None

Platform Availability

- Android
- iOS

kony.types.RawBytes.getTempPath

This API returns the path where the source files of all RawBytes are stored.

Syntax

```
kony.types.RawBytes.getTempPath()
```

Input Parameters

None

Example

```
var tempPath = kony.types.RawBytes.getTempPath();
```

Return Value

The path of source files of all RawBytes.

Exceptions

None

Platform Availability

- iOS

readAsText

This API returns the data as text represented by the RawBytes object. It returns null if the RawBytes object represents binary data.

Syntax

```
<RawBytes object>.readAsText()
```

Input Parameters

None

Example

```
 var mainLoc = kony.io.FileSystem.getDataDirectoryPath();
var myFileLoc = mainLoc + constants.FILE_PATH_SEPARATOR + "myFile.txt";
var rawBytesObj = new kony.io.File(myFileLoc).read();
if (rawBytesObj === null) {
    kony.print("rawBytes object is null");
} else {
    kony.print(rawBytesObj.readAsText());
}
```

Return Values

String - text available in this rawbytes.

Exceptions

None

Platform Availability

Available for iOS, Android, and Windows platforms.

29. Internationalization API

Internationalization is the process of enhancing an application to support multiple languages across various regions. Internationalization is abbreviated as i18n. The `kony.i18n` namespace provides a comprehensive set of functions for developing multilingual applications.

The process of designing or developing an application in such a way that it supports various languages and regions is known as *Internationalization*. For an application to support *Internationalization*, you need not make any changes in the application code or logic. *Internationalization* is abbreviated as *i18n*. *i18n* support for an application has to be defined at the time of developing the application.

The Internationalization API uses `kony.i18n Namespace` and the following API elements.

Function	Description
`kony.i18n.deleteResourceBundle`	Enables you to delete an existing resource bundle.
`kony.i18n.getCurrentLocale`	Returns the locale (string) that is currently being used by the application to populate the localized data.

Function	Description
`kony.i18n.getCurrentDeviceLocale`	Provides you the ability to fetch the current locale of the device.
`kony.i18n.getLocalizedString`	Returns the localized string that corresponds to the specified i18n Key.
`kony.i18n.getSupportedLocales`	Retrieves a list of all the locales supported by the device.
`kony.i18n.isLocaleSupportedByDevice`	Provides you the ability to view whether a locale is supported by a device.

Function	Description
kony.i18n.isResourceBundlePresent	Checks if a resource bundle exists for a given locale and returns a boolean value.
kony.i18n.setDefaultLocaleAsync	Enables you to set the specified locale as the default locale for the application.
kony.i18n.setCurrentLocaleAsync	Provides you the ability to set the specified locale as the current locale of the application.

Function	Description
`kony.i18n.setLocaleLayoutConfig`	Enables you to define the Right-To-Left (RTL) behavior for each locale in an application.
`kony.i18n.setResourceBundle`	Enables you to set the specified resource bundle for a given locale.
`kony.i18n.updateResourceBundle`	Enables you to append new key-value pairs to the given resource bundle for a specified locale.

You can find the locales supported by a device. The `kony.i18n.getSupportedLocales` function fetches all the locales supported by a device. If you want to check whether a specific locale is supported by a device, use the `kony.i18n.isLocaleSupportedByDevice` function

There are two types of locales, device-specific locales and application-specific locales. Use the `kony.i18n.getCurrentDeviceLocale` function to find the current locale of a device. Fetch the current locale supported by an application using the `kony.i18n.getCurrentLocale` function. If you want to change the locale supported by an application, use the `kony.i18n.setCurrentLocaleAsync` function. You can set a default locale for an application by using the `kony.i18n.setDefaultLocaleAsync` function. Once a locale is set, configure the layout of your locale, and define the Right-To-Left behavior of the locale by using the `kony.i18n.setLocaleLayoutConfig` function.

Every locale comprises of a resource bundle. Check whether the selected locale contains a resource bundle using the `kony.i18n.deleteResourceBundle` function. If the locale does not contain a resource bundle, use the `kony.i18n.setResourceBundle` function to set a specific resource bundle for the locale. You can then update the resource bundle with new key-value pairs, by using the `kony.i18n.updateResourceBundle` function. To obtain a localized string corresponding to an i18n key, use the `kony.i18n.getLocalizedString` function.

Before you get started with using the kony.i18n namespace and functions, you should get to know a few things. They are, Resource bundle and Implementing i18N.

Note: From V8 SP4 onwards, the i18n database data for a Kony Visualizer App Viewer child app is stored in child app data and not under the parent app. This feature is applicable for iOS, Windows, and Android platforms.

29.1 Resource Bundle

Resource bundle is a properties file that contains all the i18n Keys and their pair values. The resource bundle is locale specific, that is, you will have one resource bundle per locale. The resource bundle follows the naming convention of `<language_Country>.properties`. For example,

- For English locale in United States, the resource bundle will be `en_US.properties.`
- For French locale in Canada, the resource bundle will be `fr_CA.properties.`

The resource bundle has a list of key-value pairs that are used as internationalization keys within the application.

29.2 Implementation Details

The current practice to support *i18n* for applications is to place the text (i18n keys) in the resource bundles that are loaded into the application. Applications are built to access the resource bundles depending on the selected locale data. For an application to support multiple languages, the application should be designed to select the relevant resource bundle at run-time. The keys in the resource bundles are translated to the required languages at run-time.

An internationalized application has the following benefits:

- The same application can run on multiple locales.
- Widget's text is not hard-coded in the application. Instead, the localized keys are retrieved dynamically.
- Support for new locales does not require recompilation.
- Region-dependent data such as dates and currencies appear in formats that conform to the end user's region and language.

To view the functionality of the Internationalization API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

29.3 kony.i18n Namespace

The kony,i18n Namespace, which forms the Internationalization API, provides the following API elements.

Note: From V8 SP4 onwards, the i18n database data for a Kony Visualizer App Viewer child app is stored in child app data and not under the parent app. This feature is applicable for iOS, Windows, and Android platforms.

29.3.1 Functions

The kony.i18n namespace, which is used for internationalization, provides the following functions.

29.3.2 kony.i18n.deleteResourceBundle

This API allows you to delete an existing resource bundle. If a resource bundle does not exist, the API will return without performing any operation. Only resource bundles which are newly created using the *setResourceBundle* API will be deleted. The bundle which are created by IDE, cannot be deleted, but they can only be updated.

Syntax

```
kony.i18n.deleteResourceBundle(locale)
```

Input Parameters

Parameter	Description
locale [String] - Mandatory	Specifies the locale for which the resource bundle needs to be deleted.

Example

```
//To delete the resource bundle
deleteResourceBundle: function() {
    kony.i18n.deleteResourceBundle("de_DE");
    alert(" Resources bundle has been deleted.");
},
```

Return Values

None

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms.

29.3.3 kony.i18n.getCurrentLocale

This API returns the locale (string) that is currently being used by the application to populate the localized data. This locale might be different than the system locale. The locale follows the format *language_Country*.For example, `en_US.Country` is not mandatory.

Use Cases

You can use this API to know the current locale of the application before:

- changing the locale.
- deleting the resource bundle of the locale.

Syntax

```
kony.i18n.getCurrentLocale()
```

Example

```
//To get the Locale name that is set on your app
getCurrentLocale:functio(){
var currentLocales = kony.i18n.getCurrentLocale();
alert("CurrentLocale :" + currentLocales);
},
```

Input Parameters

None.

Return Values

Return Value	Description
Current locale [String]	The current locale that is being used by the application is returned.

Exceptions

1300 - i18n error or Locale not supported error

Implementation Details

The current locale of the application is identified using the following rules:

Device Locale	Locales Supported by the application	Default Locale	Locale returned by *getCurrentLocale* API
en_GB	en_US, zh_TW	en_TW	*en_TW* (Since the device locale is not supported by the application, the API falls back to default locale)
en_US	en_US, en , zh_TW	en_TW	*en_US* (Since the device locale is supported by the application, the API returns it as the current locale)
en_US	en, zh_TW	en_TW	*en* (Since the device locale is supported by the application, the API falls back to the country)

This API follows the rules given below to identify the current locale:

1. **When the device locale is not supported by the application:**

   For example, if the

   - application supports *fr_GR*, *nl_NL*.
   - default locale is (set from IDE) *nl_NL*.
   - device locale is *en_GB.*

   The *kony.i18n.getCurrentLocale()* API returns *nl_NL*.

2. **When the device locale supports just the language part and not the region part**

   For example,

   - application supports *fr_GR*, *en_US*, *en_GB*, *nl_NL*.
   - default locale is (set from IDE) *nl_NL*.
   - device locale is *en*..

   > ***Note:*** On iPhone and Android, the device settings mandate that a region must also be associated with a language.

   The *kony.i18n.getCurrentLocale()* API picks up the first locale that matches the language. In this example, the API returns *en_US* as it is the first locale that matches the language as specified in the device system settings.

3. **When appropriate fonts are not installed on the device for a given locale**

   For example,

   - application supports *fr_FR* and *nl_NL*

- default locale is (set from IDE) *nl_NL*
- device locale is *en_GB.*

The *kony.i18n.getCurrentLocale()* API returns *nl_NL*. If the device does not have Dutch fonts installed, the application UI displays junk characters.

Platform Availability

Available on all platforms.

29.3.4 kony.i18n.getCurrentDeviceLocale

This API provides you the ability to fetch the current locale of the device.

Syntax

```
kony.i18n.getCurrentDeviceLocale()
```

Example

```
//To get the current device locale
getCurrentDeviceLocale:function(){
    var locale = kony.i18n.getCurrentDeviceLocale();
    alert("current device locale is " + locale);
},
```

Input Parameters

None

Return Values

Return Value	Description
listOfLocales [Table]	A table with the following key-value pairs is returned: • language • country • name For example, `local myLocaleDetails = kony.i18n.getCurrentDeviceLocale()` Where myLocaleDetails={language="en", country="US", name="English US"}

Note: In iOS platform this API returns a string value.

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on iOS, Android, Windows Mango, Windows 8.

29.3.5 kony.i18n.getLocalizedString

This API returns the localized string that corresponds to the specified i18n Key.

Syntax

```
kony.i18n.getLocalizedString(i18nkey)
```

Input Parameters

Parameter	Description
i18nkey [String]- Mandatory	Specifies the internationalization key for which the localized string needs to be returned.

Example

```
//To get the localized string that corresponds to the specified i18n
Key.
getLocalizedString: function() {
    var currentLocale = kony.i18n.getLocalizedString("Hello");
    alert(" LocalizedString Method called :" + currentLocale);
},
```

Return Values

Return Value	Description
Localizedstring [String]	Returns the localized string corresponding to the key value.

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms.

29.3.6 kony.i18n.getSupportedLocales

This API retrieves a list of all the locales supported by the device.

Syntax

```
kony.i18n.getSupportedLocales()
```

Example

```
//To get all the locales supported by the device
 getSupportedLocale: function() {
    var supportedLocales = kony.i18n.getSupportedLocales();
    alert("Supported Locales :" + JSON.stringify(supportedLocales));
},
```

Input Parameters

None

Return Values

Return Value	Description
listOfLocales [Table]	A table with the following key-value pairs is returned: • language • country • name ***Note:*** In Windows platform, Country and Name are not supported. For example, `local list = kony.i18n.getSupportedLocales()`

```
Where list={{language:"en", country:"US", name:"English US"},
{language:"en", country:"UK", name:"English UK"}}
```

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms except Server Side Mobile Web, Desktop Web and SPA. *SPA and Desktop Web returns only current locale instead of all locales supported by the browser.

29.3.7 kony.i18n.isLocaleSupportedByDevice

This API provides you the ability to view whether a locale is supported by a device.

Syntax

```
kony.i18n.isLocaleSupportedByDevice(locale)
```

Input Parameters

Parameter	Description
locale [String] - Mandatory	Specifies the locale that must be verified if it is supported by the device. locale must be in one of the following formats: • language_countryCode • language For example, `kony.i18n.isLocaleSupportedByDevice ("en_US")` `k ony.i18n.isLocaleSupportedByDevice ("fr")`

Example

```
//To check whether the specified locale is supported by the device or
not.
Checkthelocale: function() {
    var a = this.view.txtbx.text;
    if (kony.i18n.isLocaleSupportedByDevice("a") === true) {
        alert("This locale is supported");
    } else {
        alert("This Locale is not supported");
    }
},
```

Return Values

Return Value	Description
status [Boolean]	Indicates the status if the specified locale is supported by the device. • *true* - if the specified locale is supported by the device. • *false* - if the specified locale is not supported by the device.

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms except SPA and Desktop Web.

29.3.8 kony.i18n.isResourceBundlePresent

This API checks if a resource bundle exists for a given locale and returns a boolean value.

Syntax

```
kony.i18n.isResourceBundlePresent (locale)
```

Input Parameters

Paramater	Description
locale [String] - Mandatory	Specifies the locale for which the resource bundle is checked.

Example

```
  isResourceBundlePresent: function() {
    try {
        var exists = kony.i18n.isResourceBundlePresent("en_GB");
        alert("English resource bundle is present " + exists);
    } catch (i18nError) {
        alert("Exception While getting isResourceBundlePresent : " +
i18nError);
    }
},
```

Return Values

Return Value	Description
status [Boolean]	*true* - if the resource bundle exists for the given locale. *false* - if no resource bundle exists for the given locale.

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms except SPA and Desktop Web.

29.3.9 kony.i18n.setDefaultLocaleAsync

Every application that has support for internationalization needs a locale to be set as a default locale. This API allows you to set the specified locale as the default locale for the application. You can also set the default locale from the IDE. However, this API allows you to perform that task dynamically.

Use Cases

You must set the resource bundle for a given locale to have internationalization support for that locale.

Syntax

```
kony.i18n.setDefaultLocaleAsync(localename, onsuccesscallback,
onfailurecallback, info)
```

Input Parameters

Parameter	Description
localename [String] - Mandatory	Specifies the locale that must be set as the default locale of the application.

Parameter	Description
onsuccesscallback [Function] - Mandatory	onsuccess callback is called by the platform after successfully setting the specified locale as default. `onsucess (oldlocalename, newlocalename){ \\code }`
onfailurecallback [Function] - Mandatory	onfailure callback will be called by the platform if the locale is not set successfully. `onfailure (errCode, errMsg){ \\code }`

Parameter	Description
info [Object] - Optional	A JavaScript object consisting of key value pairs. If info parameter is specified, it is passed to the callback function as a last parameter. If the info parameter is not specified, the callback function receives the info as null/nil. "Info" is basically a user data where in the application developers will pass it to the async API's and the platform returns this info object to the corresponding async callback. This parameter is helpful for developers to remember the context when the methods are called in asynchronous fashion.Developers can define any custom keys and values within the info Object based on the needs. These are not predefined keys with values.

Example

```
//To set the default locale for your app.
  setDefaultLocaleAsync: function(){
    kony.i18n.setDefaultLocaleAsync("fr_FR", this.onsuccesscallback,
this.onfailurecallback);
  },
```

Return Values

None

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms.

29.3.10 kony.i18n.setCurrentLocaleAsync

This API provides you the ability to set the specified locale as the current locale of the application. If the locale is not supported by the device, junk characters are displayed on the screen for the locale specific string.

Use Cases

You must set the resource bundle for a given locale to have internationalization support for that locale.

Syntax

```
kony.i18n.setCurrentLocaleAsync(localename, onsuccesscallback,
onfailurecallback, info)
```

Input Parameters

Parameter	Description
localename [String] - Mandatory	Specifies the locale that must be set as the default locale of the application.

Parameter	Description
onsuccesscallback [Function] - Mandatory	onsuccess callback will be called by the platform after a locale is set successfully as the current locale. `onsucess (oldlocalename, newlocalename){ \\code }`
onfailurecallback [Function] - Mandatory	onfailure callback will be called by the platform if the locale is not set successfully. `onfailure (errCode, errMsg){ \\code }`

Parameter	Description
info [Object] - Optional	A JavaScript object consisting of key value pairs. If info parameter is specified, it is passed to the callback function as a last parameter. If the info parameter is not specified, the callback function receives the info as null/nil. "Info" is basically a user data where in the application developers will pass it to the async API's and the platform returns this info object to the corresponding async callback. This parameter is helpful for developers to remember the context when the methods are called in asynchronous fashion.Developers can define any custom keys and values within the info Object based on the needs. These are not predefined keys with values.

Example

```
/*By using this function, you set a locale based on your input that
you select from the
  list box widget.*/
setlocaleListbox: function() {
    if (this.view.lstbx.selectedKey == "lb1") {
        kony.i18n.setCurrentLocaleAsync("en_GB",
this.onsuccesscallback, this.onfailurecallback);
    } else if (this.view.lstbx.selectedKey == "lb2") {
        kony.i18n.setCurrentLocaleAsync("es_AR",
this.onsuccesscallback, this.onfailurecallback);
    } else if (this.view.lstbx.selectedKey == "lb3") {
        kony.i18n.setCurrentLocaleAsync("fr_FR",
this.onsuccesscallback, this.onfailurecallback);
    }
},
```

Return Values

None

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms.

29.3.11 kony.i18n.setLocaleLayoutConfig

This API helps you to define the Right-To-Left (RTL) behavior for each locale in an application.

Note: When the application calls the kony.i18n.setDefaultLocaleAsync API, Kony Framework loads forms with the corresponding Locale.The forms that were loaded previously must be destroyed in application code to see the effect of the RTL/LTR feature for that particular form.

Syntax

```
kony.i18n.setLocaleLayoutConfig({ "<locale_key1>" :
{"mirrorFlexPositionProperties", "mirrorContentAlignment",
"mirrorFlowHorizontalAlignment":}
```

Input Parameters

Parameter	Description
mirrorFlexPositionProperties [Boolean] - Mandatory	This property reverses the layout properties for left and right alignment. At run time the framework iterates through all the widgets in any given form and if the layout is set to RTL (mirrorFlexPositionProperties = true), the following actions are performed: • For any widget for which the values for **Left** and **Width** are set but **Right** is empty, the Left value is then replaced with the Right value; and the Left value becomes Null. • For any widget for which the values for **Right** and **Width** are set but **Left** is empty, the Right value is then replaced with the Left value; and the Right value becomes Null. • For any widget that has both **Right** and **Left** values set, the values are swapped between the two. • For any widget for which **Padding** has a **Left** value, the Left value is then replaced with the Right value. • For any widget for which **Padding** has a **Right** value, the Right value is

Parameter	Description
mirrorContentAlignment [Boolean] - Mandatory	This property reverses the content alignment from right to left. At run time the framework iterates through all the widgets in any given form and if the layout is set to RTL(mirrorContentAlignment = true), the following actions are performed: • For any widget for which Content Alignment is set as **Left**, the content alignment is changed to **Right**. • For any widget for which Content Alignment is set as **Right**, the content alignment is changed to **Left**.
mirrorFlowHorizontalAlignment [Boolean] - Mandatory	This property converts the flow horizontal alignment of a FlexContainer from left to right. Kony Framework internally applies the **NOT** operator on the **reverseLayoutDirection** Property of horizontal FlexContainer to change the alignment.

Example

```
kony.i18n.setLocaleLayoutConfig({
    "<locale_key1>": {
        "mirrorFlexPositionProperties": true / false,
        "mirrorContentAlignment": true / false,
        "mirrorFlowHorizontalAlignment": true / false,
    },
    "<locale_key2>": {
        "mirrorFlexPositionProperties": true / false,
        "mirrorContentAlignment": true / false,
        "mirrorFlowHorizontalAlignment": true / false,
    },
});
```

Platform Availability

- iOS
- Android
- Windows
- SPA

Limitations

- For Windows, if you use the **isI18nLayoutConfigEnabled** key and the **kony.application.setApplicationLayout** API together in a single application, the application does not function as expected.

29.3.12 kony.i18n.setResourceBundle

This API allows you to set the specified resource bundle for a given locale. If the specified locale has a resource bundle already set, it is overridden with the given resource bundle.

If no resource bundle has been set previously, this API will create a new resource bundle.

Use Cases

You must set the resource bundle for a given locale to have internationalization support for that locale.

Syntax

```
kony.i18n.setResourceBundle(inputtable, locale)
```

Input Parameters

Paramater	Description
inputtable [Table] - Mandatory	Specifies the resource bundle that needs to be set for the given locale.
locale [String] - Mandatory	Specifies the locale for which the resource bundle needs to be set.

Example

```
//To set a resource bundle for the key "Hello"
  setResourceBundle: function(){
    kony.i18n.setResourceBundle({
      Hello: "Hallo Welt",
    }, "de_DE");
    kony.i18n.setCurrentLocaleAsync("de_DE",this.onsuccesscallback,
this.onfailurecallback);
  },
```

Return Values

None

Exceptions

1300 - i18n error or Locale not supported error, Locale not supported.

Platform Availability

Available on all platforms.

29.3.13 kony.i18n.updateResourceBundle

This API allows you to append new key-value pairs to the given resource bundle for a specified locale. The key-value pairs you provide will be appended at the end of the resource bundle.

If no resource bundle exists for the specified locale, a new resource bundle is created.

Syntax

```
kony.i18n.updateResourceBundle(inputtable, locale)
```

Input Parameters

Paramater	Description
inputtable [Table] - Mandatory	Specifies the resource bundle that needs to be set for the given locale.
locale [String] - Mandatory	Specifies the locale for which the resource bundle needs to be set.

Example

```
//To update the resource bundle with a different text
  updateResourceBundle: function(){
 kony.i18n.updateResourceBundle({
           Hello: "Hallo Leute",
        }, "de_DE");
  kony.i18n.setCurrentLocaleAsync("de_DE",this.onsuccesscallback1,
this.onfailurecallback);
},
```

Return Values

None

Exceptions

1300 - i18n error or Locale not supported error

Platform Availability

Available on all platforms.

29.4 Deprecated Functions

The following APIs have been deprecated from 5.0 onwards and are only supported for backward compatibility.

kony.i18n.setCurrentLocale

This API provides you the ability to set the specified locale as the current locale of the application. If the locale is not supported by the device, junk characters will be displayed on the screen for the locale specific string.

Syntax

```
kony.i18n.setCurrentLocale(locale)
```

Input Parameters

locale [String] - Mandatory

Specifies the locale that must be set as the current locale of the application

Return Values

errorcode [Number]

A number that denotes the error.

- 0 - Locale is successfully applied
- 100 - The supplied locale is not available in the device.
- 101 - The supplied locale is not available in application

Error Codes

- 100 - if the locale is not supported by device
- 101 - if the locale is supported by device but not by the application

Implementation Details

This section explains how this API is implemented. This API:

- sets the appropriate display language and region for displaying content and images in the application
- sets the specified locale as the input locale (keyboard and calendar display) if the specified locale is available - on Android
- displays junk characters for the locale specific string if the specified locale is not supported by the device
- does not re-initialize the widgets that have already loaded the i18n content. For example, assume that an application has 2 forms frmA and frmB.
    - *frmA* is the startup form and loads the content using *i18n.getlocalizedstring* API. The current locale for the device is *en_US* and English text is populated on the form.
    - *onclick* event of the button on this form invokes *i18n.setcurrentlocale("fr_FR")*
    - Invoking this API at this stage will not cause the widgets to reload the data for *frmA* widgets from *fr_FR* resource bundle.

Platform Availability

- Android
- Windows Phone/Windows Kiosk/Mango

kony.i18n.setDefaultLocale

Every application that has support for internationalization needs a locale to be set as a default locale. This API allows you to set the specified locale as the default locale for the application. You can also set the default locale from the IDE. However, this API allows you to perform that task programmatically.

Syntax

```
kony.i18n.setDefaultLocale(locale)
```

Input Parameters

locale [String] - Mandatory

Specifies the locale that needs to be set as default locale for the application

Return Values

None.

Error Codes

- 100 - if the locale is not supported by device
- 101 - if the locale is supported by device but not by the application

Implementation Details

This API is invoked in the background when the code is generated for an application to set the default locale.

Platform Availability

Available on all platforms except J2ME.

30. Keychain API

Keychain services help you to securely store passwords, keys, certificates, and notes for one or more users. The Keychain API provides your app a mechanism to store chunks of user data (such as passwords) in an encrypted database. This credential information is saved in the device's keychain, and your app can retrieve and/ or remove the data if required.

Using the Keychain API, your app can save users' credential information in the device's keychain. The app can then retrieve this information later and log users on to your app's back-end services, so that users do not have to do it manually. The keychain data should be of type String; for storing any other type of data, you can use base64 encoded string. Keychain API is available on iOS and Android devices.

The Keychain API uses `kony.keychain Namespace` and the following API elements.

Function	Description
`kony.keychain.remove`	Deletes the credential information from the device's keychain.
`kony.keychain.retrieve`	Retrieves the specified credential information from the device's keychain.
`kony.keychain.save`	Saves credential information in the device's keychain.

To save users' credential information in the device's keychain, use the `kony.keychain.save` function. Then retrieve the saved information by using the `kony.keychain.retrieve` function to log users on to your app's back-end services. This enables users to log in easily. The keychain data should be of type String; for storing any other type of data, you can use base64 encoded string. To delete any information, use the `kony.keychain.remove` function.

30.0.1 Keychain Functionality Across Various Platforms

iOS

Every iOS device has a single keychain that is available to all apps. In iOS, the keychain is automatically unlocked when the device is unlocked, so all applications always have access to their own unique keychain item data when used by the relevant user. Developers can store the data by passing a dictionary of securedata string, secure account name, and an identifier in order to identify the item in the keychain.

For more information on Keychain API for iOS, click here.

Android

In Android, items saved in one app cannot be retrieved from another app. Developers need to store the data by passing a dictionary of securedata string and an identifier in order to identify the item in the keychain. This securedata string should be less than or equal to 245 characters.

> *Important:* Even though the namespace used is kony.keychain, for Android, the underlying implementation is Keystore APIs.

For more information on Keychain API for Android, click here.

To view the functionality of the Keychain API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

30.1 kony.keychain Namespace

The kony.keychain namespace consists of functions that enable your app to access a device's keychain.

30.1.1 Functions

kony.keychain.remove

Deletes the credential information from the device's keychain. It deletes secure data from the keychain with the provided identifier.

Syntax

```
kony.keychain.remove(
    identifier);
```

Input Parameters

Parameter	Description
identifier	A JavaScript Dictionary object containing a key-value pair that specifies the identifier of the credential information to remove. The only supported key is 'identifier.' Its value must be a string that contains the credentialInfo's unique identifier.

Example

```
var cred = {
    "identifier": "Apple"
};
var operationSuccessful = kony.keychain.remove(cred);
```

```
  //Use the below function to remove sensitive data from your device keychain
  remove: function() {
    var cred = {
        "identifier": "Apple"
    };
    kony.keychain.remove(cred);
    alert("The details are removed");
}
```

Return Values

If the credential information is successfully removed, 0 is the return value. If the credentials were not removed or not found, the appropriate error code is the return value.

Android-specific Error/Success Codes

Error/Success Code	Error/Success Message
0	SUCESSCODE
-50	INVALID_PARAM_ ERRORCODE
-25311	SIZE_NOT_ALLOWED_ ERRORCODE
-25300	ITEM_NOT_FOUND_ ERRORCODE
-36	IO_ERRORCODE

Error/Success Code	Error/Success Message
-4	API_NOT_IMPLEMENTED_ ERRORCODE

Platform Availability

- iOS
- Android

kony.keychain.retrieve

Retrieves the specified credential information from the device's keychain.

Syntax

```
kony.keychain.retrieve(
    identifier);
```

Input Parameters

Parameter	Description
identifier	A JavaScript Dictionary object containing a key-value pair that specifies the identifier of the credential information to retrieve. They only supported key is "identifier". Its value must be a string that contains the credential info's unique identifier.

Example

```
var cred = {
    "identifier": "Apple"
};
var credDict = kony.keychain.retrieve(cred);
```

```
//Use the below function to retrieve sensitive data from your device keychain

 retrieve: function() {
    var cred = {
        "identifier": "Apple"
    };
    var credDetails = kony.keychain.retrieve(cred);
    alert("The details retreived are " + JSON.stringify(credDetails));
},
```

Return Values

If successful, a JavaScript Dictionary object containing the retrieved secure data is returned. If the retrieval process fails, an appropriate error code is returned. If the data does not exist in the devuce, an empty dictionary is returned.

Android-specific Error/Success Codes

Error/Success Code	Error/Success Message
0	SUCESSCODE
-50	INVALID_PARAM_ ERRORCODE
-25311	SIZE_NOT_ALLOWED_ ERRORCODE
-25300	ITEM_NOT_FOUND_ ERRORCODE

Error/Success Code	Error/Success Message
-36	IO_ERRORCODE
-4	API_NOT_IMPLEMENTED_ ERRORCODE

Platform Availability

- iOS
- Android

kony.keychain.save

Saves credential information in the device's keychain.

Syntax

```
kony.keychain.save(
    credentialInfo);
```

Input Parameters

credentialInfo

A JavaScript Dictionary object that contains a key-value pair that specifies the identifier of the credential information to save. They following keys are supported.

Key	Value
identifier	A mandatory key that holds a string that uniquely identifies the credential information.

Key	Value
secureaccount	An iOS-specific mandatory key that contains a string that specifies the account information to store in the keychain.
secureAccessControl	An iOS-specific optional key that contains information about how a keychain item can be used.
securedata	A mandatory key which stores a string that contains the secure data to store in the keychain.

Example 1 (for iOS)

```
var cred = {
    "securedata": "Appleseed",
    "secureaccount": "John",
    "identifier": "Apple",
    ”accessibility”: constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED,
    ”secureAccessControl”: constants.KONY_KEYCHAIN_ACCESS_CONTROL_USER_
PRESENCE
};
kony.keychain.save(cred);
```

```
//Use the below function to save sensitive data  to your device keychain
  save: function() {
    var cred = {
        "securedata": JSON.stringify(this.view.tbxData.text),
        "secureaccount": "John",
        "identifier": "Apple",
    };
    kony.keychain.save(cred);
    alert("The details are successfully stored");
},
```

Here, **identifier** and **securedata** are **mandatory** parameters.

secureaccount is a **mandatory iOS-specific** key, whereas **accessibility** and **secureAccessControl** are **optional iOS-specific** parameters.

The **accessibility** parameter defines how you can access a keychain item. The constant values for the **accessibility** key are as follows:

- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED : The data in the keychain item can be accessed when a device is unlocked by the user.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_UNLOCKED_THIS_DEVICE_ONLY: The data in the keychain item can be accessed only when a specific device is unlocked by the user.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_ALWAYS_THIS_DEVICE_ONLY: The data in the keychain item can always be accessed regardless of whether a specific device is locked.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_WHEN_PASSCODE_SET_THIS_DEVICE_ONLY: The data in the keychain can only be accessed when the device is unlocked. This is only available if a passcode is set on the device.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_ALWAYS: The data in the keychain item can always be accessed regardless of whether a device is locked.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_AFTER_FIRST_UNLOCK: The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.
- constants.KONY_KEYCHAIN_ITEM_ACCESSIBLE_AFTER_FIRST_UNLOCK_THIS_DEVICE_ONLY: The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.

The **secureAccessControl** parameter contains information about how a keychain item can be used. The constant values for the **secureAccessControl** key are as follows:

- constants.KONY_KEYCHAIN_ACCESS_CONTROL_USER_PRESENCE. The Touch ID feature does not have to be available or enrolled to use this constant. The keychain item is still accessible by Touch ID even if fingerprint authentication for any finger(s) is added or removed.
- constants.KONY_KEYCHAIN_ACCESS_CONTROL_TOUCHID_ANY: Constraint: The Touch ID feature must be available and at least one finger must be enrolled, otherwise kony.keychain.save throws

an exception. The keychain item is still accessible by Touch ID even if fingerprint authentication for any finger(s) is added or removed.

- constants.KONY_KEYCHAIN_ACCESS_CONTROL_TOUCHID_CURRENT_SET: Constraint: The Touch ID feature must be available and at least one finger must be enrolled, otherwise kony.keychain.save throws an exception. When fingerprint authentication for any finger(s) is added or removed, the keychain item becomes invalid.
- constants.KONY_KEYCHAIN_ACCESS_CONTROL_DEVICE_PASSCODE: Constraint: The Device passcode feature must be available and created by the user.
- constants.KONY_KEYCHAIN_ACCESS_CONTROL_OR: Constraint logic operation: When using more than one constraint, at least one of the constraints must be satisfied.
- constants.KONY_KEYCHAIN_ACCESS_CONTROL_AND: Constraint logic operation: When using more than one constraint, all constraints must be satisfied.
- constants.KONY_KEYCHAIN_ACCESS_CONTROL_APPLICATION_PASSWORD: The application-provided password to generate the data encryption key. This is not a constraint, but an additional item encryption mechanism.

Example 2 (for Android)

```
var cred = {
    "securedata": "Appleseed",
    "identifier": "Apple"
};
kony.keychain.save(cred);
```

Here, **identifier** and **securedata** are **mandatory** parameters. The 'securedata' string must be less than or equal to 245 characters. Also, items saved in one Android app cannot be retrieved from another app.

Return Values

Returns an error dictionary containing an error number and error message.

Android-specific Error/Success Codes

Error/Success Code	Error/Success Message
0	SUCESSCODE
-50	INVALID_PARAM_ ERRORCODE
-25311	SIZE_NOT_ALLOWED_ ERRORCODE
-25300	ITEM_NOT_FOUND_ ERRORCODE
-36	IO_ERRORCODE
-4	API_NOT_IMPLEMENTED_ ERRORCODE

Platform Availability

- iOS
- Android

30.1.2 Usage

To store the required credential information in the device's keychain, you must call the kony.keychain.save function. Your app can access the saved credential information by calling the kony.keychain.retrieve function. If you want to remove any information that is no longer required, you need to call the kony.keychain.remove function.

31. Language API

In JavaScript, exceptions that occur during runtime are usually handled by using try/catch blocks. The Language API is implemented for exceptions that are not handled using try/catch blocks.

The Language API uses `kony.lang Namespace` and the following API elements.

Function	Description
`kony.lang.getUncaughtExceptionHandler`	Retrieves the reference to a JavaScript function that is currently set as an exception handler for all uncaught exceptions.
`kony.lang.setUncaughtExceptionHandler`	Sets a JavaScript function as an exception handler for all uncaught exceptions.

Set an exception handler for all the uncaught exceptions using the `kony.lang.setUncaughtExceptionHandler` function. You can get the reference to a JavaScript function that is currently set as an expection handler for all uncaught exceptions using the `kony.lang.getUncaughtExceptionHandler` function.

31.1 kony.lang Namespace

31.1.1 Functions

The kony.lang namespace provides the following functions.

kony.lang.getUncaughtExceptionHandler

The kony.lang.getUncaughtExceptionHandler API retrieves the reference to a JavaScript function that is currently set as an exception handler for all uncaught exceptions.

Syntax

```
kony.lang.getUncaughtExceptionHandler()
```

Example

```
var funtionObject = kony.lang.getUncaughtExceptionHandler();
```

Return Values

JavaScript value containing the function reference.

Platform Availability

Available in iOS and Android platforms.

kony.lang.setUncaughtExceptionHandler

The kony.lang.setUncaughtExceptionHandler API sets a JavaScript function as an exception handler for all uncaught exceptions. The JavaScript function takes one argument that is the JavaScript exception object. A developer can query the properties of this object like any other JavaScript object.

Note: In Android platform ,the exceptionObject contains only **stack** and **message** properties.

Syntax

```
kony.lang.setUncaughtExceptionHandler(JavaScript Function Object)
```

Input Parameters

Parameter	Description
JavaScript Function object [Function] - Mandatory	Call back function that is called when an uncaught exception is raised by JavaScript engine.

Following is the signature of the function object:

```
Function <FuncName> ( exceptionObject)
<handler code>
End
```

Example

```
function uncaughtExceptionHandler(exceptionObject) {
    // Converting exception object into a readable string
    var exceptionString = "";

    if ("sourceURL" in exceptionObject) {
        exceptionString += exceptionObject.sourceURL;
    }
    if ("line" in exceptionObject) {
        exceptionString += " line # " + exceptionObject.line;
    }
    if ("message" in exceptionObject) {
        exceptionString += " : " + exceptionObject.message;
    }

    //Logging the exception string to console
    kony.print("Unhandled Exception:" + exceptionString);
}

kony.lang.setUncaughtExceptionHandler(uncaughtExceptionHandler);
```

Return Values

None

Platform Availability

Available in iOS and Android platforms.

32. Live Tiles API

Live Tiles enable you to represent an application as a tile on the Home Screen of your device. You can launch the application using a Live Tile. Secondary Live tiles are secondary tiles for the same application and can be used for deeplinking to forms.

Using the Live Tiles API, you can set primary and secondary tiles for an application. The Live Tiles API uses kony.application Namespace and the following API elements.

Live Tiles enable you to represent an application as a tile on the home screen of your device. You can launch an application using a Live Tile. Besides the primary live tiles, you can also create a secondary live tile to pin specific content of an app and deep-link to specific page of the app.

Using the Live Tiles API, you can configure the primary and secondary tiles for an application. The Live Tiles API uses `kony.application Namespace` and the following API elements.

Function	Description
`kony.application.setAppTile`	Sets the data for an application tile. If the user chooses to pin the application tile, the data set is visible.
`kony.application.setSecondaryTile`	Enables you to create or update data for a secondary tile for an application.
`kony.application.removeSecondaryTile`	Enables you to remove and unpin a specified secondary tile which was created earlier.

Set the data for a primary app tile by using the `kony.application.setAppTile` function. To pin a frequently used area of an application, create a secondary app tile by using the `kony.application.setSecondaryTile` function. If you want to remove and unpin a secondary application tile, use the `kony.application.removeSecondaryTile` function.

32.1 Functions

The Live Tiles API contains the following functions, which are part of the kony.application Namespace.

kony.application.setAppTile

This API enables you to set the data for an application tile. If the user chooses to pin the application tile, the data set is visible. For more information on pinning a tile, refer http://www.microsoft.com/windowsphone/en-us/howto/wp7/start/move-or-delete-tile-on-start.aspx.

Syntax

```
//Mango
kony.application.seAppTitle(frontTileData, backTileData)

Windows 8
kony.application.setAppTitle(tileTemplateType, tileTemplateData)

Windows 8
setapptile(tileTemplateType, tileTemplateData)
```

Input Parameters for Mango

frontTileData [Object] - Mandatory, If you do not set the front tile data, a default image and values will be used

Specifies the data to be displayed at the front of a tile. This hash table has the following inputs:

key	Description
title*[String]*	Specifies the title to be displayed at the front of a tile.The title appears at the bottom of the tile as white text.
backgroundImage*[String]*	Specifies the background image to be displayed at the front of a tile. ***Note:*** The backgroundImage will use the specified image to fill the entire space of the tile, regardless of its actual size.
count*[integer]*	specifies a number to be displayed at the top - right corner of the tile. The count value is displayed in a small black circle in white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

backTileData[Object] - Optional

Specifies the data to be displayed at the back of a tile. This table has the following key - value pairs:

key	Description
title*[String]*	specifies the title to be displayed at the back of a tile.The title value appears at the bottom of the tile as white text.
backBackgroundImage*[String]*	Specifies the background image to be displayed at the back of a tile.
backcontent*[String]*	specifies the content to be displayed at the back of a tile. The back content appears at the top of the tile as white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

You can also set the backTileData to be nil. In this case, no information will be set on the back of the tile and the tile will not flip.

Input Parameters for Windows 8

Parameter	Description
tileTemplateType [String] - Mandatory	tileTemplateType is a string describing which tile template to use. Refer the tile template catalog for the list of supported tile templates
tileTemplateData [Array] - Optional	tileTemplateData is an array of data for the tile, according to the tileTemplateType being used. The tile template catalog. contains details of the data required for each tile template.

Example

Example1- Mango

```
frontTileData = ["Front Tile Title", "option1.png", 20];
backTileData = ["Front Tile Title", "calbtn.png", "This is the back tile
content"];
kony.application.setAppTile(frontTileData, backTileData);
```

Example2 - Windows 8

```
kony.application.setAppTile("TileSquareBlock", ["Hello", "World!"] );
```

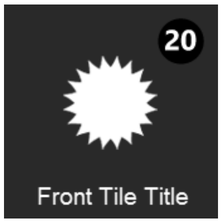
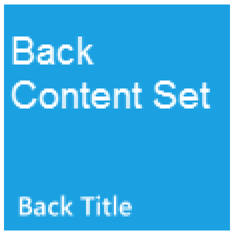
Return Values

None

UI Behavior - Mango

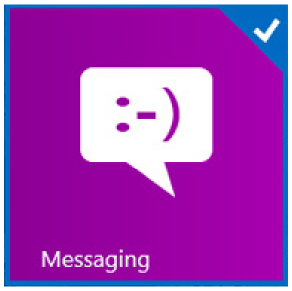
The Titles of both front and back tiles are displayed in white and this behavior cannot be changed even if the user sets a different theme or a background image. The count value for the front tile is displayed in a small black circle in white text at the top-right corner of the tile. The back content appears at the top-left corner of the tile as white text and is left justified.

Considering we have set values for both the frontTileData and the backTileData, the tiles might appear as follows:

Front Tile	Back Tile
	

UI Behavior - Windows 8

The title and the display name of the tile will be displayed in white or black depending on the setting in Kony Visualizer and this behavior cannot be changed even if the user sets a different theme or a background image. Unlike Mango. tiles cannot be flipped on Windows 8. The tile can be updated dynamically in Windows 8.



Platform Availability

Available on Windows Mango, Windows Phone 8, and Windows 8.

kony.application.setSecondaryTile

This function enables you to create or update data for a secondary tile for an application. For more information about secondary tiles, refer Secondary tiles.

Syntax

```
Mango
kony.application.setSecondaryTile(id, frontTileData, backTileData)

Windows 8
kony.application.setSecondaryTile(id, shortname, displayname, imagename)

Windows 8
setSecondaryTile(id, shortname, displayname, imagename)
```

Input Parameters for Mango

id [String] - Mandatory

Unique identifier of the secondary tile.

frontTileData [Object] - Mandatory, If you do not set the front tile data, a default image and values will be used

Specifies the data to be displayed at the front of a tile. This hash table has the following inputs:

key	Description
title*[String]*	Specifies the title to be displayed at the front of a tile.The title appears at the bottom of the tile as white text.
backgroundImage*[String]*	Specifies the background image to be displayed at the front of a tile. ***Note:*** The backgroundImage will use the specified image to fill the entire space of the tile, regardless of its actual size.
count*[integer]*	specifies a number to be displayed at the top - right corner of the tile. The count value is displayed in a small black circle in white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

backTileData[Object] - Optional

Specifies the data to be displayed at the back of a tile. This table has the following key - value pairs:

key	Description
title*[String]*	specifies the title to be displayed at the back of a tile.The title value appears at the bottom of the tile as white text.
backBackgroundImage*[String]*	Specifies the background image to be displayed at the back of a tile.
backcontent*[String]*	specifies the content to be displayed at the back of a tile. The back content appears at the top of the tile as white text. ***Note:*** The colors, sizes, and layout of the count and the title do not ever change.

You can also set the backTileData to be nil. In this case, no information will be set on the back of the tile and the tile will not flip.

Input Parameters for Windows 8

Parameter	Description
id [String] - Mandatory	Unique identifier of the secondary tile.
shortname [String] - Mandatory	A short name displayed directly on the tile. Anything over 40 characters will be truncated. The user has the option to change this value as part of the pinning process.
displayname[String] - Optional	The text specified here is displayed when you hover over the tile. There is no restriction on the display name length or the characters that it can contain.
imagename [String] - Optional	Name of the image to be displayed on the tile.

Example

The usage of this API on **Mango** is shown in the code snippet below:

```
//Setting the front and back tile data
frontTileData: {
    "Front Tile Title", "test.png", 20
}
backTileData: {
    "Front Tile Title", "test.png", "This is the back tile content"
}
//Set the secondary tile with the front and back tile data witht ile id 1234
kony.application.setSecondaryTile("1234", frontTileData, backTileData);
```

The usage of this API on **Windows 8** is shown in the code snippet below:

```
//Setting a secondary tile for an application with tile id myTile1.
kony.application.setSecondaryTile("myTile1", "title text", "display name",
"orange.png");
```

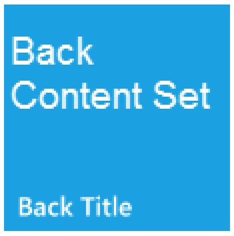
Return Values

None

UI Behavior Mango

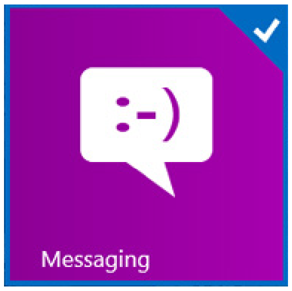
The Titles of both front and back tiles are displayed in white and this behavior cannot be changed even if the user sets a different theme or a background image. The count value for the front tile is displayed in a small black circle in white text at the top-right corner of the tile. The back content appears at the top-left corner of the tile as white text and is left justified.

Considering we have set values for both the frontTileData and the backTileData, the tiles are rendered as follows:



UI Behavior - Windows 8

The title and the display name of the tile will be displayed in white or black depending on the setting in Kony Visualizer and this behavior cannot be changed even if the user sets a different theme or a background image. Unlike Mango. tiles cannot be flipped on Windows 8. The tile can be updated dynamically in Windows 8.



Platform Availability

Available on Windows Mango, Windows Phone 8, and Windows 8.

kony.application.removeSecondaryTile

This API enables you to remove and unpin a specified secondary tile which was created earlier.

Syntax

```
kony.application.removeSecondaryTile(id)
```

Input Parameters

Parameter	Description
id [String] - Mandatory	Unique identifier of the secondary tile

Example

```
kony.application.removeSecondaryTile("12345");
```

Return Values

None

Platform Availability

Available on Windows Mango, Windows Phone 8, and Windows 8.

---

33. Local Authentication API

The Local Authentication API enables validation of credentials in iOS and Android applications. The `kony.localAuthentication Namespace` provides APIs for various authentication modes. Currently, the namespace supports the Touch ID and Face ID modes of authentication.

Touch ID is a biometric fingerprint reader that enables users to unlock the device, authenticate various payments, and so on. Touch ID is stored securely in the device and is not accessible outside. The Touch ID feature is available in the iOS from iOS 8 onwards and in the Android from Android M onwards.

Face ID is an iOS-specific facial recognition system that acts as the successor to Touch ID. This authentication mechanism allows biometric authentication to unlock a device, make payments, and access sensitive data. Furthermore, Face ID provides detailed facial expression tracking for Animoji and other features. Face ID was initially released with the iPhone X, but this mechanism has since been updated and introduced to all the latest iPhone and iPad Pro models.

Note: For information about how to enable the Face ID feature in your Kony Visualizer application, click here. For information about how to detect whether a device supports either the Touch ID or Face ID feature, refer the getBiometryType API.

The Local Authentication API contains the `kony.localAuthentication Namespace` and the following API elements.

Function	Description
`kony.localAuthentication.authenticate`	Authenticates the user.

Function	Description
kony.localAuthentication.getBiometryType	Differentiates whether a device supports either the Touch ID or Face ID feature. This API is available from iOS 11.
kony.localAuthentication.cancelAuthentication	Cancels the current authentication process.
kony.localAuthentication.getStatusForAuthenticationMode	Returns the status of the authentication mode.

33.1 Overview

The kony.localAuthentication Namespace enables authentication in iOS and Android applications. The namespace provides APIs for various authentication modes. Currently, the kony.localAuthentication namespace supports the Touch ID and Face ID modes for authentication.

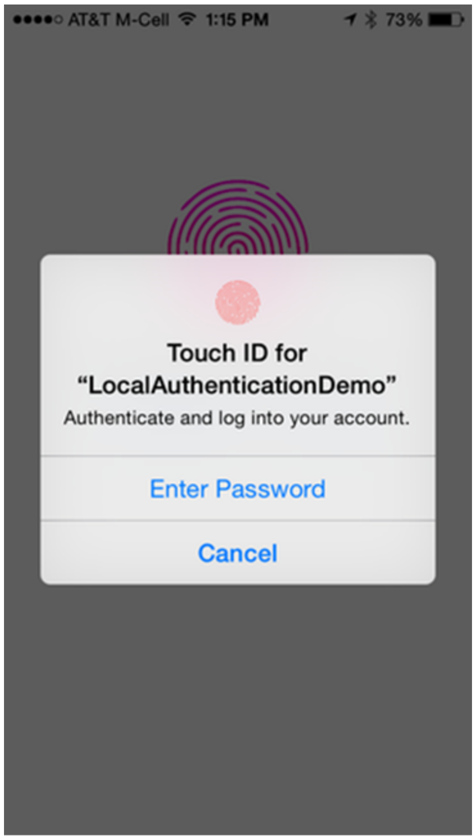
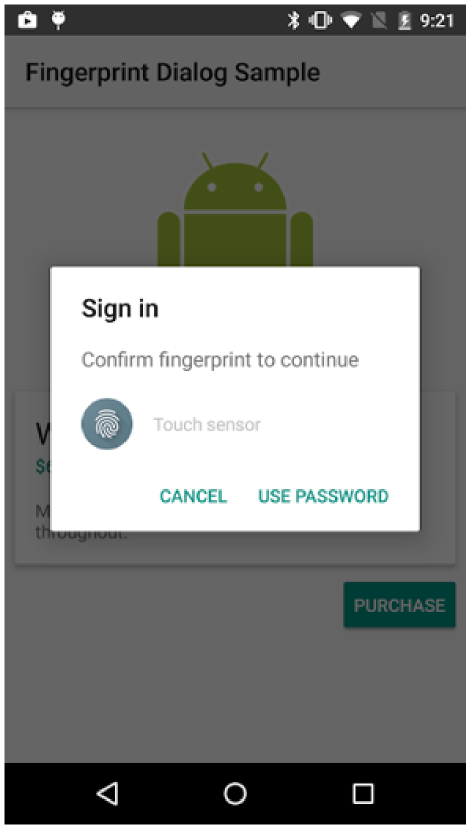
Touch ID is an authentication mechanism introduced mfor local authentication in iOS and Android devices. Touch ID is a biometric fingerprint reader that takes the place on the devices where manufactures integrated. For example, the traditional Home button on iPhone 5 and later version devices. The Touch ID feature unlocks the device, authenticates various payments, and so on. Touch ID is stored securely in the secure layer of the devices and is not accessible outside.

The Touch ID feature is available in iOS from iOS 8 onwards and in Android from Android M onwards.

Face ID is an iOS-specific facial recognition system that acts as the successor to Touch ID. This authentication mechanism allows biometric authentication to unlock a device, make payments, and access sensitive data. Furthermore, Face ID provides detailed facial expression tracking for Animoji and other features.

Face ID was initially released with the iPhone X, but this mechanism has since been updated and introduced to all the latest iPhone and iPad Pro models.

To find the mode of local authentication supported by your device, use the `kony.localAuthentication.getBiometryType` function. Use the touch ID or face ID feature and then find the state of authentication mode by using the `kony.localAuthentication.getStatusForAuthenticationMode` function. You will receive a status code indicating support for the local authentication. After you get the success status code (5000), you can now authenticate the user by using the `kony.localAuthentication.authenticate` function. If you want to cancel the authentication process, you can use the `kony.localAuthentication.cancelAuthentication` function.

iOS	Android
	

●●●●○ AT&T M-Cell 1:15 PM 73%
Touch ID for "LocalAuthenticationDemo"
Authenticate and log into your account.
Enter Password
Cancel

9:21
Fingerprint Dialog Sample
Sign in
Confirm fingerprint to continue
Touch sensor
CANCEL
USE PASSWORD
PURCHASE

In the above illustrations, a dialog box is prompted to authenticate the user to use the app. User can choose to authenticate either by fingerprint or by password. User will be authenticated by touching the Touch ID with the registered fingerprint. If the user clicks the Enter password button or the USE PASSWORD button respectively, the user can authenticate using the registered password.

In the iOS devices, the dialog box is prompted automatically by the platform itself, where as in the Android devices, the dialog box is a custom UI; you have to design and build the custom logic as per your requirement.

Users need to register fingerprint at the global settings, and the same secured data is accessed from application to authenticate. You can also set a password at the application level, which acts as an alternative method for authentication.

To use the local authentication feature in the Android platform, you need the USE_FINGERPRINT permission set to true under the Manifest Properties in the Project Settings>Native>Android tab.

> ***Note:*** The applications that use the kony.localAuthentication APIs must be in the foreground. If an application uses kony.localAuthentication APIs, the application should handle errors and respond properly with the UI to ensure an alternative method for logging on to the application.

33.2 kony.localAuthentication Namespace

The kony.localAuthentication namespace provides the functions to authenticate, get the status of the authentication mode, and cancel authentication.

> ***Note:*** For information about how to enable the Face ID feature in your Kony Visualizer application, click here. For information about how to detect whether a device supports either the Touch ID or Face ID feature, refer the getBiometryType API.

This namespace contains the following API elements.

33.2.1 Functions

The kony.localAuthentication namespace provides the following functions.

kony.localAuthentication.authenticate

The API is used to authenticate the user.

> ***Note:*** You can use this API for both the Touch ID and Face ID features. For information about how to detect whether a device supports either the Touch ID or Face ID feature, refer the getBiometryType API.

> ***Note:*** Call the `kony.localAuthentication.authenticate` API only if the `kony.localAuthentication.getStatusForAuthenticationMode` API returns the success status code (5000).

Syntax

```
kony.localAuthentication.authenticate(
    authenticationMode,
    statusCallback,
    configMap)
```

Input Parameters

Parameter	Description
authenticationMode	Specifies the authentication mode for which the status is requested. The data type is constant. For more information on authentication modes, see Authentication Modes.
statusCallBack (status, message)	A callback conveys the status of the authentication with appropriate status and message. The default value is **nil**. For status code, see the Status Codes section. ***Note:*** For Android, the `statusCallBack` parameter returns only the `5002` status code; this status code is returned when the authentication is canceled by a user.

Parameter	Description
configMap	Specifies the configuration dictionary for the specified authentication mode. The configMap parameter uses keys listed in the table below. • **promptMessage:** Message to be displayed on the screen. This is applicable only for the iOS platform. Ignored in the Android platform. • **fallbackTitle:** Allows you to edit the default text, "Enter Password" on the native pop-up, which is displayed when user authentication fails using Touch ID or Face ID. This is applicable only for the iOS platform.

Example

```
function statusCB(status, message) {
    if(status == 5000)    {
        kony.ui.Alert({
            message: "AUTHENTICATION SUCCESSFULL",
            alertType: constants.ALERT_TYPE_INFO,
            yesLabel: "Close"
        }, {});
    }
    else    {
        var messg = status + message;
        kony.ui.Alert({
            message: messg,
            alertType: constants.ALERT_TYPE_INFO,
            yesLabel: "Close"
        }, {});
    }
}

function authUsingTouchID() {
    var configMap = {
        "promptMessage": "PLEASE AUTHENTICATE USING YOUR TOUCH ID",
        "fallbackTitle": "Please enter your Password"
    };
    kony.localAuthentication.authenticate(constants.LOCAL_AUTHENTICATION_MODE_
TOUCH_ID, statusCB, configMap);
}
```

Return Values

No

Remarks

The authentication mode with the Touch ID (LOCAL_AUTHENTICATION_MODE_TOUCH_ID) requires a message to display that prompts a user to provide authentication. The key to be used is “promptMessage”. You can also pass another key-value pair, *fallbackTitle* in the configMap parameter to customize the label title of the "Enter Password" button.

Below is a sample configMap:

```
var configMap = {"promptMessage" : "PLEASE AUTHENTICATE USING YOUR TOUCH
ID",
                 "fallbackTitle" : "Please enter your Password"};
```

Note: Depending on the type of authentication available, the promptMessage is either **PLEASE AUTHENTICATE USING YOUR TOUCH ID** or **PLEASE AUTHENTICATE USING YOUR FACE ID**.

> ***Note:*** If you assign an empty string, " " to the fallbackTitle key, the Enter Password button will be hidden. If the fallbackTitle key is not defined in the configMap parameter, the default (Enter Password) value is displayed.

Platform Availability

- iOS
- Android

kony.localAuthentication.getBiometryType

This API differentiates whether a device supports either the Touch ID or Face ID feature. The kony.localAuthentication.getBiometryType API is available from iOS 11.

Syntax

```
kony.localAuthentication.getBiometryType()
```

Example

```
function getBiometryTypeOfDevice() {
    var promptMessage = "Sign in with ";
    switch (kony.localAuthentication.getBiometryType()) {
        case constants.BIOMETRY_TYPE_NONE:
            // Handle the case if the device doesn't support any biometryType
            break;
        case constants.BIOMETRY_TYPE_TOUCHID:
            promptMessage += "TouchID";
            break;
        case constants.BIOMETRY_TYPE_FACEID:
            promptMessage += "FaceID";
            break;
        case constants.BIOMETRY_TYPE_UNDEFINED:
            // Handle the case if the device is not a iOS11 device or later
            break;
    }
}
```

Return Values

Return Value	Description
constants.BIOMETRY_TYPE_NONE	If there is no biometric authentication in the device.
constants.BIOMETRY_TYPE_TOUCHID	If the device supports Touch ID authentication.
constants.BIOMETRY_TYPE_FACEID	If the device supports Face ID authentication.
constants.BIOMETRY_TYPE_ UNDEFINED	If this API is called on the device with OS earlier than iOS11.

Remarks

Face ID is the new biometric authentication that Apple has introduced with iPhoneX. This API will help to customize the prompt message in kony.localAuthentication.authenticate. Depending on the type of authentication available, the prompt message is **Sign in with FaceID** or **Sign in with TouchID**.

Platform Availability

- iOS

kony.localAuthentication.cancelAuthentication

The API cancels the current authentication process.

Syntax

```
kony.localAuthentication.cancelAuthentication()
```

Example

```
var cancelButton = kony.ui.Button({
    onClick: btnOnClick
});
function btnOnClick() {
    kony.localAuthentication.cancelAuthentication()
}
```

Return Values

Return Value	Description
status	The 5002 status code is returned indicating the authentication is canceled.

Remarks

The API is available only for the Android platform. You can use this API for the Cancel button placed in the authentication dialog box. When user taps the Cancel button, the API is called to cancel the authentication process.

Platform Availability

- Android

kony.localAuthentication.getStatusForAuthenticationMode

The API returns the status of the authentication mode.

> ***Note:*** You can use this API for both the Touch ID and Face ID features. For information about how to detect whether a device supports either the Touch ID or Face ID feature, refer the getBiometryType API.

Syntax

```
kony.localAuthentication.getStatusForAuthenticationMode(
    authenticationMode)
```

Input Parameters

Parameter	Description
authenticationMode	Specifies the authentication mode for which the status is requested. The data type is constant. For the authentication modes, see Authentication Modes.

Example

```
function isAuthUsingTouchSupported() {
    var status = kony.localAuthentication.getStatusForAuthenticationMode
(constants.LOCAL_AUTHENTICATION_MODE_TOUCH_ID);
    if (status == 5000)    {
        kony.ui.Alert({
            message: "AUTHENTICATION BY TOUCHID SUPPORTED",
            alertType: constants.ALERT_TYPE_INFO,
            yesLabel: "Close"
        }, {});
    }
    else    {
        var msg = "TOUCHID AUTHENTICATION RETURNED THE STATUS ::" + status;
        kony.ui.Alert({
            message: status,
            alertType: constants.ALERT_TYPE_INFO,
            yesLabel: "Close"
        }, {});
    }
}
```

Return Values

Return Value	Description
status	A status code is returned indicating the support for local authentication using Touch ID or Face ID. For information about how to detect whether a device supports either the Touch ID or Face ID feature, refer the getBiometryType API.

Remarks

Using the API, you can verify whether local authentication is supported on the device.

The API also returns different status codes based on the state of the device. For status codes, see the Status Codes section.

Platform Availability

- iOS
- Android

33.2.2 Authentication Modes

The following is an authentication mode and Touch ID or Face ID is used for authentication.

- constants.LOCAL_AUTHENTICATION_MODE_TOUCH_ID

Note: You can use the same constant for the Face ID feature. For information about how to detect whether a device supports either the Touch ID or Face ID feature, refer the getBiometryType API.

33.2.3 Status Codes

The following table provides a list of status codes and their descriptions.

Status Codes	Description
5000	No Error
5001	Authentication is not successful because a user fails to provide valid credentials.
5002	Authentication is canceled by a user. For example, a user taps Cancel in the dialog box.
5003	Authentication is canceled because a user taps the fallback button (Enter Password). This is applicable only for the iOS platform.

Status Codes	Description
5004	Authentication is canceled by system. This is applicable only for the iOS platform.
5005	Authentication does not start because the passcode is not set on the device.
5006	Authentication does not start because Touch ID or Face ID is not available on the device.
5007	Authentication does not start because Touch ID has no enrolled fingerprints or Face ID has no enrolled face recognition.
5008	Authentication does not start because the target device's OS does not support local authentication with Touch ID or Face ID. This is applicable only for the iOS platform.
5009	Authentication was not successful because there were too many failed Touch ID attempts, and the Touch ID feature has now been locked. This is applicable only for the iOS platform.

34. Map API

A Map widget displays geographical locations on a map in your application. Using the Map API, you can customize maps with desired content and add more capabilities to the Map widget. The Map API uses `kony.map Namespace` and the following API elements.

Function	Description
`kony.map.containsLocation`	Tests to see whether a specified location is within a circle or polygon on a map or whether it lies along a polyline on a map.
`kony.map.distanceBetween`	Finds the linear distance between two locations on a map.
`kony.map.decode`	Enables apps to to decode the encoded polyline points which are provided in search route results.

Function	Description
`kony.map.searchRoutes`	Searches for routes between the start and destination locations.

To search for all possible routes between any two points on the map, use the `kony.map.searchRoutes` function. Find the latitude and longitude values of each point in the selected search route by using the `kony.map.decode` function. Using this information, you can calculate the distance between the two locations with the `kony.map.distanceBetween` function. You can also determine if a specific location falls within a circle or along a polyline on the map using the `kony.map.containsLocation` function.

34.1 Searching Routes

You can search for routes between the source and destination locations using the kony.map.searchRoutes function. Your can also use polylines to display routes on maps and find the distance between a given set of locations.

The route searching capabilities of underlying platforms varies as each platform supports a different set of request parameter configurations. The format of the search results also varies from platform to platform.

Here are the list of request parameters available for all platforms.

Parameter	Description	Android	iOS
origin [JSObject] [Mandatory] [All]	Source address or latitude/ longitude values for which you want to get the route details. If both lat/lon and address are mentioned, the lat/lon values are considered. Address is supported only in Android & iOS. Ex: searchCriteria = {origin : {lat:17.499467, lon: 78.386760, address= "9225 Bee Cave Rd, Austin, TX 78746"} };	Y	Y
destination [JSObject] [Mandatory] [All]	Destination address or latitude/ longitude values for which you want to get the route details. If both lat/lon and address are mentioned, the lat/lon values are considered. Address is supported only in Android. Ex: searchCriteria = { destination : {lat:17.499467, lon: 78.386760, address= "9225 Bee Cave Rd, Austin, TX 78746"} };	Y	Y

Parameter	Description	Android	iOS
transportMode [String] [Optional] [All]	Mode of the transport used to calculate the directions. Here are the list of supported modes by platform. • Driving - Applies to Android, iOS, and Windows • Walking - Applies to Android, iOS, and Windows • Bicycling - Applies to Android only • Transit - applies to Android only If transportMode is not specified, underlying platforms assumes the default values. Here are the default values assumed by each platform in the absense of mode key. • iOS: Any • Android: Driving • Windows: Driving If an unsupported mode is received, platforms fallback on default mode.	Y	Y

Parameter	Description	Android	iOS
directionServiceUrl [String] [Mandatory] [Android]	URL that the Android platform uses to fetch direction details from Google Direction API service. Google offers two URLs (secure/ non-secure) for the direction service. You can use any one of them as required. Google Direction Service URLs: "https://maps.googleapis.com/maps/api/directions/ json" ***Note:*** If the application is using SSL pinning (Allow Self Signed certificates - Only Bundled), the application must obtain the certificate for the URL and bundle along with the other pinned certificates. You must be cautious about the validity of the bundled certificate as Google certificates have a limited validity period. This requirement is not applicable from Visualizer V8 SP4 onwards.	Y	N

Parameter	Description	Android	iOS
waypoints [Array] [Optional] [Android, Win]	Specifies an array of waypoints. A waypoint is specified as either a latitude/ longitude coordinate or as an address. Number of legs returned in resultant routes depends on the number of waypoints specified. If no waypoint is specified, single leg is returned covering the entire route. Since this property is not supported for iOS, search results always contains single leg for each route for iOS. For example, searchCriteria = {... waypoints: [{lat:17.495106, lon:78.324235}, {lat:17.495024, lon:78.345263}] ...}; ***Note:*** Google Map restriction for Android allows up to 8 waypoints for Google Map free API and 23 waypoints for Google Map for Work in each request. Waypoints are not available for transit directions. ***Note:*** iOS core OS does not have direct support for waypoints. To support, Kony iOS platform need to make multiple MapKit API calls internally.	Y	N
alternatives [Boolean] [Optional] [All]	Boolean flag to indicate whether to search for alternative routes or not. If true, search results may contain more than one route. Searching for alternative routes may increase the response time. For example, searchCriteria = {... alternatives:true ...}	Y	Y

Parameter	Description	Android	iOS
avoid [Array] [Optional] [Android,Win]	Indicates that the calculated route(s) should avoid certain features. • tolls, highways, ferries - avoided in iOS and Windows. • tunnels, dirtroad, motorrail - avoided in Windows, but not in Android. • Indoor is avoided in Android, but not in Windows.	Y	N
departureTime [Number] [Optional] [Android, iOS]	Desired time of departure in seconds which is calculated since January 1, 1970 UTC. This property is applicable for Android when transportMode is transit. For example, searchCriteria = {... departureTime : ...} ***Note:*** Both departureTime and arrivalTime should not present search criteria for Android.	Y	Y
arrivalTime [Number] [Optional] [Android, iOS]	Desired time of arrival in seconds which is calculated since January 1, 1970 UTC. This property is applicable for Android when transportMode is transit. For example, searchCriteria = {... arrivalTime : ...} ***Note:*** Both departureTime and arrivalTime should not present search criteria for Android.	Y	Y

Parameter	Description	Android	iOS
language [String] [Optional] [Android]	Specifies the language in which you want to return results. If the language is not set, the service will attempt to use the native language of the domain from which the request is sent. A list of latest supported language codes can be found here. For example, searchCriteria = {… language = “fr” …}	Y	N
region [String] [Optional] [Android]	Specify ccTLD country code to get the search results biased to the specified region. For more info about region specifications refer here. For example, searchCriteria = {… region : “es” …}	Y	N

Parameter	Description	Android	iOS
transitMode [Array] [Optional] [Android]	Specifies one or more preferred modes of transit. This parameter may only be specified for transit directions, and only if the request includes an API key or a Google Maps API for Work client ID. The parameter supports the following arguments: <ul><li>bus: indicates that the calculated route should prefer travel by bus.</li><li>Subway: indicates that the calculated route should prefer travel by subway.</li><li>train: indicates that the calculated route should prefer travel by train.</li><li>tram: indicates that the calculated route should prefer travel by tram and light rail.</li><li>rail: indicates that the calculated route should prefer travel by train, tram, light rail, and subway. This is equivalent to transit_ mode=train|tram|subway.</li></ul>Ex : searchCriteria = {… transitMode : [“bus”,”tram”] …};	Y	N

Parameter	Description	Android	iOS
transitRoutingPreference [Array] [Optional] [Android]	Specifies preferences for transit routes. This parameter may only be specified for transit directions, and only if the request includes an API key or a Google Maps API for Work client ID. The parameter supports the following arguments: • less_walking: indicates that the calculated route should prefer limited amounts of walking. • fewer_transfers : indicates that the calculated route should prefer a limited number of transfers. • routeOptimization [String] [Optional] [Windows] : Route optimization criteria. Here are the supported values.	Y	N
apiKey [String] [Optional] [Android]	Google Maps V2 api key that the Google Direction API uses to determine your usage quota and rate limits. If you do not have a key, you can obtain one here.	Y	N
clientID [String] [Optional] [Android]	Use clientID to access the special features of the Google Maps API for Work along with the signature property. More details about acquiring and using clientID can be found here. ***Note:*** You must mention the apiKey or [clientID + signature], otherwise, the request will fail. apiKey and clientID are mutually exclusive and using both in the same request causes the route search to fail.	Y	N

Parameter	Description	Androi d	iO S
signature [String] [Optional] [Android]	Use signature to access the special features of the Google Maps API for Work along with clientID property. More details about acquiring and using signature can be found here.	Y	N

34.2 Route Search Results

Route search results are delivered through the successCallback callback function of the searchRoute function and provided in JSON format. The *routes* parameter is a JSON array that contains one or more routes depending on the search criteria parameter *alternatives* which influences the returned results to include alternative routes.

The high level structure of a typical JSON response would be similar to the following example.

```
routes = [ {
        legs : [ {
            steps: [ {
                     }
                           …. //other speps
                   ]
                 }
                 …. //other legs
               ]
            }
            ….. //additional alternative routes
         ];
```

Note: In Android, the availability of certain platform specific keys in search results depend on the search criteria. Please refer the Google Direction API documentation for detailed information on when certain keys are included in search result.

34.2.1 Routes

Search results are returned as JSON arrays with one or more routes that indicate possible directions that are available between the source and the destination locations. Each route contains the following information.

- **startLocation [JSObject]:** Starting location for the route. It may be different from the source location given in search criteria.
- **endLocation [JSObject]:** Ending location for the route. It may be different from the destination location given in search criteria.
- **distance [Number]:** Total distance of the route in meters.
- **duration [Number]:** Total duration of the route in seconds
- **legs [Array]:** An array of legs which constitute the route. A separate leg will be present for each waypoint or destination specified. If the search criteria does not contain any waypoints, only one leg will be returned. iOS platform does not support waypoints, so it always returns single leg covering the total route.

  Each leg contains a startLocation, endLocation, distance, duration, and steps. An array of steps constitute the leg. A step is the most atomic unit of a direction's route, containing a single step describing a specific, single instruction on the journey.

  To know more about a step, refer here.

  Apart from the common properties, platforms can also provide additional information in each route.

  Android :

    - **durationInTraffic [Number]**Indicates the total duration of this leg, taking into account current traffic conditions. Availability of this key is based on certain condition. For more information, refer here.
    - **arrivalTime [String]**: Contains the estimated time of arrival for this leg.
    - **departureTime [String]** : Contains the estimated time of departure for this leg.
    - **startAddress [String]** : Contains the human-readable address (typically a street address) reflecting the start_location of this leg.
    - **endAddress [String]** : Contains the human-readable address (typically a street address) reflecting the end_location of this leg.

  Windows:

    None.

- **polylinePoints [Array]:** An Array of Lat/Lon values that can be used to draw path that follows the road closely. Developers can use addPolyline() api to draw the path.

34.2.2 Steps

Each step contains the following information

- **startLocation [JSObject]**: Starting location for the step.
- **endLocation [JSObject]**: Ending location for the step.
- **distance [Number]** : Total distance of the step in meters.
- **instruction [String]**: Instruction for step to present turn-by-turn direction hint
- **transportMode [String]** : Mode of the transport.

Apart from common properties, platforms can also provide additional information in each step.

Android:

- **duration [Number]**: Total duration of the step in seconds
- **encodedPolylinePoints[String]** : Contains an encoded polyline representation of the step. This polyline is an approximate (smoothed) path of the step. More details about encoded polyline can be found here. Developers need to decode the encoded polyline using kony.map.decode() utility function to get the series of lat/lon values which can be used to draw a smooth polyline using addPolyline() function.

iOS:

- **notice [String]** : Additional localized legal or warning notice related to this step (e.g. "Do not cross tracks when lights flash")

34.2.3 Platform Specific properties

Apart from common properties, platforms can also provide additional information in each route.

34.2.3.1 iOS

- **name [String]:** Localized description of the route's significant feature. For example, "US-101"
- **advisoryNotices [NSArray]:** Localized notices of route conditions as NSStrings. For example, "Avoid during winter storms"

34.2.3.2 Android

- **copyrights [String]:** Contains the copyrights text to be displayed for this route. You must handle and display this information yourself.
- **warnings [Array]:** Contains an array of warnings to be displayed when showing these directions. You must handle and display these warnings yourself.

Windows :

- **HasBlockedRoads [bool]:** Gets a value that indicates the route has been modified from the "best" route to avoid blocked roads.
- **IsTrafficBased [bool]:** Gets a value indicating whether the MapRoute is based on traffic.

34.2.4 Limitations

34.2.4.1 Android

Applications which use the apiKey parameter in their search criteria must enable the Directions API in the Google Developer Console. The usage quota for the Google Directions API is counted against the apiKey.

Google's usage limits are as follows:

- Free: 2,500 directions requests per 24 hour period, Up to 8 waypoints allowed in each request and 2 requests per second. Waypoints are not available for transit directions.
- Paid: 100,000 directions requests per 24 hour period.

34.2.4.2 iOS

iOS does not support directions in all countries. For a list of countries where directions are available, please refer here.

To view the functionality of the Map API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

34.3 kony.map Namespace

The kony.map namespace provides support functions for use with the Map widget. This section contains reference information about the kony.map namespace in the following topics.

- Constants
- Functions

34.3.1 Constants

The kony.map namespace defines the following constants.

Map Provider Constants

The Map Provider Constants enable your app to select which map provider to use.

Constant	Description
kony.map.MAP_PROVIDER_BING	Select Bing as the map provider. This constant is available on all platforms except SPA/Desktop Web.
kony.map.MAP_PROVIDER_GOOGLE	Select Google as the map provider.

Map View Mode Constants

Use the Map View Mode Constants to configure which map view your app selects.

Constant	Description
kony.map.MAP_VIEW_MODE_NORMAL	View the map in whatever mode is the default for the map provider.
kony.map.MAP_VIEW_MODE_SATELLITE	View the map as a satellite image.
kony.map.MAP_VIEW_MODE_STREET	View the map as a street map.
kony.map.MAP_VIEW_MODE_TRAFFIC	View traffic information on the map.

Map Widget Error Codes

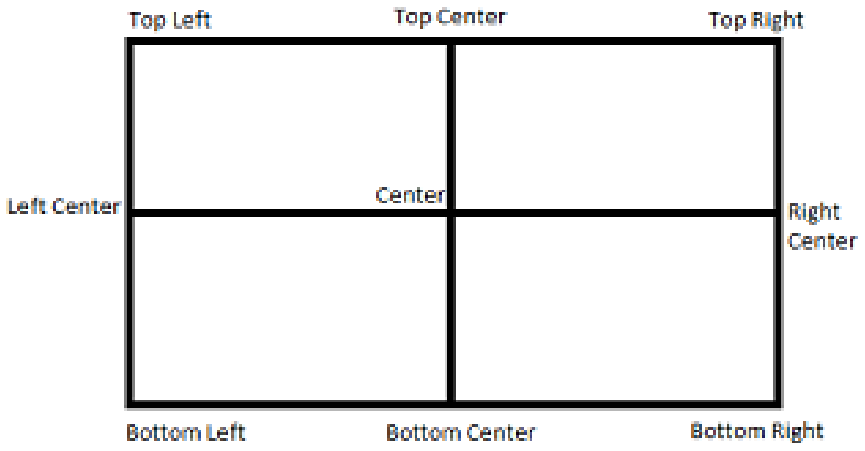
The following table lists the error codes that the Map widget generates.

Constant	Description
kony.map.ROUTE_SEARCH_INVALID_REQUEST	The format of the route search request was invalid.
kony.map.ROUTE_SEARCH_LIMIT_EXCEEDED	The service has received too many requests from your application within the allowed time period. For Android, below are the usage limits imposed by Google Map Service. • Up to 8 waypoints for Google Map free API and 23 waypoints for Google Map for Work in each request • 2500 & 100000 direction requests per 24 hour period for free API and work api respectively. • 2 and 10 requests per second for free API and work API respectively.
kony.map.ROUTE_SEARCH_NETWORK_FAILURE	The request failed due to network failure.

Constant	Description
kony.map.ROUTE_SEARCH_PLACE_NOT_FOUND	At least one of the locations specified in the request's source, destination, or waypoints could not be found.
kony.map.ROUTE_SEARCH_UNKNOWN_ERROR	An unknown error occurred.

Pin Image Anchor Constants

The Pin Image Anchor Constants define the positions that your app can anchor a pin image to on a map. The image positions are illustrated in the image below.



Constant	Description
kony.map.PIN_IMG_ANCHOR_BOTTOM_CENTER	Anchors the pin image at the bottom center position.
kony.map.PIN_IMG_ANCHOR_BOTTOM_LEFT	Anchors the pin image by its lower left corner.

Constant	Description
kony.map.PIN_IMG_ANCHOR_BOTTOM_RIGHT	Anchors the pin image by its lower right corner.
kony.map.PIN_IMG_ANCHOR_CENTER	Anchors the pin image at the center position.
kony.map.PIN_IMG_ANCHOR_MIDDLE_LEFT	Anchors the pin image at the middle left position of the image.
kony.map.PIN_IMG_ANCHOR_MIDDLE_RIGHT	Anchors the pin image at the middle right position of the image.
kony.map.PIN_IMG_ANCHOR_TOP_CENTER	Anchors the pin image at the top center position.
kony.map.PIN_IMG_ANCHOR_TOP_LEFT	Anchors the pin image by its upper left corner.
kony.map.PIN_IMG_ANCHOR_TOP_RIGHT	Anchors the pin image by its upper right corner.

Pin Image Type Constants

These constants define the types of images that can be used with maps.

Constant	Description
kony.map.PIN_IMG_SRC_TYPE_BASE64	Indicates that the pin image should be created out of given a base64 string.

Constant	Description
kony.map.PIN_IMG_SRC_TYPE_FILE_PATH	Indicates that the pin image is available in internal file system. The specified value can be either an absolute path or a File object.
kony.map.PIN_IMG_SRC_TYPE_IMAGE	Indicates that the pin image is of type Image object
kony.map.PIN_IMG_SRC_TYPE_RESOURCES	Indicates that the pin image is available in bundled resources.

Shape Type Constants

The following constants identify the shapes that can be drawn on maps.

Constant	Description
kony.map.SHAPE_TYPE_POLYGON	The shape is a polygon.
kony.map.SHAPE_TYPE_POLYLINE	The shape is a polyline.
kony.map.SHAPE_TYPE_CIRCLE	The shape is a circle.

34.3.2 Functions

The kony.map namespaces contains the following functions.

kony.map.containsLocation

This function tests to see whether a specified location is within a circle or polygon on a map or whether it lies along a polyline on a map.

Syntax

```
kony.map.containsLocation(
    shapeType,
    location,
    shapeData)
```

Input Parameters

Parameter	Description
shapeType	Contains a Shape Type Constant that defines which kind of shape the location is being tested against.
location	Holds a location object which contains lat and lon values.

Parameter	Description
shapeData	A key-value pair object that defines the shape using the following keys: • locations [Array]: List of locations that defines a given shape. Each element in Array is an Object, which contains latitude and longitude values. For Circle, only first value in Array is considered. • radius [Number]: Radius that is needed to define circle shape. This key is only valid of shapeType is Circle and ignored for other shapes. • tolerance [Number]

Examples

Example 1: Polyline

```
var shapeData = {
    locations: [{
        lat: "17.451759",
        lon: "78.380806"
    }, {
        lat: "17.473305",
        lon: "78.425191"
    }],
    tolerance: 200,
};

var location = {
    lat: "17.427789",
    lon: "78.451751"
};
var value = kony.map.containsLocation(kony.map.SHAPE_TYPE_POLYLINE, location,
shapeData);
```

```
//Defining the shapeData parameter
  var shapeData = {
    locations: [{
        lat: "17.451759",
        lon: "78.380806"
    }, {
        lat: "17.473305",
        lon: "78.425191"
    }],
    tolerance: 200,
};
//Defining the location parameter
var location = {
    lat: "17.427789",
    lon: "78.451751"
};
//Checking if the location mentioned falls on the polyline
```

```
var value = kony.map.containsLocation(kony.map.SHAPE_TYPE_POLYLINE, location,
shapeData);
```

Example 2: Circle

```
var shapeData = {
    locations: [{
        lat: “17.451759”,
        lon: “78.380806”
    }];
    radius: 1000;
};
kony.map.containsLocation(kony.map.SHAPE_TYPE_CIRCLE, location, shapeData);
```

```
//Defining the shapeData parameter
    var shapeData = {
    locations: [{
        lat: "17.451759",
        lon: "78.380806"
    }],
    radius: 1000
};
//Defining the location parameter
    var location ={
      lat: "17.451759",
      lon: "78.380806"
    };
//Checking if the location mentioned falls inside the circle
var b = kony.map.containsLocation(kony.map.SHAPE_TYPE_CIRCLE, location,
shapeData);
```

Return Values

True if the location is within the circle or polygon, or if it lies along the polyline. Otherwise, false.

Remarks

For detailed information on how to use this function and what parameter values are valid, please see Map API.

Platform Availability

Available on Android and iOS.

kony.map.distanceBetween

This function finds the linear distance between two locations on a map.

Syntax

```
kony.map.distanceBetween(
    location1,
    location2)
```

Input Parameters

Parameter	Description
location1	Contains the first location to use.
location2	Contains the second location to use.

Example

```
location1 = {
    lat: “17.451759”,
    lon: “78.380806”
};
location2 = {
    lat: “17.427789”,
    lon: ”78.451751”
};
var distanceInMeters = kony.map.distanceBetween(location1, location2);
```

```
//Defining pin 1.
var pin1 = {
    id: "id1", // id is mandatory for every pin in dictionary
    lat: "17.4947934",
    lon: "78.3996441",
    name: "KPHB",
    image: "pinb.png",
    focusImage: "focusImage.png", //focus image will be shown when map pin is
selected
    desc: "Kukatpally",
    showCallout: true,
    meta: {
        color: "green",
        label: "A"
    }
};
//Defining pin 2.
var pin2 = {
    id: "id2", // id is mandatory for every pin in dictionary
    lat: "17.3616",
    lon: "78.4747",
    name: "Charminar",
    image: "pinb.png",
    focusImage: "focusImage.png",
    //focus image will be shown when map pin is selected
    desc: "In Hyderabad",
    showCallout: true,
    meta: {
        color: "green",
        label: "B"
    }
};

//Adding pins.
this.view.MainMap.addPins([pin1, pin2]);
//Calculating the distance between the two pins.
var distanceInMeters = kony.map.distanceBetween(pin1, pin2);
```

Return Values

A number that specifies the distance between the two input locations.

Platform Availability

Available on Android and iOS.

kony.map.decode

This function enables apps to to decode the encoded polyline points which are provided in search route results. In Android, each step in the search results contains a key.

Syntax

```
kony.map.decode(
    encodedPolylinePoints)
```

Input Parameters

Parameter	Description
encodedPolylinePoints	Hold a string containing the encoded polyline points.

Return Values

An array containing only the lat/lon values.

Example

```
var polylineConfig = {
    lineColor: "0x0000ffff",
    lineWidth: "2"
};
var bool = kony.map.decode(polylineconfig);
```

Platform Availability

Available on Android only.

kony.map.searchRoutes

This function searches for routes between the start and destination locations.

Syntax

```
kony.map.searchRoutes(
    searchCriteria,
    successCallback,
    errorCallback)
```

Input Parameters

Parameter	Description
searchCriteria	A JSObject with set of search request configuration parameters that defines the search criteria for routes request.

Parameter	Description
successCallback	A callback function that receives the search results when search request succeeds. The callback function must have the following syntax. `function successCallback ( routes );` The callback function's *routes* parameter is an array with one or more routes indicating possible directions between source and destination.

Parameter	Description
errorCallback	An optional callback function that gets called when search request fails. The callback function must have the following syntax. `function errorCallback( errorCode [Number], errorMessage [String] )` The *errorCode* parameter indicates the category of error. This carries the one of the Map Error Codes defined in the kony.map namespace. The *errorMessage* parameter contains a detailed error message describes the reason for failure. These error messages are platform specific.

Example

```
//function to call the search routes
function callSearchRoutefunc() {
    var searchCriteriaObj = {
        alternatives: true,
        directionServiceUrl:
"https://maps.googleapis.com/maps/api/directions/json,"
        destination: {
            address: Madhapur,
            Hyderabad,
            Telangana,
            India,
            lon: "78.3913771",
            lat: "17.4482825"
        },
        origin: {
            address: Dilsukhnagar,
            Hyderabad,
            Telangana,
            India,
            lon: "78.52468589999999",
            lat: "17.3687747"
        },
        waypoints: [{
            address: Nampally,
            Hyderabad,
            Telangana,
            lon: "78.466502",
            lat: "17.383814"
        }],
        transportMode: "driving"
    }

    kony.map.searchRoutes(searchCriteriaObj, searchRouteSuccesCallback,
errorRouteSuccesCallback);
}

function searchRouteSuccesCallback(routes) {
    kony.print("######Succeess callback is called###");
    displySearchRoutes(routes);
}

function displySearchRoutes(Searchroutes) {
    kony.application.showLoadingScreen(
        "formskin",
        "LoadingScreen",
        constants.LOADING_SCREEN_POSITION_ONLY_CENTER,
        false,
        true, {
            enableMenuKey: true,
            enableBackKey: true,
            progressIndicatorColor: "ffffff77"
        });

    routeColors = ["0000FFFF", "FF00FFFF", "FF0000FF", "FFFF00FF",
"0x000000FF"];

    for (var i = 0; i & lt; Searchroutes.length; i++) {
        drawRoute("route" + i, Searchroutes[i].polylinePoints, routeColors[i])
    }

    kony.application.dismissLoadingScreen();
    frmMapSearchResult.show();

}

function drawRoute(routeid, polyPoints, color) {
    var steps = polyPoints;
    kony.print("###############The polyline points");
    kony.print(steps);
    ei = steps.length - 1;

    var startLoc = {
        lat: steps[0].lat,
        lon: steps[0].lon,
        image: {
            source: "pin5.png",
            anchor: kony.map.PIN_IMG_ANCHOR_CENTER
        }
    };
    var endLoc = {
        lat: steps[ei].lat,
        lon: steps[ei].lon,
        image: {
            source: "pin6.png",
            anchor: kony.map.PIN_IMG_ANCHOR_CENTER
        }
    };

    polylineData = {
        id: routeid,
        locations: steps,
        startLocation: startLoc,
        endLocation: endLoc,
        polylineConfig: {
            lineWidth: 5,
            lineColor: color
        }
    };

    frmMapSearchResult.Map1.addPolyline(polylineData);
}
```

```
* @ function callSearchRoutefunc * @description invokes searchRoutes API * /
  callSearchRoutefunc:function()
  {
    try{
      var searchCriteriaObj = {
        alternatives : true,
        directionServiceUrl : "https:/ / maps.googleapis.com / maps / api /
directions / json ",
        destination : {lat: MAPCONSTANTS.dest3Lat, lon:
MAPCONSTANTS.dest3Lon},
        origin :  {lat: MAPCONSTANTS.originLat, lon : MAPCONSTANTS.originLon},
        transportMode : "driving ",
        apiKey:""
      };
      if (this.index === null || this.index === undefined) {
        alert('Please select any one destination');
      }
      switch (this.index) {
        case "Kony Foster City ":
          searchCriteriaObj.destination = {lat: MAPCONSTANTS.dest1Lat, lon:
MAPCONSTANTS.dest1Lon};
          break;
        case "Kony Austin ":
          searchCriteriaObj.destination = {lat: MAPCONSTANTS.dest2Lat, lon:
MAPCONSTANTS.dest2Lon};
          break;
        default :
          kony.print("@@@@
destination is Orlando ");
          break;
      }

      kony.map.searchRoutes(searchCriteriaObj, this.searchRouteSuccesCallback,
this.errorRouteSuccesCallback);
    }catch(error){
      kony.print("
frmMapSearchResult Controller "+JSON.stringify(error));
    }
  },
  /**
* @function searchRouteSuccesCallback
* @description success callback for searchRoutes API
* @private
* @param routes-&gt; routes available in the given source and destination
                        */
  searchRouteSuccesCallback:function(routes)
  {
    try{
      this.displySearchRoutes(routes);
    }catch(error){
      kony.print("
frmMapSearchResult Controller "+JSON.stringify(error));
    }
  },
  /**
* @function errorRouteSuccesCallback
* @description error callback for searchRoutes API
*/
  errorRouteSuccesCallback:function(){
    try{
      alert("
Search result failed ");
    }catch(error){
      kony.print("
frmMapSearchResult Controller "+JSON.stringify(error));
    }
  },
  /**
```

Return Values

None.

Remarks

Applications which use the apiKey in search criteria must enable the “Directions API” in Google Developer Console. Google API’s usage quota is counted against the apiKey. For activating and deactivating Google API’s, please follow the below link for detailed procedure. For an overview on searching for routes on maps, please see Map API.

Platform Availability

Available on Android and iOS.

35. Map Styling API

The Map Styling API helps provide various custom styling options for maps. By using these style options, you can customize the presentation of the standard Google map styles, thereby changing the visual display of features such as roads, businesses, parks, and other points of interest.

This feature is applicable for Android platform only and is available on Kony Visualizer from V8.2 onwards.

35.1 API Details

1. Create a custom style raw resource JSON file (https://mapstyle.withgoogle.com/).
2. Import the JSON files to Kony Visualizer under Assets (mobile/native/Android/raw).

The Map Styling API helps to provide various custom styling options for maps. By using these style options, you can customize the presentation of the standard Google map styles, thereby changing the visual display of features such as roads, businesses, parks, and other points of interest.

This feature is applicable for Android platform only and is available on Kony Visualizer from V8.2 onwards.

35.2 Create custom Map style

The Map Styling API enables you to load custom map styles in JSON format by using the LoadRawResourceStyle method. To create a custom map style, follow these steps:

1. Create a custom style raw resource JSON file here.

2. Import the JSON files to assets in Kony Visualizer (mobile/native/Android/raw).

35.3 Functions

LoadRawResourceStyle

Syntax

```
mapObject.loadRawResourceStyle(rawJsonFile,stylingCallback)
```

Input Parameters

Parameter	Description
rawJsonFile	Name of the file added in the raw folder of native Android.
stylingCallback	Callback to be invoked.

Example

```
var mapObject = new kony.ui.Map({}, {}, {});
mapObject.loadRawResourceStyle("style", stylingCallback);

function stylingCallback(booleanFlag, resource) {
    if (booleanFlag == true)

        kony.print("Styling is successful-" + resource);
    else
        kony.print("Styling failed-" + resource);
}
```

Note: This API returns true if the style is successfully parsed, and false if any problems are detected with MapStyleOptions. Such problems include unparsable styling JSON, unrecognized feature type, unrecognized element type or invalid styler keys, or the provided resource file is not found. If the return value is false, the current style is left unchanged.

Return Values

Return Value	Description
Boolean	true/false. Indicates whether the styling is successful or not.
String	Name of the resource file (JSON) passed to the API.

Clearing the Custom Styling

Provide the resource file as null to clear the applied custom styling.

```
mapObject.loadRawResourceStyle(null,stylingCallback);
```

IDE/CodeGen requirements

None.

Platform Availability

Android.

36. Math API

Using the Math API, you can provide your app with the ability to perform various mathematical operations on a given set of numbers.

The Math API uses the `math Namespace` and the following API elements.

Functions

Function	Description
`Math.max`	Returns the maximum value among the given set of numbers.
`Math.min`	Returns the minimum value among the given set of numbers.
`Math.pow`	Computes the value of the first parameter raised to the power of the second parameter
`Math.sqrt`	Returns the square root of a given number.
`Math.random`	Generates a real number between 0 and 1.
`Math.floor`	Converts a float value to an integer (number before the decimal).

Properties

Property	Description
`Math.PI`	Returns the value of pi.

Find the largest and the smallest numbers from a given set of numbers, by using the Math.max and the Math.min functions. Compute the square root of a number by using the Math.sqrt function and power of a number using the Math.pow function. To return the value of pi, use the Math.pi function. To generate a random number between 0 and 1, use the Math.random function. Further, you can convert a float value to an integer by using the Math.floor function.

To view the functionality of the Math API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

36.1 math Namespace

The math namespace contains the following API elements to help you include common mathematical tasks in your apps.

36.1.1 Functions

The math namespace provides the following functions.

math.floor

This API converts a float value to an integer. The converted integer value is always the integer part of the specified float number (number before the decimal).

Syntax

```
Math.floor()
```

Return Values

Return Value	Description
integer [Number]	The return value is an integer

Example

```
onclickfloor: function(){
  var a= this.view.tbxOperations.text;
  var b= Math.floor(a);
  alert("The converted value is " +b );
},
```

Platform Availability

Available on all platforms.

---

math.max

This API returns the maximum value among the arguments.

Syntax

```
Math.max(x,y)
```

Return Values

Return Value	Description
Maximum Value [Number]	The return value is the maximum value among all the arguments

Example

```
onclickfloor: function(){
  var a= this.view.tbxOperations.text;
  var b= Math.floor(a);
  alert("The converted value is " +b );
},
```

Platform Availability

Available on all platforms.

---

math.min

This API returns the minimum value among the arguments.

Syntax

```
Math.min(x,y,x,...n)
```

Return Values

Return Value	Description
Minimum Value [Number]	The return value is the minimum value among all the arguments

Example

```
onclickmin: function(){
   var a=this.view.tbxNumber1.text;
   var b=this.view.tbxNumber2.text;
   var c=Math.min(a,b);
    alert("The minimum value is " +c);
 },
```

Platform Availability

Available on all platforms.

math.pi

This API returns the value of pi.

Note: *math.pi* is not a function, but a property in math namespace.

Syntax

```
Math.PI
```

Input Parameters

None.

Return Values

Return Value	Description
value of pi [Number]	Value of pi is returned

Example

```
onclickpi: function(){
  var pi= Math.PI;
  alert("The pi value is " +pi );
},
```

Platform Availability

Available on all platforms.

---

math.pow

This API raises the first parameter to the power of the second parameter and returns the result.

Syntax

```
Math.pow(x,y)
```

Return Values

Return Value	Description
x^y	Raises the first parameter to the power of the second parameter

Implementation Details

It is advisable to use the expression x^y as it is much faster when compared to this API.

Example

```
onclickpow: function(){
 var a= this.view.tbxOperations.text;
 var b= Math.pow(a,2);
 alert("The Squared value is " +b );
   },
```

Platform Availability

Available on all platforms.

---

math.random

This API generates pseudo-random numbers which are uniformly distributed. This API generates a real number between 0 and 1.

Syntax

```
Math.random()
```

Return Values

Return Value	Description
pseudo-random number [Number]	A pseudo-random number between the value 0 and 1 is generated

Example

```
onclickRandom: function(){
    var random= Math.random();
    alert("The random number is " +random);
  }
```

Platform Availability

Available on all platforms.

---

math.sqrt

This API returns the square root of the given number.

Syntax

```
Math.sqrt()
```

Return Values

Return Value	Description
square root [Number]	The square root of the number is returned
nan (not a number)	This value is returned when the input parameter is a negative number

Example

```
onclicksqrt: function(){
  var a= this.view.tbxOperations.text;
  var b= Math.sqrt(a);
  alert("The square root value is " +b );
},
```

Platform Availability

Available on all platforms.

Common Example

```
var ans = Math.max(10, 20);
var sqrt_num = Math.sqrt(4);
var round_off = Math.floor(2.3);
kony.print(Math.PI);
kony.print(Math.random);
kony.print(Math.pow(2, 3));
kony.print(Math.min(10, 20));
```

37. Media API

The Media API enables your app to play and record audio files. It contains the following Namespace and objects:

- kony.media Namespace

Function	Description
`kony.media.createFromFile`	Creates a media object from a media file on the device.
`kony.media.createFromUri`	Creates a media object that plays a remote audio file across the network.
`kony.media.record`	Creates a record object that your app can use to record audio.

- media Object

Method	Description
pause	Pauses the playback of a media file.
play	Plays a media file.
releaseMedia	Releases the memory and resources held by the media object.
seek	Sets the current playback position to a specific spot in the media file.
setCallbacks	Associates callback functions with the media object.

Method	Description
stop	Stops the playback of a media file.

Property	Description
data	Holds the data object that contains the sound associated with the media object.
duration	Contains the duration of the audio in seconds.
isPlaying	Contains a Boolean value that indicates whether or not the audio is currently playing.

Property	Description
`volume`	Contains the current volume level.

- record Object

Method	Description
`startRecording`	Stops the current recording.
`stopRecording`	Starts recording audio.

Create a media object from an existing audio file using the `kony.media.createFromFile` function. To control the audio output, use the methods of media object such as `play`, `pause`, and `stop`. If you want to play a remote audio file across the network, create a media object by using the `kony.media.createFromUri` function. You can move the playback position to a desired point using the `seek` method. Using the `releaseMedia` method, you can delete the resources held by the media object and save memory. Further, to see a response based on a specific event, associate callback functions with the media object using the `setCallbacks` method.

You can configure the `duration`, `volume`, `data` properties of the audio file and find whether a specific audio is playing using the `isPlaying` property.

Further, you can record audio files using the record object. To start recording an audio file, use the `startRecording` method and to stop recording the audio file, use the `stopRecording` object.

37.1 Overview

To use the Media API, your app calls functions from the kony.media Namespace to create a media object or a record object.

Specifically, your app can call the kony.media.createFromFile function to create a `media` object from an existing audio file. Next, your app calls `media1` object methods such as `play` or `pause` to control the audio output.

Alternatively, your app can invoke the kony.media.createFromUri to create a `media` object that plays an audio file remotely across the network. As with local audio files, your app uses the `media` object methods to play the remote audio, pause it, or stop the playback.

Note: Both the media and record objects have Android-specific behaviors. For more information, please see the Overview for media and record objects.

To view the functionality of the Media API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

37.2 kony.media Namespace

The `kony.media` namespace, together with the media object and the record object, implements the functionality of the Media API.

Your app uses the functions in the `kony.media` namespace to create media and record objects. It calls the createFromFile and createFromUri functions to instantiate media objects and associate media objects with files. The files can exist either locally on the device or remotely across the network or Internet.

To record audio to a file, your app uses the kony.media.record function to create a `record` object.

37.2.1 Functions

The kony.media namespace contains the following functions.

kony.media.createFromFile

Creates a media object from a media file on the device.

Syntax

```
kony.media.createFromFile(
    fileobj, psp)
```

Input Parameters

Parameter	Description
fileobj	An object of type kony.io.File that specifies the file that the created `media` object will play.
psp [Optional]	A JSON Object that contains the key-value pairs for the AVAudioSession Category and AVAudioSession Category Options parameters. ***Note:*** The psp parameter is only applicable for the iOS platform. The default value of the **AVAudioSession Category** is `kony.media.AUDIO_SESSION_CATEGORY_PLAY_AND_RECORD`. The **AVAudioSession Category Options** is an array of category options with the default value `kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS` and `kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_SPEAKER`.

Example 1

```
var fileObj = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(fileObj);
```

Example 2

```
var fileObj = new kony.io.File("MyAudioFile.mp3");
var psp = {
    "category": kony.media.AUDIO_SESSION_CATEGORY_PLAY_AND_RECORD,
    "options": [kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_
SPEAKER,kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_BLUETOOTH_
A2DP,kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS]
     }
var mediaObj = kony.media.createFromFile(fileObj,psp);
```

Return Values

Returns a media object that is associated with a specific file on the device, or null if the object was not created.

Remarks

This function has platform-specific behaviors when there are errors. In particular, when the *fileObj* parameter refers to a file that doesn't exist on iOS, iOS generates an error with the message string "Unable to play the media file". However, if this error occurs on Android, no error message is generated. Instead, this function does not create the `media` object.

Parameter Details

The *psp* parameter is an object that contains key-value pairs for the AVAudioSession Category and AVAudioSession Category Options parameters.

The structure of the psp parameter is as follows:

```
psp Structure:{
    "category" : <category_name>
    "options"  : [categoryOptions]
 }
```

> ***Note:*** If the value of the options parameter is not an array, the default value (`kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS` and `kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_SPEAKER`) is assigned to the options parameter.

The ***AVAudioSession Category*** parameter supports the following values.

Constant	Description
kony.media.AUDIO_SESSION_ CATEGORY_AMBIENT	This category indicates that the sound playback for the app is non-primary. This means that the app will work even with the sound turned-off. When you use this category, audio from other apps gets mixed with the current audio. The audio is silenced when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).
kony.media.AUDIO_SESSION_ CATEGORY_SOLO_AMBIENT	This category indicates that the audio from the app is non-mix-able. This means that when you activate an audio session, it interrupts all the other audio sessions that are non-mix-able. If you want to allow audio mixing, you must use the kony.media.AUDIO_SESSION_CATEGORY_ AMBIENT category. The audio is silenced when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).

Constant	Description
kony.media.AUDIO_SESSION_ CATEGORY_PLAYBACK	Use this category for playing recorded music or other sounds that are central to the successful use of your app. This category indicates that the audio from the app is non-mix-able. This means that when you activate an audio session, it interrupts all the other audio sessions that are non-mix-able. If you want to allow audio mixing, you must use the kony.media.AUDIO_SESSION_CATEGORY_ OPTIONS_MIX_WITH_OTHERS category. When you use this category, the audio of the app continues to play even when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).
kony.media.AUDIO_SESSION_ CATEGORY_PLAY_AND_ RECORD	Use this category in scenarios where recording (input) and playback (output) of audio must be performed simultaneously. However, you can also use this category in apps that record and then play back the audio. This category indicates that the audio from the app is non-mix-able. This means that when you activate an audio session, it interrupts all the other audio sessions that are non-mix-able. If you want to allow audio mixing, you must use the kony.media.AUDIO_SESSION_CATEGORY_ OPTIONS_MIX_WITH_OTHERS category. When you use this category, the audio of the app continues to play or get recorded even when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).
kony.media.AUDIO_SESSION_ CATEGORY_MULTIROUTE	Use this category to route distinct streams of audio data to different output devices at the same time.

The ***AVAudioSession Category Options*** parameter supports the following values.

Constant	Description
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS	An option that indicates whether the audio from the current session mixes with the audio from active sessions in other audio apps.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DUCK_OTHERS	An option that reduces the volume of other audio sessions while audio from the current session plays.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_BLUETOOTH	An option that determines whether Bluetooth hands-free devices can appear as available input routes.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_SPEAKER	An option that determines whether audio from the current session must use the default built-in speaker instead of the receiver.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_INTERRUPT_SPOKEN_AUDIO_AND_MIX_WITH_OTHERS	An option that determines whether spoken audio content from other sessions must be paused when the app plays its audio.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_BLUETOOTH_A2DP	An option that determines whether you can stream audio from the current session to Bluetooth devices that support the Advanced Audio Distribution Profile (A2DP).
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_AIR_PLAY	An option that determines whether you can stream audio from the current session to AirPlay devices.

Constant	Description
kony.media.AUDIO_SESSION_ CATEGORY_OPTIONS_ OVERRIDE_MUTED_ MICROPHONE_ INTERRUPTION	An option that indicates whether the system interrupts the audio session when the built-in microphone is muted.

Platform Availability

Windows10, Android, iOS

Platform Availability

Windows10, Android, iOS

kony.media.createFromUri

Creates a media object that plays a remote audio file across the network.

Syntax

```
kony.media.createFromUri(
    uriString, psp)
```

Input Parameters

Parameter	Description
uriString	A string containing the URI of the remote audio file.

Parameter	Description
psp [Optional]	A JSON Object that contains the key-value pairs for the AVAudioSession Category and AVAudioSession Category Options parameters. *Note:* The psp parameter is only applicable for the iOS platform. The default value of the **AVAudioSession Category** is `kony.media.AUDIO_SESSION_CATEGORY_PLAY_AND_RECORD`. The **AVAudioSession Category Options** is an array of category options with the default value `kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS` and `kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_SPEAKER`.

Example 1

```
var mediaObj = kony.media.createFromUri(url);
```

Example 2

```

var psp = {
    "category": kony.media.AUDIO_SESSION_CATEGORY_PLAY_AND_RECORD,
    "options": [kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_
SPEAKER,kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_BLUETOOTH_
A2DP,kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS]
     }
var mediaObj = kony.media.createFromUri(url,psp);
```

Return Values

Returns a media object that is associated with a remote audio file, or null if the object was not created.

Remarks

Parameter Details

The *psp* parameter is an object that contains key-value pairs for the AVAudioSession Category and AVAudioSession Category Options parameters.

The structure of the psp parameter is as follows:

```
psp Structure:{
    "category" : <category_name>
    "options"  : [categoryOptions]
}
```

> ***Note:*** If the value of the options parameter is not an array, the default value (`kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS` and `kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_SPEAKER`) is assigned to the options parameter.

The ***AVAudioSession Category*** parameter supports the following values.

Constant	Description
kony.media.AUDIO_SESSION_ CATEGORY_AMBIENT	This category indicates that the sound playback for the app is non-primary. This means that the app will work even with the sound turned-off. When you use this category, audio from other apps gets mixed with the current audio. The audio is silenced when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).

Constant	Description
kony.media.AUDIO_SESSION_CATEGORY_SOLO_AMBIENT	This category indicates that the audio from the app is non-mix-able. This means that when you activate an audio session, it interrupts all the other audio sessions that are non-mix-able. If you want to allow audio mixing, you must use the kony.media.AUDIO_SESSION_CATEGORY_AMBIENT category. The audio is silenced when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).
kony.media.AUDIO_SESSION_CATEGORY_PLAYBACK	Use this category for playing recorded music or other sounds that are central to the successful use of your app. This category indicates that the audio from the app is non-mix-able. This means that when you activate an audio session, it interrupts all the other audio sessions that are non-mix-able. If you want to allow audio mixing, you must use the kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_MIX_WITH_OTHERS category. When you use this category, the audio of the app continues to play even when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).

Constant	Description
kony.media.AUDIO_SESSION_ CATEGORY_PLAY_AND_ RECORD	Use this category in scenarios where recording (input) and playback (output) of audio must be performed simultaneously. However, you can also use this category in apps that record and then play back the audio. This category indicates that the audio from the app is non-mix-able. This means that when you activate an audio session, it interrupts all the other audio sessions that are non-mix-able. If you want to allow audio mixing, you must use the kony.media.AUDIO_SESSION_CATEGORY_ OPTIONS_MIX_WITH_OTHERS category. When you use this category, the audio of the app continues to play or get recorded even when you lock the screen or use the Silent switch (the Ring/Silent switch on iPhones).
kony.media.AUDIO_SESSION_ CATEGORY_MULTIROUTE	Use this category to route distinct streams of audio data to different output devices at the same time.

The ***AVAudioSession Category Options*** parameter supports the following values.

Constant	Description
kony.media.AUDIO_SESSION_ CATEGORY_OPTIONS_MIX_ WITH_OTHERS	An option that indicates whether the audio from the current session mixes with the audio from active sessions in other audio apps.
kony.media.AUDIO_SESSION_ CATEGORY_OPTIONS_ DUCK_OTHERS	An option that reduces the volume of other audio sessions while audio from the current session plays.

Constant	Description
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_BLUETOOTH	An option that determines whether Bluetooth hands-free devices can appear as available input routes.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_DEFAULT_TO_SPEAKER	An option that determines whether audio from the current session must use the default built-in speaker instead of the receiver.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_INTERRUPT_SPOKEN_AUDIO_AND_MIX_WITH_OTHERS	An option that determines whether spoken audio content from other sessions must be paused when the app plays its audio.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_BLUETOOTH_A2DP	An option that determines whether you can stream audio from the current session to Bluetooth devices that support the Advanced Audio Distribution Profile (A2DP).
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_ALLOW_AIR_PLAY	An option that determines whether you can stream audio from the current session to AirPlay devices.
kony.media.AUDIO_SESSION_CATEGORY_OPTIONS_OVERRIDE_MUTED_MICROPHONE_INTERRUPTION	An option that indicates whether the system interrupts the audio session when the built-in microphone is muted.

Platform Availability

Windows10, Android, iOS

Platform Availability

Windows10, Android, iOS

kony.media.record

Creates a record object that your app can use to record audio.

Syntax

The syntax for native platforms is as follows.

```
kony.media.record(fileobj,config)
```

Input Parameters

Parameter	Description
fileobj	A `kony.io.file` object into which the recording will be saved.
config	An optional object that contains configuration information for the `record` object. For more information, see Remarks below.

Example

```
function errorcallback(errorMessage) {
    var errorMesg = "Reason for failure is : " + errorMessage;
    alert(errorMesg);
}

function successcallback(fileobj) {
    // Your code goes here.
}

var fileObj = new kony.io.file("recording");

var config = {
    onSuccess: successCallback,
    onFailure: failureCallback
};

var _recordObj = kony.media.record(fileObj, config);
```

Return Values

Returns an instantiated `record` object, or null if the object was not created.

Remarks

Use the `kony.media.record` function to instantiate a record object that your app can use to record audio on the device.

Parameter Details

The *config* parameter contains an object with configuration information. Specifically, it contains key-value pairs that set callbacks which are invoked by the `kony.media.record` function. The *config* parameter supports the following keys.

Key	Description
onFailure	The callback function that is invoked when the `kony.media.record` function is not able to create a `record` object.
onSuccess	The callback function that is invoked when the `kony.media.record` function successfully creates a `record` object.

The callback for the `onFailure` key must have the following signature.

```
onFailureCallback(errorMessage);
```

where the `errorMessage` parameter is a string containing the reason for the failure.

The callback for the `onSuccess` key must have the following signature.

```
onSuccessCallback(fileobj);
```

where the `fileobj` parameter is an object of type `kony.io.file` that represents the file the audio is recorded into.

Platform-Specific Notes

The following platform-specific features should be considered when using this function.

- **iOS**: Your app must enable recording before it calls this function. To enable recording, it invokes the kony.application.checkPermission(kony.os.RESOURCE_AUDIO_RECORD,null) function. In addition, the file extension of the audio file for the recording is set to `.aiff`, irrespective of the extension specified in the user's input.
- **Android**: You must add the `RECORD_AUDIO` permission into your app's manifest. On Android 6.0 or later, this will result in the operating system displaying a dialog box asking the user to confirm this permission at runtime. In addition, the file extension of the audio file for the recording is set to `.m4a` no matter what the user input specifies.
- **Windows**: To enable your app to record audio, you must add the "Miciphone" capability in the app's properties.

Platform Availability

Windows10, Android, iOS

37.3 media Object

The `media` object is part of the Media API, and it represents an audio file. Using a `media` object , your app can play the associated audio file, pause it, and so forth. It consists of the following API elements:

- Methods
- Properties

37.3.1 Overview

Use the kony.media.createFromFile or kony.media.createFromUri functions to instantiate a `media` object. Once your app creates a `media` object, it can call the media object methods to play, pause, or stop the audio playback, and so forth.

37.3.1.1 Android-Specific Behaviors

Only one audio file can be played at a time. If your app is playing an audio file and it starts a new one, the current playback is paused until the new file is played. Your app can resume the playback of the paused audio file from the current position by calling the media object's play method.

If a `media` object is playing and a call (incoming or outgoing) or alarm occurs, the `media` object automatically pauses the playback until the call or alarm ends. At that time, playback resumes automatically from the current location in the audio file.

If a `media` object is playing and an SMS (text) message or a notification occurs, the `media` object automatically lowers the volume of the current audio file until the text message sound or notification sound has finished. At that point, the `media` object automatically raises the playback volume to its former level.

37.3.2 media Methods

The media Object consists of the following methods.

pause

Pauses the playback of a media file.

Syntax

```
pause()
```

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
mediaObj.pause();
```

Input Parameters

None.

Return Values

None.

Remarks

This method only has an effect if a media file is currently being played.

Platform Availability

Windows10, Android, iOS

play

Plays a media file.

Syntax

```
play(repeatCount)
```

Input Parameters

Parameter	Description
repeatCount	An integer specifying the number of time the media is played. The default is 1.

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj =  kony.media.createFromFile(theFile);
mediaObj.play(5);
```

Return Values

None.

Remarks

If your app calls this method and does not provide a value for the *repeatCount* parameter, this method plays the audio file once. if the value for the *repeatCount* parameter is negative, the file plays indefinitely. Setting the *repeatCount* parameter to zero stops the playback. However, the recommended way to stop playback is for your app to call the stop or pause methods.

When you call the stop method on Android and then call `play`, there may be a noticeable lag before the file starts playing again. The delay is caused by Android preparing the media again and is therefore specific to that platform only.

Platform Availability

Windows10, Android, iOS

releaseMedia

Releases the memory and resources held by the `media` object.

Syntax

```
releaseMedia()
```

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
mediaObj.releaseMedia();
// If your app tries to use the mediaObj object again, it will get an error!
```

Input Parameters

None.

Return Values

None.

Remarks

Your app can call this function to save memory, especially on devices where memory is in short supply. After your app invokes this function, the `media` object is no longer in memory and attempts to continue to use it by calling its member functions result in errors. Your app must

seek

Sets the current playback position to a specific spot in the media file.

Syntax

```
seek(position)
```

Input Parameters

Parameter	Description
position	An integer number of seconds within the timeline of the media object where playback begins.

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
mediaObj.seek(5); // Moves playback to 5 seconds from the start of the file.
```

Return Values

None.

Remarks

This method moves the current playback position to a point that is a specified number of seconds from the beginning of the media. The number of seconds is specified as an integer in the *position* parameter.

Platform Availability

Windows10, Android, iOS

setCallbacks

Associates callback functions with the `media` object.

Syntax

```
setCallbacks(
    config);
```

Input Parameters

config

A JavaScript object that contains key-value pairs specifying functions to call when `media` object events occur. The keys are as follows.

Key	Description
onMediaCompleted	A function that is called when the media is finished playing. For more information, see the **Remarks** section below.
onMediaFailed	A function that is called if the media cannot be played. For more information, see the **Remarks** section below.
onProgressCallBack	A function that is called when the media is playing. For more information, see the **Remarks** section below.

Example

```
function OnMediaProgress(Position) {
    // Your code goes here.
}

function OnMediaCompleted() {
    alert("Completed playing given song");
}
```

```
function OnMediaFailed(errorMessage) {
    alert("Unable to play the given media");
}

function SetCallbacks() {
    var mediaObj = kony.media.createFromFile(fileobj);
    mediaObj.setCallbacks({
        onProgressCallBack: OnMediaProgress,
        onMediaCompleted: OnMediaCompleted,
        onMediaFailed: OnMediaFailed
    });
}
```

Return Values

None.

Remarks

The *config* parameter of the `setCallbacks` function contains keys that specify callback functions. The callback functions are as follows.

onMediaCompleted

The `onMediaCompleted` key in the *config* parameter of the `setCallbacks` function enables your app to set a callback function that is invoked when the media is finished being played. The callback function must have the following signature.

```
onMediaCompleted();
```

onMediaFailed

The `onMediaFailed` key in the *config* parameter of the `setCallbacks` function enables your app to set a callback function that is invoked when the media cannot be played. The callback function must have the following signature.

```
onMediaFailed();
```

onProgressCallBack

The `onProgressCallBack` key in the *config* parameter of the `setCallbacks` function enables your app to set a callback function that is invoked when the media plays. The callback function must have the following signature.

```
onProgressCallBack(Position);
```

where Position contains the position of the current playback at the time the callback function is triggered.

Platform Availability

Windows10, Android, iOS

stop

Stops the playback of a media file.

Syntax

```
stop()
```

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
mediaObj.stop();
```

Input Parameters

None.

Return Values

None.

Remarks

This method only has an effect if a media file is currently being played.

When you call this method on Android and then call play, there may be a noticeable lag before the file starts playing again. The delay is caused by Android preparing the media again and is therefore specific to that platform only.

Platform Availability

Windows10, Android, iOS

37.3.3 media Properties

The media object provides the following properties.

data

Holds the data object that contains the sound associated with the media object.

Syntax

```
data
```

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
kony.print("The data inside the media object is:" + mediaObj.data);
```

Type

JavaScript object.

Read/Write

Read only.

Platform Availability

Windows10, Android, iOS

duration

Contains the duration of the audio in seconds.

Syntax

```
duration
```

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
kony.print("The duration of the media is:" + mediaObj.duration);
```

Type

Number

Read/Write

Read only.

Platform Availability

Windows10, Android, iOS

isPlaying

This property contains a Boolean value that indicates whether or not the audio is currently playing.

Syntax

```
isPlaying
```

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
kony.print("The media is being played now or not:" + mediaObj.isPlaying);
```

Type

Boolean

Read/Write

Read only

Platform Availability

Windows10, Android, iOS

volume

Contains the current volume level.

Syntax

```
volume
```

Example

```
var theFile = new kony.io.File("MyAudioFile.mp3");
var mediaObj = kony.media.createFromFile(theFile);
kony.print("The volume of the media is:" + mediaObj.volume);
```

Type

Double

Read/Write

Read+Write

Remarks

Use this property to read the current volume level or set a new volume level for playing back the audio file. Valid values for this property range from 0.0 to 1.0 inclusive.

Platform Availability

Windows10, Android, iOS

37.4 record Object

The record object is part of Media API, and it enables your app to record audio. It consists of the following API elements:

- Methods

37.4.1 Overview

Use the kony.media.record function to create a record object. Once your app creates a `record` object, it can call the `record` object methods to start or stop recording audio.

37.4.1.1 Android-Specific Behaviors

When a `record` object is recording audio, playing an audio file causes the `record` object to stop and save the recording. This occurs whenever a phone call is initiated or received, or when a notification or alarm sound is played. If the recording is successfully saved, the `onSuccess` callback that your app set in the *config* parameter of the kony.media.record function is automatically invoked. If the recording cannot be saved, the `onFailure` callback, which your app also set in the *config* parameter of the `kony.media.record` function, is invoked instead.

It is possible, for whatever reason, for the device to begin recording audio while your app is recording. In such a case, the `record` object in your app stops recording and saves the audio it has captured to a file.

37.4.2 record Methods

The `record`object supports the following methods.

startRecording

Starts recording audio.

Syntax

```
startRecording();
```

Example

```
function errorcallback(errorMessage) {
    var errorMesg = "Reason for the failure is: " + errorMessage;
    alert(errorMesg);
}
```

```
function successcallback(fileobj) {}

var fileObj = new kony.io.file("recording");

var config = {
    onSuccess: successCallback,
    onFailure: failureCallback
};

var recordObj = kony.media.record(fileobj, config);
recordObj.startRecording();
```

Parameters

None.

Return Values

None.

Platform Availability

Windows10, Android, iOS

stopRecording

Stops the current recording.

Syntax

```
stopRecording();
```

Example

```
function errorcallback(errorMessage) {
    var errorMesg = "Reason for the failure is: " + errorMessage;
    alert(errorMesg);
}
```

```
function successcallback(fileobj) {}

var fileObj = new kony.io.file("recording");

var config = {
    onSuccess: successCallback,
    onFailure: failureCallback
};

var recordObj = kony.media.record(fileobj, config);
recordObj.startRecording();

// More code goes here.

recordObj.stopRecording();
```

Parameters

None.

Return Values

None.

Remarks

This method has no effect if recording is not in progress.

Platform Availability

Windows10, Android, iOS

38. Network API

The Network API provides functions for network calls via HTTP or HTTPS. The Network API enables your app to request data from remote sources, such as your data servers, over the Internet. In also enables you to validate that the data requests and the responses have not been tampered with as they traveled between the client app and the server. The Network API comprises of the following modules:

- Network Calls
- Asynchronous Network Calls

The Network API uses the `kony.net Namespace` and the following API elements

Function	Description
`kony.net.cancel`	cancels only async network calls. Synchronous calls have a platform-specific cancellation mechanism provided by the platform
`kony.net.clearCookies`	Removes cookies from the client that are associated with the specified domain.

Function	Description
`kony.net.FormData`	Call this function to create a FormData object.
`kony.net.getActiveNetworkType`	Retrieves the currently-active network type.
`kony.net.getCookies`	Retrieves cookies associated with the specified domain.
`kony.net.HttpRequest`	Creates an HttpRequest object.
`kony.net.isNetworkAvailable`	Enables you to check whether a network is available for data transport on a device.
`kony.net.loadClientCertificate`	Sets a client certificate to be used for HTTPS client authentication.

Function	Description
`kony.net.mfintegrationsecureinvokerasync`	Invokes a Kony Fabric service asynchronously.
`kony.net.removeClientCertificate`	Removes already loaded client certificate.
`kony.net.removeAllCachedResponses`	Clears the default cache of an application by removing all responses received from URLs.
`kony.net.removeIntegrityCheck`	Disables integrity checks for HTTP calls between the client and the server.
`kony.net.setCookies`	Adds cookies, or replaces an existing cookie in the cookie storage.

Function	Description
`kony.net.setIntegrityCheck`	Enables the addition of checksums to HTTP calls for HTTP integrity checking.
`kony.net.setNetworkCallbacks`	Allows the developer to register for network status changes.
`setResponse`	Sets a callback function that handles network events.
`kony.net.urlDecode`	Converts a URL string from application/x-www-form-urlencoded format in the UTF-8 encoding.

Function	Description
kony.net.urlEncode	Converts a URL string into application/x-www-form-urlencoded format using the UTF-8 encoding.

38.1 Overview

The Network API enables your app to request data from remote sources, such as your data servers, over the Internet. In also enables you to validate that the data requests and the responses have not been tampered with as they traveled between the client app and the server.

To set a callback function to be executed during network changes by using the `kony.net.setNetworkCallbacks` function. Also, set a call back function to configure the response for any network events by using the `setResponse` function. Before establishing a network connection, ensure if a network is available by using the `kony.net.isNetworkAvailable`function. To find the active network available, use the `kony.net.getActiveNetworkType` function.

To verify the identity of the server and establish a secure network connection through HTTPS, use the `kony.net.loadClientCertificate` function. A client certificate for HTTPS client authentication is set. You can remove the client certificate by using the `kony.net.removeClientCertificate` function.

To start with establishling a connection using HTTP, create an Http request object by using the `kony.net.HttpRequest` function. After creating an object, invoke a required Kony Fabric service by using the `kony.net.mfintegrationsecureinvokerasync` function. Then use the `kony.net.setIntegrityCheck` function to validate HTTP messages passing between the client app and the servers. To disable the HTTP integrity checking, use the `kony.net.removeIntegrityCheck` function.

Further, a URLs can be sent over the internet in ASCII format only. To convert a URL into valid ASCII format, use the `kony.net.urlEncode` function. Then use the `kony.net.urlDecode` function to convert the ASCII characters to readable format.

Please note that the `kony.net.invokeServiceAsync` and `kony.net.invokeservice` are deprecated and should not be used in new software. However, documentation for them is still available to help with the maintenance of legacy software.

> ***Note:*** From V8 SP4 onwards, for etag caching of Network APIs, the data in a Kony Visualizer App Viewer child app is stored in child app data and not under the parent app. This feature is applicable for iOS, Windows, and Android platforms.

38.2 Network Calls

Any network call goes through the following phases:

- Phase 1: preparing network request.
- Phase 2: making a network connection.
- Phase 3: handling the returned response.

> ***Note:*** All network APIs use the *post* method to fetch data from a specified URL. The post method is not configurable.

On all Apple platforms (iOS, OS X, etc), Apple requires that all apps use App Transport Security (ATS) for all network calls. As a result, all calls should use HTTPS rather than HTTP as the transport protocol. Apps developed with Kony technologies for Apple platforms have ATS enabled by default. Attempts to manually bypass ATS result in an exception being thrown.

For more information about ATS, please see the Apple Documentation.

38.3 Asynchronous Network Calls

Asynchronous services can be invoked over existing web protocols (http or https). An asynchronous service does not wait for a response, but continues its work and handles the response later during the execution of the application. When you invoke an asynchronous service call, the user interface (UI) is not blocked, and you can perform other actions such as invoking other services such as updating data on the forms. Use asynchronous service calls if you have long running tasks in the application (like querying a database, making another service call etc).

You need to define a separate callback function to handle return values of an asynchronous service call. An asynchronous call always returns a handle to the service call as an immediate response, and the callback function handles logic to process the return values and error codes (if any).

On the Mobile Web platform, where the virtual machine runs the server, an asynchronous network call is also treated like a synchronous call. When the asynchronous network call API is used, the Mobile Web platform invokes the network call with the URL and the parameters provided. While the native applications continue processing of the next line of code after the network call (without waiting for the network call to return), the Mobile Web platform waits until the response is fetched from the network call, executes the callback function, and then executes the next line of code after the network call.

In native applications, the print statements appear in the following order:

1. Hello World - Before invoking network call.
2. Hello World - After invoking network call.
3. Hello World - Inside callback function.

On the Mobile Web platform, the print statements appear in the following order:

1. Hello World - Before invoking a network call.
2. Hello World - Inside a callback function.
3. Hello World - After invoking a network call.

Advantages

With an asynchronous service call, you can:

- Make multiple service calls at the same time based on the application requirement.
- Perform other actions because code execution and user interface are not blocked.
- Make your application perform better and be more responsive.
- Use the resources to the fullest extent (of the system on which the application is running).

Disadvantages

You need to define appropriate actions within the callback functions, and manage the application states and the UI.

Use Cases

You should use asynchronous services in the following scenarios:

1. When the service calls are independent of each other, you can use the asynchronous service calls. For example, you can search for flights, hotels, cars, and more (all these tasks are independent of each other).
2. When the response of a service call is not consumed immediately in the application flow, but is used later.

38.4 kony.net Namespace

The kony.net namespace, which is a part of the Network API, contains the following API elements.

38.4.1 Functions

The kony.net namespace contains the following functions.

kony.net.cancel

This API cancels only async network calls. Synchronous calls have a platform-specific cancellation mechanism provided by the platform (mechanism may vary from platform to platform).

Syntax

```
kony.net.cancel(
    connHandle);
```

Input Parameters

Parameter	Description
connHandle	The handle to the asynchronous service. connHandle is returned by invokeserviceasync.

Example

```
function cancelService() {
    kony.net.cancel(connHandle);
}
```

Return Values

None.

Exceptions

1200 - Network Error.

Remarks

Invalid parameters to this function are ignored.

Platform Availability

Available on all platforms except Server-Side Mobile Web, Windows 8, and Windows Phone 8 channels.

kony.net.clearCookies

Removes cookies from the client that are associated with the specified domain.

Syntax

```
kony.net.clearCookies(
    url,
    cookieName)
```

Input Parameters

Parameter	Description
url	An optional string that specifies a valid URL of a domain from where the cookies were obtained.
cookieName	An optional array of strings that specifies the cookie names that are to be removed from the current application.

Example

```
kony.net.clearCookies("http://www.google.com");
```

Return Values

None

Remarks

In an application where service calls are made through Browser Widget or through native service calls, cookies are sent from the servers to maintain the session. These cookies are stored in the cookie store on the client side. Using this API you can remove the cookies of a particular domain obtained through all of HttpRequest, invokeServiceAsync and browser widget.

If you specify the optional parameters `url` and `cookieName`, then all the cookies with the specified name related to that URL are removed from your application. The parameter `url` should be the complete URL. For example, http://www.gooogle.com. If the URL is not valid, then this API will not remove the cookies. An exception is shown of type KonyError with the error code as 1005 and error name/message as "invalid input url".

If the scheme of the URL is "https" then all the cookies of `https` as well as `http` will be removed. If the scheme of the URL is "http" then only the cookies stored for `http` will be removed. If the cookies are not present in the application cookie store in the provided array of cookieNames, then this API will not perform any action and does not raise an exception.

Platform Availability

Available on iOS, Android, Windows, and SPA platforms.

Limitations

- Android
  - The kony.net.clearCookies API does not remove the cookie from the Browser widget's cookie store; it replaces the cookie value with an empty string.

- SPA
    - You can remove the cookies related to a currently loaded domain in a web page.
    - Cookies with the `httpOnly` flag cannot be removed. The secured cookies (https) cannot be removed when application is accessed with non-secured protocol.

kony.net.FormData

The FormData object represents an ordered collection of entries with name-value pairs. Call this function to create a FormData object.

Syntax

```
kony.net.FormData();
```

Example

```
var httpinputparams1 = new kony.net.FormData()
```

Input Parameters

None.

Return Values

Returns an object of type FormData.

Platform Availability

Android, iOS, Windows, and SPA

kony.net.getActiveNetworkType

Retrieves the currently-active network type.

Syntax

```
kony.net.getActiveNetworkType();
```

Example

```
function checkActiveNetwork() {
    return kony.net.getActiveNetworkType() ;
}
```

Input Parameters

None.

Return Values

If the device is offline, it will return null. Otherwise, it will return the appropriate connection type.

Return Value	Description
constants.NETWORK_TYPE_3G	when 3G network is used.
constants.NETWORK_TYPE_WIFI	when Wi-Fi is used.
constants.NETWORK_TYPE_ANY	only when SPA , Mobile Web, and Desktop Web applications are used.

For SPA , Mobile Web, and Desktop Web, it will always return constants.NETWORK_TYPE_ANY.

Remarks

SPA and Desktop Web depend upon the navigator.onLine property to detect if the device is offline or online. The implementation of this property across browsers is uneven. For more information, refer to Mozilla documentation for browser support.

Platform Availability

Available on all platforms.

kony.net.getCookies

Retrieves cookies associated with the specified domain.

Syntax

```
kony.net.getCookies(
    URL)
```

Input Parameters

Parameter	Description
URL	The URL of the domain for which the cookie is retrieved. If the URL in the *URL* parameter is not fully formed, this function returns `Null`. For details, see **Remarks** below.

Parameter	Description

cookieFormat (optional)	The format in which the cookies must be retrieved.
	
	Note: This parameter is only available on the iOS platform.
	
	The cookieFormat parameter can have the following two values:
	
	• **constants.COOKIES_IN_JSON**: Returns an array of cookies in JSON format.
	
	Sample JSON dictionary format:
	
	`{`
	`  "Version": 1,`
	`  "Name":"appName",`
	`  "Value": "App01",`
	`  "ExpiresDate":'(null)', // in milli seconds`
	` "Domain": "app.example.com",`
	`  "Path": "/"`
	`  "HttpOnly"=TRUE;`
	`}`
	
	• **constants.COOKIES_IN_STRING**: Returns an array of cookies in String format.
	
	This is the default format in which cookies are retrieved.
	
	Sample String format:
	
	`appName=App01; Version=1; Domain=app.example.com; HttpOnly=TRUE; Path=/`

Example

```
var cookies = kony.net.getCookies("http://www.google.com", constants.COOKIES_
IN_STRING);
for (index = 0; index < cookies.length; index++) {
    cookie = cookies[index];
    kony.print("Cookie is: " + cookie);
}
```

Return Values

Returns an array of cookies. For details, see **Remarks** below.

```
//For example
SSID=Ap4P….GTEq; Domain=foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01
GMT; Secure; HttpOnly
```

Remarks

In an application where service calls are made through Browser Widget or through native service calls, cookies are sent from the servers to maintain the session. These cookies are stored in cookie store on the client side. The kony.net.getCookies API enables access to the cookies for a particular domain. The URL is given as an input parameter to kony.net.getCookies API, then all cookies related to that domain are returned.

In the current application context, you can also access cookies for a particular domain stored in cookie store on the client side or device.

The kony.net.getCookies function returns all the cookies stored in local cookie store on the client side or device for that particular domain. The returned cookies can vary based on the domain (partial or full domain) and as well as the path.

The URL is passed as input and must be a fully formed, valid URL like `http://www.google.com`.

If the URL is invalid or not formed properly, `Null` is returned.

Each cookie that this function returns is a string that follows the format `key=value`. Each cookie string can also contain additional information following the Standard HTTP Cookie Format, such as Domain, Path, Expires, Secure, and HttpOnly. This function returns `Null` if there are no cookies for the domain.

If the scheme of the URL is `https`, then this function returns all the cookies of `https` cookies as well as the `http` cookies. If the scheme of the URL is `http`, then only the cookies stored for `http` are returned.

Platform Availability

Android, iOS, Windows, and SPA

Limitations

- In Windows, the cookies stored in the Browser Widget cookie store are returned. Cookies stored through native service calls are not returned.
- In SPA, the cookies related to a currently loaded domain in a web page are accessible.
- In SPA, cookies with the `httpOnly` flag cannot be accessed.

---

kony.net.getDomainVerificationUserState

Determines the verification status of the specified domain.

Syntax

```
kony.net.getDomainVerificationUserState (domainString);
```

Input Parameters

Parameter	Description
domainString	The domain for which the verification status is to be determined.

Example

```
var state = kony.net.getDomainVerificationUserState("temenos.com");
if (state == constants.DOMAIN_STATE_NONE) {
    // show the dialog to provide some context to user to navigate settings
screen.
} else if (state == constants.DOMAIN_STATE_VERIFIED) {

}
```

Return Values

A constant that determines the verification status of the specified domain. Following are the supported constants:

Constant	Description
kony.constants.DOMAIN_STATE_VERIFIED	Indicates that the domain has passed the Android App Links verification.
kony.constants.DOMAIN_STATE_SELECTED	Indicates that the domain has not passed Android App Links verification. However, it indicates that the user has associated with an app.
kony.constants.DOMAIN_STATE_NONE	Indicates that the domain has neither passed the Android App Links verification nor has a user associated with an app

Platform Availability

Available on the Android platform.

kony.net.HttpRequest

Creates an HttpRequest object.

Syntax

```
kony.net.HttpRequest(requestOptions)
```

Example

```
var httpRequestNew = kony.net.HttpRequest(({
       "timeoutIntervalForRequest": 60,
       "timeoutIntervalForResource": 600
   }
);
```

Input Parameters

requestOptions - An optional argument that can have the following parameters.

> ***Note:*** The requestOptions parameter is only available on the Android and iOS platforms.

Parameter	Description
timeoutIntervalForRequest [integer]	An optional parameter that specifies the maximum connection time out value (in seconds) for the request to establish an HTTP connection. If the request is not established before the timeout interval is reached, a timeout error occurs. The timeoutIntervalForRequest parameter is only applicable for the Android and iOS platforms. On the Android platform, the default timeout value is zero, which means that the timeout is infinite. On the iOS platform, the timeoutIntervalForRequest parameter is only applicable for background network calls (when the backgroundTransfer property is set to true). If the timeout value is not specified, the default timeout value of 60 seconds is set by the OS.
timeoutIntervalForResource [integer]	An optional parameter that specifies the maximum time out value (in seconds) for which the network must wait for a response from the server resource. If the resource is not retrieved before the timeout interval is reached, a timeout error occurs. The timeoutIntervalForResource parameter is only applicable for the Android and iOS platforms. On the Android platform, the default timeout value is zero, which means that the timeout is infinite. On the iOS platform, the timeoutIntervalForResource parameter is only applicable for background network calls (when the backgroundTransfer property is set to true). If the timeout value is not specified, the default timeout value of 1 week (7 days) is set by the OS.

Return Values

None

Remarks

The HttpRequest Object supports an HTTP or HTTPS request to any resource on the network and fetches the response.

> ***Note:*** SNI (Server Name Indication) is supported from Android 4.2 onwards. That is, from API level >=17. From Android 4.0 to 4.2 versions, the support of SNI depends on device capability. That is 14<= API Level <17.

kony.net.isNetworkAvailable

This API enables you to check whether a network is available for data transport on a device.

Syntax

```
kony.net.isNetworkAvailable(
    networkType)
```

Input Parameters

networkType

An integer constant that specifies the network type on the device for data transport. You can specify any of the following.

Constant	Description

constants.NETWORK_TYPE_3G	Indicates whether the device has sufficient network coverage to send data using the 3G channel. Available on Android, iOS, and Windows. Other platforms always return `false` when queried using `kony.net.isNetworkAvailable (constants.NETWORK_TYPE_3G)`. Some platforms such as Android do not differentiate between 3G and 2G connectivity. In this case, even if the device is on a 2G connection, the API would still indicate that the device is on 3G connectivity.
constants.NETWORK_TYPE_WIFI	Indicates whether the device has sufficient network coverage to send data using the Wi-Fi channel. Available on Android, iOS, and Windows. Other platforms always return `false` when queried using `kony.net.isNetworkAvailable (constants.NETWORK_TYPE_WIFI).`
constants.NETWORK_TYPE_ETHERNET	Indicates whether the device has sufficient network coverage to send data using the ETHERNET channel. Available on Desktop Web, but it will be ignored for Mobile Web and Tablet. Other platform will always return `false` when queried using `kony.net.isNetworkAvailable (constants.NETWORK_TYPE_ETHERNET).`
constants.NETWORK_TYPE_ANY	Indicates whether the device has sufficient network coverage to send data over any supported data channels. Available on Android, iOS, Windows, HTML5 SPA, Mobile Web, and Desktop Web.

Example

```
function checkIfNetworkIsAvailable() {
return kony.net.isNetworkAvailable(constants.NETWORK_TYPE_ANY);
}
```

Return Values

Returns `true` if the specified data network is available, or `false` if not.

Platform Availability

Available on all platforms.

kony.net.loadClientCertificate

Sets a client certificate to be used for HTTPS client authentication.

Syntax

```
kony.net.loadClientCertificate(
    certParamsTable)
```

Input Parameters

certParamsTable

A list of following key-value pairs required to load client certificate for client authentication in a mutually authenticated HTTPS connection.

Key	Description
cert	Can be of type RawBytes or base64String. Specifies the certificate to be loaded. The certificate should be of type PKCS12 certificate encoding format, which holds both client certificate and private key.
pass	A string containing the password that is protecting the PKCS12 certificate. If PKCS12 certificate is not password protected, use an empty string; that is, “ ”.

Example

```
function loadCert() {
  try {
  var certFileName = kony.io.FileSystem.getCacheDirectoryPath() + "/" +
clientCertFileName;
  var certFile = new kony.io.File(certFileName);
  var certStream = certFile.read()

  var certParamTable = {
  cert: certStream,
  pass: "password"
  };
  var res = kony.net.loadClientCertificate(certParamTable);
  kony.print("loadClientCertificate status = " + res)
  } catch (e) {
  alert(e);
  }
 }
```

Return Values

This API returns a Boolean value whether the client certificate is loaded successfully or not.

Remarks

The following functions use the client authentication feature:

kony.net.invokeServiceAsync

kony.net.send

If loading the client certificate succeeds, the previously-loaded certificate is replaced with new one.

If loading the client certificate fails, the previously-loaded certificate is retained as is.

Any certificate loaded using this function is automatically unloaded as soon as the application exits.

> ***Note:*** When you use the kony.net.HttpRequest API in sync mode, both server and client pinning do not work in the iOS platform.

Exceptions

If the mandatory parameters are missing, following error codes occur.

100 - invalid type of parameters

101 - invalid number of arguments

Platform Availability

- iOS
- Android

kony.net.mfintegrationsecureinvokerasync

Invokes a Kony Fabric service asynchronously.

Syntax

```
kony.net.mfintegrationsecureinvokerasync(
    inputParams,
    serviceName ,
    operationName,
    callbackFunction)
```

Input Parameters

Parameter	Description
inputParams	An object containing the Parameters for the Kony Fabric service. The format of this object is dependent on the input requirements of the Kony Fabric service being invoked.
serviceName	A string that holds the name of the service to invoke.
operationName	A string that specifies the name of the service's operation.

Parameter	Description
callbackFunction	A callback function to handle the service's response.

Example

```
function callbackFunction(status, response) {
    //application specific code goes here.
}

var serviceName = "ServiceTwo";
var operationName = "HealthOperation";

// inputParams here is just a sample. Your Parameters will look different.
// The Parameters are dependent on the input information that your
// Kony Fabric service's operation requires.
// <place-holder> is a placeholder for the atual values of your parameters.
var inputParams = {
    "q": "<place-holder>",
    "httpheaders": {
        "api-key": "<place-holder>"
    }
};
mfintegrationsecureinvokerasync(inputParams, serviceName, operationName,
callbackFunction);
```

Return Values

None.

Remarks

Use this service to call a Kony Fabric service from your app. When the service sends a response to your app, the response is processed by the callback function that your app passes through the callbackFunction parameter of this function.

The format of the *inputParams* object parameter is not specified here because its format depends entirely on the information that the Kony Fabric service's operation requires for input.

The function passed through the *callbackFunction* parameter must match the following Syntax.

```
callbackFunction(status,response);
```

The callback function takes two parameters. The first, called *status*, is an integer that indicates the status of the response from the service's operation The value contained in the *status* parameter depends on the service itself.

The second parameter to the callback function is called *response*. It is a JavaScript object that contains the response from the service's operation. The format of the object and the data it contains is dictated by the service's operation.

kony.net.removeClientCertificate

Removes already loaded client certificate.

Syntax

```
kony.net.removeClientCertificate()
```

Example

```
function removeCert() {
    kony.net.removeClientCertificate();
}
```

Input Parameters

None.

Return Values

None.

Exceptions

None.

Platform Availability

- Android

kony.net.removeAllCachedResponses

Clears the default cache of an application by removing all responses received from URLs.

Syntax

```
kony.net.removeAllCachedResponses()
```

Example

```
kony.net.removeAllCachedResponses();
```

Input Parameters

None.

Return Values

None.

Platform Availability

iOS

kony.net.removeIntegrityCheck

Disables intregity checks for HTTP calls between the client and the server.

Syntax

```
kony.net.removeIntegrityCheck()
```

Example

```
kony.net.removeIntegrityCheck();
```

Input Parameters

None.

Return Values

None.

kony.net.setCookies

Adds cookies in the main app cookie storage. If a cookie with the same name, domain, and path already exists in storage, this API replaces the specified cookie in the cookie storage.

Syntax

```
kony.net.setCookies(url,cookiesList)
```

Input Parameters

Parameter	Description
url	A string that specifies a valid URL of a domain where the cookies are to be defined. If the url is empty, a specific cookie is stored based on the cookie acceptance policy.

Parameter	Description

cookiesList	An array of objects, that each contains details of a cookie in the dictionary format.
	
	Sample dictionary format:
	
	`{`
	`  "Version": 1,`
	`  "Name":"appName",`
	`  "Value": "App01",`
	`  "ExpiresDate":'(null)', // in milli seconds`
	`  "Domain": "app.example.com",`
	`  "Path": "/"`
	`}`

Example

```
function setCookiesFunc() {
    var lisOfCookies = [{
        "Name": "appName",
        "Value": "App01",
        "Domain": "app.example.com",
        "Path": "/"
    }]
    kony.net.setCookies("https://www.example.com", listOfCookies)
}
```

Return Values

None

Remarks

To successfully create a cookie, you must provide values for (at least) the **Path**, **Name**, **Value**, and the **Domain** keys. All the other valid keys must start with capital letters.

The parameter `url` must be the complete URL. For example, http://www.gooogle.com.

Platform Availability

Available on iOS platform.

kony.net.setIntegrityCheck

Enables the addition of checksums to HTTP calls for HTTP integrity checking.

Syntax

```
kony.net.setIntegrityCheck(
    propertiesTable)
```

Input Parameters

propertiesTable

A JavaScript object that contains the following key-value pairs.

Key	Description
algo	A string that specifies the hashing algorithm to use for the checksum. The following values are supported: • MD5 • SHA1 • SHA224 (Supported on Android API 21 and later) This is not supported for Windows 10 and Windows Desktop. • SHA256 • SHA384 • SHA512

Key	Description
salt	A string containing an app-supplied random data value that is used as additional input to the hash function. The `salt` argument has a maximum string length of 1024 characters. If it is longer, the excess characters are truncated. Your app can pass an empty string for this argument if you feel that a salt value is not necessary.
headerName	A non-empty string that specifies the name of the header. The maximum length of this string is 64 characters. If more characters are passed, the `headerName` argument is truncated to 64 characters.
validateResp	A Boolean value that indicates whether the response should be validated. A value of `true` means that the response checksum needs to be validated, while `false` indicates that it does not. For important details, see the **Remarks** section below.
hostNameList	An optional array of strings that specify the URLs of servers. When the client app on the device communicates with the specified servers, it adds integrity checking headers to each HTTP request. For more information, see the **Remarks** section below.

Example

```
var propertiesTable {
    algo: “SHA256”,
    salt: “secret_123”,
    headerName: “X - Checksum”,
    validateResp: true,
    hostNamesList: [“mail.kony.com”, “cloud.kony.com”]

};

kony.net.setIntegrityCheck(propertiesTable);
```

Return Values

None

Exceptions

This function throws the following exceptions.

Error Code	Description
100	Invalid argument or parameter name.
101	Missing argument or parameter,
102	Invalid number of arguments.

Remarks

Use this function to cause the Kony Visualizer API Framework to use checksums to validate HTTP messages passing between the client app and the servers.

The beginning and ending whitespace is automatically trimmed on all string arguments in the *propertiesTable* parameter .

If the `validateResp` argument in the *`propertiesTable`* parameter is set to `true`, your app uses an HttpRequest object to receive the response. When it does, it uses the integrityStatus property of the HttpRequest object to check whether the integrity check is successful or not.

If the optional `hostNameList` argument is omitted from the *`propertiesTable`* parameter, then the integrity checking header is added to all outgoing HTTP and HTTPS calls. However, if the host names are specified in the `hostNameList` argument, then the integrity checking header is only added to calls to the specified hosts. Also, only basic validation is performed on the named hosts' parameters if the hostname strings are limited to the set [a-z, A-Z, 0-9, -]. In addition, the wildcard character, which is the asterisk (*), can only be used as a prefix. For example, *.kony.com is valid, but kony.* is not. Finally, you must ensure that all of the host names you pass are valid. The `setIntegrityCheck` function does not validate the format of the host names for you. All host names must follow the standard delineated in the Wikipedia Hostname topic.

Platform Availability

- Andoid
- iOS
- Windows 10
- Windows Desktop

kony.net.setNetworkCallbacks

This function allows the developer to register for network status changes.

Syntax

```
kony.net.setNetworkCallbacks(
    callbackconfig)
```

Input Parameters

Parameter	Description
callbackconfig	A JSON object that contains a property called `statusChange`. • statusChange: A callback function that is invoked when the device goes offline or online. This callback receives an input parameter that indicates whether the device was online or offline when this callback was invoked.

Example

```
var config = {};
config.statusChange = function(isOnLine) {
    if (isOnLine) {
        alert("Device is online");
    } else {
        alert("Device is offline");
    }
};
kony.net.setNetworkCallbacks(config);
```

Return Values

None

Remarks

This function allows the developer to register for network status changes. You can then change the user experience according to the network availability.

Platform Availability

Available on all platforms except Windows Phone 8, Windows Phone 8.1, Windows 8, and Mobile Web.

kony.net.urlDecode

Converts a URL string from application/x-www-form-urlencoded format in the UTF-8 encoding.

Syntax

```
kony.net.urlDecode(
    queryParams,
    exemptionString)
```

Input Parameters

Parameter	Description
queryParams	A string in application/x-www-form-urlencoded format in the UTF-8 to decode.
exemptionString	Optional. Specify the characters in the string that should be exempted from decoding. The parameter is used only on iOS . The Android and Windows platforms ignore this parameter.

Example with exemptionString parameter

```
var result = kony.net.urlDecode("hello*_%40+world");
kony.print(result);

//iOS output
  hello*_%40+worl%64

//Android output
  hello*_%40+world
```

Example without exemptionString parameter

```
var result = kony.net.urlDecode("hello*_%40+world");
kony.print(result);

//Output for all platforms
  "hello*_@ world"
```

Return Values

Returns a string containing the decoded URL.

Platform Availability

- Android
- iOS
- Windows

kony.net.urlEncode

Converts a URL string into application/x-www-form-urlencoded format using the UTF-8 encoding.

Syntax

```
kony.net.urlEncode(
    queryParams
    exemptionString)
```

Input Parameters

Parameter	Description
queryParams	A string containing a URL to encode.
exemptionString	Optional. A string that specifies the characters in the string that should be exempted from encoding. The parameter is used only on iOS. The Android and Windows platforms ignore this parameter.

Example with exemptionString parameter

```
var exemptionString =  "helloworl-*_.";
var result = kony.net.urlEncode(“hello*_@ world”, exemptionString);
kony.print(result);

//iOS output
   hello*_%40+worl%64

//Android output
  hello*_%40+world
```

Example without exemptionString parameter

```
var result = kony.net.urlEncode(“hello*_@ world”);
kony.print(result);

//Output for all platforms
  hello*_%40+world
```

Return Values

Returns the encoded string.

Remarks

The string is encoded based on the following rules.

- The following characters remain the unchanged.
    - a-z
    - A-Z
    - 0-9
    - Period (**.**), hyphen (**-**), asterisk (*****), and underscore (**_**)

- The space character " " is converted to a plus symbol (**+**).
- Other characters are unsafe and are first converted to one or more bytes using an encoding scheme. Each byte is represented as 3-character string, *%xy*, where *xy* is the two-digit hexadecimal representation of the byte.

Platform Availability

- Android
- iOS
- Windows

38.4.2 Deprecated Functions

The following functions in the kony.net namespace are deprecated. Their references are provide to assist in maintaining legacy software. However, these functions should not be used for new development.

kony.net.invokeServiceAsync

This API allows you to invoke a service asynchronously with the specified parameters.

Note: These APIs are generated by Kony Visualizer for the code generated using mapping editor. Even though these APIs have been retained as is they have not been published as standard APIs provided by the Kony runtime libraries. These APIs will not be available to a developer not using Kony Visualizer in future for development.

Note: SNI (Server Name Indication) is supported from Android 4.2 onwards. That is, from API level >=17. From Android 4.0 to 4.2 versions, the support of SNI depends on device capability. That is 14<= API Level <17.

```
function asyncCallback(status, news)
                         {
                         // populate UI
                         };
```

```
var news_inputparam = {};
news_inputparam["serviceID"] = "news";
news_inputparam["httpheaders"] = {};
news_inputparam["httpconfig"] = {timeout:5};
news = appmiddlewareinvokerasync(news_inputparam, asyncCallback);
```

Syntax

```
kony.net.invokeServiceAsync (url,inputParamTable,callback,info)
```

Input Parameters

url [String] - Mandatory

The URL containing the remote location from where the content is to be retrieved.

inputParamTable [Object] - Mandatory

The inputParamTable is the list of parameters that need to be passed to the remote service. The *inputParamTable* contains the following key-value pairs:

- *appID* - Unique ID of the application.
- *serviceID* - Unique ID of the service.
- *channel* - Indicates the mobile application channel for which the request is been raised. Valid values are "rc" for native clients and "wap" for Mobile Web and SPA clients.

  Important:

  - The above fields **appID**, **serviceID**, and **channel** are mandatory only when **middleware** is used.
  - If you use a third-party URL for invokeServiceAsync API, the URL should return only JSON value.

- *httpheaders* [Object] - The *httpheaders* is an optional key in the parameter table . Use this key when you want to pass custom headers to the remote service.

For example:

```
httpheaders={authkey:"myauthkey",
authtoken:"myauthtoken"}
```

Certain headers like useragent are overridden by the underlying SDK and cannot be set using the httpheaders key.

If the remote service requires basic authentication, you can set the header as follows:

```
Authorization:"Basic" <base64encoding of the
username:password>". For example, httpheaders={Authorization:
"Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ=="}
```

Note:

- Make sure there is a space between `Basic` and the actual base64 encoded string.
- In SPA and Desktop Web, if you use third-party URLs, then apply xmlHTTPRequest APIs and enable Cross-Origin Resource Sharing (CORS) headers on the target servers (that is, the server to which the service request is made).

If you make a third-party URL call without CORS headers, it results in security exception saying that browser has prevented cross origin request.

The following is an example on how to use this parameter:

```
local myhttpheaders={authkey:
"myauthkey",authtoken:"myauthtoken"}
local inputParamTable={appID:"SampleApp",
serviceID:"accountLogin",channel:"rc",
httpheaders:myhttpheaders};
```

- *httpconfig* [Object] - The *httpconfig* is an optional key in the parameter table . Use this key when to pass custom configurations to set the timeout in seconds for network calls.

For example:

```
httpconfig={timeout:5}
```

Important: On SPA and Desktop Web, if you are using third-party URLs, use the HTTP request method.

To configure the HTTP request method, use "method." The possible values are:

- *Get*
- *Post* (default value)

Note: An invalid value to *method* leads to undefined behavior.

For example:

```
httpconfig = { timeout: 5, method: "get"}
```

callback [Function] - Mandatory

The callback function that is called to handle the return values of the asynchronous network call (in case of success) and error messages (in case of failure). The following is the signature:

callback(status, resulttable)

- **status** - an integer value - indicating the status
  - 100 - network call initiated successfully. The resultset is not available, and it is nil.
  - 200 - network is in progress (when you start receiving the first byte). The resultset is not available, and it is nil.
  - 300 - network call canceled. The resultset is not available and it is nil.
  - 400 - network call is finished (called in success and failure scenarios). Actual state can be queried using opstatus in the resultset.

Note: On Mobile Web, the callback function is always invoked with a status of 400. Other intermediate status codes are not applicable on Mobile Web.

- **resulttable** - an object with key-value pairs. Follows the same structure (opstatus, errcode, and errmsg along with the actual network returned data).

The resulttable represents the object returned by the service. This object contains three values:

opstatus

errcode

errmsg

If the *opstatus* is 0, the service call is a success while a non-zero value indicates a failure.

If the *opstatus* is a non-zero value, an errcode is generated. The following are the error codes:

- 1000- Unknown error while connecting (if the platform cannot differentiate between network errors, the platform reports error code 1000 by default).
- 1011 - Device has no Wi-Fi or mobile connectivity. Try the operation after establishing connectivity.
- 1012 - Request failed.
- 1013 - Middleware returned invalid JSON string.
- 1014 - Request timed out.
- 1015 - Cannot find host.
- 1016 - Cannot connect to host.
- 1200 - SSL - Certificate related error codes.

The error message corresponding to each error code is captured in the *errmsg* parameter.

Important: The following information is applicable only for iOS and Android platforms.

The resulttable also contains the httpresponse key. The httpresponse contains a value as a table containing responsecode, url, headers, and cookies keys.

```
httpresponse:{
    "headers":{
        "xxxx":"Basic realm=\"Authentication required\"",
        "Content-Language":"en",
        "Content-Type":"text/html;charset=utf-8",
        "Date":"Tue, 29 Jul 2014 10:17:04 GMT",
        "Server":"Apache-Coyote/1.1",
        "Content-Length":"951"},
    "responsecode":401,
    "url":"http://x.x.x.x",
    "integrityStatus":"constants.HTTP_INTEGRITY_CHECK_SUCCESSFUL"}.
```

The `integrityStatus` field can be one of the following values.

- constants.HTTP_INTEGRITY_CHECK_NOT_DONE
- constants.HTTP_INTEGRITY_CHECK_SUCCESSFUL
- constants.HTTP_INTEGRITY_CHECK_FAILED

For more information, see HTTP Integrity Checking.

info [Object] - Optional

A JavaScript object consisting of key value pairs. If the info parameter is specified, it is passed to the callback function as a last parameter. If the info parameter is not specified, the callback function receives the info as null. The info object represents user data where in the application developers will pass it to the async API's, and the platform returns this info object to the corresponding async callback.This parameter helps developers remember the context when the methods are called in asynchronous fashion.Developers can define any custom keys and values within the info object based on the needs. Custom keys are not predefined keys with values.

Return Values

connHandle

The connHandle represents the handle to the underlying URL connection. The connHandle helps cancel the network call using the ***cancel*** API.

null

If the input parameters are invalid.

Implementation Details

To ensure security, all JavaScript functions within an application are executed in a single thread. There is no separate thread for the asynchronous service callback functions. The callback function for the service call executes serially in the thread, only after the execution of other JavaScript functions (queued up earlier).

Any subsequent user actions are queued if they occur during the execution of the callback function. Therefore, parallel execution of multiple asynchronous calls (even the operation of waiting for the response) cannot be guaranteed.

API Usage

Based on the application requirement and logic, a developer should decide whether to invoke a service synchronously or asynchronously.

UI Behavior

There is no noticeable difference in the user interface because the user can perform other actions while the asynchronous network call is in progress.

Rules and Restrictions

Do not use other service calls or functions that use/update the same data structure that is applied by the asynchronous network call. If you invoke more than one asynchronous service call that accesses the same data structure, the behavior of the application is unpredictable. You should handle such validations or error handling mechanisms in the callback function that is executed when an asynchronous service call is made.

> ***Note:*** You must sanitize the data such as user input or http response before setting them as http header values. If the data is not sanitized it can lead to various types of header manipulation attacks such as an HTTP response splitting attack, cross-site scripting, browser hijacking, cookie manipulation, and cross-user defacement.

Exceptions

1200 - Network Error.

Platform Availability

Available on all platforms.

Example

```
function callbackfunction(status, resulttable)
                      {
                      if(status == 400)
                      {
                      if(resulttable["opstatus"] == 0)
                      {
                      alert("opstatus is zero");
                      }
                      }
                      }

                      function startup()
                      {
                      try
                      {
                      var myhttpheaders={authkey:"myauthkey", authtoken:"myauthtoken"};
                      var inputParamTable=
                      {
                      appID:"SampleApp",
                      serviceID:"accountLogin",
                      channel:"rc",
                      httpheaders:myhttpheaders
                      };
                      var connHandle = kony.net.invokeServiceAsync(
                      "http://www.test.kony.com",
                      inputParamTable,
                      callbackfunction);

                      }
                      catch(err)
                      {
                      alert("Error"+err);
```

```
                                }
                    }
```

kony.net.invokeService

This API allows you to invoke a service synchronously with the specified parameters.

Syntax

```
kony.net.invokeService(url,inputParamTable,isblocking)
```

Input Parameters

url [String] - Mandatory

The URL containing the remote location from where the content is to be retrieved.

inputParamTable [None.]

It is the list of parameters that need to be passed to the remote service. The *parametermap* is a table that has the following key-value pairs:

- *appID* - Unique ID of the application
- *serviceID* - Unique ID of the service
- *httpheaders* [None.] - The *httpheaders* is an Optional key in the parameter table . You can use this key when you want to pass custom headers to the remote service.

  For example,

  ```
  httpheaders={authkey:"myauthkey",
  authtoken="myauthtoken"}
  ```

  Certain headers like User-Agent will be overridden by the underlying SDK and hence cannot be set using this key.

  If the remote service requires basic authentication, you can set the header as follows:

```
Authorization="Basic <base64encoding of the username:password>".
For example, httpheaders={Authorization=
"Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ=="}
```

Note: Make sure there is a space between `Basic` and the actual base64 encoded string.

The following is an example on how to use this parameter:

```
local myhttpheaders={authkey=
"myauthkey",authtoken="myauthtoken"}
local inputParamTable={appID="SampleApp",
serviceID="accountLogin",
httpheaders=myhttpheaders};
```

isblocking [Boolean] - Mandatory

Indicates if further actions on the user-interface needs to be blocked or not. If set to *true*, further clicks or actions on the form are blocked. If set to *false*, the user can to scroll through the user-interface but cannot perform any actions.

Return Values

resulttable

This table contains three values:

- *opstatus*
- *errcode*
- *errmsg.*

If the *opstatus* is 0, it indicates that the service call is a success while a non-zero value indicates a failure.

If the *opstatus* is a non-zero value, it is captured in *errcode*. The following are the possible error codes:

- *1000*- Unknown Error while connecting (If the platform cannot differentiate between the various kinds of network errors, the platform reports this error code by default)

- *1011* - Device has no WIFI or mobile connectivity. Please try the operation after establishing connectivity.
- *1012* - Request Failed
- *1013* - Middleware returned invalid JSON string
- *1014* - Request Timed out
- *1015* - Cannot find host
- *1016* - Cannot connect to host
- *1022* - Service call is canceled. You can customize the behavior of the application if a service call is canceled.

  Note: When you explicitly call **form.show**, then the iOS device will cancel the service call.

- *1200* - SSL - Certificate related error codes.

The error message corresponding to each error code is captured in the *errmsg* parameter.

API Usage

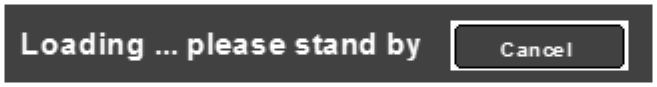
Based on the application requirement and logic, the developer should decide whether to invoke a service synchronously or asynchronously. For a synchronous service call, you should use the *net.invokeservice* API. For an asynchronous service call, you should use the *net.invokeserviceasync* API.

UI Behavior

When the synchronous service call is in progress, you can scroll through the user-interface but cannot perform any actions until the application receives the response for the synchronous service from the server. A progress indicator (something similar to) appears on the screen indicating to the user that a service call is in progress.

Note: Currently, the iPhone platform allow you to cancel a synchronous service call while it is in progress. When a service call is canceled, 1022 error code is thrown and you have to handle the logic that needs to be executed for that specific error code.

On iPhone, if the *isblocking* parameter is set to *true*, a message similar to the following message appears at the bottom of the screen allowing you to cancel the service call.


Loading ... please stand by

If the *isblocking* parameter is set to *false*, this message does not appear.

Note: On iPhone platform, the default value of *isblocking* parameter is *true*.

Rules and Restrictions

Use synchronous service calls only when you need the service output to be consumed immediately for other actions in the application. In other words, use synchronous service calls when you want the code in the application to be executed sequentially.

Platform Availability

Available on all platforms.

Example

```
local myhttpheaders={authkey="myauthkey",authtoken="myauthtoken"}
local params={appID="12345",serviceID="test1",httpheaders=myhttpheaders};
local myresulttable=net.invokeservice("http://test.konylabs.net",
params,true);
if (myresulttable.opstatus == 0)
then
// display some other form
else// display error
end;
```

39. Notifications

The notification system allows users to keep informed about relevant and timely events in your app, such as new chat messages from a friend or a calendar event. For example, if you have installed news app in your devices, you will receive timely news updates for that app.

Notifications are of two types: local notifications and push notifications.

Notifications are the means to keep the users informed about relevant events in your app such as a calendar event and a news update.

Notifications are of two types:

- Local Notifications: These notifications are scheduled in the app locally and delivered to the same device. Example, calendar event.
- Push Notifications: These notifications are sent from a remote server to the device on which the app is installed. For example, if you have installed a news app in your device, you will receive timely news updates for that app.

For more information on notification settings, refer Notification Settings.

Notifications comprise of the following sections:

- Local Notifications
- Remote Notifications
- Notification Settings

The Notifications API comprises of the following Namespaces and functions:

kony.localnotifications Namespace

Function	Description
`kony.localnotifications.cancel`	Cancels the specified notifications.
`kony.localnotifications.create`	Creates a local notification.
`kony.localnotifications.getNotifications`	Retrieves the pending local notifications.
`kony.localnotifications.setCallbacks`	Associates online and offline callbacks for local notifications.

kony.push Namespace

Function	Description
`kony.push.deRegister`	Allows an application on a device to deregister from Push Notifications.
`kony.push.register`	Allows you to register the application and the mobile device for Push Notifications.
`kony.push.setCallbacks`	You can specify the functions to be executed for Push Notification in an Object for this API.

kony.notificationsettings Namespace

Function	Description
`kony.notificationsettings.createAction`	Creates an action that can be used with category.
`kony.notificationsettings.createCategory`	Creates a category with a group of created actions.
`kony.notificationsettings.registerCategory`	Registers the created category with the application.

Function	Description
`kony.notificationsettings.pickTitleAndDescriptionFromPushPayload`	The Title and Description details of the payload are considered when the value is set to true.

Function	Description
`kony.notificationsettings.setShowBadge`	Enables or disables notification badges for push or local notifications that are only supported by Kony Framework.

The Notifications API enables you to set local notifications, register for push notifications and configure various notification settings.

To design a local notification, use the `kony.localnotifications.create` function. Use the `kony.localnotifications.setCallbacks` function to configure callbacks depending on the type of the notification that is triggered. If you want to see pending notifications, use the `kony.localnotifications.getNotifications` function. To cancel a local notification, use the `kony.localnotifications.cancel` function.

To enroll an application on your device for Push Notifications, use the `kony.push.register` function. Then configure the callbacks by using the `kony.push.setCallbacks` function. If you want to stop receiving push notifications for an application use the `kony.push.deRegister` function.

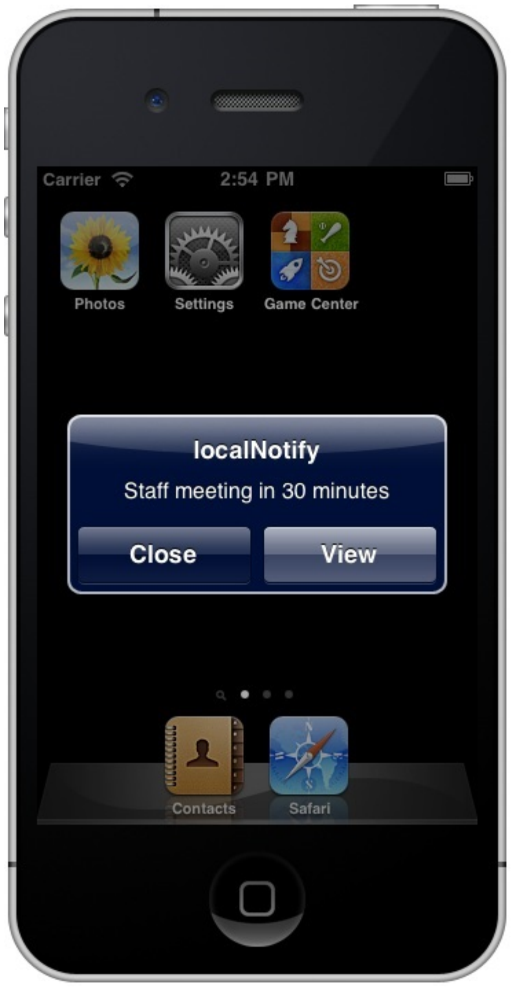
You can also configure various notification settings. Create an actions on the notification interface such as accept or decline a calendar event by using the `kony.notificationsettings.createAction` function. Then give a unique ID to the group of actions created by using the `kony.notificationsettings.createCategory` function. Register the created category with the application by using the `kony.notificationsettings.registerCategory` function. You can also display a badge on the push or local notifications by using the `kony.notificationsettings.setShowBadge` function.

39.1 Local Notifications

Local notifications are scheduled by an app and delivered on the same device. They are suited for apps with time-based behaviors, such as calendar events.

When you run your app on a device with Android OS 8.0 or above, Kony uses default channels that are mentioned in the `localnotificationconfig.xml` file.

Note: Each app on a device is limited to 64 scheduled local notifications. The system discards scheduled notifications exceeding this limit. Recurring notifications are treated as a single notification.



The local notification system consists of the following namespace and related functions:

- kony.localNotifications Namespace

39.1.1 Initializing Local Notifications

To initialize your app to receive local notifications, it must invoke the kony.localnotifications.setCallbacks function in the Notifications API and pass it the callback handler functions. The following sample code demonstrates how to initialize local notifications.

```
/
```

```
*********************************************************************
****************
 * Function:localNotCallBacks()
 * Description: Initializes local notifications.
 * Author: Kony

*********************************************************************
****************/
function localNotCallBacks() {
    try {
        kony.localnotifications.setCallbacks({
            "offlinenotification": offlinenotification,
            "onlinenotification": onlinenotification
        });
    } catch (err) {
        kony.print("Error Code " + err.errorCode + " Message " +
err.message);
    }
}

/* Notification callback handlers. These are invoked automatically
when their respective notifications are fired. */
function offlinenotification(notificationobject, actionid) {
    alert("offline notification callback inkvoked");
    alert("notification object is :" + JSON.stringify
(notificationobject) + " action id is " + actionid);
}

function onlinenotification(notificationobject, actionid) {
    alert("onlinenotification notification callback inkvoked");
    alert("notification object is :" + JSON.stringify
(notificationobject) + " action id is " + actionid);
}
```

39.1.2 Creating Local Notifications

Your app creates local notifications that are triggered then their corresponding events occur.

```
/
**********************************************************************
****************
 * Function:createLocalnotification()
 * Description: Creates local notifications.
 * Author: Kony

 **********************************************************************
****************/
function createLocalnotification() {
    var notificationId = "01";
    var date = "05 jan 2017 16:42:00 +0530";
    var format = "dd MMM yyyy HH:mm:ss Z";
    var message = "Local notification Received";
    var title = "Title";
    var categoryId = "calendar";

    kony.localnotifications.create({
        "id": notificationId,
        "dateTime": {
            "date": date,
            "format": format
        },
        "message": message,
        "title": title,
        "categoryId": categoryId,
        "pspConfig": {
            "badge": 1,
            "sound": kony.localnotifications.DEFAULT_SOUND
        }


    });


}
```

Note: The local notification API, kony.localnotification.create

- Can be called directly in Android devices.
- In iOS devices you must first call kony.notificationsettings.registerCategory API in order to call the create API.

39.1.3 Canceling Local Notifications

To stop receiving local notifications, your all calls the kony.localnotifications.cancel function, as shown in the code sample below.

```
/
*********************************************************************
***************
 * Function:cancelLocalnotifications()
 * Description: Cancels local notifications.
 * Author: Kony

 *********************************************************************
***************/
function cancelLocalnotifications() {
    notificationIdArray = [];
    notificationIdArray.push("01");
    kony.localnotifications.cancel(notificationIdArray);
}
```

39.1.4 Creating Actions and Categories

Your app can create categories of notifications and set actions for those categories. To do so, it call the kony.notificationsettings.registerCategory function, as illustrated in the following sample code.

```
/
**********************************************************************
****************
 * Function:registerActions()
 * Description: Creates Actions and a Category.
 * Author: Kony

 **********************************************************************
****************/
function registerActions() {
    var accept = kony.notificationsettings.createAction({
        "id": "Accept",
        "label": "Accept",
        "pspConfig": {
            "authenticationRequired": true,
            "destructive": true,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var reject = kony.notificationsettings.createAction({
        "id": "Reject",
        "label": "Reject",
        "pspConfig": {
            "authenticationRequired": false,
            "destructive": false,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var decline = kony.notificationsettings.createAction({
        "id": "Decline",
        "label": "Decline",
        "pspConfig": {
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_BACKWARDS,
            "authenticationRequired": true,
            "destructive": false,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });


    var defaultActionContextArr = [accept, reject, decline];
    var minimalActionContextArr = [accept, reject];

    var categoryObj = kony.notificationsettings.createCategory({
        "categoryId": "invitation",
        "actions": defaultActionContextArr,
        "pspConfig": {
            "minimalActions": minimalActionContextArr
        }
    });


    //Using kony.notificationsettings.registerCategory
    var categoryArr = [categoryObj];

    var registerCategory = kony.notificationsettings.registerCategory
({
        "categories": categoryArr,
        "pspConfig": {
            "types": [0, 1, 2]
        }
    });

}
```

39.1.5 Important Consideration for Android Platform

The default settings are located in the `localnotificationconfig.xml` file (available after the application is built) in the 'dist{APP-ID}\res\values' location. You can modify these settings by using a regular expression replacement task that is written in the androidprecompiletask.xml file.

For example, if you want to update the value of the 'notify_local_msg_notifications_count' key from '1' to '5 ', you must use the following code snippet, which replaces the value directly in the file.

```
<replace file="$
{app.dir}/res/values/localnotificationconfig.xml"
token = "<string name="notify_local_msg_notifications_count">1"
value = "<string name="notify_local_msg_notifications_count">5"/>
```

Similarly, you can configure other keys by using the *androidprecompiletask.xml* file.

39.1.6 Modifying localnotificationconfig.xml File in Android Platform

The default behavior of the local notification message can be customized by modifying the `localnotificationconfig.xml` file. The `localnotificationconfig.xml` file contains key value pairs that allow applications to configure individual keys to override the default behavior.

You can customize the notification using the keys provided in `localnotificationconfig.xml`.

The table below shows a list of key value pairs, each with a brief description.

Notification ID	Default Value	Description
notify_local_msg	true	Enable or disable new local message notifications. If false, no status bar notification is shown when application is running in background.
notify_local_msg_channel_title	Local Notifications	Channel title for local notifications.
notify_local_msg_channel_desc	All local notifications will be displayed under this category	Channel description for local notifications.
notify_local_msg_notifications_count	1	By default, only the latest local message notification is kept in the status bar. Use this option to control the number of messages to be stored. The maximum limit allowed in Android is 50. ***Note:*** After exceeding the storage limit, older messages are replaced with newer messages. However, sometimes older notifications are replaced with newer messages even before the storage reaches the specified count (this happens when the app is closed by the Android system).

Notification ID	Default Value	Description
notify_local_msg_icon	icon	Specify icon resource without extension. For example, set a logo to use an image for the message icon for which the file name is logo.png. The default is icon.png.
notify_local_msg_sound	true	Enable or disable sound for local message notification.
notify_local_msg_vibrate	true	Enable or disable vibrate for local message notification.
notify_local_msg_lights	true	Enable or disable lights for local message notification.
notify_local_msg_clear	true	Indicates if a user can clear the unread local message notification.

39.1.7 kony.localnotifications Namespace

The `kony.localnotifications` namespace provides your app the ability to create and receive local, on-device notifications that do not rely on off-device information coming across a network. The kony.localnotifications namespace contains the following API elements.

39.1.7.1 Functions

The kony.localnotifications namespace contains the following functions.

kony.localnotifications.cancel

Cancels the specified notifications.

Syntax

```
kony.localnotifications.cancel(notificationId)
```

Input Parameters

Parameter	Description
notificationId	An array of notification IDs that selects the notifications to cancel.

Example

```
/
**********************************************************************************
*******
* Function:cancelLocalnotifications()
* Description: function is used to cancel local notifications.
* Author: Kony

 **********************************************************************************
*******/
function cancelLocalnotifications(){
    notificationIdArray = [];
    notificationIdArray.push("01");
    kony.localnotifications.cancel(notificationIdArray);
 }
```

Return Values

None

Platform Availability

Available on iOS and Android platforms.

kony.localnotifications.create

Creates a local notification.

Syntax

```
kony.localnotifications.create(
    notificationId,
    datetime,
    message,
    title,
    categoryId,
    pspConfig)
```

Input Parameters

Parameter	Description
notificationId	A string that specifies a unique ID for the notification.
datetime	A string that specifies the date and time when the notification must be triggered. Must follow the unicode date, time, and Timezone pattern.
message	A string that specifies the message for the notification.
title	A string that specifies the title for the notification.
categoryId	A string that specifies the category ID to associate this local notification with, or null in case no actions are to be displayed.

Parameter	Description
pspConfig	An optional JavaScript object containing key-value pairs that set the platform-specific options. Used on iOS platform only. The following keys are supported. • badge: An optional number that displays the number of notifications on the app icon. • sound: An optional string that specifies the sound to play. For more information, see the **Remarks** section below.

Example

```
/
*****************************************************************************
*******
* Function:createLocalnotification()
* Description: Creates local notifications.
* Author: Kony

 *****************************************************************************
*******/
function createLocalnotification(){
    var notificationId = "01";
    var date = "05 jan 2017 16:42:00 +0530";
    var format = "dd MMM yyyy HH:mm:ss Z";
    var message = "Local notification Received";
    var title = "Title";
    var categoryId ="calendar";

    kony.localnotifications.create({
        "id": notificationId,
        "dateTime": {
            "date": date,
            "format": format
        },
        "message": message,
        "title": title,
        "categoryId": categoryId,
        "pspConfig":{
            "badge":1,
            "sound": kony.localnotifications.DEFAULT_SOUND
        }

    });

}
```

Return Values

None

Remarks

To play a sound when a notification arrives, your app must specify the sound using the "sound" key in the JavaScript object contained in the *pspConfig* parameter. Your app must assign a sound file from the app's main bundle. If the sound file is not available or the provided file name is incorrect, it will play a default sound from the system (`kony.localnotifications.DEFAULT_SOUND`). To know the see file formats, click here

Platform Availability

Available on

- iOS (The kony.notificationsettings.registerCategory API must be called before the create API for it to work)
- Android

kony.localnotifications.getNotifications

Retrieves the pending local notifications.

Syntax

```
kony.localnotifications.getNotifications(notificationObjects)
```

Input Parameters

Parameter	Description
notificationObjects	An optional array of notification objects if there are pending notifications, or an empty array if there are not. This parameter is only used on iOS.

Example

```
// iOS Example. The callback function is not used on Android.
function callback(arrayOfNotificationObjects)
{
    // Your logic
    if(arrayOfNotificationObjects.length == 0)
    {
        kony.print("No pending notifications");;
    }
}

kony.localnotifications.getNotifications(callback);
```

Return Values

None.

Platform Availability

Available on iOS and Android platforms.

---

kony.localnotifications.setCallbacks

Associates online and offline callbacks for local notifications.

Syntax

```
kony.localnotifications.setCallbacks(onlinenotification,offlinenotification)
```

Input Parameters

Parameter	Description
onlinenotification	A callback function that is invoked when the online notification is triggered. For more information, see the **Remarks** section below.
offlineNotification	A callback function that is invoked when the offline notification is triggered. For more information, see the **Remarks** section below.

Example

```
/
*****************************************************************************
*******
* Function:localNotCallBacks()
* Description: Initializes local notifications.
* Author: Kony

 *****************************************************************************
*******/
function localNotCallBacks() {
    try {
        kony.localnotifications.setCallbacks({
            "offlinenotification": offlinenotification,
            "onlinenotification": onlinenotification
    });
    } catch (err) {
        kony.print("Error Code " + err.errorCode + " Message " + err.message);
    }
}

/* Notification callback handlers. These are invoked automatically when their
respective notifications are fired. */
function offlinenotification(notificationobject, actionid ){
    alert("offline notification callback inkvoked");
    alert("notification object is :"+JSON.stringify(notificationobject) +"
action id is "+actionid);
}

function onlinenotification(notificationobject,actionid){
    alert("onlinenotification notification callback inkvoked");
    alert("notification object is :"+JSON.stringify(notificationobject)+"
action id is "+actionid);
}
```

Return Values

None

Remarks

The callbacks that this function sets are invoked when notifications are raised by the underlying system. The online notification callback is involved when the application is running. The offline callbacks are invoked when the application is not running. This function should be called inside the `post app init` event handler when your app starts.

The callback function in the *onlinenotification* parameter must have the following signature.

```
onlineNotificationCallback(actionID,notificationObject);
```

where *actionId* is a unique ID for the action, and *notificationobject* is a JavaScript object that contains platform-specific notification information.

The callback function in the *offlinenotification* parameter must have the following signature.

```
offlineNotificationCallback(actionID,notificationObject);
```

where *actionId* is a unique ID for the action, and *notificationobject* is a JavaScript object that contains platform-specific notification information.

Platform Availability

Available on iOS and Android platforms.

39.1.7.2 Properties

The Local Notifications API contains the following properties:

backgroundColor

An optional String value (for example, ffffff for white) that fires a notification with the provided background color for the notification icon.

Syntax

```
backgroundColor
```

Type

String

Remarks

- You can configure the background color of the notification icon at the app level in localnotificationconfig.xml and pushconfig.xml.
-

notify_local_msg_background_color	empty	Notification icon background color for local message notification.
notify_push_msg_background_color	empty	Notification icon background color for push message notification.

Example

```
pspconfig:{
"backgroundColor" :  "ffffff"
}
```

Platform Availability

- Android

priority

An optional constant integer value that fires a notification with the specified priority.

Syntax

```
priority
```

Type

int

Supported Values

- constants.NOTIFICATION_PRIORITY_MIN: This is the lowest notification priority. For this priority value, the notifications may not be shown to the user, except for special circumstances such as detailed notification logs.
- constants.NOTIFICATION_PRIORITY_MAX: This is the highest notification priority. This priority value should be used for your application's most important notifications that require the user's prompt attention or input.
- constants.NOTIFICATION_PRIORITY_LOW: This is a lower notification priority. This priority value should be used for those notifications that are comparatively less important. The UI may choose to display these items in a smaller font or at a different position in the list, as compared to your app's PRIORITY_DEFAULT notifications.
- constants.NOTIFICATION_PRIORITY_HIGH: This is a higher notification priority. This priority value should be used for more important notifications or alerts. The UI may choose to show these items in a larger font or at a different position in notification lists, as compared to your app's PRIORITY_DEFAULT notifications.
- constants.NOTIFICATION_PRIORITY_DEFAULT: This is the default notification priority. If your application does not prioritize its own notifications, use this value for all notifications. This priority value is available from Android O and later. The priority is applied to the Notification Channel. The priority of an existing channel is changed only if the new priority is lower than the current value and the user has not altered any settings on this channel.
  You can configure the priority of notifications at the app level in localnotificationconfig.xml and pushconfig.xml.
-

notify_local_msg_priority	empty	Priority for local message notification.
notify_push_msg_priority	empty	Priority for push message notification.

Example

```
pspconfig:{
"priority" :  constants.NOTIFICATION_PRIORITY_DEFAULT
 }
```

Platform Availability

- Android

repeats

An optional Boolean value that displays a notification repeatedly after a specific time interval. You must use the **repeats** function along with a **timeInterval** value.

Syntax

```
repeats
```

Type

Boolean

Remarks

- You can only set **repeats** to true if **timeInterval** is equal to or greater than 60 seconds.
- If **repeats** is set to true, the notification gets repeated multiple times.
- If **repeats** is set to false, the notification is fired just once.
- By default, the value of **repeats** is false.

Example

Repeat a notification after every 300 seconds.

```
pspconfig:{
“timeInterval”: 300
“repeats”: true
}
```

Platform Availability

- iOS

timeInterval

An optional constant integer value that fires a notification at a time relative to the current time.

Syntax

```
timeInterval
```

Type

int

Remarks

- You should always specify the timeInterval value in seconds.
- If values for both timeInterval and dateTime are provided, timeInterval is given higher precedence.

Example

Fire a notification in the next 300 seconds.

```
pspconfig:{
“timeInterval”: 300
}
```

Platform Availability

- iOS

vibrate

An optional Boolean value that fires a notification with or without the vibration of the device.

Syntax

```
vibrate
```

Type

Boolean

Example

```
pspconfig:{
“vibrate”: ”true”
}
```

Platform Availability

- Android

visibility

An optional constant integer value that fires a notification with the given visibility.

Syntax

```
visibility
```

Type

int

Supported Values

- constants.NOTIFICATION_VISIBILITY_PRIVATE: This notification visibility value does not reveal any part of the notification on the secure lock-screen of a user's device.
- constants.NOTIFICATION_VISIBILITY_PUBLIC: This notification visibility value displays the entire notification on all the lock-screens of a user's device.
- constants.NOTIFICATION_VISIBILITY_SECRET: This notification visibility value displays the notification on all lock-screens of a user's device, but conceals sensitive or private information on secure lock-screens.
- You can configure the visibility of notifications at the app level in localnotificationconfig.xml and pushconfig.xml.

-

notify_local_msg_visibility	empty	Visibility for local message notification.
notify_push_msg_visibility	empty	Visibility for push message notification.

Example

```
pspconfig:{
"visibility" :  constants.NOTIFICATION_VISIBILITY_PRIVATE
}
```

Platform Availability

- Android

39.2 Push Notifications

Push Notifications is a mechanism using which you can initiate a message delivery to a device without receiving a request. Push notifications are sent by Kony Fabric Engagement Services to the Apple Push Notification Service (APNS), which pushes the notification to devices.

Note: Push Notifications will not work in the simulator.

The Notification Payload For Push Notification

Each push notification includes a payload In iOS 8 and later, the maximum size allowed for a notification payload is 2 kilobytes. Apple Push Notification service refuses any notification that exceeds this limit. (Prior to iOS 8, the maximum payload size is 256 bytes.)

Note: Delivery of notifications is a “best effort”, not guaranteed. It is not intended to deliver data to your app.

When you run your app on a device with Android OS 8.0 or above, Kony uses default channels that are mentioned in the `pushconfig.xml` file.

For a more hands-on approach in the understanding and implementation of Push Notifications API, you can import and preview the Events app by using Kony Visualizer.

39.2.1 How it Works

The following are the primary processes involved in an application being setup for Push Notifications:

1. Enabling Push Notifications: When an application needs to use the Push Notifications service, it uses the kony.push.setCallbacks function and specifies the functions to be executed for Push Notifications.
   The application then uses the kony.push.register API to register the application and the device with the corresponding Push Notification Service provider of the device OS.

2. Sending Push Notifications: If the registration is successful, the application can receive Push Notifications on the device from the third-party application server.

3. Receiving a Message: The third-party application server sends a Push Notification to the application and the device extracts the data from the message and passes it to the application.

4. Disabling Push Notifications: The application can deregister from Push Notification service by using the kony.push.deRegister function.

39.2.1.1 Platforms Supporting Push Notifications

Push Notifications are currently supported only by Android, iPhone, and Windows Phone platforms.

For Push Notifications to work, the application and the device must first register with the corresponding Push Notification Service provider of the device os.

The following is the list of platforms and their corresponding Push Notifications Service providers they must register with:

Platform	Push Notifications Service
Android/Android Tablet	Google Cloud Messaging (GCM) Deprecated:Android Cloud to Device Messaging (C2DM)
iPhone	Apple Push Notification Service (APNS)
Windows Phone	Microsoft Push Notification Service (MPNS)
Desktop Web	Google Firebase Cloud Messaging (FCM)

39.2.1.2 Enabling Push Notifications

To enable Push Notifications for an application do the following:

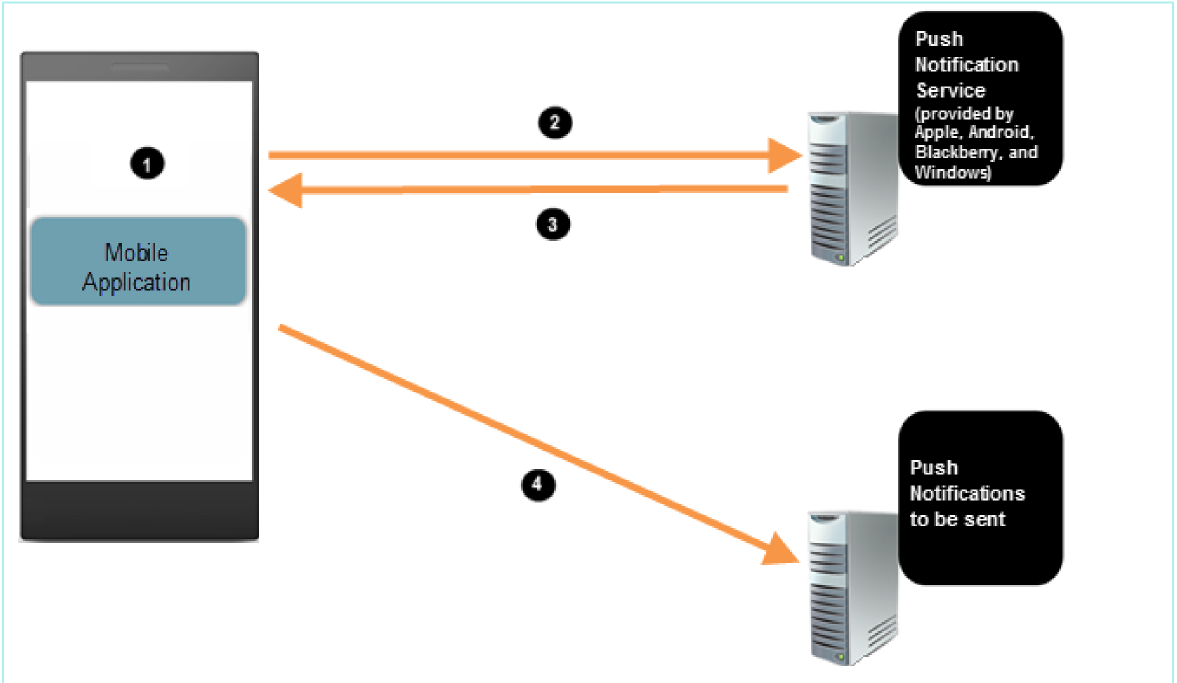
1. The application uses the kony.push.setCallbacks function and specifies the functions to be executed for Push Notifications.
2. The application uses the kony.push.register API to register for Push Notifications with the corresponding Push Notification Service provider of the device OS.
3. If the registration is successful, the Push Notification Service provider returns a unique identifier to the device.

   The following is the list of Push Notification Service providers and the unique identifier they return:

Push Notifications Service	Returned Unique Identifier
Google Cloud Messaging(GCM) Deprecated: Android Cloud to Device Messaging (C2DM) Firebase Cloud Messaging (FCM)	Registration ID
Apple Push Notification Service (APNS)	DeviceToken
Microsoft Push Notification Service (MPNS)	Unique URL

4. The application sends the identifier to the third-party application server. The third-party application server can use this identifier to send Push Notifications to it.

The following image illustrates the steps involved in enabling Push Notifications for an application on a device:


1
Mobile Application
2
3
4
Push Notification Service (provided by Apple, Android, Blackberry, and Windows)
Push Notifications to be sent

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

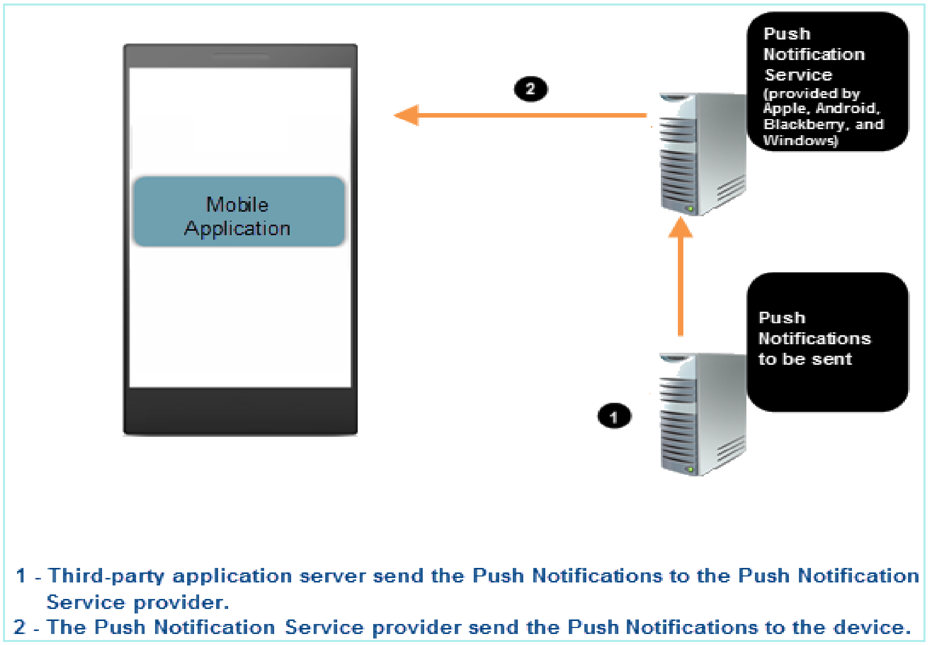
39.2.1.3 Sending Push Notifications

The following are the steps involved in a third-party application server sending Push Notifications to an application on a device:

1. The third-party application server uses the unique identifier (sent by the application) in sending the Push Notifications to the Push Notification Service provider of the device.

2. The Push Notification Service provider then sends the Push Notification message to the application on the device.

The following image illustrates the steps involved in sending Push Notifications to an application on a device:


Push Notification Service (provided by Apple, Android, Blackberry, and Windows)
2
Mobile Application
Push Notifications to be sent
1
1 - Third-party application server send the Push Notifications to the Push Notification Service provider.
2 - The Push Notification Service provider send the Push Notifications to the device.

39.2.1.4 Receiving a Message

When an application on a device receives a message, the following sequence of events occur:

1. The device receives the incoming Push Notification and extracts the data from the message payload.
2. The device passes the data to the application.
3. The application processes the data and displays it to the user.

39.2.1.5 Disabling Push Notifications

You can use the kony.push.deRegister API and deregister the application and the device from the Push Notifications service.

39.2.1.6 Silent Push Notifications

Push notifications can occur in the background without triggering a notification that is visible to the app's user through its UI. These are known as silent notifications and are often used to download data in the background while the user continues with other tasks.

To enable your app to receive silent notifications, your app must call the kony.push.setCallbacks function and pass it the `onlinenotification` callback function that is invoked whenever a silent notification arrives from a Kony Fabric server. The `onlinenotification` function then checks the payload from the server to see if it has received a silent notification. The payload has the following format.

```

{
    "data": {
        "mid": "830400256629909033 8",
        "content-available": “1”
    }
}
```

Note in particular that the data contains the key `"content-available"`, which is set to `"1"`. This specifies to your app that a silent notification has been received and that it should process the notification silently. Therefore, it should not display UI alerts to the user or put itself into foreground mode if it is running in the background. Instead, the callback function your app sets through the `onlinenotification` parameter of the kony.localnotifications.setCallbacks function is invoked automatically so that it can process the notification without disturbing the user. The callback function is invoked irrespective of the current state of the app. So the app can be running, not running, in the foreground, or in the background and still receive the silent notification.

To enable silent push notifications, your app must be able to receive push messages from the server. It must also call the kony.localnotifications.setCallbacks function when processing either the pre-appinit or post-appinit events.

39.2.2 API Functions for Push Notifications

Kony Platform provides APIs that you can use to enable Push Notifications for an application on a device and also an API to deregister from the Push Notifications service.

The following are the APIs for Push Notifications:

1. kony.push.setCallbacks
2. kony.push.register
3. kony.push.deRegister

Note: The following are the generic error codes for Push Notification APIs:

Error Code	Error Message
1400	Invalid number of arguments.
1401	Illegal arguments.
1402	Unable to connect to push service - PNS service is not available
1406	Platform-specific issue. Full details are available in the errormessage. For example, received payload but payload is in incorrect format.

Important: Use *Error Codes* to refer and do not rely on *Error Messages* as each message may differ from platform to platform.

39.2.3 Important Considerations for Android

The following are the important considerations you must be aware for Android platform:

- Before you build the application for Android, navigate to the Project Properties of the application and navigate to *Native App* -> *Android* tab, and select the *GCM* under the *Push Notification* section. Selecting this option will copy the required Push Notification libraries into the application during build time."
- For C2DM to GCM conversion refer, http://developer.android.com/guide/google/gcm/c2dm.html
- Google may occasionally refresh the Registration ID. Hence, you must design the application to update the third-party Application server with the new ID.

  If the Kony Android platform receives a new registration ID, the following takes place:

  - If the application is running in the foreground or the background - *onsuccessfulregistration* function is called.
  - If the application is not running - a status bar notification is displayed. If you select the notification, the application is launched and *onsuccessfulregistration* function is called.
- If the Kony Android platform receives a new Push notification message, the following takes place:
  - If the application is running in the foreground:
    - GCM - Online callback is triggered without any notification.
    - FCM - Online callback is triggered without any notification.

- If the application is running in the background:
    - GCM - A status bar notification is displayed. If you click the notification, the application is brought to the foreground and its *onlinenotification* function is called.
    - FCM - A status bar notification is displayed. If you click the notification, the application is brought to the foreground and its *onlinenotification* function is called.
- If the application is not running:
    - GCM - A status bar notification is displayed. If you click the notification, the application is launched and its *offlinenotification* function is called.
    - FCM - Any of the following can occur:
        - If the payload contains a notification key (for example, when the 'Enable GCM v3.0 Payload for Android' check box is selected in KMS console), a status bar notification is displayed by the system. If you click the notification, the application is launched and its *onlinenotification* function is called.
        - If the payload does not contain a notification key (for example, when the 'Enable GCM v3.0 Payload for Android' check box is not selected in KMS console), a status bar notification is displayed by Framework. If you click the notification, the application is launched and its *offlinenotification* function is called.

- The status bar notification is displayed by the platform with the default settings.
  The default settings are in the *pushconfig.xml* file (available after the application is built) in the 'dist{APP-ID}\res\values' location. You can modify these settings by using a regular expression replacement task that is written in the *androidprecompiletask.xml* file.
  For example, if you want to update the value of the 'notify_push_msg_title_keys' key to 'title,' you must use the following code snippet, which replaces the value
  directly in the file.

```
<replace file="${app.dir} /res/values/pushconfig.xml"
token="<string name="notify_push_msg_title_keys">"
value="<string name="notify_push_msg_title_keys">title"/>
```

  Similarly, you can configure other keys by using the *androidprecompiletask.xml* file.

- To customize GCM broadcast receiver, refer Kony Visualizer User Guide.

39.2.4 Important Considerations for Desktop Web

- From Project Settings of your Visualizer project, you must go to the Desktop Web tab and perform the following actions:
  - Select the **Enable PWA** checkbox.
  - Select the **Enable push notifications** check box.
  - Specify your messagingSenderId in the **FCM Sender ID** field.
- Push notifications will work only when you build the app in Release mode.
- Push notifications work only in HTTPS environment, which means that you must publish the app in HTTPS server.

> *Important:* Desktop web push notifications are not supported on localhost.

- The Push notifications feature for the Desktop Web channel is supported from V8 SP4 onwards.
- Push notifications for the Desktop Web channel is not supported in:
  - Mac Safari
  - All web browsers in iOS
- If a web browser has the support for service worker, only then push notifications will work in that browser.
- If the Kony Desktop Web platform receives a new Push notification message, the following takes place:

- If the application is running in the foreground, an online callback is triggered without any notification.
- If the application is running in the background or if the application is not running, a status bar notification is displayed.

39.2.5 Modifying pushconfig.xml File in Android Platform

The default behavior of the push notification message can be customized by modifying the `pushconfig.xml` file. The `pushconfig.xml` file contains key value pairs that allow applications to configure individual keys to override the default behavior.

You can customize the notification using the keys provided in `pushconfig.xml`.

The table below shows a list of key value pairs, each with a brief description.

Notification ID	Default Value	Description
notify_push_msg_channel_title	Push Notifications	Channel title for push notifications.
notify_push_msg_channel_desc	All push notifications will be displayed under this category	Channel description for push notifications.
use_same_channel_details_for_local_notifications	false	If configured to true, same channel details will be used for local notifications.

39.2.6 Reference Links

You can get detailed information regarding the implementation of Push Notifications on various platforms, by visiting the following links:

- **Desktop Web**: https://firebase.google.com/docs/cloud-messaging/js/client
- **Windows Phone**: http://msdn.microsoft.com/en-us/library/ff402537(VS.92).aspx

39.2.7 kony.push Namespace

The kony.push namespace provides the following API elements.

39.2.7.1 Functions

The kony.push namespace contains the following functions.

39.2.7.2 kony.push.deRegister

This API allows an application on a device to deregister from Push Notifications. This API takes an empty object as the parameter.

If the deregistration is successful, the platform invokes the *onsuccessfulderegistration* function. If the deregistration is a failure, the platform invokes the *onfailurederegistration* function.

Syntax

```
kony.push.deRegister(emptyObject)
```

Input Parameters

Parameter	Description
emptyObject [Object]- Mandatory	Is an empty Object.

Example

To deregister from Push Notifications, use the following code snippet:

```
/**
 * Name                    :       deRegisterMyPush
 * Author          :       Kony
 * Purpose         :       This function helps to de-register from push
notifications.
 **/
function deRegisterMyPush() {
    var myhash = new Hash();
    kony.push.deRegister(myhash);
}
```

Return Values

None

Exceptions

The following error codes are applicable to onfailureregistration and onfailurederegistration functions:

- 1402 - Unable to connect to push service - PNS service is not available
- 1403 - Registration failed by PNS - Account related or device restrictions (thrown by push.register() API during an onfailureregistration callback).
- 1404 - Deregistration failed - Unable to close channel or PNS internal error while deregistration (thrown by push.deregister() API during an onfailurederegistration callback).
- 1405 - Duplicate Registration - Actual registration details should be provided? (thrown by kony. push.register() API during an onfailureregistration callback)
- 1406 - Platform-specific issue. Full details are available in the errormessage. For example, received payload but payload is in incorrect format.

- An error is thrown if input type is invalid or does not follow the expected structure.
    - 102-Invalid input error

API Usage

You must use this API to deregister the application and the device from the Push Notifications service.

Platform Availability

- Android
- iOS
- Windows
- Desktop Web

39.2.7.3 kony.push.register

This API allows you to register the application and the mobile device for Push Notifications. This API takes an object as a parameter (the values in the table differ across platforms).

If the registration is successful, the platform invokes the *onsuccessfulregistration* function. If the registration is a failure, the platform invokes the *onfailureregistration* function.

Syntax

```
kony.push.register(configObject)(iOS/Android/Desktop Web/Windows)
```

Input Parameters

Parameter	Description
configObject [Array] - Mandatory (iOS)	The Array for iOS must contain **any** or **all** of the following: • 0 - Specifies the Notification type as Badge. • 1 - Specifies the Notification type as Sound. • 2 - Specifies the Notification type as Alert.
configObject [Object] - Mandatory (Android)	A Hash table for Android must contain the following key value pairs: • **senderid** - Specifies the project ID of the account registered to use FCM. The projectID is the ID for a created API project. For more information on FCM , refer here.

Parameter	Description
configObject [Object] - Mandatory (Desktop Web)	A Hash table for Desktop Web must contain the following key value pairs: • **senderid** - Specifies the sender ID of the registered FCM application. For more information on FCM, click here. • **publicKey** - This is the value that is generated under the key-pair tab for the registered FCM application.

Parameter	Description
configObject [Object]- Mandatory (Windows)	The Array for Windows must contain the following key value pairs: • **enableraw**: This is a Boolean parameter and it specifies if the device must receive raw messages or not. The default value is *true* (you can receive raw messages). If you do not want the device to receive raw messages, set the value to *false*. • **enabletile**: This is a Boolean parameter and it specifies if the device must receive tile messages or not. The default value is *true* (you can receive tile messages). If you do not want the device to receive tile messages, set the value to *false*. • **enabletoast**: This is a Boolean parameter and it specifies if the

Example

The following are the examples of this API on various platforms:

Android

The following code snippet uses the *senderid* to register for Push Notifications:

```
/**
 * Name                 :       registerMyAndroidPush
 * Author       :       Kony
 * Purpose      :       This function registers the senderID on the Google cloud.
 **/

function registerMyAndroidPush() {
    var config = {
        senderid: "4815162342"
    };
    kony.push.register(config);
}
```

iOS

The following code snippet uses the *notification types* to register for Push Notifications:

```
/**
 * Name                 :       registeriPhonePush
 * Author       :       Kony
 * Purpose      :       This function register the senderID on the Google cloud.
 **/
function registeriPhonePush() {
    var config = [0, 1, 2];
    kony.push.register(config);
}
```

Desktop Web

```
/**
 * Name                    :       registerDesktopWebPush
 * Author       :       Kony
 * Purpose      :       This function registers the senderID on the Google cloud.
 **/
function registerDesktopWebPush() {
    var configRegister = {
        messagingSenderId: "88888888888",
        publicKey: "BKf0xO2plAvCNtblOcgeTeyMleGcOnhetKe3Birx4aqhR-Wh3-
D8Px7kPYa1YyMBIECg_tKz7droMbGNjFwyUMw"
    };
    kony.push.register(configRegister);
}
```

Windows

```
/**
 * Name                    :       registerWindowsPush
 * Author       :       Kony
 * Purpose      :       This function registers the senderID on the Google cloud.
 **/

function registerWindowsPush() {
    var config = {
        enableraw: true,
        enabletile: true,
        enabletoast: true,
    };
    kony.push.register(config);
}
```

Note: Only network URLs are allowed. MPNS allows only images from trusted domains (registered to MPNS) to be displayed to avoid image policies. There is no such restriction on local images.

Exceptions

The following error codes are applicable to onfailureregistration and onfailurederegistration functions:

- 1402 - Unable to connect to push service - PNS service is not available
- 1403 - Registration failed by PNS - Account related or device restrictions (thrown by push.register() API during an onfailureregistration callback).
- 1405 - Duplicate Registration - Actual registration details should be provided? (thrown by kony. push.register() API during an onfailureregistration callback)
- 1406 - Platform-specific issue. Full details are available in the errormessage. For example, received payload but payload is in incorrect format.
- An error is thrown if input type is invalid or does not follow the expected structure.
    - 102-Invalid input error

API Usage

You must use this API to register the application and the device for Push Notifications with the Push Notifications service provider.

Platform Availability

- Android
- iOS
- Windows
- Desktop Web

39.2.7.4 kony.push.setCallbacks

When an application on a device registers or deregisters for Push Notifications, or if the device receives a notification, the device executes the function of your choice.
You can specify the functions to be executed for Push Notification in an object of the kony.push.setCallbacks API.

Syntax

```
kony.push.setCallbacks(Object)
```

Input Parameters

Object [Object] - Mandatory

A Hash table has the following key-value pairs:

Key	Description
onsuccessfulregistration [Function] - Mandatory	Specifies the function to be executed when the Push Notifications registration is successful.
onfailureregistration [Function] - Mandatory	Specifies the function to be executed when the Push Notifications registration fails.

Key	Description
onlinenotification[Function]	Mandatory for iOS, Android, and Windows. Specifies the function to be executed when the device receives a message when the application is running. The message types for which this function is executed varies. The message type on various platforms are as follows: **iOS:** Sounds, Alerts, or Badges; **Windows:** raw or toast.
offlinenotification [Function]	Mandatory for iOS, Android, and Windows. Specifies the function to be executed when the device receives a message when the application is not running. The message types for which this function is executed varies. The message type for iOS is as follows: **iOS:** Sounds, Alerts, or Badges

Key	Description
onsuccessfulderegistration [Function] - Mandatory	Specifies the function to be executed when the Push Notification deregistration is successful.
onfailurederegistration [Function]	Mandatory for Android, and Windows. Optional for iOS. Specifies the function to be executed when the Push Notification deregistration is a failure.
offlinecallback [Function] - Mandatory.	Specifies the function to be executed when an action is invoked. • payload • actionId [String] - A unique id to identify the action.

Example

```
/**
 * Name                    :       regSuccessAndroidCallback
 * Author       :       Kony
 * Purpose      :       This function is the callback for the successful
registration of the device to the FCM server. It returns the callerID.
 **/

function regSuccessAndroidCallback(regId) {
    kony.print("** JavaScript regSuccessCallback() called **");
    kony.print(regId);
}

/**
 * Name                :       regFailureAndroidCallback
 * Author      :       Kony
 * Purpose     :       This function is the callback for the registration
failure to the FCM server.
 **/

function regFailureAndroidCallback(errormsg) {
    kony.print("* JavaScript regFailureCallback() called *");
    kony.print(errormsg.failurereason);
kony.print(errormsg.description);
}

/**
 * Name                :       onlinePushNotificationAndroidCallback
 * Author      :       Kony
 * Purpose     :       This function is the callback for the registration
failure to the FCM server.
 **/

function onlinePushNotificationAndroidCallback(msg) {
    kony.print("* JavaScript onlinePushNotificationCallback() called
*");
    kony.print(msg);
    kony.print(msg.message);
}

/**
 * Name                :       offlinePushNotificationAndroidCallback
 * Author       :       Kony
 * Purpose      :       This function is the callback for the received push msg
event while offline.
 **/

function offlinePushNotificationAndroidCallback(msg) {
    kony.print("* JavaScript offlinePushNotificationCallback() called
*");
    kony.print(msg);
}

/**
 * Name                 :       unregSuccessAndroidCallback
 * Author       :       Kony
 * Purpose      :       This is the callback for the successful unregistration
from the FCM server.
 **/

function unregSuccessAndroidCallback() {
    kony.print("* JavaScript unregSuccessCallback() called *");
}

/**
 * Name                 :       unregFailureAndroidCallback
 * Author       :       Kony
 * Purpose      :       This is the callback for the unsuccessful deregistration
from the FCM server.
 **/

function unregFailureAndroidCallback(errormsg) {
    kony.print("* JavaScript unregFailureCallback() called *");
    kony.print(errormsg.errorcode);
    kony.print(errormsg.errormessage);
}



/**
 * Name                    :       callbackAndroidSetCallbacks
 * Author       :       Kony
 * Purpose      :       This function sets the callback for registration,
deregistration, and pushnotification events.
 **/

function callbackAndroidSetCallbacks() {
    kony.push.setCallbacks({
        onsuccessfulregistration: regSuccessAndroidCallback,
        onfailureregistration: regFailureAndroidCallback,
        onlinenotification: onlinePushNotificationAndroidCallback,
        offlinenotification: offlinePushNotificationAndroidCallback,
        onsuccessfulderegistration: unregSuccessAndroidCallback,
        onfailurederegistration: unregFailureAndroidCallback
    });
}
```

Properties

The kony.push.setCallbacks API contains the following property:

defaultRemoteNotificationCallbackBehaviour

This property is used to trigger online and offline notification callbacks based on whether your application is in the background, foreground, or is in an unused (dead) state. The default value of this property is false, and it is available only for the iOS platform.

The following table illustrates when online and offline notification callbacks are triggered depending on the value of the `defaultRemoteNotificationCallbackBehaviour` property.

Application Status	When defaultRemoteNotificationCallbackBehaviour is true/yes	When defaultRemoteNotificationCallbackBehaviour is false/no
When the application is in the foreground	Online notification callback is triggered	Online notification callback is triggered
When the application is in the background	Online notification callback is triggered	Offline notification callback is triggered
When the application is unused (dead state)	Offline notification callback is triggered	Offline notification callback is triggered

Return Values

None

Exceptions

The following error codes are applicable to onfailureregistration and onfailurederegistration functions:

- 1403 - Registration failed by PNS - Account related or device restrictions (thrown by push.register() API during an onfailureregistration callback).
- 1404 - Deregistration failed - Unable to close channel or PNS internal error while deregistration (thrown by push.deregister() API during an onfailurederegistration callback).

API Usage

You must call this API after the application is launched as this API will not be explicitly called by the user.

Implementation Details

The following are the implementation details of this API:

Based on the status of the registration or deregistration attempt (success or failure) and the state of the application on the device (active or inactive) when the Push Notifications arrive, use the following snippet to call the associated functions:

```
object = {
    onsuccessfulregistration: onsuccess,
    onfailureregistration: onfailure,
    onlinenotification: onlineCallback,
    offlinenotification: offineCallback,
    onsuccessfulderegistration: onderegsuccess,
    onfailurederegistration: onderegfailure
};
kony.push.setCallbacks(object);
```

onsuccessfulregistration

If the registration for Push Notifications is successful, this callback is executed by the underlying platform.

The callback takes string as the parameter.

The following code snippet is an example of the *onsuccessfulregistration* callback.

```
function onsuccess(identifier) {
    kony.print("Registered SUCCESSFULLY :" + identifier);
    //Send the identifier to the Push Notifications Sender.
}
```

Note: If there are spaces in the identifier (device token), you must replace the spaces with an empty string to successfully communicate with the APNS Server.

onfailureregistration

If the registration for Push Notifications is a failure, this callback is executed by the underlying platform.

The callback takes an Object as the parameter.

The following code snippet is an example of the *onfailureregistration* callback:

```
function onfailure(errortable) {
    kony.print("Registration Failed");

    kony.print(errortable.errorcode + errortable.errormessage);
}
```

onlinenotification

If the device receives a message when the application is running, this callback is executed by the underlying platform.

This callback can be used for silent push notifications, as described in the overview Silent Push Notifications. With silent push notifications, your app can perform operations in the background, such as downloading data from across the Internet, and not interrupt users by displaying notifications about the download.

The callback takes an Object as the parameter.

The following code snippet is an example of the *onlinenotification* callback:

```
function onlineNotification(payload) {
    /*payload is an Object that contains a set of key-value pairs
provided by the respective Push Notification Server*/
}
```

offlinenotification

If the device receives a message when the application is *not* running, this callback is executed by the underlying platform.

The callback takes an Object as the parameter.

The following code snippet is an example of the *offlinenotification* callback:

```
function offlineNotification(payload) {
    /*payload is an Object that contains a set of key-value pairs
provided by the respective Push Notification Server*/
}
```

onsuccessfulderegistration

If the deregistration for Push Notifications is successful, this callback is executed by the underlying platform.

The callback takes string as the parameter.

The following code snippet is an example of the *onsuccessfulderegistration* callback:

```
function onderegsuccess() {
    kony.print("Deregistered Successfully :");
}
```

onfailurederegistration

If the deregistration for Push Notifications is a failure, this callback is executed by the underlying platform.

The callback takes an Object as the parameter.

> ***Note:*** This function is not applicable on iOS platform.

The following code snippet is an example of the *onfailurederegistration* callback:

```
function onderegfailure(errortable) {
    kony.print("Deregistration Failed");
    kony.ui.Alert("Message : " + errortable["errorcode"] + errortable
["errormessage"], null, "info", null, , "Info");
}
```

Callback

permissionStatusCallback

Specifies the permission status of the resource authorization.

Syntax

```
permissionStatusCallback(response)
```

Parameters

Parameter	Description
response	A dictionary that contains the authorization status of the requested resource. This argument contains the following key: **status [Constant]** Resource status constant that indicates the status of the resource authorization. The possible values for status are as follows: • kony.application.PERMISSION_GRANTED • kony.application.PERMISSION_DENIED

Registering the Callback in kony.push.setCallbacks API

```
Kony.push.setCallbacks
({authorizationCallback:permissionStatusCallback, : : : : : : : : });
```

Example

```
function permissionStatusCallback(response) {
    if (response.status == kony.application.PERMISSION_GRANTED) {
        //PERMISSION_GRANTED Logic
    } else if (response.status == kony.application.PERMISSION_DENIED)
{
        //User-defined PERMISSION_DENIED Logic
    }
}
```

Note: This callback is called when you invoke the kony.push.register(notificationtypes) API.

Platform Availability

- Android
- iOS
- Windows
- Desktop Web

39.3 Notification Settings

Note: Notification setting APIs work on the following OS versions.

- iOS8 and above
- Android 4.1 and above

You can send actionable notifications using the notification setting APIs. User notifications can have additional custom actions. Two actions can be displayed on the lock screen, in a banner, and in the Notification Center. In alerts, notifications can display up to four actions when a user taps the Options button. To use notification actions in your app, you need to register the actions, schedule a local notification or a remote notification, and handle the action chosen by the user.

Following are the types of notifications:

- Alert or Banner: Based on the preferences a user sets in the devices, a notification is shown using either an alert view or a banner. Both the alert and banner should contain the message of the notification.
- Sound: A predefined or a custom sound is played when the notification is fired. In most cases, users will not look at the device all the time, so any notifications could be easily overlooked.
- Badge: A badge number can also be shown in the app's icon when a new notification is fired. The badge number must be increased by one with a new notification, and then decreased when the notification is handled. iOS automatically shows and hides the badge, depending on whether there is a value other than zero to display.

39.3.1 Actions, Categories, Settings, and Badges

Actions: Specifies the actions that any user can perform to interact with the notification message. Actions are presented as buttons and defined by the `kony.notificationsettings.createAction` API.

Categories: Specifies a group of actions into a category. They categories are defined by the `kony.notificationsettings.createCategory` API.

Settings: The settings must be registered by the application, and they contain all the categories and notification types that we create. These settings are defined by the `kony.notificationsettings.registerCategory` API.

Badges: Notification badges (also known as notification dots) appear on a launcher icon when the associated app has an active notification. Users can long-press the app icon to reveal the notifications (alongside any app shortcuts). These dots appear by default in launcher apps that support them, and there is nothing that your app needs to do. However, there might be situations in which you don't want the to notification dot to appear or you want to control exactly which notifications to appear there, so you can disable them on a per-channel basis. These settings are defined by the `kony.notificationsettings.registerCategory.setShowBadge` API.

Following are the notification settings APIs:

- kony.notificationsettings.createAction
- kony.notificationsettings.createCategory
- kony.notificationsettings.registerCategory
- kony.notificationsettings.setShowBadge

39.3.2 kony.notificationsettings Namespace

The kony.notificationsettings namespace provides the following API elements.

39.3.2.1 Functions

The kony.notificationsettings namespace provides the following functions.

39.3.2.2 kony.notificationsettings.createAction

This API helps you create an action that can be used with category.

Syntax

```
kony.notificationsettings.createAction (actionId, label, pspConfig)
```

Input Parameters

Parameter	Description
actionId [String] - Mandatory	Specifies a unique ID to identify the action chosen by a user.
label [String] - Mandatory	Specifies a label that needs to be associated with a action.

Parameter	Description
pspConfig [JSObject] - Mandatory	Specifies the configuration function. The parameters follow: • image [Number] - Mandatory: This parameter is applicable only on the Android platform. Set the image to be displayed for notification settings. • activationMode [Number] - Mandatory: Specifies the activation mode. The options are: • kony.notificationsettings.ACTIVATION_MODE_FORWARDS: If you set this option, the action will occur in the foreground and launches the application interface. • kony.notificationsettings.ACTIVATION_MODE_BACKWARDS: If you set this option, the action will occur in the background without launching the application interface. ***Note:*** The activationMode property is applicable only for the iOS platform. • authenticationRequired [Boolean] - Mandatory: This parameter is applicable only on iOS platform. This parameter is applicable only when you set activationMode as kony.notificationsettings.ACTIVATION_MODE_BACKWARDS. The user needs to unlock the device, and action occurs in the background. • destructive [Boolean] - Mandatory: This parameter is applicable only on iOS platform. Set this value to true, if the notification is self destructive. Otherwise, set it to false. • visibleOn [String] - Mandatory: Set this parameter to display the notification on a watch,

Example

```
/
**********************************************************************
****************
 * Function:registerActions()
 * Description: Creates Actions and a Category.
 * Author: Kony

 *********************************************************************
****************/
function registerActions() {
    var accept = kony.notificationsettings.createAction({
        "id": "Accept",
        "label": "Accept",
        "pspConfig": {
            "authenticationRequired": true,
            "destructive": true,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var reject = kony.notificationsettings.createAction({
        "id": "Reject",
        "label": "Reject",
        "pspConfig": {
            "authenticationRequired": false,
            "destructive": false,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var decline = kony.notificationsettings.createAction({
        "id": "Decline",
        "label": "Decline",
        "pspConfig": {
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_BACKWARDS,
            "authenticationRequired": true,
            "destructive": false,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });



    var defaultActionContextArr = [accept, reject, decline];
    var minimalActionContextArr = [accept, reject];

    var categoryObj = kony.notificationsettings.createCategory({
        "categoryId": "invitation",
        "actions": defaultActionContextArr,
        "pspConfig": {
            "minimalActions": minimalActionContextArr
        }
    });



    //Using kony.notificationsettings.registerCategory

    var categoryArr = [categoryObj];

    var registerCategory = kony.notificationsettings.registerCategory
({
        "categories": categoryArr,
        "pspConfig": {
            "types": [0, 1, 2]
        }
    });

}
```

Return values

This API returns action as a JavaScript object.

Platform Availability

Available on iOS and Android platforms.

39.3.2.3 kony.notificationsettings.createCategory

This API helps you create a category with a group of created actions.

Syntax

```
kony.localnotifications.createCategory (categoryId, actions,
pspConfig:{minimalActions:[actions] , presentationOptions});
```

Input Parameters

Parameter	Description
categoryId [String] - Mandatory	Specifies a unique ID for the group of actions that are defined.

Parameter	Description
actions [array] - Mandatory	Specifies the actions that are associated with the category. Pass null, if no actions are associated with the category. ***Note:*** In Default Context, you cannot add more than four actions on the iOS platform and three actions in the Android platform.
pspConfig [JSObject] - Mandatory	Specifies the configuration function. The parameter is: • minimalActions [array] - Mandatory: This parameter is applicable only on the iOS platform. Specify the actions that are to be associated for minimal actions for the category. • presentationOptions [array] - You can use this parameter to configure the presentation options of notifications. This is a iOS-specific parameter. This parameter is available from V8 SP3 onwards. You can pass an array of the following values: ◦ UNNotificationPresentationOptionBadge - 0: adds the notification badge icon. ◦ UNNotificationPresentationOptionSound - 1: plays the sound. ◦ UNNotificationPresentationOptionAlert - 2: displays the notification banner. ***Note:*** In default mode, you cannot add more than two actions.

Return Values

This API returns category as a JavaScript object.

Remarks

- If the presentationOptions parameter is not set, it is considered as no options are set and onlineNotification will not be shown.
- Only the mentioned values of presentationOptions must be passed as array.
- The input values for presentationOptions must consist of an array of minimum one value to a maximum of all three values. These options are applicable for all kinds of notifications.
- Onclick of notification banner triggers the onlinenotification callback when the app is in foreground/active state, whereas the offlinenotification callback is triggered when the app is in background/inactive state.

Platform Availability

Available on iOS and Android platforms.

Example

Example 1:

```
var categoryObj = kony.notificationsettings.createCategory({
    "categoryId": "invitation",
    "actions": defaultActionContextArr,
    "pspConfig": {
        "minimalActions": minimalActionContextArr,
        "presentationOptions": [0, 1, 2]
    }
});
```

Example 2:

```
/
*********************************************************************
***************
```

```
 * Function:registerActions()
 * Description: Creates Actions and a Category.
 * Author: Kony

*********************************************************************
****************/
function registerActions() {
    var accept = kony.notificationsettings.createAction({
        "id": "Accept",
        "label": "Accept",
        "pspConfig": {
            "authenticationRequired": true,
            "destructive": true,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var reject = kony.notificationsettings.createAction({
        "id": "Reject",
        "label": "Reject",
        "pspConfig": {
            "authenticationRequired": false,
            "destructive": false,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var decline = kony.notificationsettings.createAction({
        "id": "Decline",
        "label": "Decline",
        "pspConfig": {
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_BACKWARDS,
            "authenticationRequired": true,
            "destructive": false,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });


    var defaultActionContextArr = [accept, reject, decline];
    var minimalActionContextArr = [accept, reject];

    var categoryObj = kony.notificationsettings.createCategory({
        "categoryId": "invitation",
        "actions": defaultActionContextArr,
        "pspConfig": {
            "minimalActions": minimalActionContextArr
        }
    });


    //Using kony.notificationsettings.registerCategory

    var categoryArr = [categoryObj];

    var registerCategory = kony.notificationsettings.registerCategory
({
        "categories": categoryArr,
        "pspConfig": {
            "types": [0, 1, 2]
        }
    });

}
```

39.3.2.4 kony.notificationsettings.registerCategory

This API helps you register the created category with the application.

Syntax

```
kony.notificationsettings.registerCategory (categories, pspConfig)
```

Input Parameters

Parameter	Description
category [array] - Mandatory	Specifies the category objects that are to be registered.

Parameter	Description
pspConfig [JSObject] - Optional	This parameter is applicable on iOS platform only. It specifies the platform specific configuration function. The arguments are: • types [String] - Mandatory: Configure this parameter to register the type of notification. The Array for iPhone must contain **any** or **all** of the following: • 0 - Specifies the notification type as Badge. • 1 - Specifies the notification type as Sound. • 2 - Specifies the notification type as Alert. ***Note:*** If the types are not passed or pspConfig is not defined, then all the

Example

```
/
**********************************************************************
****************
 * Function:registerActions()
 * Description: Creates Actions and a Category.
 * Author: Kony

 **********************************************************************
****************/
function registerActions() {
    var accept = kony.notificationsettings.createAction({
        "id": "Accept",
        "label": "Accept",
        "pspConfig": {
            "authenticationRequired": true,
            "destructive": true,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var reject = kony.notificationsettings.createAction({
        "id": "Reject",
        "label": "Reject",
        "pspConfig": {
            "authenticationRequired": false,
            "destructive": false,
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_FORWARDS,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });

    var decline = kony.notificationsettings.createAction({
        "id": "Decline",
        "label": "Decline",
        "pspConfig": {
            "activationMode": kony.notificationsettings.ACTIVATION_
MODE_BACKWARDS,
            "authenticationRequired": true,
            "destructive": false,
            "visibleOn": kony.notificationsettings.BOTH
        }
    });



    var defaultActionContextArr = [accept, reject, decline];
    var minimalActionContextArr = [accept, reject];

    var categoryObj = kony.notificationsettings.createCategory({
        "categoryId": "invitation",
        "actions": defaultActionContextArr,
        "pspConfig": {
            "minimalActions": minimalActionContextArr
        }
    });



    //Using kony.notificationsettings.registerCategory

    var categoryArr = [categoryObj];

    var registerCategory = kony.notificationsettings.registerCategory
({
```

```
            "categories": categoryArr,
            "pspConfig": {
                "types": [0, 1, 2]
            }
        });

}
```

Return Values

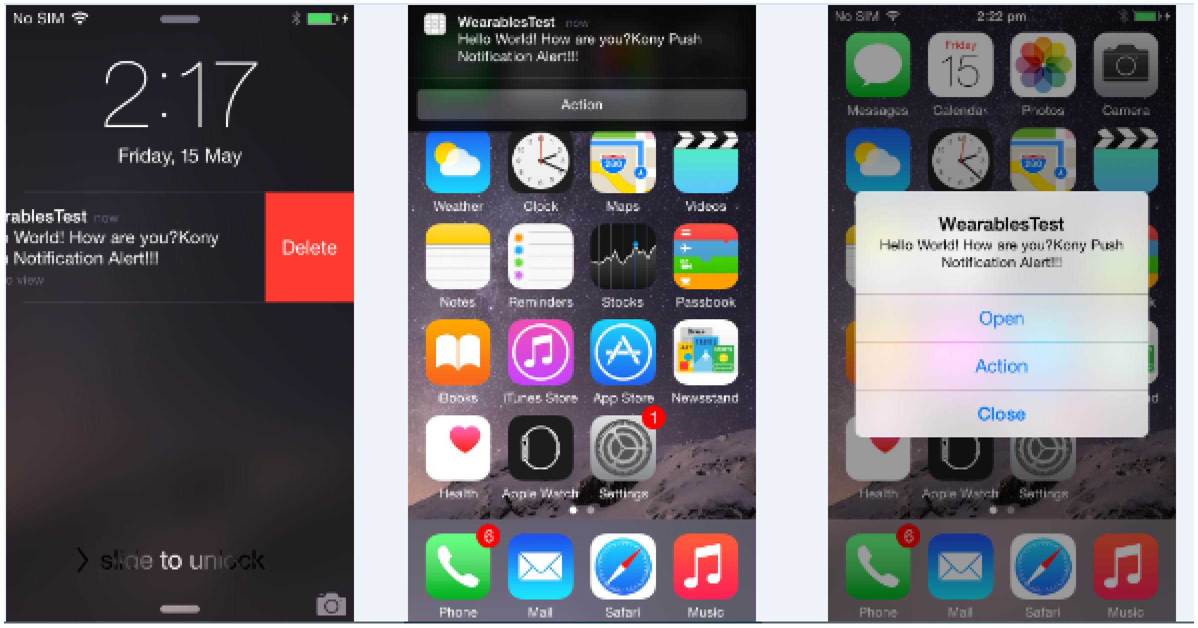
None

Platform Availability

Available on iOS and Android platforms.

Handling Local and Remote Notifications Callbacks:

- If the app is in the foreground state (app instance alive), an onlinecallback should be invoked.
- If the app is in background state (app instance alive or not alive), an offlinecallback should be invoked.


WearablesTest
Hello World! How are you?Kony Push Notification Alert!!!

39.3.2.5 kony.notificationsettings.pickTitleAndDescriptionFromPushPayload

Once you set the kony.notificationsettings.pickTitleAndDescriptionFromPushPayload API as true, the Title and Description details of the payload are considered. By default, the Title and Description information of the payload are not respected.

Syntax

```
kony.notificationsettings.pickTitleAndDescriptionFromPushPayload
(pickTitleAndDescriptionFromPushPayloadKey)
```

Input Parameters

Parameter	Description
pickTitleAndDescriptionFromPushPayloadKey [Boolean] - Mandatory	Set this input parameter as true to ensure that the Title and Description data of the payload are taken into consideration.

Example

```
var pickTitleAndDescriptionFromPushPayloadKey = true;
kony.notificationsettings.pickTitleAndDescriptionFromPushPayload
(pickTitleAndDescriptionFromPushPayloadKey);
```

Remarks

- The Boolean value that is passed in the pickTitleAndDescriptionFromPushPayloadKey input parameter is stored in Shared Preferences.

Platform Availability

- Android

39.3.2.6 kony.notificationsettings.setProperties

You can use the kony.notificationsettings.setProperties API to set various properties of notifications that belong to a category.

Syntax

```
setProperties(categoryId, properties)
```

Input Parameters

Parameter	Description
categoryId	The category ID of the notification for which the priority is set. The value type of categoryId is String.

Parameter	Description
properties	The configuration object of the category. The value type of properties is JSON object. Currently, the JSON object can contain only the priority key. **priority:** A property for setting priority to the notification object related to a category. The value type of priority is constant. The priority key contains one of the following constants: • constants.NOTIFICATION_PRIORITY_DEFAULT: Default notification priority. • constants.NOTIFICATION_PRIORITY_HIGH: Higher notification priority is set for important notifications. • constants.NOTIFICATION_PRIORITY_LOW: Lower notification priority is set for notifications that are less important. • constants.NOTIFICATION_PRIORITY_MAX: Highest notification priority is set for an app's most important items that require the end user's prompt inputs. • constants.NOTIFICATION_PRIORITY_MIN: Lower notification priority is set for items that are less important. You can choose

Example

```
propTable = {};
propTable.priority = constants.NOTIFICATION_PRIORITY_HIGH;
setProperties({
    categoryId: "invitation",
    properties: propTable
});
```

Platform Availability

Android

39.3.2.7 kony.notificationsettings.setShowBadge

This API helps you to enable or disable notification badges for push/remote or local notifications that are only supported by Kony Framework. By default, the visibility of the notification badge is set to true on Android 8.0 devices.

Syntax

```
kony.notificationsettings.setShowBadge(notificationChannelName,
showBadgeValue)
```

Input Paramters

Parameter	Description
notificationChannelName [String] - Mandatory	The following values are applicable for this parameter: • "localNotification" • "pushNotification"

Parameter	Description
showBadgeValue [Boolean] - Mandatory	Set this value to false to disable the visibilty of the app notification badge.

Example

```
var notificationChannelName = "pushNotification";
var showBadgeValue = false;
kony.notificationsettings.setShowBadge(notificationChannelName,
showBadgeValue);
```

Remarks

- You must call this API in **preapp init**, as the badge visibility can only be set after the first notification is received. This is because the channel creation is done after the first notification is initiated; once the channel creation is done, the visibility of the notification badge cannot be modified.
- Some launchers may not support the Notification Badge feature, even on devices with Android 8.0 and later.

Platform Availability

- Android 8.0 and later

40. Offline Data Access API

These APIs allow you to store data onto the device data store persistently (available after application or device restart). Such data can be accessed even if the device is offline (not connected to any wireless network [cellular or WIFI]. The API libraries are as follows:

The Offline Data Access API allows you to store data onto the device store persistently (data is made available even after application or device restart). You can access this data even when the device is offline, i.e, the device is not connected to any wireless network (cellular or WiFi). The Offline Data Access API has the following API libraries:

40.1 Local Storage API Library

LocalStorage APIs allow the users to store and retrieve data in HTML pages from the same domain. The data can be retrieved from all the windows in the same domain even if the browser is restarted.

Allows users to store and retrieve data in HTML pages from all the windows in the same domain even if the browser is restarted.

The data stored using the LocalStorage APIs is accessible only from the client-side as opposed to the data stored in cookies (accessible from both client-side and server-side). You can store huge data using these APIs.

> ***Note:*** From V8 SP4 onwards, the Local Storage APIs' data for a Kony Visualizer App Viewer child app is stored in child app data and not under the parent app. This feature is applicable for iOS, Windows, and Android platforms.

The Local Storage API Library contains the `kony.store Namespace` and related functions:

Function	Description
`kony.store.clear`	Allows you to empty the database by clearing all the key-value pairs. If there are no key-value pairs, then the API does not do anything.

Function	Description
`kony.store.getItem`	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** for JavaScript.
`kony.store.removeItem`	Removes the item identified by the key, if it exists. If no item with that key exists, the method does not perform any action.
`kony.store.key`	Returns the name of the nth key in the list. If n is greater than the number of key/value pairs in the object, then this method returns **null** for JavaScript.
`kony.store.setItem`	Creates a structured clone of the given value. If this raises an exception then the list associated with the object is left unchanged.
`kony.store.length`	Returns the length of the local storage.

To store data items in a key-value format in the Local Storage, use the `kony.store.setItem` function. Use the `kony.store.key` function to retrieve the name of an existing key. Using the `kony.store.length` function, you can find the amount of the local storage. To retrieve a stored value, use the `kony.store.getItem` function. Use the `kony.store.removeItem` function to remove a data item from the local storage. To empty the database, use the `kony.store.clear` function.

To view the functionality of the Local Storage API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

40.2 Web SQL API Library

WebSQL APIs allow you to manipulate client-side databases using SQL. You can use these APIs to store data in databases that can be queried using a variant of SQL.

Allows users to manipulate client-side databases using SQL. Use these APIs to store data in databases that can be queried using a variant of SQL.

The Web SQL API Library contains the `kony.db Namespace` and related functions:

Function	Description
`kony.db.changeVersion`	Allows scripts to automatically verify the version number and change it at the same time during a schema update.
`kony.db.executeSql`	Allows you to execute a specified SQL statement on the given database. This is an asynchronous API.
`kony.db.openDatabase`	Allows you to open the specified version of a database. This is an asynchronous API.
`kony.db.readTransaction`	Allows you to read a specified transaction. This API creates a *SQLTransaction* object for read-only transactions.
`kony.db.sqlResultsetRowItem`	Returns the row available at the specified index. If there is no such row, then the API returns **null** for JavaScript.
`kony.db.transaction`	Allows you to execute the specified transaction on the given database. When you invoke this API, it returns immediately and asynchronously executes the transaction.

Open a specific version of the database using the `kony.db.openDatabase` function. You can then create an SQLTransaction object using the `kony.db.transaction` function. Use the `kony.db.readTransaction` function to read a specific transaction. Execute an SQL statement using the `kony.db.executeSql` function. Use the `kony.db.sqlResultsetRowItem` function to retrieve the row available at a specific index. You can also verify if there is any schema update and then change the version of the database being used by using the `kony.db.changeVersion` function.

Important: WebSQL implementation utilizes the database engines bundled along with the underlying SDK (SQLite in most of the cases). Kony Platform ensures that the WebSQL semantics are same across platforms, but the uniformity of SQL semantics across native platforms is not guaranteed. For example, the WebSQL APIs have no control over the way CONSTRAINTS are supported as per the SQL92 specification. For more information on SQLite refer Appendix.

Important: WebSQL APIs are not supported on Internet Explorer and Firefox web browsers.

Important: The framework does not store the password argument to **kony.db.openDatabase()** function, it is the responsibility of application developer to secure the password.

The actual database used for storing the data and its visibility to other applications is defined in the following table:

Platform	Client Database Engine	Accessible to other applications
iPhone/iPad	SQLite	The storage area is unique to the application.
Android/Android Tablet	SQLite	The storage area is unique to the application.

Platform	Client Database Engine	Accessible to other applications
BlackBerry	SQLite	SDCard. Data is accessible to other applications.
Windows 6x	SQL Server CE	File System. Data is accessible to other applications.
Windows Phone 7.5	SQLite	The storage area is unique to the application.
Windows Phone 8	SQLite	The storage area is unique to the application.
Windows Phone 8.1	SQLite	The storage area is unique to the application.
Windows Desktop/Kiosk	SQLite	The storage area is unique to the application.
Windows 8	SQLite	The storage area is unique to the application.
Windows 8.1	SQLite	The storage area is unique to the application.
Symbian	SQLite	The storage area is unique to the application.
Single Page Application (SPA)	SQLite	Determined by the browser implementation.

The following table indicates the JavaScript datatypes and their corresponding SQLite datatypes:

JavaScript Datatype	SQLite Datatype
string	TEXT
number	REAL
byte array (rawbytes)	BLOB
boolean	BOOLEAN

40.3 Data Store API Library

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

Data Store is a storage area that can be shared with multiple applications. The Data store enables applications to synchronize the data from the data store into the application.

Note: From V8 SP4 onwards, the Data Store APIs' data for a Kony Visualizer App Viewer child app is stored in child app data and not under the parent app. This feature is applicable for iOS, Windows, and Android platforms.

The Data Store API contains the `kony.ds Namespace` and the following API elements:

Function	Description
`kony.ds.read`	Provides you an ability to read the data stored on the data store of the mobile device using the given name identifier.
`kony.ds.remove`	Allows you to delete the data stored on the data store that corresponds to a specific name identifier.
`kony.ds.save`	Allows you to save the given data using the specified name identifier in the data store of the mobile device.

You can store the given data using the `kony.ds.save` function, fetch the data that is stored using the `kony.ds.read` function, and remove the data associated with a specific name identifier using the `kony.ds.remove` function.

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

- the application accesses the data
- the device reboots
- the battery is changed
- the application upgrades:
    - **Native Clients -** data stored earlier is not deleted when an application upgrades. On some of the J2ME phones, the user is prompted with an option to delete the data during the application upgrade process. If the end user chooses the delete option then the data will be lost even on application upgrade.
    - **Mobile Web when storage mode is cookie -** the data is lost when you delete the cookies on the client browser.
    - **Mobile Web when storage mode is cache -** the data is lost every time you restart the application server.
    - **Mobile Web when storage mode is session-** the data is lost as soon as the server-side session is terminated/expired.

The actual storage used to store this data depends upon the underlying Operating System:

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

Android/Android Tablet	Within a file in the application sandbox. The storage area is unique to the application.	No
BlackBerry	RMS (Record Management Store)	No
J2ME	RMS (Record Management Store)	No
Windows 6x	Within a file	Yes
Windows Phone	Within a file in the application sandbox. The storage area is unique to the application.	No
Symbian	Within a file in the application sandbox. The storage area is unique to the application.	No
Mobile Web (mode set as cookie)	Browser Cookie	No
Mobile Web (mode set as cache)	Application Context	No (but shared across all the users for the same application)
Mobile Web (mode set as session)	HTTP Session	No

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

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

40.3.0.1 Limitations

This section describes the limitations of WebSQL APIs:

- Table names and Column names are case-sensitive. For example, "Employee" and "employee" refer to two different tables. It is observed that table names and columns name are case-insensitive on Android platform, but case-sensitive on iPhone platform. For a cross platform uniform behavior, assume that the table names and column names are case-sensitive.
- When using the binding "?" placeholders, ensure that the number of binding parameters and the number of values are the same. Else, the database can be inconsistent.

  For example, the following statement can cause the database to be inconsistent as the number of binding parameters are 6 and the number of values are 3:

  ```
  local stmt3 = "insert into EMP_DETAILS values(?,?,?,?,?,?)"
  ```

  ```
  kony.db.executeSql(tx,stmt3,{
  '1', '2', '3'
  }, successCallback,errorCallback);
  ```

- Nested database transactions are not supported across platforms. Avoid starting a new transaction in the successcallback function of an existing transaction.
- The error codes might vary between the platforms. This is due to the fact that underlying SDKs do not expose all the errors that can be mapped to HTML5 WebSQL Specification. The application should not build its business logic around these error codes. These error codes are only for diagnostic purpose.
- Accessing *insertid* key in the resultset rows leads to an error in SPA.
- The WebSQL APIs are not supported on Internet Explorer and Firefox.

To view the functionality of the Data Store API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

40.4 kony.store Namespace

The kony.store Namespace provides the following API elements.

40.4.1 Functions

The kony.store namespace provides the following functions.

40.4.2 kony.store.clear

This API allows you to empty the database by clearing all the key-value pairs. If there are no key-value pairs, then the API does not do anything.

Syntax

```
kony.store.clear()
```

Input Parameters

None

Example

```
try {
    kony.store.clear();
    alert("store is cleared");
} catch (err) {
    alert("error occurred in clear() and the error is :" + err);
}
```

Return Values

None.

Exceptions

LocalStorageError

Error

Implementation Details

For implementation details, see http://www.w3.org/TR/webstorage/#the-localstorage-attribute.

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

40.4.3 kony.store.getItem

This API 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** for JavaScript.

Syntax

```
kony.store.getItem(keyname)
```

Input Parameters

Parameter	Description
keyname [String] - Mandatory	Specifies the keyname from which the item needs to be fetched.

Example

```
var myValue = kony.store.getItem("name");
alert("name is " + myValue);
```

Return Values

Return Value	Description
myitem [Object]	Returns the item located at the specified index.

Exceptions

LocalStorageError

Error

Implementation Details

For implementation details, see http://www.w3.org/TR/webstorage/#the-localstorage-attribute.

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

40.4.4 kony.store.removeItem

This API removes the item identified by the key, if it exists. If no item with that key exists, the method does not perform any action.

Syntax

```
kony.store.removeItem(keyname)
```

Input Parameters

Parameter	Description
keyname [String] - Mandatory	Specifies the keyname for which the item needs to be removed.

Example

```
kony.store.removeItem("name");
alert("name removed");
```

Return Values

This API does not return a value.

Exceptions

LocalStorageError

Error

Implementation Details

For implementation details, see http://www.w3.org/TR/webstorage/#the-localstorage-attribute.

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

40.4.5 kony.store.key

This API returns the name of the nth key in the list. If n is greater than the number of key/value pairs in the object, then this method returns **null** for JavaScript.

Syntax

```
kony.store.key(index)
```

Input Parameters

Parameter	Description
index [Number] - Mandatory	Specifies the index for which the key name is to be returned.

Example

```
var keyName = kony.store.key(0);
alert("first key name is " + keyName);
```

Return Values

Return Value	Description
keyname [String]	Returns the keyname of the specified index.
null/nil	Returns null/nil when the specified index is greater than the number of key/value pairs in the object.

Exceptions

LocalStorageError

Error

Implementation Details

For implementation details, see http://www.w3.org/TR/webstorage/#the-localstorage-attribute.

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

40.4.6 kony.store.setItem

This API creates a structured clone of the given value. If this raises an exception then the list associated with the object is left unchanged.

Syntax

```
kony.store.setItem(key, value)
```

Input Parameters

Parameter	Description
key [string] - Mandatory	Specifies the keyname for which the item needs to be set.
value [object] - Mandatory	Specifies the value that must be set at the given index. This value can be a number, string, Boolean.

Example

```
kony.store.setItem("keyValue5", "this is a key value");
kony.store.setItem("keyValue4", true);
```

Return Values

None.

Exceptions

LocalStorageError

Error

Implementation Details

For implementation details, see http://www.w3.org/TR/webstorage/#the-localstorage-attribute.

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

40.4.7 kony.store.length

This API returns the length of the local storage.

Syntax

```
kony.store.length()
```

Input Parameters

None

Example

```
mylength = kony.store.length();
alert("length is " + mylength);
```

Return Values

Return Value	Description
mylength [String]	Returns the length of the local storage.

Exceptions

LocalStorageError

Error

Implementation Details

For implementation details, see http://www.w3.org/TR/webstorage/#the-localstorage-attribute.

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

40.5 kony.db Namespace

The kony.db namespace contains the following API elements.

40.5.1 Functions

The kony.db namespace contains the following functions.

40.5.2 kony.db.changeVersion

The *changeVersion* method allows scripts to automatically verify the version number and change it at the same time during a schema update. This method creates an SQLTransactionSync object (executeSQL) for a read/write transaction. The database's actual version changes to newVersion only if the first argument (oldVersion) exactly matches the database's actual version, otherwise throws a SQLException.

When you invoke this API, it returns immediately and asynchronously reads the transaction.

Syntax

```
kony.db.changeVersion(dbaseObjectId, oldVersion, newVersion, 
SQLTransactionCallback, SQLTransactionErrorCallback, 
SQLTransactionVoidCallback)
```

Input Parameters

Parameter	Description
dbaseObjectId [String] - Mandatory	Specifies the unique ID of the database
oldVersion [String] - Mandatory	Specifies the older version of the database.
newVersion [String] - Mandatory	Specifies the newer version of the database.

SQLTransactionCallback [Function] - Mandatory	Specifies the callback function that contains the transactions.
	
	For example:
	
	`function callback(dbId){`
	`//SQLTransaction contains implementation of executeSql method`
	`// invoke database.executesql method for sql trasaction`
	`}`

SQLTransactionErrorCallback [Function] - Optional	Specifies the callback that must be executed if there is an error in executing the transaction. This callback function is used to roll back the updates to the database.
	
	For example:
	
	`function errorCallback(SQLError){`
	`//code`
	`}`

Parameter	Description

SQLVoidCallback [Function] - Optional	Specifies the callback that must be executed if the transaction is successful.
	
	For example:
	
	`function successCallback(){`
	`//code`
	`}`

Example

```
kony.db.changeVersion("1.0", "1.1", transactioCall, errorCall,
sucessCall);

function transactioCall() {
    //respective code
}

function sucessCall() {
    //respective code
}

function errorCall() {
    //respective code
}
```

```
changeVersion: function() {
    this.baseObjectId = kony.db.openDatabase("webSqlDB",
        "1.0",
        "Sample SQL Database",
        5 * 1024 * 1024);
    kony.db.changeVersion("1.0", "1.1", null,
this.commonErrorCallback, this.commonErrorCallback);
```

```
    kony.print("");
}
```

Return Values

None

Platform Availability

Available on all platforms* except SPA in Windows Phone 7.5 and windows Phone 8 browsers, for desktop IE8, IE9, IE10 browsers. *Dummy implementation on Mobile Web.

40.5.3 kony.db.executeSql

This API allows you to execute a specified SQL statement on the given database. This is an asynchronous API.

Important: The table names and column names are case sensitive.

Syntax

```
kony.db.executeSql(transactionId,SQLStatement,
arguments,SQLStatementSuccessCallback,SQLStatementErrorCallback)
```

Input Parameters

Parameter	Description
transactionId [String] - Mandatory	Specifies the unique ID of the transaction.
SQLStatement [String] - Mandatory	Specifies the SQL statement that must be executed.

Parameter	Description
arguments [Object] - Optional	Specifies the arguments for executing the SQL statement. If this parameter is not specified or is nil, then the statement is executed without any arguments. The supported argument types are: • NULL in JavaScript • REAL • TEXT • BLOB. No type affinity is performed.
SQLStatementSuccessCallback [Function] - Optional	Specifies the callback function that must be executed when the execution of the SQL statement is a success.
SQLStatementErrorCallback [Function] - Optional	Specifies the callback function that must be executed when the execution of the SQL statement is a failure. This callback returns a boolean value. • *true* - ends the execution of the transaction. *true* is returned if there is no callback function specified as a parameter. *Important:* When *true* is returned, the transaction is rolled back. • *false* - continues executing the transaction. The default return value is *false* for this callback.

Example

```
//The below function specifies the callback function that must be
executed when the execution of the SQL statement is a success
function sql_success(transactionId, resultset) {
    //logic to process the resultset
}
//The below function specifies the callback function that must be
executed when the execution of the SQL statement is a failure.
function sql_errorCallBack(err) {
    alert("Error processing sql statement error code=" + err.code);
}
//The below function specifies the callback function that contains the
transactions also invokes executeSql
function myTransactionCallback(dbId) {
    //SQLTransaction contains implementation of executeSql method
    // invokekony.db.executesql method for sql trasaction
    var sqlStatement = "SELECT * FROM employee";
    kony.db.executeSql(transactionID, sqlStatement, null, sql_success,
sql_errorCallBack);
}
```

```
The below function inserts 3 rows into the 'employee_details' table.
var insertTable = [
    ["Siberius", 10],
    ["Clark", 10],
    ["Richard", 20]
];
for (i = 1;
    ((insertTable) != null) & amp; & amp; i & lt; =
insertTable.length; i++) {
    var v = insertTable[kony.decrement(i)];
    var sqlStatement = "INSERT INTO emp_details VALUES (" +
(this.count + 1000) + ",\"" + v[kony.decrement(1)] + "\"," + v
[kony.decrement(2)] + ")";
    this.count = this.count + 1;
    kony.db.executeSql(transactionID,
        sqlStatement,
        null,
        this.success_insertData,
        this.commonErrorCallback);
}
```

Return Values

None

Error Codes

The following table lists the error codes along with its corresponding error messages:

Error Code	Error Message	Description
1	UNKNOWN_ ERR	The statement failed for database reasons not covered by any other error code.
2	DATABASE_ ERR	The operation failed because the actual database version was not what it should be. For example, a statement found that the actual database version no longer matched the expected version of the `Database` or `DatabaseSync` object, or the `Database.changeVersion()` or `DatabaseSync.changeVersion()` methods were passed a version that doesn't match the actual database version.
3	TOO_LARGE_ ERR	The statement failed because the data returned from the database was too large. The SQL "LIMIT" modifier might be useful to reduce the size of the result set.

Error Code	Error Message	Description
4	QUOTA_ERR	The statement failed because there was not enough remaining storage space, or the storage quota was reached and the user declined to give more space to the database.
5	SYNTAX_ERR	The statement failed because of a syntax error, or the number of arguments did not match the number of ? placeholders in the statement, or the statement tried to use a statement that is not allowed, such as BEGIN, COMMIT, or ROLLBACK, or the statement tried to use a verb that could modify the database but the transaction was read-only.
6	CONSTRAINT_ERR	An INSERT, UPDATE, or REPLACE statement failed due to a constraint failure. For example, because a row was being inserted and the value given for the primary key column duplicated the value of an existing row.
7	TIMEOUT_ERR	A lock for the transaction could not be obtained in a reasonable time.

For more information, refer SQL Error Codes

Platform Availability

Available on all platforms* except for SPA in Windows Phone 7.5 and windows Phone 8 browsers, for desktop IE8, IE9, IE10 browsers.*Dummy Implementation on Mobile Web.

40.5.4 kony.db.openDatabase

This API allows you to open the specified version of a database. This is an asynchronous API.

Important: The passphrase parameter in this API is not supported in 6.5 plugins, so database encryption is not available in 6.5 plugins.

Syntax

```
kony.db.openDatabase(dbname,version,displayName, estimatedSize,
passphrase)
```

Input Parameters

Parameter	Description
dbname [String] - Mandatory	Specifies the actual name of the database that you want to open.
version [String] - Mandatory	Specifies the version of the database that you want to open.
displayName [String] - Mandatory	Specifies the display name of the database that you want to open.
estimatedSize [Number] - Optional	Specifies the approximate size of the database in bytes. ***Note:*** This parameter is applicable only on the SPA platform.

passphrase

This is applicable for iOS and Android

iOS

passphrase[Array] - Optional

In iOS, this argument is used to specify Pragma statements (Array) to encrypt database.

Usage

- To link the SQLCipher library, extract the KAR file with **-sqlcipher** option. After the extraction, pass the **passphrase** for the SQL engine to encrypt the database.

```
perl extract.pl /Users/PLATFROM/Downloads/konyappipad.KAR –
sqlcipher
```

> ***Note:*** For more information on this parameter and Pragma statements, see https://www.zetetic.net/sqlcipher/sqlcipher-api/.

Android

passphrase [String] - Optional

In Android, this argument is used to specify password (String) to encrypt database.

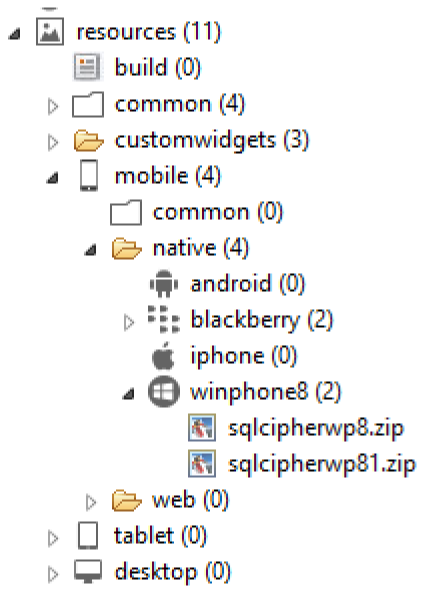
The developer needs to enable **Support SQL DB Encryption (FIPS)** option in **Kony Visualizer > Application Properties > Native > Android** section in order to support database encryption using passphrase.

Windows Phone 8/8.1

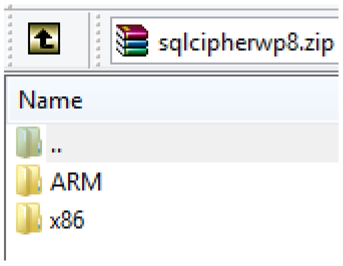
If a password is provided to the openDatabase API, the database will be encrypted.

For database encryption a set of licensed components must be obtained from Zetetic inc. for Windows Phone 8 and Windows Phone 8.1. For more information, see https://www.zetetic.net/sqlcipher/buy/.

The dll/class library is provided by Zenetic Inc as a zipped file. It contains the sqlcipher dlls built for arm and x86 architectures and should be copied to the windowsphone8 folder, as shown in the following figure.


```
resources (11)
    build (0)
    common (4)
    customwidgets (3)
    mobile (4)
        common (0)
        native (4)
            android (0)
            blackberry (2)
            iphone (0)
            winphone8 (2)
                sqlcipherwp8.zip
                sqlcipherwp81.zip
        web (0)
    tablet (0)
    desktop (0)
```

During the build process these dlls are unzipped and added to the xap file used for application deployment. The name of the zip file for Windows Phone 8 must be sqlcipherwp8.zip. The zip file for Windows Phone 8.1 must be sqlcipherwp81.zip. The zip files should have folders for each architecture type, where the dlls for each architecture are placed.


```
sqlcipherwp8.zip
Name
..
ARM
x86
```

Each architecture folder should contain the following 3 files:

- Sqlite.winmd
- Sqlite.dll
- sqlite3.dll

Windows 8.1 and Windows Kiosk

A free version of the sqlite library is available to support encryption. Therefore a license is not needed and you do not need to copy the zip files.

Important:
The database will not be encrypted in the following conditions:
1. If **Support SQL DB Encryption (FIPS)** option is enabled and empty string is passed as passphrase.
2. If **Support SQL DB Encryption (FIPS)** option is not enabled ,passphrase will be ignored and database is non encrypted.
3. If passphrase is not passed.

Important: Encrypted and non-encrypted databases cannot be interchangeably opened using this API.

Important: The other Web SQL APIs operate on encrypted DB seamlessly if the `databaseObjectId` passed to them is opened using `kony.db.openDatabase()` API with passphrase.

Important: The framework does not store the passphrase argument to `kony.db.openDatabase()` API, it is responsibility of application developer to secure the password.

Example

```
//The below function will invoke openDatabase
function openDatabase() {
    var dbName = "konytestDB";
    var version = "1.0";
    var displayName = "demo web SQL Database";
    var estimatedSize = 5 * 1024 * 1024; //5*1024*1024 indicates 5 MB
    //Note: Use only one of the following passphrases depending on
platform
    //Passphrase is string in case of android.
    var passphrase = "samplepassword";

    //Passphrase is pragma array in case of iOS.
    passphrase = ["PRAGMA key = 'old passphrase';"];

    var databaseObjectId = kony.db.openDatabase(dbName, version,
displayName, estimatedSize, passphrase);
    //databaseObjectId contains the unique ID of the database
}
```

Return Values

Return Value	Description
databaseObjectId [userdata]	Returns the unique ID of the database. The identifier is userdata (platform specific instance) and is not specifically String. Developers must not rely on the type of the identifier.

Platform Availability

Available on all platforms(dummy implementation on MobileWeb) except for SPA in Windows Phone 7.5 browser, and for desktop IE8, IE9, and IE10 browsers.

Using the Pre Bundled Database

Bundling: For information about how to bundle a database, refer to the Pre Bundling the Files.

Accessing: The kony.io.FileSystem.getDatabaseDirectoryPath API returns the path where kony.db.openDatabase API opens a specified database file.

You need to copy pre bundled database files using the kony.io.FileSystem.copyBundledRawFileTo API to the path returned by kony.io.FileSystem.getDatabaseDirectoryPath API and then you can read or edit a pre bundled database information using the kony.db.openDatabase API just by opening database using the file name.

```
//Example for copying and opening the pre bundled database
//The destination file name can be different from the source.
var destFilePath = kony.io.FileSystem.getDatabaseDirectoryPath()
+”test.db”;
var fileObj = null;
try{
    var file = new kony.io.File(destFilePath);
    //copyBundledRawFileTo API overrides the destination file with new
one.
    //Hence check before copying
    if(!file.exists()){
      fileObj = kony.io.FileSystem.copyBundledRawFileTo(dbName,
destFilePath);
    }else{
      fileObj = file;
      alert("File is already available");
      return;
    }
} catch(e) {
    kony.print(“Exception “+e);
  }
  if(fileObj == null){
    kony.print(“Copy failed”);
  }else{
    kony.print(“Copy Success”);
  }
//Opening the copied DB using openDatabase API
dbObject = kony.db.openDatabase("test.db", "1.0", "Prebundled SQL
Database", 5 * 1024 * 1024);
```

40.5.5 kony.db.readTransaction

This API allows you to read a specified transaction. This API creates a *SQLTransaction* object for read-only transactions.

When you invoke this API, it returns immediately and asynchronously reads the transaction.

Important: You can only read queries using this API. The *kony.db.readTransaction* API results in a SYNTAX_ERR in case of DML commands in the SQL statement. DML commands are not supported in this API.

Important: The table names and column names are case sensitive.

Syntax

```
kony.db.readTransaction
(dbaseObjectId,TransactionCallback,TransactionErrorCallback,
SuccessCallback)
```

Input Parameters

Parameter	Description
dbaseObjectId [String] - Mandatory	Specifies the unique ID of the database.

TransactionCallback [Function] - Mandatory	Specifies the callback function that contains the transactions.
	For example:
	`function callback(dbId){`
	`//Code`
	`}`

TransactionErrorCallback [Function] - Optional	Specifies the callback that must be executed if there is an error in executing the transaction. This callback function is used to roll back the updates to the database.For example:
	`function errorCallback(SQLError){`
	`//Code`
	`}`

SuccessCallback [Function] - Optional	Specifies the callback that must be executed if the transaction is successful.
	For example:
	`function successCallback(){`
	`//code`
	`}`

Example

```
//The below function specifies the callback function that contains the
transactions.
function myTransactionCallback(dbId) {
    //SQLTransaction contains implementation of executeSql method
    // invokekony.db.executeSql method for sql trasaction
}
//The below function specifies the callback that must be executed if
there is an error in executing the transaction. This callback function
is used to roll back the updates to the database.
function myTransactionErrorCallback(SQLError) {
    // proceed with the logic
}
//The below function specifies the callback that must be executed if
the transaction is successful.
function mySuccessCallback() {
    // proceed with the logic
}
//The below function will invoke readTransaction
function readTransaction() {
    var dbName = "konytestDB";
    var version = "1.0";
    var displayName = "demo web SQL Database";
    var estimatedSize = 5 * 1024 * 1024; //5*1024*1024 indicates 5 MB
    var databaseObjectId = kony.db.openDatabase(dbName, version,
displayName, estimatedSize);
    //databaseObjectId contains the unique ID of the database
    kony.db.readTransaction(databaseObjectId, myTransactionCallback,
myTransactionErrorCallback, mySuccessCallback);
}
```

Return Values

None

Platform Availability

Available on all platforms* except for SPA in Windows Phone 7.5 and windows Phone 8 browsers, for desktop IE8, IE9, IE10 browsers.*Dummy Implementation on Mobile Web.

40.5.6 kony.db.sqlResultsetRowItem

This API returns the row available at the specified index. If there is no such row, then the API returns **null** for JavaScript.

Important: The table names and column names are case sensitive.

Syntax

```
kony.db.sqlResultsetRowItem(transactionID,SQLResultSet,index)
```

Input Parameters

Parameters	Description
transactionID [String] - Mandatory	Specifies the unique ID of the transaction.
SQLResultSet [Array of JS Objects] - Mandatory	Specifies the name of the SQL result set.
index [Number] - Mandatory	Specifies the index from which the row is to be retrieved.

Example

```
//The below function specifies the callback function that must be
executed when the execution of the SQL statement is a success also
invokes sqlResultsetRowItem.
function sql_success(transactionId, resultset){
       //logic to process the resultset
       for (var i=0; i<resultset.length; i++){
               var rowItem =kony.db.sqlResultsetRowItem(transactionId,resultset,
i);
               alert("empID:" + rowItem["empID"] + " empName:" + rowItem["empName
+ " depID:" + rowItem["depName"]);
       }
}

//The below function specifies the callback function that must be
executed when the execution of the SQL statement is a failure.
 function sql_errorCallBack(err){
       alert("Error processing sql statement error code=" + err["code"]);
}
```

Return Values

Return Value	Description
rowitem [Array of Objects]	Returns the row item at the specified index
null/nil	Returns null/nil if there is no row available at the specified index

Platform Availability

Available on all platforms* except for SPA in Windows Phone 7.5 and windows Phone 8 browsers, for desktop IE8, IE9, IE10 browsers.*Dummy Implementation on Mobile Web.

40.5.7 kony.db.transaction

This API allows you to execute the specified transaction on the given database. When you invoke this API, it returns immediately and asynchronously executes the transaction.

This is an asynchronous API. This API creates an `SQLTransaction` object.

Important: The table names and column names are case sensitive.

Syntax

```
kony.db.transaction
(dbaseObjectId,transactionCallback,transactionErrorCallback,
successCallback)
```

Input Parameters

Parameter	Description
dbaseObjectId [String] - Mandatory	Specifies the unique ID of the database

Parameter	Description

transactionCallback [Function] - Mandatory	Specifies the callback function that contains the transactions.
	
	For example:
	
	`function callback(dbId){`
	`//SQLTransaction contains implementation of executeSql method`
	`// invoke database.executesql method for sql trasaction`
	`}`

transactionErrorCallback [Function] - Optional	Specifies the callback that must be executed if there is an error in executing the transaction. This callback function is used to roll back the updates to the database.
	
	For example:
	
	`function errorCallback(SQLError){`
	`//write code here`
	`}`

successCallback [Function] - Optional	Specifies the callback that must be executed if the transaction is successful.
	
	For example:
	
	`function successCallback(){`
	`//code here`
	`}`
	
	When the transaction is successful, this callback is executed along with the callback function of the *kony.db.executeSql* API.

Example

```
//The below function specifies the callback function that contains the
transactions.
function myTransactionCallback(dbId) {
    //SQLTransaction contains implementation of executeSql method
    // invokekony.db.executeSql method for sql trasaction
}
//The below function specifies the callback that must be executed if
there is an error in executing the transaction. This callback function
is used to roll back the updates to the database.
function myTransactionErrorCallback(SQLError) {
    // proceed with the logic
}
//The below function specifies the callback that must be executed if
the transaction is successful.
function mySuccessCallback() {
    // proceed with the logic
}
//The below function will invoke transaction
function transaction() {
    var dbName = "konytestDB";
    var version = "1.0";
    var displayName = "demo web SQL Database";
    var estimatedSize = 5 * 1024 * 1024; //5*1024*1024 indicates 5 MB
    var databaseObjectId = kony.db.openDatabase(dbName, version,
displayName, estimatedSize);
    //databaseObjectId contains the unique ID of the database

    kony.db.transaction(databaseObjectId, myTransactionCallback,
myTransactionErrorCallback, mySuccessCallback);
}
```

Return Values

None

Platform Availability

Available on all platforms* except for SPA in Windows Phone 7.5 and windows Phone 8 browsers, for desktop IE8, IE9, IE10 browsers.*Dummy Implementation on Mobile Web.

40.6 kony.ds Namespace

The kony.ds namespace provides data storage functionality. It contains the following API elements.

40.6.1 Functions

The kony.ds namespace contains the following functions.

40.6.2 kony.ds.read

This API provides you an ability to read the data stored on the data store of the mobile device using the given name identifier. A reference to the data table is returned.

Use Cases

You can use this API when you want to:

- fetch the data stored on the mobile device and re-use it.
- know the data associated with a name identifier before you delete it.

Syntax

```
kony.ds.read(name,storeContext )
```

Input Parameters

Parameter	Description
name [String] - Mandatory	Specifies the unique name identifier that represents the data on the data store
storeContext [String] - Optional	Specifies the unique identifier that of the data store if the data store is shared between apps. The storeContext parameter is a dictionary that holds the following keys. • **storeAsUserPreference** [Boolean]: Contains true to access the data in an app group. Otherwise, set to false. • **konyAppGroupID** [string]: Holds a unique identifier that specifies the app group to be accessed.

Example

```
kony.ds.read("friends");
```

Return Values

Return Value	Description
Reference [String]	A reference to the table is returned
nil [nil]	nil is returned if no record is found

Error Codes

The following error codes are returned:

- 704 Unknown error
- 705 Invalid parameters
- 706 Invalid parameter type. A non-string type specified as the name of the datastore

Implementation Details

If you use secure=true, while saving the data in ds.save, then the ds.read API fetches the correct values only if you are invoking the API on a form enabled for Secure Submit. Else, it will return nil.

Platform Availability

Available on all platforms.

40.6.3 kony.ds.remove

This API allows you to delete the data stored on the data store that corresponds to a specific name identifier.

Syntax

```
kony.ds.remove (name, storeContext)
```

Input Parameters

Parameter	Description
name [String] - Mandatory	Specifies the name identifier using which the table was saved.

Parameter	Description
storeContext [String] - Optional	Specifies the unique identifier that of the data store if the data store is shared between apps. The storeContext parameter is a dictionary that holds the following keys. • **storeAsUserPreference** [Boolean]: Contains true to access the data in an app group. Otherwise, set to false. • **konyAppGroupID** [string]: Holds a unique identifier that specifies the app group to be accessed.

Example

```
kony.ds.remove("friends");
```

Return Values

Return Value	Description
status [Boolean]	true - if the table is deleted successfully false - if the table is not deleted, or if no datastore exists with the key specified

Platform Availability

Available on all platforms.

40.6.4 kony.ds.save

This API allows you to save the given data using the specified name identifier in the data store of the mobile device. In case of Mobile Web applications, you have the option to specify where to store the data (cache/session/cookie).

Use Cases

You can use this API to store data on the mobile device that you may require even after quitting the application.

Syntax

```
kony.ds.save(inputtable, name, metainfo, storeContext)
```

Input Parameters

Parameter	Description
inputtable [Array] - Mandatory	Specifies the table that needs to be saved (array of values)
name [String] - Mandatory	Specifies the name identifier for the table. The data is stored on the data store with this name identifier. You must use this identifier if you want to retrieve data from the data store.

Parameter	Description

metainfo [Table] - Optional	Specifies a table with key-value pairs. These key-value pairs are useful for Mobile Web and indicate where to store the data. You have an option to store the data in the user session or cache or create a cookie to be placed on the device.
	
	Important: The following points are applicable only to **metainfo [Table]** parameter:
	• The parameter is mandatory for Server Side Mobile Web.
	• Example code to use:
	`kony.ds.save (["John","Joe","Jack"],"friends", {dsmode:"session"})`
	
	secure means that the data stored on the device store is stored using a secure mode.
	
	Note: secure is applicable only on Mobile Web and must be used in conjunction with dsmode="cookie". For example, {dsmode="cookie", secure="true"}.
	
	You should provide a key value in the metainfo as key:
	`"dsmode" value:"cache/session/cookie"`
	• cache: Data is stored in the application context and shared across all users of the Web Application
	• session: Data is stored in the HTTP session created for a particular user
	• cookie: Data is stored in client browser's cookie store.
	
	Based on the mode you specify here, the availability of the data after you quit the application varies.
	
	For example, `dsmode="session"`.

Parameter	Description
storeContext [String] - Optional	Specifies the unique identifier that of the data store if the data store is shared between apps. The storeContext parameter is a dictionary that holds the following keys. • **storeAsUserPreference** [Boolean]: Contains true to access the data in an app group. Otherwise, set to false. • **konyAppGroupID** [string]: Holds a unique identifier that specifies the app group to be accessed.

Example

```
kony.ds.save(["John","Joe","Jack"],"friends",{dsmode:"session"});
```

Return Values

None.

Error Codes

The following error codes are returned:

- 700 Unknown error
- 701 Invalid parameters specified
- 702 Invalid parameter type.
- 703 Invalid parameter type. A non-string type specified as input
- 704 Unknown error
- 705 Invalid parameters
- 706 Invalid parameter type. A non-string type specified as the name of the datastore

Implementation Details

If you save some other data on the data store with the same name identifier used earlier, data is overridden.

API Usage

We recommend you not to store save sensitive data like passwords, account numbers, and so on using this API.

Platform Availability

Available on all platforms.

41. Operating System API

The Operating System API enables you to access various features provided by the operating system of the device. The Operating System API uses the `kony.os Namespace` and related functions and constants.

Function	Description
`kony.os.addHiddenField`	This API helps the developers to pass dynamic values when the form needs to be submitted to external sites.
`kony.os.addMetaTag`	This API adds a meta tag in html header. This API result will effect only on header reload.
`kony.os.createUUID`	This API returns a string that contains a formatted UUID value.
`kony.os.detectDynamicInstrumentation`	This API helps your application to detect the presence of any Dynamic Instrumentation instance.
`kony.os.deviceInfo`	This API allows the developers to get information about the device in which the application is launched.
`kony.os.endSecureTransaction`	This API can be invoked on an event of a widget.
`kony.os.freeMemory`	This API provides the ability to query and fetch the system-wide memory available on the mobile device for allocation.

Function	Description
kony.os.getAppContext	This API allows the developers to get information about the mode in which the application is launched.
kony.os.getBatteryLevel	Retrieves the current percentage charge level of the device battery, as an integer value.
kony.os.getBatteryState	Retrieves the current state of the battery.
kony.os.getDeviceId	This API returns the unique ID of a device.
kony.os.getDeviceCurrentOrientation	This API returns the current orientation of the device.
kony.os.hasAccelerometerSupport	This API returns whether accelerometer is supported on a device.
kony.os.hasCameraSupport	This API returns whether Camera is supported on a device.
kony.os.hasGPSSupport	This API returns whether GPS is supported on a device.
kony.os.hasOrientationSupport	This API returns whether Orientation is supported on a device.
kony.os.hasTouchSupport	This API returns whether Touch is supported on a device.

Function	Description
kony.os.print	When invoked without any parameter, this API prints the entire form that is currently in view.
kony.os.readHiddenField	This API allows the developers to read the hidden fields added by the os.addHiddenField API.
kony.os.registerBatteryService	Registers for the battery monitoring service of the device operating system.
kony.os.registerSpeechRecognizer	Registers callbacks for speech recognition events.
kony.os.removeAllMetaTags	This API removes all the user defined meta tags from a html header.
kony.os.removeMetaTag	This API removes a specific meta tag from a html header.
kony.os.startSecureTransaction	This API can be invoked on an event of a widget. When this API is invoked it makes all the data and subsequent transactions of the application secure.
kony.os.startSpeechRecognition	Starts the speech recognition process.
kony.os.stopSpeechRecognition	Stops existing (already started with kony.os.startSpeechRecognition API) speech recognition operations.

Function	Description
kony.os.toCurrency	This API allows you to convert the given number to represent currency. At present, only USA currency is supported.
kony.os.toNumber	This API converts the argument to a number.
kony.os.unregisterBatteryService	This API stops the monitoring process of the device battery.
kony.os.unregisterSpeechRecognizer	Deregisters existing (already registered with kony.os.registerSpeechRecognizer API) callbacks for speech recognition events.
kony.os.userAgent	This API returns a unique identifier of the mobile device that is extracted from the user agent.

To view the functionality of the Operating System API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

41.1 kony.os Namespace

The kony.os namespace provides functions for the Operating System API. It contains the following API elements:

- Constants
- Functions
- Objects

41.1.1 Constants

The kony.os Namespace contains the following types of constants.

Battery State Constants

These constants specify the current state of the device battery.

Constant	Description
BATTERY_STATE_CHARGING	Indicates that the state of the device battery as being charged.
BATTERY_STATE_DISCHARGING	Indicates that the state of the device battery as being discharged.
BATTERY_STATE_FULL	Indicates that the state of the device battery charge is completely full.
BATTERY_STATE_UNKNOWN	Indicates that the state of the device battery charge as not known.

Example

When you query for the state of the device battery as shown in this example, any of the four available battery states is returned.

```
var batteryState = kony.os.getBatteryState();

if (kony.os.BATTERY_STATE_CHARGING == batteryState) {

    kony.print("Battery State: Charging");
}
```

Platform Availability

- iOS
- Android
- Windows

41.1.2 Functions

The kony.os namespace contains the following functions.

kony.os.addHiddenField

This API helps the developers to pass dynamic values when the form needs to be submitted to external sites.

Syntax

```
addHiddenField(key,value,private)
```

Input Parameters

Parameter	Description
key [String] - Mandatory	Specifies the key of the hidden field that you would like to add.
value [String] - Mandatory	Specifies the value that corresponds to the specified key in the hidden field.
private [String] - Optional	Specifies if the hidden field can be read using the `kony.os.readHiddenField` API. The expected values for this field are either "private" or "public". The default value of this field id "public". ***Note:*** Only when this value is specified, you can read the hidden value through `kony.os.readHiddenField` API.

Example

```
function addHiddenField() {
    kony.os.addHiddenField("myhiddenfield", "myvalue", "private");
    // private value is specified and hence this hiddenfield can be read
through the kony.os.readHiddenField API
    kony.os.addHiddenField("myhiddenfield1", "myvalue");
    // As private value is not specified and hence this hiddenfield cannot be
read through the kony.os.readHiddenField API
}
```

Return Values

None.

Platform Availability

Available only on Mobile Web.

kony.os.addMetaTag

This API adds a meta tag in html header. This API result will effect only on header reload.

Syntax

```
addMetaTag (key, value)
```

Input Parameters

Parameter	Description
key [String] - Mandatory	Adds a meta tag in the HTML header
value [Object] - Mandatory	Adds meta tag attributes in the html header as key value pairs.

Example

```
kony.os.addMetaTag("test2", {
    "http-equiv": "refresh",
    "content": "30"
});
```

Return Values

This API has no return values.

Platform Availability

Applicable only on Mobile Web.

kony.os.createUUID

UUID (Universally Unique Identifier) is a universally unique value that can be used to identify types, interfaces, and other items. This API returns a string that contains a formatted UUID value. For example, E621E1F8-C36C-495A-93FC-0C247A3E6E5F.

Syntax

```
kony.os.createUUID()
```

Input Parameters

None

Example

```
function createMyUUIDFunc() {
var uuid = kony.os.createUUID();
kony.print("The created UUID is : " + uuid);
}
```

Return Type

String

Platform Availability

- Android
- iOS

kony.os.detectDynamicInstrumentation

This API helps your application to detect the presence of any Dynamic Instrumentation instance. Currently, this API only supports the detection of Frida server presence.

Syntax

```
kony.os.detectDynamicInstrumentation(object)
```

Input Parameters

Parameter	Description
object [Object] - **Mandatory**	This is a dictionary that contains the following keys: • *detectedCallback* [Function] - **Optional** This callback is executed when the presence of Dynamic Instrumentation is detected. If you do not specify this key, the application safely exits by default whenever it detects Dynamic Instrumentation. • *undetectedCallback* [Function] - **Optional** This callback is executed when Dynamic Instrumentation is not detected. • *type* [String] - **Mandatory** This parameter specifies the type of detection that is to be performed. You can specify any one of the following values for this parameter: • fridaquickscan - Returns results quickly and can be used synchronously in JS code. • fridadeepscan - Performs a deep scan to search for the presence of Frida server, and it takes about six or more seconds to return the result. For this reason, you must use the `fridadeepscan` option in WorkerThread.

Example

```
var didobject = {
    detectedCallback: detectedCallbackFunction,
    undetectedCallback: undetectedCallbackFunction,
    "type": "fridaquickscan"
};
kony.os.detectDynamicInstrumentation(didobject);
```

Return Value

None

Platform Availability

- Android

kony.os.deviceInfo

This API allows the developers to get information about the device in which the application is launched.

You can view a video on using Device Info API here.

Syntax

```
deviceInfo()
```

Input Parameters

None

Example

```
var deviceInfo = kony.os.deviceInfo();
alert(deviceInfo);

for (var key in deviceInfo) {
    if (deviceInfo.hasOwnProperty(key)) {
        alert(key + ":" + deviceInfo[key]);
        }
}
```

Return Values

Return Value	Description
values[Object]	Returns a DeviceInfo object.

Platform Availability

Available on all platforms.

---

kony.os.endSecureTransaction

This API can be invoked on an event of a widget. This API should be invoked on forms of the application where user validation is not required or cross site request forgery is not a concern.

Syntax

```
kony.os.endSecureTransaction()
```

Input Parameters

None

Example

```
kony.os.endSecureTransaction();
```

Return Values

None

Platform Availability

Available only on Mobile Web.

---

kony.os.freeMemory

This API provides the ability to query and fetch the system-wide memory available on the mobile device for allocation.

You can use this API to:

- Check the amount of free memory on the mobile device before you go ahead with installation of any software or applications.

- Find out the free memory on the mobile device, clear unwanted objects, and thus improve the performance of the application.

Syntax

```
kony.os.freeMemory()
```

Input Parameters

None

Example

In the following example, kony.os.freeMemory returns the freememory available for allocation.

```
var freememory = kony.os.freeMemory();
kony.print(freememory);
//After the kony.os.freeMemory operation, the memory available for allocation
is printed.
//For example, 1070404 (indicates that 1046 KB of memory is available for
allocation)
```

Return Values

Return Value	Description
Free memory[Number]	The available memory for allocation is returned. The returned memory always indicates the number of **bytes** available.

Platform Availability

Available on all platforms* except Windows Phone 8, Windows Phone 8.1, Windows Tablet 8, and Windows 7 / Kiosk. *Dummy implementation on Server Side Mobile Web, SPA, and DesktopWeb and returns a dummy value.

kony.os.getAppContext

This API allows the developers to get information about the mode in which the application is launched.

Syntax

```
kony.os.getAppContext()
```

Input Parameters

None

Example

```
function getAppContext() {
    var mycontext = kony.os.getAppContext();
    kony.print(mycontext);
    /*prints {launchmode=0} if the application was launched in normal mode
,prints {launchmode=1} if the application was launched in full screen mode*/
}
```

Return Values

Return Value	Description
contextDetails[Object]	Returns an object with key-value pairs: <ul><li>`launchmode:0` indicates that the application is launched in normal mode.</li><li>`launchmode:1` indicates that the application is launched in full screen mode.</li></ul>

Platform Availability

Applicable only on Mobile Web.

kony.os.getBatteryLevel

Retrieves the current percentage charge level of the device battery, as an integer value.

Syntax

```
kony.os.getBatteryLevel()
```

Input Parameters

None

Example

```
kony.os.registerBatteryService();
var batteryLevel = kony.os.getBatteryLevel();
kony.os.unregisterBatteryService();
```

```
getBatteryLevel: function() {
    kony.os.registerBatteryService(this.batterySuccessCallback);
    var battery = kony.os.getBatteryLevel();
    kony.os.unregisterBatteryService();
    this.view.lblDisplay.text = battery + "%";
},
```

Return Values

Returns an integer that ranges from 0-100 (inclusive) that specifies the battery's current charge level in percentage. For example, a return value of 30 specifies that the current charge level of the battery is 30%.

Platform Availability

- iOS
- Android
- Windows

kony.os.getBatteryState

Retrieves the current state of the battery.

Syntax

```
kony.os.getBatteryState()
```

Input Parameters

None

Example

```
var batteryState = kony.os.getBatteryState();
if (kony.os.BATTERY_STATE_CHARGING == batteryState) {
    kony.print(“This is the battery state: charging”);
}
```

```
//This code is used to obtain your device battery state
getBatteryState: function() {
    kony.os.registerBatteryService(this.batterySuccessCallback);
    var batteryState = kony.os.getBatteryState();
    if (kony.os.BATTERY_STATE_CHARGING == batteryState) {
        alert("The Device is charging");
        kony.os.unregisterBatteryService();
    } else if (kony.os.BATTERY_STATE_DISCHARGING == batteryState) {
        alert("The Device is discharging");
        kony.os.unregisterBatteryService();
    } else if (kony.os.BATTERY_STATE_FULL == batteryState) {
        alert("The Device is completely charged");
        kony.os.unregisterBatteryService();
    } else if (kony.os.BATTERY_STATE_UNKNOWN == batteryState) {
        alert("The Device charging state is unkonwn");
        kony.os.unregisterBatteryService();
    }
},
```

Return Values

Returns a constant from the Battery State Constants.

Remarks

The battery state indicates whether it is charging, discharging, and so forth.

Platform Availability

- iOS
- Android
- Windows

kony.os.getDeviceId

This API returns the unique ID of a device: IMEI for GSM phones and MEID or ESN for CDMA phones from Telephony Manager, based on the slot index that you provide. This API works when used on devices that have an operating system of API Level 23 Marshmallow or later, and when the application targets an API Level of 23 or later.

As this API requires the READ_PHONE_STATE permission, the getDeviceId API returns the device ID from Telephony Manager after properly handling permissions and prompts the permission dialog to acquire the Android permission. If you deny the READ_PHONE_STATE permission, a Permission error is displayed.

You must set the READ_PHONE_STATE permission in the Android Manifest file to retrieve the device ID.

> ***Note:*** In Android, the property returns null when used on devices running OS API Level 29 (Android-Q) and above without showing any permission dialog.
>
> Starting in Android Q, apps must have the READ_PRIVILEGED_PHONE_STATE privileged permission (which cannot be granted to a regular app) to access the device's non-resettable identifiers, which include both IMEI and serial number. For more information, see Android Documentation.

Syntax

```
kony.os.getDeviceId(int slot)
```

Input Parameters

Slot Index as an integer value.

Example

```
function getMyDeviceIDFunc() {
    var devId = kony.os.getDeviceId(0); // param is sim slot
    kony.print("The device ID of the first SIM slot is: " + devId);
}
```

Return Type

String

Unsupported Cases

- If your application is running on a device with an operating system earlier than API level 23, the API returns "Null."
- Invalid slot indexes that are not supported by a device are ignored and the API returns "Null."
- The API returns "Null" on some devices, such as tablets, where Telephony Manager is not available.

Platform Availability

- Android

kony.os.getDeviceCurrentOrientation

This API returns the current orientation of the device. The possible values are portrait or landscape.

Syntax

```
kony.os.getDeviceCurrentOrientation()
```

Input Parameters

None

Example

```
kony.os.getDeviceCurrentOrientation();  Returns the device orientation
```

Return Values

This API returns whether the device orientation is landscape or portrait.

- constants.DEVICE_ORIENTATION_PORTRAIT
- constants.DEVICE_ORIENTATION_LANDSCAPE

Platform Availability

Available on iPhone, iPad, Windows 8, Android, Windows 7 / Kiosk, and Desktop Web (Always returns DEVICE_ORIENTATION_PORTRAIT in Desktop Web Channel)

kony.os.hasAccelerometerSupport

This API returns whether accelerometer is supported on a device.

Syntax

```
hasAccelerometerSupport()
```

Input Parameters

None.

Return Values

Return Value	Description
value[Boolean]	Returns whether accelerometer is supported on a device.

Platform Availability

Applicable only on iPhone, Android, and Windows platforms.

Example

```
kony.print(kony.os.hasAccelerometerSupport());
//prints true if device has accelerometer support
```

kony.os.hasCameraSupport

This API returns whether Camera is supported on a device.

Syntax

```
hasCameraSupport()
```

Input Parameters

None.

Example

```
kony.print(kony.os.hasCameraSupport());
//prints true if device has  camera support
```

Return Values

Return Value	Description
value[Boolean]	*true*: the platform supports Camera *false*: the platform does not support Camera

Platform Availability

Available on all platforms* except Mobile Web and Windows 7 / Kiosk. *Dummy implementation for SPA and Desktop Web that always returns False.

kony.os.hasGPSSupport

This API returns whether GPS is supported on a device.

Syntax

```
kony.os.hasGPSSupport()
```

Input Parameters

None.

Example

```
kony.print(kony.os.hasGPSSupport());
//prints true if device has GPS support
```

Return Values

Return Value	Description
value[Boolean]	*true*: the platform supports GPS. *false*: the platform does not support GPS.

Platform Availability

Available on all platforms except Mobile Web and Windows 7 / Kiosk.

kony.os.hasOrientationSupport

This API returns whether Orientation is supported on a device.

You can view a video on using Display Orientation here.

Syntax

```
kony.os.hasOrientationSupport()
```

Input Parameters

None.

Example

```
var orientation = kony.os.getDeviceCurrentOrientation();

if (orientation == constants.DEVICE_ORIENTATION_PORTRAIT) {
    alert("PORTRAIT");
} else if (orientation == constants.DEVICE_ORIENTATION_LANDSCAPE) {
    alert("LANDSCAPE");
} else {
    alert("UNKNOWN");
}
```

Return Values

Return Value	Description
value[Boolean]	*true*: the platform supports orientation. *false*: the platform does not support orientation.

Platform Availability

Available on all platforms except Mobile Web.

kony.os.hasTouchSupport

This API returns whether Touch is supported on a device.

Syntax

```
kony.os.hasTouchSupport()
```

Input Parameters

None.

Example

```
kony.print(kony.os.hasTouchSupport());
//prints true if device is a touch device
```

Return Values

Return Value	Description
value[Boolean]	*true*: the platform supports touch. *false*: the platform does not support touch.

Platform Availability

Available on all platforms except Mobile Web and Desktop Web.

kony.os.print

When invoked without any parameter, this API prints the entire form that is currently in view.

Syntax

```
kony.os.print(containerID)
```

Input Parameters

Parameter	Description
containerID [Number] - Optional	The containerID can be a ID of any container widget that can be directly referenced from a form. ***Note:*** • If the user is typing in a text box, the typed content will not be printed. • Print API can be used only after the target Form, Popup or Datagrid is rendered on the browser. • Print API cannot be used in pre-show, post-show, pre and post app init or any other functionality that is run before the target Form, Popup, or Datagrid is fully rendered. • When printing the form with widgets like ScrollBox, Image Strip, the print functionality prints only those widgets that are in the view or potentially occupy the available print space in portrait or landscape views.

Example

```
kony.os.print()
```

Return Values

None.

Platform Availability

Applicable only on Desktop Web.

kony.os.readHiddenField

This API allows the developers to read the hidden fields added by the `os.addHiddenField` API.

Syntax

```
readHiddenField(key)
```

Input Parameters

Parameters	Description
key [String] - Mandatory	Specifies the key of the hidden field that you would like to read.

Example

```
function readHiddenField() {
    kony.os.readHiddenField("myhiddenfield");
    // Reads the value that corresponds to the myhiddenfield key, i.e.,
myvalue
}
```

Return Values

Return Value	Description
value[String]	Returns the value that corresponds to the specified key.
nil	nil is returned if there is no value assigned to the corresponding key.

Platform Availability

Available only on Mobile Web.

kony.os.registerBatteryService

Registers for the battery monitoring service of the device operating system. The callback is delivered to the most recent registered battery service.

Note: Whenever the battery state changes or for every 1% change in the battery level, a callback to the registerBatteryService function is triggered.

Syntax

```
kony.os.registerBatteryService(callbackMethod)
```

Input Parameters

Parameter	Description
callbackMethod	A JavaScript function that is automatically invoked when you register to the battery monitoring service of the device OS.

Example

```
kony.os.registerBatteryService(callBackservice);
kony.os.registerBatteryService(mybatterychangecallback);
var batterylevel = kony.os.getBatteryLevel();

function mybatterychangecallback(var batteryInfo) {
    var batterylevel = batteryinfo.batterylevel;

    var batteryState = batteryinfo.batterystate;

    if (batterylevel >= 30) {

        // User-defined logic

    }

    if (batterylevel >= 50) {

        kony.os.unregisterBatteryService();

        //You must unregister the battery monitoring service callback to
reduce the overhead.

    }

}
//Here, callbacks are only delivered to the mybatterychangecallback function.
```

```
//This code is used to register a battery service and deregister the service
based on your battery level
registerBatteryService: function() {
    kony.os.registerBatteryService(this.mybatterychangecallback);
    var batterylevel = kony.os.getBatteryLevel();
},
mybatterychangecallback: function(batteryInfo) {
    var batterylevel = batteryInfo.batterylevel;
    if (batterylevel <= 20) {
        alert("The Battery Level is below 20%, make sure that you charge your
device");
    } else {
        kony.os.unregisterBatteryService();
        alert("We are unregistering the Battery Service as it might cause an
overhead");
    }
},
```

Return Values

None

Limitations

- The callback for the registered battery service is delivered only when the application is running; this is because, you can only receive notifications when the application is in the foreground for the iOS, Windows, and Android platforms.
- The callback to the registered battery service is delivered after every one minute duration for iOS; whereas in case of in Android and Windows, the callback is delivered for every 1% change in the battery charge.

Platform Availability

- iOS
- Android
- Windows

kony.os.registerSpeechRecognizer

Registers callbacks for speech recognition events.

Syntax

```
kony.os.registerSpeechRecognizer(callbackTable)
```

Input Parameters

Parameters	Description
callbackTable [JSON object] - Mandatory	The callbackTable is a mandatory key-value pair, that helps you to register JavaScript callbacks, which are triggered when any speech to text recognition events occur.Following are the key-value pairs: ◦ **resultGenerated [callback]** Triggered continuously whenever a phrase is generated out of the recognized speech. • **Parameters for callback [JSON object]** • **result [String]** The recognized phrase of the speech recognition session. **confidence [Number]** • The confidence level of the speech recognition result. For example, if the speech includes a word such as "weight," the confidence level is the certainty with which the app recognizes the word as "weight" and not as "wait." ◦ 0 - High ◦ 1 - Medium ◦ 2 - Low ◦ 3 - Rejected • **status [Number]** The status of the result. ◦ 0 - Success ◦ 1 - TopicLanguageNotSupported ◦ 2 - GrammarLanguageMismatch ◦ 3 -

Example

```
function registerCallbacks() {
    kony.os.registerSpeechRecognizer({
        "resultGenerated": resultGenCallback,
        "timeouts": {
            "initialSilenceTimeout": 5,
            "autoStopSilenceTimeout": 60,
            "endSilenceTimeout": 80
        }
    });
}

function resultGenCallback(data) {
    frmHome.txtareaSpeech.text = "Text: " + data.result + "confidence: " +
data.confidence + "Status: " + data.status;
}
```

Return Values

None.

Platform Availability

Available only on Windows 10.

kony.os.removeAllMetaTags

This API removes all the user defined meta tags from a html header. This API result will effect only on header reload.

Syntax

```
removeAllMetaTags()
```

Input Parameters

None.

Example

```
kony.os.removeAllMetaTags()
```

Return Values

None.

Platform Availability

Applicable only on Mobile Web.

kony.os.removeMetaTag

This API removes a specific meta tag from a html header. This API result will effect only on header reload.

Syntax

```
removeMetaTag (key)
```

Input Parameters

Parameters	Description
key [String] - Mandatory	Removes a meta tag with the specific key in html header.

Example

```
kony.os.removeMetaTag("test1")
```

Return Values

This API has no return values.

Platform Availability

Applicable only on Mobile Web.

kony.os.startSecureTransaction

This API can be invoked on an event of a widget. When this API is invoked it makes all the data and subsequent transactions of the application secure. For example, the login page of an application has the following: user name field, password field, and a button. On the onclick event of the button, the user is verified and navigated to pages with sensitive information. If you want to prevent cross site request forgery or double submissions, you can invoke this API ensuring that all the subsequent transactions are secure.

Syntax

```
startSecureTransaction(callback, scope)
```

Input Parameters

Parameters	Description
callback [Function] - Mandatory	If there are instances where cross site request forgery is attempted, this parameter should comprise a session/request expiry function.

Parameters	Description
scope [Integer] - Mandatory	Specifies whether this API will be valid per request or per session of the application. The possible values are as follows: • 0 - Request Scope: The data in the application is secure only on a per request basis. *Note:* If you press the browser back button on BJS, the token is rendered invalid on the browser back request. • 1 - Session Scope. The data is secure for an entire user session. This is the default value.

Example

```
function callback() {}
kony.os.startSecureTransaction(callback, 1)
```

Return Values

This API has no return values.

Remarks

Whenever os.startsecuretransaction is invoked, a krfid for that session or request is generated internally as a hidden field. The krfid is validated for each transaction/request. If the krfid is invalid, the callback function of os.startsecuretransaction API is invoked, and the request processing fails or a message appears stating that the session has expired.

Platform Availability

Available only on Mobile Web.

kony.os.startSpeechRecognition

Starts the speech recognition process.

Note: Speech recognition callback(s) must be registered before invoking this API. Refer kony.os.registerSpeechRecognizer API for more information.

Syntax

```
kony.os.startSpeechRecognition(successCallback, errorCallback)
```

Input Parameters

Parameters	Description
successCallback [JS Function] - Optional	Triggered when speech recognition has started successfully.
errorCallback [JS Function] - Optional	Triggered if there is an error while starting the speech recognition operation or if the speech recognition operation is already in progress.

Example

```
function startSpeech() {
    kony.os.startSpeechRecognition(successCallback, errorCallback);
}

function successCallback(result) {
    alert("Success " + result);
}

function errorCallback(error) {
    alert("Failure " + error);
}
```

Return Values

None.

Platform Availability

Available only on Windows 10.

kony.os.stopSpeechRecognition

Stops existing (already started with **kony.os.startSpeechRecognition** API) speech recognition operations.

Note: Speech recognition callback(s) must be registered before invoking this API. Refer kony.os.registerSpeechRecognizer API for more information.

Syntax

```
kony.os.stopSpeechRecognition(successCallback, errorCallback)
```

Input Parameters

Parameters	Descrption
successCallback [JS Function] - Optional	Triggered when speech recognition has stopped successfully.

Parameters	Descrption
errorCallback [JS Function] - Optional	Triggered if there is an error while stopping the speech recognition operation or if there is no speech recognition operation in progress to stop.

Example

```
function stopSpeech() {
    kony.os.stopSpeechRecognition(successCallback, errorCallback);
}

function successCallback(result) {
    alert("Success " + result);
}

function errorCallback(error) {
    alert("Failure " + error);
}
```

Return Values

None.

Platform Availability

Available only on Windows 10.

kony.os.toCurrency

This API allows you to convert the given number to represent currency. At present, only USA currency is supported.

Syntax

```
kony.os.toCurrency(number)
```

Input Parameters

Parameters	Description
number [Number] - Mandatory	Specifies the number that must be converted to represent currency. If the input number is a negative number, the negative number is treated as a positive number (this is because a currency does not have any negative symbol) and the converted value is returned.

Example

Perform a *kony.os.toCurrency* operation on the number "*10000*".

```
var tocurrency = kony.os.toCurrency(10000);
kony.print(tocurrency);
//prints $ 10,000
```

Return Values

Return Value	Description
Currency [String]	A string with the number formatted as currency. If the input string has decimal points, the return value is truncated till two decimal points.

Exceptions

An error is thrown if input is invalid or does not follow the expected structure.

102 - Invalid input error

Platform Availability

Available on all platforms.

kony.os.toNumber

This API converts the argument to a number. If the argument is already a number or a string convertible to a number, then the API returns this number; otherwise, it returns **null** for JavaScript.

Syntax

```
kony.os.toNumber(argument)
```

Input Parameters

Parameters	Description
argument [String or Number] - Mandatory	The argument that must be converted to a number.

Example

In this example, only the string which can be converted to a number returns a number otherwise it returns n.

```
kony.os.toNumber (ms34rd);
//returns null as the string passed cannot be converted to a number
kony.os.toNumber ("58");
//returns 58 as the string could be converted to a number
```

Return Values

Return Value	Description
Converted Number [Number]	The input string or number that has been converted to a number and returned.
null/nil	The argument cannot be converted to a number.

Remarks

The input parameter must be a number or a string.

Exceptions

An error is thrown if input is invalid or does not follow the expected structure.

102 - Invalid input error

Platform Availability

Available on all platforms.

kony.os.unregisterBatteryService

This API stops the monitoring process of the device battery. You must call this API when the use of the battery monitoring service has been completed, to reduce the overhead.

> ***Note:*** After your app calls the kony.os.unregisterBatteryService API, the callback function registered by the kony.os.registerBatteryService API is no longer invoked.

Syntax

```
kony.os.unregisterBatteryService()
```

Input Parameters

None

Example

```
kony.os.registerBatteryService(callBackservice);
kony.os.registerBatteryService(mybatterychangecallback);
var batterylevel = kony.os.getBatteryLevel();

function mybatterychangecallback(var batteryInfo) {
    var batterylevel = batteryinfo.batterylevel;
    var batteryState = batteryinfo.batterystate;

    if (batterylevel >= 30) {

        // User-defined logic

    }

    if (batterylevel >= 50) {

        kony.os.unregisterBatteryService();

        //You must unregister the battery monitoring service callback to
reduce the overhead.

    }

}
```

```
//This code is used to register a battery service and deregister the service
based on your battery level
registerBatteryService: function() {
    kony.os.registerBatteryService(this.mybatterychangecallback);
    var batterylevel = kony.os.getBatteryLevel();
},

mybatterychangecallback: function(batteryInfo) {
    var batterylevel = batteryInfo.batterylevel;
    if (batterylevel <= 20) {
        alert("The Battery Level is below 20%, make sure that you charge your
device");
    } else {
        kony.os.unregisterBatteryService();
        alert("We are unregistering the Battery Service as it might cause an
overhead");
    }
},
```

Return Values

None

Platform Availability

- iOS
- Android
- Windows

kony.os.unregisterSpeechRecognizer

Deregisters existing (already registered with **kony.os.registerSpeechRecognizer** API) callbacks for speech recognition events.

Syntax

```
kony.os.unregisterSpeechRecognizer()
```

Input Parameters

None.

Example

```
function unRegisterCallbacks() {
kony.os.unregisterSpeechRecognizer();
}
```

Return Value

None.

Platform Availability

Available only on Windows 10.

kony.os.userAgent

This API returns a unique identifier of the mobile device that is extracted from the useragent. This unique ID represents the device model and the manufacturer. Kony Application Server uses this information to adjust the content to match the screen resolution of the device. For example, the content is adjusted to fit the screen width, height, or memory required etc.

The useragent contains the following information:

- device model
- manufacturer
- OS version
- browser version
- Java capabilities, and so on.

The following are a few sample useragents:

Useragent	Description
`Nokia6230i/2.0 (03.25) Profile/MIDP-2.0 Configuration/CLDC-1.1`	Nokia 6230i model mobile device
`SonyEricssonT610/R501 Profile/MIDP-1.0 Configuration/CLDC-1.0`	SonyEricsson T610 model mobile device
`BlackBerry9500/4.7 Profile/MIDP-2.0 Configuration/`	BlackBerry 9500 mobile device
`BlackBerry9700/5.0 Profile/MIDP-2.1 Configuration/`	BlackBerry 9700 mobile device

Useragent	Description
`Mozilla/4.0 (compatible; MSIE 7.0; Windows Phone OS 7.0)`	Windows Phone mobile device
`OPWV-SDK/62 UP.Browser/6.2.2.1.208 (GUI) MMP/2.0`	Openwave Mobile Browser 6.2.2
`Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)`	Microsoft Internet Explorer 6
`Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7.12) Gecko/20050915 Firefox/1.0.7`	Mozilla Firefox 1.0.7 running on Windows 2000
`Mozilla/5.0(iPad; U; CPU iPhone OS 3_2 like Mac OS X; en-us) AppleWebKit/531.21.10 (KHTML, like Gecko) Version/4.0.4 Mobile/7B314 Safari/531.21.10`	iPad
`Device Model. For example, Galaxy Nexus.`	Android and Android Tablet.

You can use this API in the following scenarios when you need to identify:

- The model of a specific mobile device.
- The mobile devices based on the manufacturer.
- If the device is a web browser, mobile device, micro browser, or a computer.

Syntax

```
kony.os.userAgent()
```

Input Parameters

None

Example

In the following example, the uid returned by the kony.os.userAgent is displayed in the alert.

```
var devID = kony.os.userAgent();
alert("User Agent return value is::" + devID);
```

Return Values

Device ID [String]

Any of the available Device ID is returned in the order Device Model, OS Version, Browser Version, Java Capabilities, and Manufacturer.

> ***Note:***
>
> - For Android and Android Tablet, device model is returned as an user agent.
> - For iOS Devices, user agent is returned as a string.

Platform Availability

Available on all platforms except Windows 7.

41.1.3 Objects

The kony.os namespace contains the following objects.

- DeviceInfo

41.1.3.1 DeviceInfo Object

The DeviceInfo object is a hash table in JavaScript and with the following keys. The following return values are available on all platforms:

- **screenWidth**: Specifies the screen width of the device by considering the device orientation at the time of query. The returned value is in device independent pixels (Dp). This attribute is supported in iOS, Android, Windows, and SPA platforms.
- **screenHeight**: Specifies the screen height of the device by considering the device orientation at the time of query. The returned value is in device independent pixels (Dp). This attribute is supported in iOS, Android, Windows, and SPA platforms.

> ***Note:*** The values returned by the screenWidth and screenHeight properties describe the screen space where widgets can be placed using layout parameters. This includes app menu height, headers, and footers, but excludes the screen space occupied by status bar.

- **deviceID**: The unique device identifier. The following are the device IDs returned by the platforms:
  - Windows Phone: Device Unique ID (20 bytes in length), a unique hash for the device. This value is constant across all applications and does not change even if the phone is updated with a new version of the operating system.
  - iOS 7 and above: A 32-bit unique ID that is generated by using a private hashed constant and the bundle identifier of the application.

    > ***Note:*** From V8 ServicePack4 Fixpack 93, the MAC address is not used to identify the device due to privacy issues. Therefore, instead of using the MAC address, use the identifierForVendor property. For more information on how to use the **identifierForVendor** property in your app, refer the Base Camp article.

  - It internally uses the MAC address of the iOS device and the Bundle Identifier of the application to generate a 32-bit id.

- From V8 SP4 onwards, the deviceID key is available for Desktop Web and SPA platforms with the following features:
    - If a new device is detected for the same user, you can use deviceID to register the user's device and map it with the logged-in user in order to perform an action, such as an email alert on new device login or send a push notification.
    - If a user clears the cache, the next time the user launches the web app on the same device, the user is treated as unique. The deviceID value does not persist after cache cleanup.
    - If a user opens two different web apps on the same device, the deviceID value is also different for both the apps.
    - If a user opens a web app in Incognito mode, deviceID continues to work.
- Android: Returns the unique device ID, for example, the IMEI for GSM and the MEID or ESN for CDMA phones. For application running on APILevel>=23 (Marshmallow) devices, you must use the **uid** property in deviceInfo to get the IMEI for GSM and the MEID or ESN for CDMA phones. For application running on APILevel >= 23 (Marshmallow) devices **ANDROID_ID** is returned as **deviceID** to get rid of acquiring READ_PHONE_STATE permission for device id from Telephony Manager.
- For application running on APILevel <= 22, this property tries to retrieve (Device ID) IMEI for GSM and the MEID or ESN for CDMA phones if available from Telephony Manager.If IMEI/MEID/ESN is not available, in any of the devices such as tablets, **SERIAL_NO** of the device is returned. If both are not present then **ANDROID_ID** is returned as a device-id.
- Set the READ_PHONE_STATE permission in the Android Manifest file to view the device ID. For more information on DeviceiD, http://developer.android.com/reference/android/telephony/TelephonyManager.html#getDeviceId()

- **SERIAL_NO**: Is a hardware serial number which is Alphanumeric and case-insensitive.This is defined for APILevel >= 9. For more information refer, http://developer.android.com/reference/android/os/Build.html#SERIAL. For applications targeting 26 and above and running on OS API Level26 (Oreo) and above, a prompt to acquire android.permission.READ_PHONE_STATE permission appears if the permission is not previously granted by the user. If the user denies the permission, a permission error appears. In this case user also needs to add READ_PHONE_STATE permission in the Android Manifest file. For apps targeting OS API Level25 and below permissions are not required.

> ***Note:*** In Android, the property returns null when used on devices running OS API Level 29 (Android-Q) and above without showing any permission dialog.
>
> Starting in Android Q, apps must have the READ_PRIVILEGED_PHONE_STATE privileged permission (which cannot be granted to a regular app) to access the device's non-resettable identifiers, which include both IMEI and serial number. For more information, see Android Documentation.

- **ANDROID_ID**: In versions of the platform lower than Android 8.0 (API level 26), a 64-bit number (expressed as a hexadecimal string) that is randomly generated when the user first sets up the device and should remain constant for the lifetime of the user's device. On devices that have multiple users, each user appears as a separate device, so the ANDROID_ID value is unique to each user. For more information refer, http://developer.android.com/reference/android/provider/Settings.Secure.html#ANDROID_ID

  Android IDE behavioral changes on Android 8.0 (API level 26) and higher versions of the platform:

  - A 64-bit number (expressed as a hexadecimal string), unique to each combination of app-signing key, user, and device. Values of ANDROID_ID are scoped by signing key and user. The value may change if a factoryreset is performed on the device or if an APK is signing key changes. As a result, apps with different signing keys running on the same device no longer see the same Android ID (even for the same user).

    - For apps that were installed prior to an OTA to a version of Android 8.0 (API level 26), the value of ANDROID_ID remains the same unless uninstalled and then reinstalled after the OTA.
    - The value of ANDROID_ID does not change on package uninstall or reinstall, as long as the signing key is the same (and the app was not installed prior to an OTA to a version of Android 8.0).
    - The value of ANDROID_ID does not change even if a system update causes the package signing key to change.
    - On Android 8.0 (API level 26) and higher versions of the platform, a 64-bit number (expressed as a hexadecimal string), unique to each combination of app-signing key, user, and device. Values of ANDROID_ID are scoped by signing key and user. The value may change if a factory reset is performed on the device or if an APK is signing key changes.
    - In versions of the platform lower than Android 8.0 (API level 26), a 64-bit number (expressed as a hexadecimal string) that is randomly generated when the user first sets up the device and should remain constant for the lifetime of the user's device. On devices that have multiple users, each user appears as a completely separate device, so the ANDROID_ID value is unique to each user.
    - For apps installed on a device running Android 8.0, the value of ANDROID_ID is now scoped per app signing key, as well as per user. The value of ANDROID_ID is unique for each combination of app-signing key, user, and device. As a result, apps with different signing keys running on the same device no longer see the same Android ID (even for the same user).
- **uid**: Returns the unique device IDthe IMEI for GSM and the MEID or ESN for CDMA phones from Telephony Manager. Returns null on some devices such as tablets where Telephony Manager is not available is not available. As this requires READ_PHONE_STATE permission, the uid returns device id from Telephony Manager after properly handling permissions. Using the uid identifier (when used on devices running OS APILevel 23 Marshmallow and above, and when application targets APILevel 23 and above) prompts permission dialog to acquire

android.permission.READ_PHONE_STATE permission if not already granted by the user. If the user denies the permission, a **PermissionError** appears. If your application target is less than OS API level 23, then the above permission dialog would not appear. Set the READ_PHONE_STATE permission in the Android Manifest file to view the device ID.

> ***Note:*** In Android, the property returns null when used on devices running OS API Level 29 (Android-Q) and above without showing any permission dialog.
>
> Starting in Android Q, apps must have the READ_PRIVILEGED_PHONE_STATE privileged permission (which cannot be granted to a regular app) to access the device's non-resettable identifiers, which include both IMEI and serial number. For more information, see Android Documentation.

- **customdeviceid**: For iPhone, a custom device identifier is returned. It internally uses the MAC address of the iOS device and the Bundle Identifier of the application to generate this.

> ***Note:*** The new iPhone retina display contains double the pixels (640) as that of the standard display (320), but iOS treats the resolution as a standard display for measurement and screen output. Websites built for non-retina versions will look very tiny and zoomed out.

- **identifierForVendor**: An alphanumeric string that uniquely identifies a device to the app’s vendor. The value of this property is the same for apps that come from the same vendor running on the same device. A different value is returned for apps on the same device that come from different vendors, and for apps on different devices regardless of vendor.

```
if (iOS version is less than 6.0){
        kony.os.deviceInfo().customdeviceid
} else {
        kony.os.deviceInfo().identifierForVendor
}
```

> ***Important:*** This property is applicable only for iOS 6 and above.

- **name**: Specifies the name of the platform as iPhone, iPad, Windows Phone, thin client (The name thin client is used for SPA/ Desktop Web / Mobile Web), and Android.
- **model**: Specifies the model number of the platform. For example 8900. Not applicable for Mobile Web.
- **deviceWidth**: The device width in pixels. Valid only on Native Apps channels and not applicable for Mobile Web.
- **deviceHeight**: The device height in pixels. Valid only on Native Apps channels and not applicable for Mobile Web.
- **hasforcetouchsupport**: A Boolean value that specifies whether force touch is supported on the device. This value is true if force touch is supported; otherwise, false.

```
function isForceTouchAvailble() {
    var deviceInfo = kony.os.deviceInfo();
    return deviceInfo.hasforcetouchsupport;
}
```

- **APILevel**: Provides the Android API Level based on the SDK Version.

```
function getDeviceAPILevel() {
    var devInfo = kony.os.deviceInfo();
    return devInfo.APILevel;
}
```

The following are the mappings between Android SDK Version and the API Level that is returned:

  - SDK 1.5 - *3*
  - SDK 1.6 - *4*
  - SDK 2.1 - *7*
  - SDK 2.2 - *8*
  - SDK 2.3 - *9*

    - SDK 2.3.3 - *10*
    - SDK 3.0 - *11*

    > ***Note:*** The API Level is returned only for Android.

- For Mobile Web, the following values are returned:
    - type [String] - Specifies if the Mobile Web is Advanced or Basic.
    - category [String] - Provides the Mobile Web category.
      For Advanced Mobile Web, the return value can be iPhone or Android.
    - tcsession [String] - Provides the session ID of the current user.
    - imagecat [String] - Image category based on the device width available in Kony Device Database or returns a default value.

42. Passbook API

Passbook is used to keep things like airline boarding passes, movie tickets, and gift cards all in one place, letting you scan your iPhone or iPod touch to check in for a flight, get into a movie, redeem a coupon, and more.

Use a passbook to store items like boarding passes, movie tickets, and gift cards. The Passbook API lets you scan your iPhone or iPod touch to check-in for a flight, get into a movie, redeem a coupon, and more.

Important: Ensure that you have the iOS Passbook FFI imported into your project before you use Kony's Passbook API. The Passbook FFI is mandatory for the steps in this topic to work.

The Passbook API contains the following objects and related methods:

PassLibrary Object

Method	Description
`addPassWithCompletionCallback`	Presents UI to add pass.
`addPassesWithCompletionCallback`	Presents UI to add multiple passes.
`containsPass`	Returns whether the user's pass library contains a pass.
`getPassWithTypeIdentifierAndSerialNumber`	The proximity value gives a general sense of the relative distance to the beacon.
`getPasses`	Returns the passes in the user's pass library.

Method	Description
`removePass`	Removes the pass from the user's pass library.
`replacePassWithPass`	Replaces a pass in the user's pass library with the given pass.

You can add passes to a pass library using the `addPassWithCompletionCallback` or `addPassesWithCompletionCallback` functions. You can retrieve the available passes using the `getPasses` or `getPassWithTypeIdentifierAndSerialNumber` functions. Use the `replacePassWithPass` function to replace a pass with another pass, and the `removePass` function to remove a pass.

Pass Object

Method	Description
`getAuthenticationToken`	The token used to authenticate with the web service.
`getLocalizedDescription`	The localized description of the pass's kind.

Method	Description
getLocalizedName	Use this property to provide accessibility information for a UI element that represents a pass, such as a cell in a table view.
getLocalizedValueForKeyForDeviceLocale	Returns the localized value for specified field of the pass.
getOrganizationName	The name of the organization that created the pass.
getPassTypeIdentifier	The pass’s type identifier.

Method	Description
`getPassURL`	The URL that opens the pass in the Passbook app.
`getSerialNumber`	A value that uniquely identifies the pass.
`getUserInfo`	Developer-specific custom data.

AddPassesViewController Object

Method	Description
`dismissAnimated`	Used to dismiss the view of add passes view controller.
`getLocalizedDescription`	The localized description of the pass’s kind.

Method	Description
`getLocalizedName`	Use this property to provide accessibility information for a UI element that represents a pass, such as a cell in a table view
`getLocalizedValueForKeyForDeviceLocale`	Returns the localized value for specified field of the pass.
`getOrganizationName`	The name of the organization that created the pass.
`getPassTypeIdentifier`	The pass’s type identifier.

Method	Description
getPassURL	The URL that opens the pass in the Passbook app.
getSerialNumber	A value that uniquely identifies the pass.
getUserInfo	Developer-specific custom data.

You can retrieve the authentication token for a pass using the getAuthenticationToken function. You can retrieve localized information about the pass using the getLocalizedDescription, getLocalizedName, or getLocalizedValueForKeyForDeviceLocale functions. Using the getOrganizationName function, you can retrieve the name of the organization that created the pass. You can retrieve information about the pass using getPassTypeIdentifier, getPassURL, or getSerialNumber functions, and retrieve information about the user using the getUserInfo function. You can dismiss the add passes view using the dismissAnimated function.

Here is an example for creating a **isPassLibraryAvailable** with callbacks:

```
var pass1 = new com.kony.isPassLibraryAvailable();
```

Input Parameters

None

Return Values

Boolean - True if available, false if not available.

Platform Availability

Available only on iOS.

42.1 Interacting with Passes

The KonyPassLibrary helps you interact with the Passes in iOS devices. All the Passes are contained within a PassLibrary.

The PassLibrary object represents the library of passes, and a Pass object represents an individual pass.

42.1.1 Determine the Availability of a Pass Library

The developer must check the availability of the iOS Pass library in the current device.

Use the following snippet to determine the availability of a pass library:

```
var isPassLibraryAvailable = com.kony.isPassLibraryAvailable();

var passLibrary = null;

if (isPassLibraryAvailable) {

   passLibrary = new com.kony.PassLibrary();

}
```

42.1.2 Check whether a Pass is in the Pass Library

Use the containsPass method to determine whether a pass is in the user pass library.

```
var aPass = new com.kony.Pass(path_to_pkpass_file);
var passLibrary = new com.kony.PassLibrary();

if (passLibrary.containsPass(aPass)) {
    // Pass is available in the pass library

}
```

42.1.3 Access Passes from the Pass Library

Use the getPasses method to get all the passes that a developer's application is entitled to access.

42.1.4 Read a Pass from the Pass Library

Use the getPassWithTypeIdentifierAndSerialNumber method to read a pass from pass library.

Developer can use methds such as getOrganizationName, getLocalizedDescription of Pass class to access information about the pass.

Use the getLocalizedValueForKeyForDeviceLocale method to access specific field data for a key.

```
pass = passLibrary.getPassWithTypeIdentifierAndSerialNumber
(identifier, serialNumber);


var organizationName = pass.getOrganizationName();

var localizedDescription = pass.getLocalizedDescription();
```

42.1.5 Add a Pass to the Pass Library

AddPassesViewController is a view controller which can be used to display passes and let users add them to their pass library.

Create a pass with pass data file path, use a AddPassesViewController object to add passes to pass library.

```
function statusCallback(status) {
    if (status == "KonyPKAddPassesViewControllerFinished") {
        // view controller finished adding passes
    } else if (status == "KonyPKAddPassesViewControllerShown") {
        // view controller shown to add passes

    } else if (status == "KonyPKAddPassesViewControllerDismissed") {
        // view controller dismissed
    }


}

var aPass = new com.kony.Pass(path_to_pkpass_file);

var addpassesview = com.kony.AddPassesViewController
([aPass],statusCallback);


addpassesview.presentAnimated(true);
```

42.1.6 Modify a Pass

Application can download a new pass(signed) from server and replace it in the pass library using replacePassWithPass method.

```
var newPass = new com.kony.Pass(path_to_new_pass);
var passLibrary = new com.kony.PassLibrary();
passLibrary.replacePassWithPass(newPass);
```

42.1.7 Remove a Pass from the Pass Library

Use the removePass method of PassLibrary class to remove a pass from pass library.

```
var pass = new com.kony.Pass(path_to_pass);
var passLibrary = new com.kony.PassLibrary();
passLibrary.removePass(pass);
```

42.1.8 Open a Pass in the Passbook application

Use getPassURL property to present the pass in Passbook.

42.2 PassLibrary Object

The PassLibrary Object represents the library of passes. It provides an interface to the user’s library of passes.

Here is an example for creating a PassLibrary with callbacks:

```
var passLibrary1 = new com.kony.PassLibrary();
```

Return Value

Object - com.kony.PassLibrary

42.3 Methods

The com.kony.PassLibrary class contains the following methods.

42.3.1 addPassWithCompletionCallback

Presents UI to add pass.

If the status is KonyPKPassLibraryShouldReviewPasses in completionCallback, present an instance of com.kony.AddPassesViewController with the same pass, to let the user review and add them.

Syntax

```
<<PassObject>>.addPassWithCompletionCallback(pass, completionCallback)
```

Input Parameters

Parameter	Description
pass [Object] - com.kony.Pass - Mandatory	The pass being added.
completionCallback [Function] - Mandatory	The completion callback called after the user. • status [String] Status of the passes adding task to PassLibrary. One of the below values: • KonyPKPassLibraryDidAddPasses • KonyPKPassLibraryShouldReviewPasses • KonyPKPassLibraryDidCancelAddPasses

Example

```
passbook1.addPassWithCompletionCallback(pass, completionCallback);
```

Return Values

None

Platform Availability

Available only on iOS

42.3.2 addPassesWithCompletionCallback

Presents UI to add multiple passes.

If the status is KonyPKPassLibraryShouldReviewPasses in completionCallback, present an instance of com.kony.AddPassesViewController with the same pass, to let the user review and add them.

Syntax

```
<<PassObject>>.addPassesWithCompletionCallback()
```

Input Parameters

Parameter	Description
passes [Array] - Mandatory	The pass being added.

Parameter	Description
completionCallback [Function] - Mandatory	The completion callback called after the user. • status [String] Status of the passes adding task to PassLibrary. One of the below values: • KonyPKPassLibraryDidAddPasses • KonyPKPassLibraryShouldReviewPasses • KonyPKPassLibraryDidCancelAddPasses

Example

```
passbook1.addPassWithCompletionCallback(passes, completionCallback);
```

Return Values

Number

Platform Availability

Available only on iOS

42.3.3 containsPass

Returns whether the user's pass library contains a pass. This method lets you determine that the pass library contains a pass even though your app wouldn't be able to read or modify the pass.

Syntax

```
<<PassObject>>.containsPass()
```

Input Parameters

Parameter	Description
pass [Object] - com.kony.Pass - Mandatory	The pass being queried.

Example

```
passbook1.addPassWithCompletionCallback(passes, completionCallback);
...
var passbook1 = passbook1.addPassWithCompletionCallback(passes,
completionCallback);
```

Return Values

Return Value	Description
Boolean	True if pass is in the PassLibrary.

Platform Availability

Available only on iOS

42.3.4 getPassWithTypeIdentifierAndSerialNumber

The proximity value gives a general sense of the relative distance to the beacon. Use it to quickly identify beacons that are nearer to the user rather than farther away.

Syntax

```
<<PassObject>>.getPassWithTypeIdentifierAndSerialNumber(identifier,
serailNumber)
```

Input Parameters

Parameter	Description
identifier [String] - Mandatory	The pass’s pass type identifier.
serailNumber [String]	The pass’s serial number.

Example

```
passbook1.getPassWithTypeIdentifierAndSerialNumber(identifier,
serailNumber);
...
var passbook1 = passbook1.getPassWithTypeIdentifierAndSerialNumber
(identifier, serailNumber);
```

Return Values

Return Value	Description
com.kony.Pass [Object]	The pass with the given pass type identifier and serial number, or nil if there is no such pass or if the app doesn’t have the appropriate entitlement.

Platform Availability

Available only on iOS

42.3.5 getPasses

Returns the passes in the user’s pass library.

Your app only has access to certain passes, based on its entitlements. Passes that your app doesn’t have access to are not returned. The ordering of the passes is not fixed; calling this method multiple times may return the same passes but in a different order.

Syntax

```
<<PassObject>>.getPasses()
```

Example

```
var passbook1 = passbook1.getPasses();
```

Return Values

Return Value	Description
Array	An array of com.kony.Pass objects.

Platform Availability

Available only on iOS

42.3.6 removePass( )

Removes the pass from the user's pass library. This method does nothing if your app doesn't have the appropriate entitlement. The new pass replaces the existing pass with the same pass type identifier and serial number. If there is no such pass in the user's pass library, the replacement fails.

Syntax

```
<<PassObject>>.removePass()
```

Input Parameters

Parameter	Description
identifier [String] - Mandatory	The pass's pass type identifier.
serailNumber [String]	The pass's serial number.

Example

```
passbook1.removePass(identifier, serailNumber);
```

Return Values

None

Platform Availability

Available only on iOS

42.3.7 replacePassWithPass

Replaces a pass in the user’s pass library with the given pass.

Syntax

```
<<PassObject>>.replacePassWithPass()
```

Input Parameters

Parameter	Description
pass [Object] - Mandatory	The new pass.

Example

```
passbook1.replacePassWithPass(pass);
...
var passbook1 = passbook1.replacePassWithPass(pass);
```

Return Values

Return Value	Description
Boolean	True if the pass was replaced successfully; otherwise false.

Platform Availability

Available only on iOS

42.4 Pass Object

A Pass Object represents a Pass in iOS.

Here is an example for creating a Pass with callbacks:

```
var pass1 = new com.kony.Pass();
```

42.4.0.1 Return Values

Object - com.kony.Pass

42.5 Methods

The com.kony.Pass class contains the following methods.

42.5.1 getAuthenticationToken( )

The token used to authenticate with the web service.

Syntax

```
<<PassObject>>.getAuthenticationToken()
```

Input Parameters

None

Example

```

var pass1 = pass1.getAuthenticationToken();
```

Return Values

Return Value	Description
String	The authentication token of Pass

Platform Availability

Available only on iOS

42.5.2 getLocalizedDescription( )

The localized description of the pass’s kind. You can use this property to provide accessibility information for a UI element that represents a pass, such as a cell in a table view.

Syntax

```
<<PassObject>>.getLocalizedDescription()
```

Input Parameters

None

Example

```

var pass1 = pass1.getLocalizedDescription();
```

Return Values

Return Value	Description
String	localized description of Pass.

Platform Availability

Available only on iOS

42.5.3 getLocalizedName( )

You can use this property to provide accessibility information for a UI element that represents a pass, such as a cell in a table view.

Syntax

```
<<PassObject>>.getLocalizedName()
```

Input Parameters

None

Example

```

var pass1 = pass1.getLocalizedName();
```

Return Values

Return Value	Description
String	localized description of Pass.

Platform Availability

Available only on iOS

42.5.4 getLocalizedValueForKeyForDeviceLocale( )

Returns the localized value for specified field of the pass.

Syntax

```
<<PassObject>>.getLocalizedValueForKeyForDeviceLocale()
```

Input Parameters

Parameter	Description
key [String] - Mandatory	The field's key, as specified in the pass.

Example

```
pass1.getLocalizedValueForKeyForDeviceLocale(key);
...
var pass1 = pass1.getLocalizedValueForKeyForDeviceLocale(key);
```

Return Values

Return Value	Description
String	The localized value for the pass’s field.

Platform Availability

Available only on iOS

42.5.5 getOrganizationName( )

The name of the organization that created the pass.

Syntax

```
<<PassObject>>.getOrganizationName()
```

Input Parameters

None

Example

```

var pass1 = pass1.getOrganizationName();
```

Return Values

Return Value	Description
String	organization name of Pass.

Platform Availability

Available only on iOS

42.5.6 getPassTypeIdentifier( )

The pass’s type identifier.

Syntax

```
<<PassObject>>.getPassTypeIdentifier()
```

Example

```

var pass1 = pass1.getPassTypeIdentifier();
```

Return Values

Return Value	Description
String	The pass’s type identifier.

Platform Availability

Available only on iOS

42.5.7 getPassURL( )

The URL that opens the pass in the Passbook app.

Syntax

```
<<PassObject>>.getPassURL()
```

Input Parameters

None

Example

```

var pass1 = pass1.getPassURL();
```

Return Values

Return Value	Description
String	absolute string of passURL

Platform Availability

Available only on iOS

42.5.8 getSerialNumber( )

A value that uniquely identifies the pass.

Syntax

```
<<PassObject>>.getSerialNumber()
```

Input Parameters

None

Example

```

var pass1 = pass1.getSerialNumber();
```

Return Values

Return Value	Description
String	The serial number of Pass.

Platform Availability

Available only on iOS

42.5.9 getUserInfo( )

Developer-specific custom data.

Only available in iOS7 and above.

Syntax

```
<<PassObject>>.getUserInfo()
```

Input Parameters

None

Example

```
var pass1 = pass1.getUserInfo();
```

Return Values

Return Value	Description
String	userInfo dictionary of Pass.

Platform Availability

Available only on iOS

42.6 AddPassesViewController Object

The AddPassesViewController object provides a view to add passes to the user’s library of passes.

Here is an example for creating a AddPassesViewController with callbacks:

```
var AddPassesViewController1 = new com.kony.AddPassesViewController
(an, statusCallback);
```

Input Parameters

Parameter	Description
an [Array] - Optional	***Note:*** Respects only first pass in the array if the iOS is below 7.0.
statusCallback [Function] - Optional	callback to get the status of adding passes view controller. ◦ *status* [String] - status of the AddPassesViewController.

Return Values

Object - com.kony.AddPassesViewController

42.6.1 Methods

The com.kony.AddPassesViewController class has the following methods:

dismissAnimated

This API is used to dismiss the view of add passes view controller.

Syntax

```
dismissAnimated(
    animated)
```

Input Parameters

Parameter	Description
animated	Set True to dismiss with animation.

Example

```
AddPassesViewController1.dismissAnimated(animated);
...
var AddPassesViewController1 = AddPassesViewController1.dismissAnimated
(animated);
```

Return Values

Returns the authentication token of the Pass.

Platform Availability

Available only on iOS

getLocalizedDescription

The localized description of the pass’s kind. You can use this property to provide accessibility information for a UI element that represents a pass, such as a cell in a table view.

Syntax

```
getLocalizedDescription()
```

Input Parameters

None

Example

```
var AddPassesViewController1 =
AddPassesViewController1.getLocalizedDescription();
```

Return Values

Returns a string containing the localized description of Pass.

Platform Availability

Available only on iOS

getLocalizedName

You can use this property to provide accessibility information for a UI element that represents a pass, such as a cell in a table view.

Syntax

```
getLocalizedName()
```

Input Parameters

None

Example

```
var AddPassesViewController1 = AddPassesViewController1.getLocalizedName();
```

Return Values

Returns a string containing the localized description of Pass.

Platform Availability

Available only on iOS

getLocalizedValueForKeyForDeviceLocale

Returns the localized value for specified field of the pass.

Syntax

```
getLocalizedValueForKeyForDeviceLocale()
```

Input Parameters

Parameter	Description
key	A string that holds the field's key, as specified in the pass.

Example

```
AddPassesViewController1.getLocalizedValueForKeyForDeviceLocale(key);
...
var AddPassesViewController1 =
AddPassesViewController1.getLocalizedValueForKeyForDeviceLocale(key);
```

Return Values

Returns a string that contains the localized value for the pass's field.

Platform Availability

Available only on iOS

getOrganizationName

The name of the organization that created the pass.

Syntax

```
getOrganizationName()
```

Input Parameters

None

Example

```
var AddPassesViewController1 = AddPassesViewController1.getOrganizationName();
```

Return Values

Organization name of Pass.

Platform Availability

Available only on iOS.

getPassTypeIdentifier

The pass’s type identifier.

Syntax

```
getPassTypeIdentifier()
```

Example

```
var AddPassesViewController1 = AddPassesViewController1.getPassTypeIdentifier
();
```

Return Values

Returns a string that contains the pass’s type identifier.

Platform Availability

Available only on iOS

getPassURL

The URL that opens the pass in the Passbook app.

Syntax

```
getPassURL()
```

Input Parameters

None

Example

```
var AddPassesViewController1 = AddPassesViewController1.getPassURL();
```

Return Values

Returns the absolute string of passURL

Platform Availability

Available only on iOS

getSerialNumber

A value that uniquely identifies the pass.

Syntax

```
getSerialNumber()
```

Input Parameters

None

Example

```

var AddPassesViewController1 = AddPassesViewController1.getSerialNumber();
```

Return Values

Returns a string that holds the serial number of the Pass.

Platform Availability

Available only on iOS

getUserInfo

Developer-specific custom data.

Syntax

```
getUserInfo()
```

Input Parameters

None.

Example

```
var AddPassesViewController1 = AddPassesViewController1.getUserInfo();
```

Return Values

Returns the userInfo dictionary of a Pass.

Platform Availability

Available only on iOS7 and above.

43. Payment API

The Payment API facilitates online transactions in various Kony applications.

Use the payment API to execute online transactions on various Kony applications.

The Payment API uses the `kony.payment Namespace` and the following API elements.

Function	Description
`kony.payment.canWeMakePayment`	Determines if users can make payments on their device.
`kony.payment.getPaymentData`	Returns paymentResponseData, which contains the necessary information to complete a payment transaction, in the successCallback.
`kony.payment.getSupportedPaymentNetworks`	Returns the list of available payment networks that are supported by Apple Pay.
`kony.payment.updateTransactonResponse (konyconstant)`	Completes the end-to-end transaction by using the payment token and informing the result of the transaction to the native platform.

43.1 Using Payment APIs

Invoke the `kony.payment.canWeMakePayment` function to check whether payments can be made on a particular device. If the callback function of the canWeMakePayment function returns true, configure the paymentData Object and then invoke the `kony.payment.getPaymentData` function. This invocation displays the Payment Sheet UI of the native platform. The user must fill required data in the Payment Sheet and submit the data. If the payment data is valid, the data along with a payment token is returned in the successCallback. To check the list of available payment networks, use the `kony.payment.getSupportedPaymentNetworks` function. Use the `kony.payment.updateTransactonResponse` to complete the end-to-end transaction by using the payment token and informing the result of the transaction to the native platform.

You can use Payment APIs by following these steps:

1. Call the canWeMakePayment API to check whether payments can be made on a particular device.
2. If the canWeMakePayment API callback returns true, you can configure the paymentData Object and then invoke the getPaymentData(paymentData) asynchronous API. This invocation displays the native platform Payment Sheet UI to users.
3. Users can then fill the Payment Sheet UI and submit the data. After users submit the data and if the payment data is valid, the data along with a payment token is returned to you in successCallback.
4. You can complete the transaction by using the payment token.
5. In Android, the Payment Sheet UI is dismissed when you invoke the getPaymentData API. In IOS, however, the Payment Sheet UI is dismissed only when you invoke the updateTransactonResponse API to complete the end-to-end transaction by using the payment token and informing the result of the transaction to the native platform. The native platform then dismisses the Payment Sheet UI and displays an appropriate message.

43.2 Usage Guidelines

43.2.1 For iOS

1. You must have a paid Apple Developer account to use Apple Pay. To make use of the capabilities of Xcode, you must enable Apple Pay and link the required merchant ID.

2. The PaymentButton class is not available in JavaScript. You can alternatively download the Apple Pay button resources and use them on a Kony button to achieve the look desired for an Apple Pay button. The Apple Pay Buttons and Resources link at the Apple Pay developer site provides you with a zip file that contains an extensive collection of approved button resources for Apple Pay. The Apple Pay Guidelines provide explanations of the permissible modifications that you can make to the provided buttons, as well as guidelines on what colors you may choose to provide the best contrast between the button and your view’s background. In general, you are not allowed to create your own button artwork, but you can stretch the provided artwork to be wider if required.

3. If payment processing is not completed within 30 seconds, the transaction times out and the user is notified. If a timeout occurs, you must cancel the in-progress payment.
   You can refer the Apple Pay Guidelines for more information.

43.2.2 For Android

1. You must install Google Play services version 11.4.x or later on your device.

2. You must add the following tag in the Project Application Tag Child Entries section.
   ```
    <meta-data
   android:name="com.google.android.gms.wallet.api.enabled"
   android:value="true" />
   ```

3. You must add the following tag in the Project Gradle Suffix section.
   ```
     dependencies {
     implementation 'com.google.android.gms:play-services-
   ```

```
wallet:11.8.0' implementation 'com.android.support:support-
v4:26.0.0'
 }
```

4. You can refer Google Test and Deploy for more information on testing and deployment.

For a more hands-on approach on the respective features of various Payments APIs, import and preview the **Kony Payments Feature** sample app by using Kony Visualizer.

DOWNLOAD THE APP

43.3 kony.payment Namespace

The Payment API contains the kony.payment Namespace and the following API elements:

43.3.1 Functions

The kony.payment Namespace contains the following functions.

kony.payment.canWeMakePayment

Determines if users can make payments on their device. When users call this function without specifying the paymentRequirements parameter, it verifies the minimum platform requirements and these requirements differ from platform to platform. Users can enforce additional requirements apart from the minimum platform requirements by using the optional parameter: paymentRequirements.

Syntax

```
kony.payment.canWeMakePayment(canWeMakePaymentCallback,paymentRequirements);
```

Input Parameters

Parameter	Description
paymentRequirements[JSON map object]	It is the only optional parameter supported by this function. Once this parameter is set, the canWeMakePayment function returns true if a user has an existing payment method that matches the criteria specified in paymentRequirements. **Supported Values** `paymentRequirements: {"paymentCardNetworks" :[], "paymentMethodType" :[]}` 1.1 *paymentCardNetworks[JSON List Object]* This key is used to check whether any of the specified network cards is available in a user's payment account. If no payment cards have been added, the canWeMakePayment function always returns false. If no cards are specified ,the default supported card networks will be kony.payment.NETWORK_ MASTERCARD, kony.payment.NETWORK_VISA, kony.payment.NETWORK_DISCOVER, and kony.payment.NETWORK_AMEX. The following constants are applicable for this object: • kony.payment.NETWORK_ MASTERCARD • kony.payment.NETWORK_VISA • kony.payment.NETWORK_DISCOVER • kony.payment.NETWORK_AMEX • kony.payment.NETWORK_JCB • kony.payment.NETWORK_INTERAC

Parameter	Description
canWeMakePaymentCallback [Function Object]	It is a mandatory parameter. If user can execute a payment transaction on their device or if the paymentRequirements criteria is met, the callback is executed with boolean value as true; else, the callback returns false.

Example

```
paymentRequirements = {
    paymentCardNetworks: [kony.payment.NETWORK_MASTERCARD],
    paymentMethodType: [kony.payment.METHODTYPE_CREDIT]
};

function canWeMakePaymentCallback(result) {
    alert(result);
}
kony.payment.canWeMakePayment(canWeMakePaymentCallback, paymentRequirements);
```

Platform Availability

- Android
- iOS

kony.payment.getPaymentData

This asynchronous API returns paymentResponseData, which contains the necessary information to complete a payment transaction, in the successCallback. This generally requires the UI to be shown to the users so that they can select the payment method, shipping address, and other transaction information. Based on the paymentRequestData object values, the Payment Sheet UI input fields and the values in paymentResponseData are framed.

Syntax

```
kony.payment.getPaymentData
(paymentRequestData,successCallback,errorCallback);
```

Input Parameters

1. *paymentRequestData[JSON Map Object]*

Based on this object, the Payment Sheet UI input fields and the values in paymentResponseData are framed.

It has the following key-value pairs:

paymentRequestData : { "paymentCardsInfo" : { }, "shippingAddressInfo" : { }, "billingAddressInfo" : { }, "merchantInfo" : { }, "paymentSummary": { } }

1.1 *paymentCardsInfo[JSON Map Object]*

The information on supported payment cards is configured in this object. It has the following key-value pairs:

paymentCardsInfo : { "paymentCardNetworks" : [ ], "paymentMethodType" : [ ], "supportedCountries" : [ ] }

1.1.1 *paymentCardNetworks[JSON List Object]*

It has the list of card networks to limit payments to specific network cards. The values allowed for this key are as follows:

- kony.payment.NETWORK_MASTERCARD
- kony.payment.NETWORK_VISA
- kony.payment.NETWORK_DISCOVER
- kony.payment.NETWORK_AMEX
- kony.payment.NETWORK_JCB
- kony.payment.NETWORK_INTERAC

Note: If no cards are specified, the default supported card networks will be kony.payment.NETWORK_MASTERCARD, kony.payment.NETWORK_VISA, kony.payment.NETWORK_DISCOVER, and kony.payment.NETWORK_AMEX.

iOS-specific Constants

- kony.payment.NETWORK_SUICA
- kony.payment.NETWORK_QUICPAY
- kony.payment.NETWORK_IDCREDIT
- kony.payment.NETWORK_PRIVATELABEL
- kony.payment.NETWORK_CHINAUNIONPAY
- kony.payment.NETWORK_CARTESBANCAIRES
- kony.payment.NETWORK_CARTEBANCAIRES
- kony.payment.NETWORK_CARTEBANCAIRE

Supported Values

```
paymentCardNetworks : [kony.payment.NETWORK_MASTERCARD, kony.payment.NETWORK_
VISA]
```

1.1.2 *paymentMethodType[JSON List Object]*

It has a list of payment methods to limit payments to specific cards. The values allowed for this key are as follows:

Note: In iOS, 3DS cards are supported by default.

iOS-specific Constants

- kony.payment.METHODTYPE_EMV: For Prepaid payment method type.
- kony.payment.METHODTYPE_CREDIT: For Credit payment method type.
- kony.payment.METHODTYPE_DEBIT: For Debit payment method type.

Example

```
paymentMethodType : [kony.payment.METHODTYPE_CREDIT,kony.payment.METHODTYPE_
DEBIT,kony.payment.METHODTYPE_EMV]
```

Android-specific Constants

- kony.payment.METHODTYPE_PREPAID: For prepaid cards.
- kony.payment.METHODTYPE_ANDROIDPAY: For cards added in Android Pay.
- kony.payment.METHODTYPE_GOOGLE : For cards added in Google Account.

Note: If no payment methods are specified , the default supported payment methods will be kony.payment.METHODTYPE_GOOGLE and kony.payment.METHODTYPE_ANDROIDPAY.

Example

```
paymentMethodType : [kony.payment.METHODTYPE_PREPAID]
```

1.1.3 *supportedCountries[JSON List Object]*

It contains a list of ISO 3166 country codes to limit payments to cards from specific countries. Only values of type String are allowed for this key.

Platform Availability

- iOS

Example

```
supportedCountries : [“IN”,“US”,“AU”]
```

1.2 *shippingAddressInfo[JSON Map Object]*

This object enables the Shipping Address fields in the Payment Sheet UI and shipping address values in paymentDataResponse. All the values entered by users in the Payment Sheet UI are returned to you in paymentDataResponse.

shippingAddressInfo has the following key value pairs:

shippingAddressInfo : { "name" : true, "postalAddress" : true , "email" : true, "phoneNumber" : false, "shippingType" : [ ], "shippingMethod" : [ ] }

1.2.1 *name[boolean]*

If this key value is true, the Name field is enabled under Shipping Address in the Payment Sheet.

Note: By default, the **name** key value is true.

For Android: The name and postalAddress fields are enabled if the **name** key value is true.

Example

```
name : true
```

1.2.2 *postalAddress[boolean]*

If this key value is true, the postalAddress field is enabled under Shipping Address of the Payment Sheet.

Note: By default, the **postalAddress** key value is true.

For Android: The values for the name and postalAddress fields are enabled if the **postalAddress** key value is true.

Example

```
postalAddress : true
```

1.2.3 *email[boolean]*

The default value is false.

For iOS: If this key value is true, the Email field is enabled under Shipping Address of the Payment Sheet.

For Android: By default, your registered email ID for your Google Play Store account is retrieved and that email ID is auto-populated in the Payment Sheet UI. The billingAddressInfo email and the shippingAddressInfo email fields are enabled if the email value is true.

Example

```
email : true
```

1.2.4 *phoneNumber[boolean]*

The default value is false. If this key value is true, the phoneNumber field is enabled under Shipping Address of the Payment Sheet.

For Android: This value is taken into account only if the name or postal address field is enabled. The billingAddressInfo phoneNumber and the shippingAddressInfo phoneNumber fields are enabled if the phoneNumber value is true.

Example

```
phoneNumber : true
```

1.2.5 *shippingType[JSON list object]*

This key specifies the shipping type of the purchased item. It can be any of the following values. The value of this key should be of type "Kony Constant."

Platform Availability

- iOS

iOS-specific Constants

- kony.payment.SHIPPINGTYPE_SHIPPING
- kony.payment.SHIPPINGTYPE_ DELIVERY
- kony.payment.SHIPPINGTYPE_ STOREPICKUP
- kony.payment.SHIPPINGTYPE_ SERVICEPICKUP

Example

```
shippingType : [kony.payment.SHIPPINGTYPE_SERVICEPICKUP]
```

1.2.6 *shippingMethod[JSON List Object]*

It defines the shipping method for delivering physical goods. The value of this key should be of type String. App developer-defined custom values are allowed for this key.

Platform Availability

- iOS

Example

```
"shippingMethod":

[

    {
        "label": "Free shipping",

        "price": "0.00",

        "id": "free",

        "detail": "free delivery"
    },

    {
        "label": "Express shipping",

        "price": "10.00",

        "id": "express",

        "detail": "delivery in 3-4 days"
    }

]
```

1.2.7 *allowedShippingCountryCodes[JSON List Object]*

Adds a collection of ISO 3166-2 formatted country codes of the countries to which shipping is allowed in this transaction. If not specified, all the countries are considered to be allowed.

Platform Availability

- Android

Example

```
allowedShippingCountryCodes : ["US", "CA"]
```

1.3 *billingAddressInfo[JSON Map Object]*

For iOS: This object enables/disables the Billing Address fields in the Payment Sheet UI. By default, the details entered by a user while adding a card is considered for billing address. All the values entered by a user in the Payment Sheet UI are returned to the app developer in paymentDataResponse.

For Android: The details entered by a user while adding a card are taken into consideration for the billing address. This object enables/disables the availability of billing address fields in paymentDataResponse.

It contains the followings key-value pairs:

billingAddressInfo : { "name" : true, "postalAddress" : true ,"email" : true, "phoneNumber" : false }

1.3.1 *name[boolean]*

By default, the Name field is enabled in the Payment Sheet UI.

For iOS: If this key value is true, the Name field is enabled under Billing Address of the Payment Sheet UI.

For Android: If the **name** key value is true, **name** and **postalAddress** are returned in the PaymentDataResponse billing address.

Example

```
name : true
```

1.3.2 *postalAddress[boolean]*

The default value is true.

For iOS: If this key value is true, the postalAddress field is enabled under Billing Address of the Payment Sheet UI.

For Android: If the **postalAddress** key value is true, **name** and **postalAddress** are returned in the PaymentDataResponse billing address.

Example

```
postalAddress : true
```

1.3.3 *email[boolean]*

The default value is false.

For iOS: If this key value is true, the email field will be available under Billing Address of the Payment Sheet UI.

For Android: If the **email** key value is true, the billingAddressInfo email and the shippingAddressInfo email fields are enabled.

Example

```
email : true
```

1.3.4 *phoneNumber[boolean]*

The default value is false.

For iOS: If this key value is true, the phoneNumber field is enabled under Billing Address of the Payment Sheet UI.

For Android: This value is taken into consideration only if name or postalAddress is enabled. The billingAddressInfo phone number and the shippingAddressInfo phone number fields are enabled if the phoneNumber key value is true.

Example

```
phoneNumber : true
```

1.3.5 *billingAddressFormat[Kony Constant]*

This sets the billing address format, which is returned in paymentDataResponse .

The following constants are allowed:

- kony.payment.BILLING_ADDRESS_FORMAT_MIN : When this format is used, the billing address returned contains only minimal information, including name, country code, and postal code. Note that some countries do not use postal codes, so the postal code field will be empty for those countries.
- kony.payment.BILLING_ADDRESS_FORMAT_FULL : When this format is used, the complete address details are returned as the billing address. You must only select this format when it is required to process the order, since it can increase friction during the checkout process and can lead to a lower conversion rate.

Note: By default, the kony.payment.BILLING_ADDRESS_FORMAT_MIN format is applied.

Platform Availability

- Android

Example

```
billingAddressFormat : kony.payment.BILLING_ADDRESS_FORMAT_FULL
```

1.4 *merchantInfo[JSON Map Object]*

This object configures the information of the merchant who is responsible for processing transactions.

It contains the following key-value pairs:

merchantInfo : { "merchantName" : "", "merchantID" : "", "countryCode" : "", "additionalParameters" : { } }

1.4.1 *merchantName[String]*

You must specify the name of the merchant here. It should be of type String. Merchants authorized by native platforms are allowed here. If kony.payment.TOKENIZATION_GATEWAY is selected for tokenizationType, merchantName is mandatory.

Platform Availability

- Android

Example

```
merchantName :  "Stripe"
```

1.4.2 *merchantID[String]*

You must specify the merchant registration ID here. It should be of type String. This is a mandatory field. If merchantID is not specified, error code 103 is returned in errorcallback.

For Android : If kony.payment.TOKENIZATION_GATEWAY is selected for tokenizationType, merchantID is mandatory.

Example

```
merchantID : "pk_test_DCYXN0nOheeRbbf4KlNdUB9I"
```

1.4.3 *countryCode[String]*

Two-letter ISO 3166 country code where the transaction will be processed. Value of this key should be of type String. This field is mandatory. If countryCode is not specified, error code 105 is returned in errorcallback.

Platform Availability

- iOS

Example

```
countryCode :  "IN"
```

1.4.4 *additionalParameters[JSON Map Object]*

This key is used when an app developer wants to add custom options to a merchant's information. You can add custom option key-value pairs here.

Example

```
additionalParameters : {"stripe:version" : 1.5, "TOKENIZATION_TYPE" : GATEWAY}
```

1.4.5 *tokenizationType[Kony Constant]*

This key configures the tokenization of paymentToken that is received in paymentResponseData. The following constants are applicable:

- kony.payment.TOKENIZATION_GATEWAY: This is the default constant. The card selected by the buyer will be tokenized by using Payment Gateway API.
- kony.payment.TOKENIZATION_DIRECT: When this constant is used, the payment method selected by the buyer is returned directly to the integrator.

Notes:

- When kony.payment.TOKENIZATION_GATEWAY is selected, merchantName and merchantID are mandatory.
- When kony.payment.TOKENIZATION_DIRECT is selected, publicKey is mandatory.

Platform Availability

- Android

Example

```
tokenizationType : kony.payment.TOKENIZATION_DIRECT
```

1.4.6 *publicKey[String]*

If kony.payment.TOKENIZATION_DIRECT is selected, publicKey is mandatory. This public key is used to encrypt the returned token.

Example

```
publicKey : "abc123"
```

1.5 *paymentSummary[JSON Map Object]*

This object summarizes the amount, type, and currency code of the payment. It contains the following key-value pairs:

paymentSummary : { currenyCode : "", priceDetails : { } }

1.5.1 *currencyCode[String]*

You must specify the ISO 4217 currency code of the transaction here. This is a mandatory field. If currencyCode is not specified, error code 102 is returned in errorcallback.

Example

```
currencyCode : “USD”
```

1.5.2 *priceDetails[JSON Map Object]*

This object configures the price details of a transaction. It has the following keys:

i. **price**: Price of the transaction. It should follow the regex format: [0-9]+(\.[0-9][0-9])? (for example, "10.45"). This is a mandatory field. If price is not specified, error code 106 is returned in errorcallback.

ii. **priceStatus**: Indicates whether or not the amount is final. The following constants are allowed for priceStatus:

   a. kony.payment.PRICE_STATUS_ESTIMATED: The total price is an estimated price. The final price may still change depending on the selected shipping address and other factors.

b. kony.payment.PRICE_STATUS_FINAL: This is the default value. The total price is the final total price of the transaction, and it does not change based on the selection made by the buyer.

c. kony.payment.PRICE_STATUS_UNKNOWN : This is used when the total price is not known at the time.

iii. **label**: Specific to iOS. A short, localized description of the item. This is a mandatory field. If label is not specified, error code 106 is returned in errorcallback.

iv. **Android** : Total price of the transaction and its status must be specified here.
Example: priceDetails : { "price" : 10.45 , "priceStatus" : kony.payment.PRICE_STATUS_FINAL }

v. **iOS**: The array of payment items are specified here. Payment items can be tax, discount, or amount. Example: priceDetails : [{ "label" : tax , "price" : 10.45, "priceStatus" : kony.payment.PRICE_STATUS_FINAL }, { "label" : discount, "price" : 5.45, "priceStatus" : kony.payment.PRICE_STATUS_FINAL }]

2. *successCallback(paymentResponseData)*

When the getPaymentData call is successful, this callback is invoked with the paymentResponseData Object. The paymentResponseData Object contains shippingAddress, selected card details, and payment token. This information is used to complete the transaction with the merchant.

2.1 *billingAddressInfo*

The billing postal address, name, email, and phone number details are returned in this key.

Platform Availability

- Android

2.1.1 *name*

The billing postal address information is returned in this key.

Example

```
var billingName = paymentResponseData.billingAddressInfo.name;
```

2.1.2 *email*

The billing email address is returned in this key.

Example

```
var billingemail = paymentResponseData.billingAddressInfo.email;
```

2.1.3 *phoneNumber*

The billing phone number is returned in this key.

Example

```
var billing_phone = paymentResponseData.billingAddressInfo.phoneNumber;
```

2.1.4 *locality*

The billing locality is returned in this key. If the city, town, etc., details are not specified, the value defaults to "".

Example

```
var billing_locality = paymentResponseData.billingAddressInfo.locality;
```

2.1.5 *administrativeArea*

The billing administrativeArea is returned in this key. If the state, province, etc., details are not specified, the value defaults to "".

Example

```
var billing_state = paymentResponseData.billingAddressInfo.administrativeArea;
```

2.1.6 *country*

The billing country is returned in this key. If the country details are not specified, its value defaults to "".

For Android : The 2-letter ISO-3166 country code.

For iOS: The country name.

Example

```
var billing_country = paymentResponseData.billingAddressInfo.country;
```

2.1.7 *postalCode*

The billing postalCode is returned in this key. If the postal, zip code, etc., details are not specified, its value defaults to "".

Example

```
var billing_postalCode = paymentResponseData.billingAddressInfo.postalCode;
```

2.1.8 *area*

The billing local area is returned in this key. If it is not specified, the value defaults to "".

Example

```
var billing_area = paymentResponseData.billingAddressInfo.area;
```

2.2 *shippingAddressInfo*

The shipping postal address, name, email, and phone number details are returned in this key.

2.2.1 *name*

The shipping address information is returned in this key.

Example

```
var billingName = paymentResponseData.shippingAddressInfo.name;
```

2.2.2 *email*

The shipping email address is returned in this key.

Example

```
var billingemail = paymentResponseData.shippingAddressInfo.email;
```

2.2.3 *phoneNumber*

The shipping phone number is returned in this key.

Example

```
var billing_phone = paymentResponseData.shippingAddressInfo.phoneNumber;
```

2.2.4 *locality*

The shipping locality is returned in this key. If the city, town, etc., details are not specified, the value defaults to "".

Example

```
var shipping_locality = paymentResponseData.shippingAddressInfo.locality;
```

2.2.5 *administrativeArea*

The shipping administrativeArea is returned in this key. If the state, province, etc., details are not specified, the value defaults to "".

Example

```
var shipping_state =
paymentResponseData.shippingAddressInfo.administrativeArea;
```

2.2.6 *country*

The shipping country is returned in this key. If it is not specified, the value defaults to "".

For Android : The 2-letter ISO-3166 country code.

For iOS: The country name.

Example

```
var shipping_country = paymentResponseData.shippingAddressInfo.country;
```

2.2.7 *postalCode*

The shipping postalCode is returned in this key. If the postal, zip code, etc., details are not specified, the value defaults to "".

Example

```
var shipping_postalCode = paymentResponseData.shippingAddressInfo.postalCode;
```

2.2.8 *area*

The shipping local area is returned in this key. If it is not specified, the value defaults to "".

Example

```
var shipping_area = paymentResponseData.shippingAddressInfo.area;
```

2.3 *selectedCardInfo*

The selected card network, payment method type, and card number are returned in this key.

2.3.1 *cardNetwork*

The selected card network is present in this key. Kony constants are returned in this key.

Example

```
var card_network = paymentResponseData.selectedCardInfo.cardNetwork;
```

2.3.2 *cardPaymentMethod*

The selected card pay method (Credit, Debit, Prepaid) is present in this key. Kony constants are returned in this key.

Example

```
var card_class =  paymentResponseData.selectedCardInfo.cardPymentMethod;
```

2.4 *paymentToken*

Transaction token is returned in this key.

Example

```
var token = paymentResponseData.paymentToken;
```

2.5 *transactionID*

The unique ID assigned to this payment transaction by native platform.

Example

```
var token = paymentResponseData.transactionID;
```

3. *errorCallback[JSON Object]*

When the getPaymentData() call fails, this callback is invoked with any of the error codes:

- 101 for not specifying "price".
- 102 for not specifying "currencyCode".
- 103 for not specifying "merchantName".
- 104 for not specifying "merchantID".
- 105 for not specifying "countryCode".
- 106 for not specifying "label".
- 107 for not specifying "publicKey"

Return Values

If the getPaymentData call is successful, successCallback is invoked with paymentDataResponse; else, errorCallback is invoked.

Example

```
var paymentCardsInfo = {
    "paymentCardNetworks": [kony.payment.NETWORK_MASTERCARD,
kony.payment.NETWORK_VISA],
    "paymentMethodType": [kony.payment.METHODTYPE_PREPAID,
kony.payment.METHODTYPE_ANDROIDPAY,
        kony.payment.METHODTYPE_GOOGLE
    ]
};
var shippingAddressInfo = {
    "name": true,
    "postalAddress": true,
    "email": true,
    "phoneNumber": true,
    "allowedShippingCountryCodes": ["US", "CA"]
};
var billingAddressInfo = {
    "name": true,
    "postalAddress": true,
    "email": true,
    "phoneNumber": true,
    "billingAddressFormat": kony.payment.BILLING_ADDRESS_FORMAT_FULL
};
var merchantInfo = {
    "merchantName": "stripe",
    "merchantID": "pk_test_DCYXN0nOheeRbbf4KlNdUB9I",
    "additionalParameters": {
        "stripe:publishableKey": "pk_test_DCYXN0nOheeRbbf4KlNdUB9I",
        "stripe:version": "5.1.0"
    }
};
var paymentSummary = {
    "currencyCode": "USD",
    "priceDetails": [{
        "label": "Tax",
        "price": 10.45,
        "priceStatus": kony.payment.PRICE_STATUS_FINAL
    }]
};
var paymentRequestData = {
    "paymentCardsInfo": paymentCardsInfo,
    "shippingAddressInfo": shippingAddressInfo,
    "billingAddressInfo": billingAddressInfo,
    "merchantInfo": merchantInfo,
    "paymentSummary": paymentSummary
};
kony.payment.getPaymentData(getPaymentDataSuccessCallback,
getPaymentDataErrorCallback, paymentRequestData);
```

Platform Availability

- Android
- iOS

kony.payment.getSupportedPaymentNetworks

Returns the list of available payment networks that are supported by Apple Pay.

Syntax

```
getSupportedPaymentNetworks()
```

Example

```
var supportedNetworksList = kony.payment.getSupportedPaymentNetworks();
```

Return Values

An array of constants that represent the available payment networks. The values returned by this function are as follows:

- AmEx
- CarteBancaire
- CarteBancaires
- CartesBancaires
- ChinaUnionPay
- Discover
- Interac
- iD
- JCB
- MasterCard
- PrivateLabel
- QUICPay
- Visa

Platform Availability

- iOS

kony.payment.updateTransactonResponse(konyconstant)

For iOS, on receiving the payment token, the Payment Sheet UI is not automatically dismissed. You must invoke this API to complete the end-to-end transaction by using the payment token and informing the result of the transaction to the native platform. The native platform then dismisses the Payment Sheet UI and displays any of the appropriate messages as follows:

- kony.payment.TRANSACTION_SUCCESS : If the transaction is successful.
- kony.payment.TRANSACTION_FAILED : If the transaction is unsuccessful.

Syntax

```
updateTransactionResponse()
```

Example

```
kony.payment.updateTransactionResponse(kony.payment.TRANSACTION_SUCCESS);
```

Platform Availability

- iOS

44. Phone API

The Phone API provides your applications with access to the device's phone functionality, if the device supports it. Use the Phone API to access default applications in the underlying platform of the mobile device.

The Phone API provides your apps with access to the device's phone functionality, if the device supports it.

The Phone API provides you with the following Namespaces and the respective API elements therein:

kony.contact Namespace

Function	Description
`kony.contact.add`	Adds a contact to the address book of the mobile device.
`kony.contact.details`	Displays the details of a given contact and returns a new instance of the contact structure.

Function	Description
`kony.contact.find`	Parses through the address book of the mobile device and looks for contacts that match the input string.
`kony.contact.remove`	Deletes the contact (s) that was returned by the contact.find operation.

To add contacts to the phones address book, use the `kony.contact.add` function. From the list of contacts, you can search for and retrieve a contact using the `kony.contact.find` function, and view details of a specific contact using the `kony.contact.details` function. Use the `kony.contact.remove` function to delete a contact.

To view the functionality of the `kony.contact` Namespace in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

kony.phone Namespace

Function	Description
`kony.phone.addCalendarEvent`	Adds a new event to the device calendar.
`kony.phone.cancelVibration`	Stops the ongoing vibration pattern on a device.
`kony.phone.clearSMSListeners`	Clears the broadcast receiver and the existing listeners that are registered to read the one-time password (OTP) from an SMS message received from the server.
`kony.phone.dial`	Enables your app to call the specified telephone number.

Function	Description
kony.phone.findCalendarEvents	Finds an event in the device's calendar.
kony.phone.generateAppHashKey	Generates a hash key that must be included in the SMS message sent from the server to the user's device.
kony.phone.getRemoveEventOptions	Returns an array of constants that indicates possible actions for removing a particular event depending on the native support on the device.
kony.phone.hasVibratorSupport	Returns whether vibrator is supported on a device.

Function	Description
kony.phone.openEmail	Enables the application to launch the native email client with predefined email addresses, subject, body, and attachments.
kony.phone.openMediaGallery	Enables you to open the picture gallery of the mobile device and pick an existing picture.
kony.phone.performHapticFeedback	Provides various haptic feedback to users.

Function	Description
kony.phone.removeCalendarEvent	Deletes a single event or a series of repeat events starting from the date specified in the eventHandle from the device calendar, depending on the provided removeOption parameter.
kony.phone.retrieveSMS	Reads the one-time password from an SMS message received from the server. The one-time code will then be sent back to the server to complete the verification process.

Function	Description
`kony.phone.sendSMS`	Provides access to the Messaging service of the underlying platform
`kony.phone.startVibration`	Starts the vibration feature on a device, provided that device supports vibration.

Use the `kony.phone.dial` function to call a specific telephone number. To send messages to a specific telephone number, use the `kony.phone.sendSMS` function. Use the `kony.phone.openEmail` function to launch the native email client app, , and open the picture gallery of the device using the `kony.phone.openMediaGallery` function.

To add new Calendar events, use the `kony.phone.addCalendarEvent` function, find calendar events using the `kony.phone.findCalendarEvents` function, and set actions to remove calendar events using the `kony.phone.getRemoveEventOptions` function. Use the `kony.phone.removeCalendarEvent` function to delete events from the Calendar.

On Android and iOS platforms, an app must have the end users' runtime permission to use the **openMediaGallery**, **addCalendarEvent**, **findCalendarEvent**, and the **removeCalendarEvent** functions of the Phone API. If these functions are invoked without the end users' permission, the platform automatically displays a system permission dialog box requesting permission from the end user.

To check whether a device supports vibration, use the `kony.phone.hasVibratorSupport` function. Provide haptic feedback using the `kony.phone.performHapticFeedback` function. To start and stop a devices' vibrate feature, use the `kony.phone.startVibration` and the `kony.phone.cancelVibration` functions respectively.

To view the functionality of the `kony.phone` Namespace in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

44.1 Overview

The functions in this API help you to access default applications in the underlying platform of the mobile device and perform operations. The following are a few operations that you can perform using the APIs in this library:

- Send text messages (SMS).
- Call a specified number.
- Capture location details (GPS or Non-GPS).
- Add/find/delete contacts (from the underlying native Address Book.
- Add/find/delete events (from the native device calendar).
- Access the maps application of the underlying platform.
- Access the media from the media gallery of the device or from an external URL.

To use the addCalendarEvent, findCalendarEvent, openMediaGallery, and removeCalendarEvent phone APIs, your app needs runtime permission from the end-user (to add, find, and remove a calendar event from the device and to open the device's photo gallery). If you call the API without obtaining the permission, platforms automatically pops up a system permission dialog box with "Allow" and "Deny" options, asking the end-user to grant permission to add a contact to the device. This is applicable only for Android and iOS platforms.

If the end-user taps the "Allow" option, platform proceeds to access the underlying OS API. If the end-user taps the "Deny" option, the PermissionError exception is thrown with error code, 2300.

44.1.1 Error Code

2300 - Permission denied.

Example

```
try {
    kony.contact.find('John', false);
    catch (e) {
        if (e instanceof KonyError) {
            if (e.name == “PermissionError” & amp; & amp; e.errorCode
== 2300) alert(“Failed to find contacts due to permission denial”);
        }
    }
```

Note: In Windows Phone 7.5, the calendar event is only added at the application level and not on the phone's default calendar application.

44.2 kony.contact Namespace

The kony.contact namspace, which is a part of the Phone API, provides the following API elements:

44.2.1 kony.contact Functions

The `kony.contact` namespace contains the following functions:

Note: The runtime permissions are applicable only in the iOS and Android platforms.

kony.contact.add

This API adds a contact to the address book of the mobile device.

Important: On Windows Phone 7.5, you have to manually click the Save button to add the contact.

Syntax

```
kony.contact.add(contactDetails)
```

Input Parameters

contactDetails [Table] - Mandatory

A table that has the following key-value pairs.

Key	Description
firstname	specifies the first name of the contact
lastname	specifies the last name of the contact

Key	Description
phone	array of arrays containing *Label* and *number* ***Note:*** The phone number can contain a + (for country code), - (hyphen) (to separate country code, STD code, and the number), and space ((to separate country code, STD code, and the number). No other characters are allowed within a phone number. For example, +919999999999, +91 99999 99999, or +91-99999-99999.

Key	Description
email	array of arrays containing *Label* and *emailID*
postal	array of arrays containing *Label* and address in a table with defined keys
company	array of arrays containing *Label* followed by table with company name and title keys

Key	Description
website	array of arrays containing *url* and *type*. *type* is not applicable for Windows platform. So any *type* value that may be provided for Windows is ignored. The possible types for the website key are as follows. • Android: home, homepage, work, and others • iOS: home, homepage, work, ftp, blog, profile, and others • Windows: Not supported

Example (For website key)

```
{
    "website": [{
        "url": "www.google.com",
        "type": "homepage"
    }, {
        "url": "www.myblog.com",
        "type": "blog"
    }, {
        "url": "www.kony.com",
        "type": "work"
    }, {
        "url": "www.myprofile.com",
        "type": "profile"
    }, {
        "url": "www.facebook.com",
        "type": "profile"
    }]
}
```

Note: firstname and phone fields are mandatory with at least one phone number in phone field

The following are the Labels for different fields in the table:

Keys in the Table	Possible Labels
phone	•Mobile •Home •Work •Other
email	•Home •Work •Other
postal	•Home •Work •Other
company	•Work •Other

Example

```
var contactDetails = {
    "firstname": "Kony",
    "lastname": "User",
    "phone": [{
        "number": "+91 8886558889",
        "name": "mobile"
    }],
    "email": [{
        "id": "kony@gmail.com",
        "name": "work"
    }],
    "postal": [{
        "country": "INDIA",
        "name": "home"
    }],
    "company": [{
        "company": "kony",
        "name": "work"
    }],
    "website": [{
        "url": "www.google.com",
        "type": "homepage"
    }, {
        "url": "www.myblog.com",
        "type ": "blog"
    }, {
        "url": "www.kony.com",
        "type": "work"
    }, {
        "url": "www.myprofile.com",
        "type": "profile"
    }, {
        "url": "www.facebook.com",
        "type": "profile"
    }]
};
kony.contact.add(contactDetails);
```

```
//Contact Details.
var mycontact = {
    firstname: "John",
    lastname: "Steve",
    phone: [{
        name: "mobile",
        number: "9999999999"
    }, {
        name: "home",
        number: "9999999999"
    }],
    email: [{
        name: "home",
        id: "abc@yahoo.com"
    }, {
        name: "work",
        id: "def@kony.com"
    }],
    postal: [{
        name: "home",
        street: "Raheja",
        city: "hyderabad",
        state: "AP",
        zipcode: "500310"
    }],
    company: [{
        name: "work",
        company: "kony",
        title: "architect"
    }]
};
//Adding the contact to your device.
kony.contact.add(mycontact);
```

```
this.view.lblDevContact.text = "Contact is added with firstname = 'John' and
lastname = 'Steve' . Please Check the device contacts.";
```

Return Values

The following are the return values for this API:

Return Value	Description
Reference [Table]	Table containing the information related to the newly created contact is returned.
nil	If the contact was not created.

Error Codes

If any of the field is not stored, the field is ignored. If no field is stored, an error is raised.

Implementation Details

The following are the implementation details:

- firstname and phone field with empty string or nil does not create a contact and the API call is ignored.
- Other fields with empty string or nil or invalid data type assignment are ignored.
- id field is *read only* property and must not be modified by the application.

API Usage

When you use this API with Android emulator that uses Android SDK 2.0 and above, you will be prompted to create an account to add contacts.

Creating an account (Google/GMail) is a mandate to add contacts. This is an Android platform limitation.

Ensure that the following permissions are set for Android:

- READ_CONTACTS
- WRITE_CONTACTS
- GET_ACCOUNTS.

You can set these permissions for Android under **Manifest Properties** in **Project Properties -> Native App -> Android**. For more information about Android Manifest permissions, see *Kony Visualizer User Guide*.

Platform Availability

- iOS
- Android
- Windows

kony.contact.details

This API displays the details of a given contact and returns a new instance of the contact structure.

The API is introduced to improve the performance and for effective utilization of memory. In platforms like Android, contact information is stored in multiple database tables, one table contains the contact id, first name, and lastname, and the another table contains the entire contact details. The contacts application displays only firstname, lastname in the initial screen, and when the user selects a particular contact, the details are shown in next screen.

When you to accommodate the contact details in one call to *contact.find* API, the memory is not sufficient when the contacts are numerous. This results in an out of memory issue and results in performance issues in many applications.

Syntax

```
kony.contact.details(ReferenceTable)
```

Input Parameters

Parameter	Description
ReferenceTable - Mandatory	Specifies the reference table returned by the contact.find API

Example

```
function form1_button15681600315097_onClick_seq0(eventobject) {
    var array = kony.contact.find('John', false);
    var dict = array[0];
    kony.print(kony.contact.details(array[0]));
}
```

```
//Use the below function to retrieve the contact details.
getDetails: function() {
    //Finding the contact whose details are to be retrieved.
    var findContacts = kony.contact.find("John");
    if (findContacts === null || findContacts === "" || findContacts ===
undefined) {
        this.view.lblDevContact.text = "No contacts with the first name is
'John' ";
    } else {
        //Retrieving the contact details.
        var a = kony.contact.details(findContacts);
        this.view.lblDevContact.text = a;
    }

}
```

Return Values

Return Value	Description
Reference [Table]	Table of the contacts retrieved. The reference table retrieved contains a field *photorawbytes*. This is a new property in the existing contact structure that should be populated with raw bytes, just like camera or gallery raw bytes.
nil	If no contact is retrieved.

Example of the reference table returned:

```
{
 firstname = "John", middleName = "Cena"
lastname = "Xyz",

//array of arrays containing "Label" and "number"
phone = {
    {
        name = "mobile", number = "99999999999"
    },


    ////array of arrays containing "Label" and "emailid"
    {
        name = "home", number = "99999999999"
    }, {
        name = "home", number = "5555555555"
    }, .....
},
email = {
    {
        name = "home", id = "abc@yahoo.com"
    }, {
        name = "work", id = "def@kony.com"
    }, ....
},

////array of arrays containing "Label" and address in a table with
defined keys.

postal = {
    name = "home", street = "ABC", city = "hyderabad", state = "AP", zipcode =
"500310"
}, {
    name = "work", street = "XYZ", city = "hyderabad", state = "AP", zipcode =
"500010"
}, ...
},

//array of arrays containing "Label" followed by table with company name
and title keys.
company = {
    {
        name = "work", company = "kony", title = "Tester"
    }, {
        name = "other", company = "abc", title = "VP"
    }, ...
},
id = "platform specific identifier for the contact"
photorawbytes = rawbytes
}
```

Platform Availability

Available on all platforms except SPA, Desktop Web, Mobile Web, and Kiosk,.

kony.contact.find

This API parses through the address book of the mobile device and looks for contacts that match the input string.

Syntax

```
kony.contact.find(firstName, needdetails, filterKeys)
```

Input Parameters

Parameter	Description
firstName [String] - Mandatory	A string that represents the firstname of the contact.

Parameter	Description
needdetails [Boolean] - Optional	When the *needdetails* parameter is not passed, the default value is true. When *needdetails* is set to true, this API returns the complete details of the contact. If *needdetails* is false, *contact.find* returns a list of contacts with a new field called *displayname* in the existing contact structure. All other fields of contact structure are not populated. *displayname* is a combination of firstname, middlename, and lastname.

Parameter	Description
filterKeys [Array of strings] - Optional	You can use the filterKeys parameter to filter your search by passing any of the available keys of this parameter. If filterKeys is specified and needDetails is false, only the display name is returned. The available filter keys are as follows: "firstname", "lastname", "phone", "postal", "email", "company", "photorawbytes", "displayname", and "id". The following platform-specific keys are available for the filterKeys parameter: • For Windows and Android: "middlename" • For Windows: "nickname" and "title"

Example

```
var array = kony.contact.find('*', true, ["email", "phone"]);
var dict = array[0];
kony.print(kony.contact.details(array[0]));
```

```
//Finding the contact whose details are to be retrieved.
var findContacts = kony.contact.find("John");
```

Return Values

Return Value	Description
Reference [Object]	Table containing the list of all the contacts that contain the input string
nil	If there is no contact with the specified first name

Implementation Details

This API searches and matches only the contacts that have the firstname as the specified input string.

Platform Availability

Available on all platforms except SPA, Desktop Web, Mobile Web, Kiosk, and Win Mobile 6x.

kony.contact.remove

This API deletes the contact (s) that was returned by the *contact.find* operation.

Syntax

```
kony.contact.remove(ReferenceTable)
```

Input Parameters

Parameter	Description
ReferenceTable - Mandatory	Specifies the reference table returned by the contact.find API

Example

```
var array = kony.contact.find('John', false);
kony.contact.remove(array);
```

Return Values

Return Value	Description
Reference [Table]	Table of the contacts that is deleted
nil	If there is no contact with the specified first name

Platform Availability

Available on all platforms except SPA, Desktop Web, Mobile Web, and Kiosk.

44.3 kony.phone Namespace

The `kony.phone` namespace provides your apps with the ability to access the phone functionality of the user's device, if it is present on the device. The `kony.phone` namespace provides you with the following API elements:

44.3.1 Functions

The `kony.phone` namespace contains the following functions:

kony.phone.addCalendarEvent

Adds a new event to the device calendar.

Syntax

```
kony.phone.addCalendarEvent(
    eventDetails);
```

Input Parameters

eventDetails

A JavaScript object that contains the following key-value pairs.

Key	Value
summary	A string that contains a short description of the event.

Key	Value
start	A string that holds the start date and time for the event. The format for date is "dd/mm/yyyy", and time is "hh:mm:ss". This must be less than the date and time in the `finish` key,
finish	A string that holds the finish date and time for the event. The format for date is "dd/mm/yyyy", and time is "hh:mm:ss". This must be greater than the date and time in the `start` key.
alarm	An optional number that specifies the time in seconds before the `start` time when the phone must trigger an alarm. This key can be omitted if your app does not need to specify an alarm time.
note	A string that contains the long description of the event.
access	A string that indicated the privacy setting for the event. The values can be `public`, `private`, or `confidential`

repeatConfig **- Optional**

A dictionary that indicates the repeat frequency and endRecurrence values for the event. The dictionary contains two values: repeat and endRecurrence.

repeat

A constant that indicates the repeat frequency. The values are as follows:

- CALENDAREVENT_REPEAT_NONE
- CALENDAREVENT_REPEAT_DAILY
- CALENDAREVENT_REPEAT_WEEKLY
- CALENDAREVENT_REPEAT_MONTHLY
- CALENDAREVENT_REPEAT_YEARLY

endRecurrence [Optional]

A string that stores the recurrence end date and time for the event. The format for the end date and time is "dd/mm/yyyy hh:mm:ss". This date must be greater than the start date and time. For native iOS, if you do not specify the **endRecurrence** value, the events will repeat for 2 years.

Example

```
function addCalendarEvent() {
    try {
        alert("calendar event is about to start");
        var evtobj = {
            summary: "Event started",
            start: "15/04/2012 11:23:45",
            finish: "16/04/2012 12:59:45",
            alarm: 40,
            note: "Event will end at 12.59 PM",
            access: "public",
            repeatConfig: {
                repeat: constants.CALENDAREVENT_REPEAT_MONTHLY,
                endRecurrence: ”21/12/2013 13:00:00”
            }
        }
        kony.phone.addCalendarEvent(evtobj);
        //Adds an event to the device calendar
    } catch (PhoneError) {
        alert("error in addCalendarEvent:: " + PhoneError);
    }


}
```

```
// To add the calendar event to the device using 'kony.phone.addCalendarEvent'
APICalendarAddEvent: function(eventObj) {
    try {

        var today = new Date();
        var dd = today.getDate();
        var mm = today.getMonth() + 1;
        var yyyy = today.getFullYear();
        var h = today.getHours();
        var m = today.getMinutes();
        var s = today.getSeconds();
        if (dd &amp; lt; 10) {
            dd = '0' + dd
        }
        if (mm &amp; lt; 10) {
            mm = '0' + mm
        }
        var date = dd + '/
' + mm + ' / ' + yyyy;
        var stime = h + ":" + (parseInt(m) + 2).toString() + ":" + s;
        this.startTime = date + " " + stime;
        var ftime = (parseInt(h) + 1).toString() + ":" + m + ":" + s;
        this.finishTime = date + " " + ftime;
        if (eventObj["text"] == "Add calendar event") {
            var evtobj = {
                summary: "Event started",
                start: this.startTime,
                finish: this.finishTime,
                alarm: 40,
                access: "public"
            };
            kony.phone.addCalendarEvent(evtobj);
            alert("Calendar event is added with start time = " +
this.startTime + ". Please open device calendar to observe this.");
```

```
        } else {
            var evtobj = {
                summary: "Event started",
                start: this.startTime,
                finish: this.finishTime,
                alarm: 40,
                access: "confidential"
            };
            kony.phone.addCalendarEvent(evtobj);
            alert("Calendar event is added in confidential mode with start
time = " + this.startTime + ".");

        }
    } catch (PhoneError) {
        alert("error in addCalendarEvent:: " + PhoneError);
    }
}
```

Return Values

None

Exceptions

This function throws the following phone exceptions.

Exception	Description
2100	Unable to send the Message
2101	Insufficient Permissions
2102	Cannot open mail, mail not configured

© 2013 by Kony, Inc. All rights reserved

Exception	Description
2103	Cannot open media gallery

This function also throws the following general exceptions.

Constant	Description
100	Invalid parameter type.
101	Invalid number of arguments.
102	Invalid input - thrown when the input is invalid based on the context.
103	Invalid operation - thrown when the operation is invalid based on the context.
104	Not supported error - thrown when the method is not supported at all.

Constant	Description
105	Index out of range.
106	Unknown error

Remarks

You can view a video on using the Calendar here.

Events cannot be added in the past.

When your app calls this function on Android, it must have both the READ_CALENDAR and WRITE_CALENDAR permissions. Due to Android limitations, when your app sets an alarm with this function the number of seconds are rounded to the nearest minute. Events are added to the local calendar with the calendar name as the application name. They cannot be synched to email accounts.

The time zone of events that your app adds is the same as the device's current time zone.

Platform Availability

- iOS
- Android version 4.0 and later
- Windows 10

kony.phone.cancelVibration

This API stops the ongoing vibration pattern on a device. This API is available from V8 SP3 onwards.

Important: For Android, you must define the VIBRATE permission under Manifest Properties.
<uses-permission android:name="android.permission.VIBRATE" />

Syntax

```
kony.phone.cancelVibration()
```

Input Parameters

None.

Example

```
kony.phone.cancelVibration();
```

Return Values

None.

Limitations/Requirements

- Windows
    - The device must have vibration hardware.
    - Device family (Windows OS build version) must be: Windows 10 Creators Update (introduced v10.0.15063.0) or later.
    - API contract must be: Windows.Foundation.UniversalApiContract (introduced v4) or later.

Platform Availability

- Android
- Windows

---

kony.phone.clearSMSListeners

When invoked, this API clears the broadcast receiver and the existing listeners that are registered to retrieve the one-time password (OTP) from the SMS message received.

Syntax

```
kony.phone.clearSMSListeners()
```

Input Parameters

Parameter	Description

SMSListenerType [Constant] - Optional	The SMSListenerType value can be any of the following: • kony.phone.SMS_WITH_USER_CONSENT • kony.phone.SMS_WITHOUT_USER_CONSENT Depending on the parameter value, a specific listener type is cleared. If no type is specified, both type of listeners(if any) are cleared.

Example

```
clearAll: function() {
try {
kony.phone.clearSMSListeners();
} catch (err) {
alert("Error:: " + err);
}
}
```

Return Values

None.

Exceptions

- 101 - Invalid type of arguments to the kony.phone.clearSMSListeners API

Platform Availability

- Android

kony.phone.generateAppHashKey

App hash key is a hash string composed of the app's package name and the app's public key certificate. When invoked, this API generates a hash key that must be included in the SMS message sent from the server to the user's device.

For an app to receive the message callback, the hash key has to be correct.

Note: This API must be invoked only if type of SMS listener triggered is kony.phone.SMS_WITHOUT_
USER_CONSENT

Syntax

```
kony.phone.generateAppHashKey()
```

Input Parameters

None.

Example

```
generateAppHashKey: function() {
    var key = kony.phone.generateAppHashKey();
    kony.print("AppHashkey: " + key);
}
```

Return Values

Return Value	Description
appHashKey[String]	Returns a hash key that has to be appended to the SMS message by server.

Platform Availability

- Android

kony.phone.dial

Enables your app to call the specified telephone number.

You can use this API to make calls to other numbers without leaving the application.

Syntax

```
kony.phone.dial(
    number);
```

Input Parameters

Parameter	Description
number	A string containing the phone number to call. Only numbers, +, - and space are treated as valid characters in this string.

Example

```
function CallTheNumber() {

    try {
        var number = "123456789";
        kony.phone.dial(number);
    } catch (err) {
        alert("error in dial:: " + err);
    }
}
```

```
dial: function() {
    var number = this.view.tbxDial.text;
    kony.phone.dial(number);
},
```

Return Values

None

Exceptions

2101 - Insufficient Permissions

Remarks

When you make a call to the specified number using this API, the underlying OS is used to make the call. The call charges as imposed by the service provider are applicable.

On the iPhone simulator, this function is a dummy call. You need to test this function on an actual iPhone device.

Platform Availability

Not supported on Windows Kiosk,Windows 8, iPad, Android Tablets, Desktop Web, and Mobile Web platforms. You can use the phone widget to achieve the same functionality on Mobile Web.

kony.phone.findCalendarEvents

This function finds an event in the device's calendar.

Syntax

```
kony.phone.findCalendarEvents(
    evtobj);
```

Input Parameters

Parameter	Description
evtobj [Object] - Mandatory	A JavaScript object that can contain the following values: • type: "starting" or "ending" or "occuring" • start: The start date and time. • finish: The end date and time.

Example

```
function findCalendarEvent() {
    try {
        alert("Finding the calendar events");
        var evtobj = {
            type: "starting",
            start: "15/03/2012 12:46:45",
            finish: "24/12/2012 12:59:45"
        };
        var events = kony.phone.findCalendarEvents(evtobj);
        alert("Found calendar event(s)");
    } catch (err) {
        alert("error in findCalendarEvents:: " + err);
    }

}
```

```
var evtobj = {
    type: "starting",
    start: this.startTime,
    finish: this.finishTime
};
var events = kony.phone.findCalendarEvents(evtobj);
```

Return Values

If no matching event is found, this function returns null. If matching events are found, this function returns an array of JavaScript objects containing the matching events. Each JavaScript object contains the following key-value pairs.

Key	Value
summary	A string that contains a short description of the event.

Key	Value
start	A string that holds the start date and time for the event. The format for date is "dd/mm/yyyy", and time is "hh:mm:ss". This must be less than the date and time in the `finish` key,
finish	A string that holds the finish date and time for the event. The format for date is "dd/mm/yyyy", and time is "hh:mm:ss". This must be greater than the date and time in the `start` key.
alarm	An optional number that specifies the time in seconds before the `start` time when the phone must trigger an alarm. This key can be omitted if your app does not need to specify an alarm time.

Key	Value
note	A string that contains the long description of the event.
access	A string that indicated the privacy setting for the event. The values can be `public`, `private`, or `confidential`

Exceptions

This function throws the following phone exceptions.

Exception	Description
2100	Unable to send the Message
2101	Insufficient Permissions
2102	Cannot open mail, mail not configured
2103	Cannot open media gallery

This function also throws the following general exceptions.

Constant	Description
100	Invalid parameter type.
101	Invalid number of arguments.
102	Invalid input - thrown when the input is invalid based on the context.
103	Invalid operation - thrown when the operation is invalid based on the context.
104	Not supported error - thrown when the method is not supported at all.
105	Index out of range.
106	Unknown error

Remarks

The search criteria in the type key in the *evtobj* parameter can be any of the following values.

Constant	Description
starting	Searches for events starting between "*start*" and "*finish*".
ending	Searches for events ending between "*start*" and "*finish*".
occurring	Searches for events that have any part of the event occurring during the period specified between "*start*" and "*finish*". This is the default.

READ_CALENDAR permission is needed to use this function.

This function is supported on Android 4.0 version and above (that is, Android API 14 and later). Invocation on Android version less than 4.0 (that is, an Android API level earlier than 14) is ignored.

This function searches only those events that are added by the application.

Platform Availability

- iOS
- Android 4.0 or later
- Windows 10

kony.phone.getRemoveEventOptions

This API returns an array of constants that indicates possible actions for removing a particular event depending on the native support on the device. One of the values that this API returns must be passed as removeOption for kony.phone.removeCalendarEvent API.

Syntax

```
kony.phone.getRemoveEventOptions(eventHandle)
```

Input Parameters

Parameter	Description
eventHandle - Mandatory	Single event from the array of events that the kony.phone.findCalendarEvents API returns.

Example

```
var removeOptions = kony.phone.getRemoveEventOptions(eventHandle);
```

Return Values

An array with possible constant values depending on native support in devices. The possible constants that will be returned as part of the array are as follows:

Constant	Description
CALENDAREVENT_REMOVE_THIS	Indicates that only one event must be deleted.
CALENDAREVENT_REMOVE_THIS_AND_FUTURE	Indicates that the event as well as future events must be deleted.

Note: Depending on the OS configuration of certain Android devices, any of these two values may not be applicable. So before passing any of these two values in the API, you must verify if the Android device supports the required value.

Platform Availability

- iOS
- Android
- Windows 10

kony.phone.hasVibratorSupport

This API returns whether vibrator is supported on a device. This API is available from V8 SP3 onwards.

Important: For Android, you must define the VIBRATE permission under Manifest Properties.
<uses-permission android:name="android.permission.VIBRATE" />

Syntax

```
kony.phone.hasVibratorSupport()
```

Example

```
kony.phone.hasVibratorSupport();
```

```
hasVibratorSupport: function() {
    if (kony.phone.hasVibratorSupport() === true) {
        alert("The device contains a Vibration motor");
    }
},
```

Input Parameters

None.

Return Values

Boolean is the return value.

Return Value	Description
true	The device supports vibrator.
false	The device does not support vibrator.

Limitations/Requirements

- Windows
    - The device must have vibration hardware.
    - Device family (Windows OS build version) must be: Windows 10 Creators Update (introduced v10.0.15063.0) or later.
    - API contract must be: Windows.Foundation.UniversalApiContract (introduced v4) or later.

Platform Availability

- Android
- Windows

---

kony.phone.openEmail

This API allows the application to launch the native email client with predefined email addresses, subject, body, and attachments.

Syntax

```
kony.phone.openEmail(torecipients,ccrecipients, bccrecipients, subject,
messagebody, ismessagebodyhtml, attachments, viewMode, filterEmailAppsOnly)
```

Input Parameters

Parameter	Description
torecipients [Array of Strings] - Mandatory	List of email addresses to be included in the "to" list. For example, "john@example.com","stephen.joseph@kony.com", and "test@somecompany.com".

Parameter	Description
ccrecipients [Array of Strings] - Optional	List of email addresses to be included in the "cc" list. For example, "john@example.com","stephen.joseph@kony.com", and "test@somecompany.com". If you do not want to use this parameter, you can pass null value.
bccrecipients [Array of Strings] - Optional	List of email addresses to be included in the "bcc" list. For example, "john@example.com", "stephen.joseph@kony.com", and "test@somecompany.com". If you do not want to use this parameter, you can pass null value.
subject [String]- Optional	Subject to be part of the email. If you do not want to use this parameter, you can pass null value.
messagebody [String] - Optional	Body to be part of the email. If you do not want to use this parameter, you can pass null value.
ismessagebodyhtml [Boolean] - Optional	If you do not want to use this parameter, you can pass null value. When you pass null value, this parameter defaults to false. • **true**: message body must be treated as HTML content • **false**: message body must not be treated as HTML content ***Note:*** BlackBerry and Windows Phone 8/8.1 platforms do not support HTML body. This is an underlying SDK limitation.

Parameter	Description
attachments [Array of Objects] - Optional	Each attachment is a Hash-table with the following key-value pairs. If you do not want to use the ***attachments*** parameter, you can pass null value. • **mimetype [String]** Standard mime types like image/png or image/jpg or image/* etc. • **attachment [Object]** Rawbytes received from the camera, image widget, or openmediagallery api. • **filename (Optional) [String]** name of the file to appear as attachment. *Note:* • In Android, *filename* property is not supported.The Android SDK does not provide any provision for giving file name in attachments while launching native email Application. • For Windows 8 applications, attachments cannot be added programmatically. This is a limitation from Microsoft. • For more information on how to attach files in Android, click here.

Parameter	Description
viewMode [Integer] - Optional	Defines the possible view preferences for an email client window. This is a Windows-specific parameter. The values of viewMode are as follows: • 0 - Default value Defaults to half, which means, the window uses 50% (half) of the available horizontal screen pixels. • 1 - UseLess The window uses less than 50% of the available horizontal screen pixels. • 2 - UseHalf The window uses 50% (half) of the available horizontal screen pixels. • 3 - UseMore The window uses more than 50% of the available horizontal screen pixels. • 4 - UseMinimum The window uses the minimum horizontal pixel width (either 320 or 500 pixels) specified in the app's manifest file. • 5 - UseNone The window uses no visible component. If you pass any value other than an integer, the "openEmail : mode must be integer" error message is displayed. ***Note:*** viewMode is specific only to Windows 8 tablets and is ignored for all other platforms. For all other devices, the viewMode parameter falls back to its default behavior.

Parameter	Description
filterEmailAppsOnly [Boolean] - Optional	Set this parameter to true to filter and list only email applications, which are shown in the Chooser dialog box. By default, the value of this parameter is false. This is an Android-specific parameter. If you do not want to use this parameter, you can pass null value.

Example

```
var to = ["abc@kony.com", "some.one@somecompany.com"];
var cc = ["manager@somecompany.com"];
var bcc = null; // bcc is not required to setting null
var sub = "Test Message";
var msgbody = "This is a test message";
var viewMode = 0; // just need default behavior for windows so setting to 0
var filterEmailAppsOnly = true; // to filter list to show only email
applications
// Additional set up is required to enable sharing app private files in
Android. Click here to learn more about the setup procedure.
// http://docs.kony.com/konylibrary/visualizer/viz_api_dev_
guide/Default.htm#sharefilesandroid.htm
var path = kony.io.FileSystem.getDataDirectoryPath();
var sharedDir = path + constants.FILE_PATH_SEPARATOR + "images";
var sharefolder = new kony.io.File(sharedDir).createDirectory();
var myFileLoc = sharedDir + constants.FILE_PATH_SEPARATOR + "testfile.txt";
var myFileName = new kony.io.File(myFileLoc).createFile();
try {
    var writing = new kony.io.File(myFileLoc).write("How are you?");
    var reading = new kony.io.File(myFileLoc).read();
    var attachFile =

    {
        mimetype: "text/plain",
        attachment: reading,
        filename: "testfile.txt"
    };
    kony.phone.openEmail(to, cc, bcc, sub, msgbody, true, attachFile,
viewMode, filterEmailAppsOnly);
} catch (e) {}
```

```
emailSend: function() {
    kony.phone.openEmail(["kitchensinkapp@kony.com"], [""], [""], "Feedback on
KitchenSink Application 1.1", "", false, []);
}
```

Return Values

None

Exceptions

PhoneError

The following error codes are defined for Phone APIs

- 2100 - Unable to send the Message
- 2101 - Insufficient Permissions
- 2102 - Cannot open mail, mail not configured
- 2103 - Cannot open media gallery

Platform Availability

Available on all platforms except Server Side Mobile Web, SPA, Kiosk, and Desktop Web.

For these platforms you can launch the native email client by using Richtext widget with an *href* tag similar to the one shown below:

<a href="mailto:email@kony.com?subject=Foo&body=Bar">Email Me</a>

kony.phone.openMediaGallery

This API allows you to open the picture gallery of the mobile device and pick an existing picture.

You can use this API to use any of the multimedia resources (image files, audio files, or video files) available in the device media gallery within your application.

For iPad, openMediaGallery will show the Photos/Videos in the native popover. The popover can be anchored to a widget. The widget to which the popup should be anchored will be provided as a third parameter to openMediaGallery API.

You can view a video on using iPad Popover here.

Syntax

```
kony.phone.openMediaGallery(onselectioncallback, querycontext, PSP)
```

Input Parameters

Parameter	Description
onselectioncallback [Function] - Mandatory	This callback function is invoked when a media is selected. The function receives rawbytes and permStatus as parameters. • **rawbytes:** Raw bytes of a selected file. • **permStatus:** Permission status whether an app has permission to access the media gallery of the device. **For iOS**: Generally, an app needs [runtime permission]() from the end-user to access the media gallery. If you call the API without obtaining the permission, platform automatically pops up a system permission dialog box with "Allow" and "Deny" options, asking the end-user to grant permission to add a contact to the device. If the end-user taps the "Allow" option, platform proceeds to access the underlying OS API. If the end-user taps the "Deny" option, the rawbytes parameter carries null value and the permStatus parameter holds the kony.application.PERMISSION_ DENIED constant value indicates the permission to access the media gallery is denied. **For Android**: This parameter reads the external storage permission that is required to read contents from the media gallery. If you call the API without obtaining the required permission, the platform automatically pops up a system permission dialog containing 'Allow' and 'Deny' options, asking the end user to grant the necessary permission. If the end user taps the 'Allow' option, the platform proceeds to access the underlying OS API. If the end user taps the 'Deny'

Parameter	Description

querycontext [Table] - Optional	Query context is an Object that can be populated with key-value pairs for fine tuning the media items for display (currently only one key ***"mimetype"*** supported to query the gallery items). The possible values of the mimetype key can be ***image/*, video/*, audio/**** .
	
	rawbytes.getRawbytesType ()
	
	This API returns the type of rawbytes that are obtained from onSelectionCallback of openMediaGallery. Rawbytes can be of different types such as image, audio, video, file, bytes, and livePhoto. This API is available from V8 SP3 onwards.
	
	`var rawbytesType = rawbytes.getRawbytesType();`
	
	To check the obtained rawbyte type, the following constants can be used:
	
	• constants. RAWBYTES _IMAGE: rawbytes related to image
	• constants. RAWBYTES _VIDEO: rawbytes related to video
	• constants. RAWBYTES _AUDIO: rawbytes related to audio
	• constants. RAWBYTES _FILE: rawbytes related to files
	• constants. RAWBYTES _BYTES: rawbytes related to bytes
	• constants. RAWBYTES _LIVEPHOTO: rawbytes related to livePhoto
	
	`if(rawbytes.getRawbytesType() == constants. RAWBYTES_LIVEPHOTO) {`
	`    // app logic`

Parameter	Description
PSP [Table] - Optional	The PSP parameter is a dictionary with the following properties. • *widgetref [Widget Reference]*: Applicable only on iPad. The widget to which the pop-up will be anchored to. For example, *formname.widgetname*. • *compressionlevel*: Compression level is a float value. Specify 0.0 for most compressed images and 1.0 for least compressed images.

Example

```
function openMediaGallery() {
    try {

        function onselectioncallback(rawbytes) {
            alert("in selection callback");
            if (rawbytes == null) {
                alert("nothing selected");
                return;
            }
            alert("return status is " + returnStatus);
            //(convert the rawbytes to base64 and can be assigned to a image
widget or use base64 to upload)
        var base64 = convertToBase64(rawbytes);
        frmTest.imgone.base64 = base64;
        //Assigning rawbytes directly to a image widget
        /*formid.imageid.rawdata = rawbytes;*/
    }
    var querycontext = {
        mimetype: "video/*"
    };
    returnStatus = kony.phone.openMediaGallery(onselectioncallback,
querycontext);

} catch (err) {
    alert("error in openMediaGallery:: " + err);
}

}
```

```
openMediaGalleryForEmail: function() {
    kony.phone.openMediaGallery(this.openMediaGallaeryCallBck, {
        mimetype: "image/*"
    });
}
```

Return Values

None

Exceptions

PhoneError

The following error codes are defined for Phone APIs

- 2100 - Unable to send the Message
- 2101 - Insufficient Permissions
- 2102 - Cannot open mail, mail not configured
- 2103 - Cannot open media gallery

Platform Availability

Supported on all platforms except Mobile Web, SPA, Kiosk, and Desktop Web.

kony.phone.performHapticFeedback

This API provides various haptic feedback to users, and is available from V8 SP3 onwards.

Syntax

```
kony.phone.performHapticFeedback(hapticFeedbackValue)
```

Input Parameters

hapticFeedbackValue [constant] - Mandatory

Indicates the type of haptic feedback. Its possible values are as follows:

hapticFeedbackValue	Description
0 - kony.hapticFeedback.SUCCESS	Triggers a haptic feedback for a successful event.
1 - kony.hapticFeedback.WARNING	Triggers a haptic feedback that represents a warning to users.
2 - kony.hapticFeedback.FAILURE	Triggers a haptic feedback for a failure event.
3 - kony.hapticFeedback.LIGHT	Triggers a haptic feedback of light intensity.
4 - kony.hapticFeedback.MEDIUM	Triggers a haptic feedback of medium intensity.
5 - kony.hapticFeedback.HEAVY	Triggers a haptic feedback of high intensity.
6 - kony.hapticFeedback.SELECTION	Triggers a haptic feedback that represents a selection/state change.

Example

```
kony.phone.PerformHapticFeedback(kony.hapticFeedback.SUCCESS);
```

```
hapticFeedback: function(){
  kony.phone.performHapticFeedback(5);
}
```

Return Values

None.

Remarks

- iOS
  - The Haptic Feedback feature is available on iPhone 7 devices and later. These devices have Taptic Engine hardware and users can enable/disable Haptics from Device Settings-> Sounds & Haptics-> System Haptics.
- Android
  - Users can enable the Vibrate on touch feature from Settings-> Sound & notification-> Other sounds.

Platform Availability

- iOS

kony.phone.removeCalendarEvent

This API either deletes a single event or a series of repeat events starting from the date specified in the eventHandle from the device calendar, depending on the provided removeOption parameter. While deleting a series of recurring events, the API deletes all subsequent events beginning from the date specified in the eventHandle; however, the API does not delete any repeat event that is before the specified date.

Note: To delete an event, you must first search for the event by using the findCalendarEvents API and delete the event.

Syntax

```
kony.phone.removeCalendarEvent(eventHandle, removeOption)
```

Input Parameters

Parameter	Description
eventHandle [Object] - Mandatory	Event handle is returned by the find operation.
removeOption - Optional	This parameter is considered only for recurring events. If you do not specify a value for this parameter or pass an invalid value for a recurring event, the current event as well as future recurrences are deleted. Default fallback value of removeOption is constants.CALENDAREVENT_REMOVE_THIS_AND_FUTURE. You can fetch possible removeOption values for delete events by using the kony.phone.getRemoveEventOptions API. The following constants are available for removeOption: • CALENDAREVENT_REMOVE_THIS : Indicates that only one event should be deleted. • CALENDAREVENT_REMOVE_THIS_AND_FUTURE : Indicates the event as well as future events should be deleted.

Example

```
function removeCalendarEvent() {
    try {
        var evtobj = {
            type: "starting",
            start: "20/03/2014",
            finish: "21/04/2014"
        };
        var events = kony.phone.findCalendarEvents(evtobj);
        kony.phone.removeCalendarEvent(events[1]), constants.CALENDAREVENT_
REMOVE_THIS_AND_FUTURE);
    alert("Event deleted successfully");
    // It deletes an event that matches the specified
    // event handle from the device calendar
} catch (err) {
    alert("error in removeCalendarEvent:: " + err);
}

}
```

```
//To remove the calendar event from the device using
'kony.phone.removeCalendarEvent' API

CalendarRemoveEvent: function() {
    if (this.startTime === "undefined" || this.startTime === undefined) {
        this.view.lblPhone.text = "Please create the calendar event.";
        return;
    }
    var evtobj = {
        type: "starting",
        start: this.startTime,
        finish: this.finishTime
    };
    var events = kony.phone.findCalendarEvents(evtobj);
    kony.phone.removeCalendarEvent(events[0]);
    alert("Calendar event is removed. Please open device calendar to observe
this");
}
```

```
kony.phone.removeCalendarEvent(events[1],constants.CALENDAREVENT_REMOVE_THIS);
kony.phone.removeCalendarEvent(events[1],constants.CALENDAREVENT_REMOVE_THIS_
AND_FUTURE);
kony.phone.removeCalendarEvent(events[1]); //If it is non-recurring, the
current event is deleted; however, if it is a recurring event, the event as
well as future events will be deleted.
```

Return Values

None.

Exceptions

PhoneError

The following error codes are defined for Phone APIs

- 2100 - Unable to send the Message
- 2101 - Insufficient Permissions
- 2102 - Cannot open mail, mail not configured
- 2103 - Cannot open media gallery

Error

- 100 - invalid type of parameters
- 101 - invalid number of arguments
- 102 - invalid input - thrown when the input is invalid based on the context.
- 103 - invalid operation - thrown when the operation is invalid based on the context.
- 104 - not supported error - thrown when the method is not supported at all.

- 105 - index out of range.
- 106 - unknown error.

Platform Availability

- iOS
- Android
- Windows 10

Remarks

- READ_CALENDAR and WRITE_CALENDAR permissions are needed for this API.
- This API is supported for Android 4.0 version and above (that is, Android API level later to 13). Invocation on Android version less than 4.0 (that is, Android API level earlier to 14) will be ignored.
- If this API removes last calendar event added to local calendar, then the local calendar account will be deleted.

kony.phone.retrieveSMS

When invoked, this API reads an SMS message received from the server. The API then retrieves the one-time password (OTP) from the message if it matches with the config provided.

Syntax

```
kony.phone.retrieveSMS(callback, SMSConfig);
```

Input Parameters

SMSConfig [Object] - Optional

SMSConfig supports the following key-value pairs:

Key	Description

OTPSizeLimit [Number] - Optional	Expected/desired size limit of the OTP. If the OTPSizeLimit is not specified, the default size limit is 6.
OTPType [Constant] - Optional	Expected/desired type of OTP. [kony.phone.SMS_ALPHANUMERIC_OTP or kony.phone.SMS_NUMERIC_OTP] The default type is Numeric.
OTPRegex [String] - Optional	Customized regex to retrieve OTP or PIN as required. If customized regex is provided, all the other criteria such as OTPSizeLimit and OTPType are neglected.

© 2013 by Kony, Inc. All rights reserved

SMSListenerType [Constant] - Optional	Type of SMS listener to be triggered. Following are the SMS listener types: **kony.phone.SMS_WITHOUT_USER_CONSENT:** Using this constant, OTP would be retrieved automatically without any user interaction. In this case, the message that you send from the server to the user's device must fulfill the following: • Be no longer than 140 bytes • Contain a one-time code that the client sends back to your server to complete the verification flow • Include an 11-character hash string that identifies your app To generate a hash key, use any of the following ways: • The command-line procedure to generate hash. For more information on computing an app's hash string, refer here. • The generateAppHashKey API to generate the hash key when signed in with the production keystore. Here is a sample message format: 123ABC78 FA+9qCX9VSu (123ABC78 is the one-time code and FA+9qCX9VSu is the hash string) **kony.phone.SMS_WITH_USER_CONSENT:** The User Consent Screen is displayed to users after the SMS with an OTP is received. The OTP would be retrieved only after user grants access to the SMS through the User Consent Screen. In this case, the message that you send from the server to the user's device must fulfill the following: • Contain a 4-10 character alphanumeric string with at least one number. • Is sent by a phone number that's not in the user's contacts.

SMSSenderDetails [String] - Optional	SMS sender details may be specified only if `SMSListenerType = kony.phone.SMS_WITH_USER_CONSENT.` In this case, OTP will be retrieved only if the SMS was sent by that phone number specified in SMSSenderDetails key. The sender's phone number should be of E.164 format.

callback [Function] - Mandatory

The callback executed to communicate the SMS Listener Registration status and response of API. The syntax of the callback function is:

```
callback(callbackObject)
```

The callbackObject contains SMS and status key value pairs.

SMS key: SMS object, the value of the SMS key is populated only when code is equal to kony.phone.SMS_ RETRIEVED_SUCCESSFULLY. In case of any error, the SMS object is null.

The SMS object contains the following key-value pairs:

Key	Description
message [String]	SMS received
OTP [String]	Retrieved OTP from SMS

> ***Note:*** The OTP value returned can be null/empty if the message doesn't contain any OTP matching the SMSConfig.

status key: Value of the status key contains the following key-value pairs:

Key	Description
code[Number]	status code
message[String]	status message

The code key can have any of the following values. A code value is populated when the scenario in the corresponding description occurs.

The following codes specify if the framework is successful in starting the SMS listener.

Code	Description
kony.phone.SMS_ LISTENER_ REGISTRATION_ SUCCESS	SMS listener registered successfully. You must request the server for OTP only after the successful registration status is conveyed in a message callback.
kony.phone.SMS_ LISTENER_ REGISTRATION_ FAILED	SMS listener registration failed.

The following codes are applicable only after kony.phone.SMS_LISTENER_REGISTRATION_SUCCESS is received.

Code	Description
kony.phone.SMS_ RETRIEVED_ SUCCESSFULLY	SMS retrieved successfully.
kony.phone.SMS_ TIMEOUT	The SMS listener has timed out as No SMS is received within 5mins of trigger of API.
kony.phone.SMS_ DEVELOPER_ERROR	Caller app has incorrect number of certificates. Only one certificate is allowed. This can occur only when SMSListenerType is set to kony.phone.SMS_WITHOUT_USER_CONSENT.
kony.phone.SMS_ ERROR	App-code collides with other installed apps. This can occur only when SMSListenerType is set to kony.phone.SMS_WITHOUT_USER_ CONSENT.
kony.phone.SMS_ USER_CANCELLED	While SMSListenerType = kony.phone.SMS_WITH_USER_ CONSENT, user denies OTP retrieval. This can occur only when SMSListenerType is set to kony.phone. .SMS_WITH_USER_ CONSENT.
kony.phone.ACTIVITY_ NOT_FOUND	While SMSListenerType = kony.phone.SMS_WITH_USER_ CONSENT, an activity cannot be found to launch the Consent dialog. This can occur only when SMSListenerType is set to kony.phone. .SMS_WITH_USER_CONSENT.
kony.phone.SMS_ UNKNOWN_ERROR	Any error with unknown root cause.

Example

```
retrieveOTPFromSMS: function() {



    SMSConfig={}
    SMSConfig.OTPLength = 4;
    SMSConfig.OTPType = kony.phone.SMS_NUMERIC_OTP;
    SMSConfig.SMSListenerType = kony.phone.SMS_WITH_USER_CONSENT;
    SMSConfig.SMSSenderDetails = null;
    try{
        kony.phone.retrieveSMS(callback,SMSConfig);
    }
    catch(err){
        kony.print("ERROR:"+err);
    }



function callback(callbackObj){
statusCode = callbackObj.status.code;

kony.print("STATUS_CODE::"+statusCode);

if(statusCode == kony.phone.SMS_LISTENER_STARTED_SUCCESS){
//ADD CODE TO INDICATE SERVER TO SEND OTP
 }

if(statusCode==kony.phone.SMS_RETRIEVED_SUCCESSFULLY){
kony.print("message:"+callbackObj.SMS.message)
kony.print("otp:"+callbackObj.SMS.OTP)
alert(callbackObj.SMS.OTP)
}


}
```

Return Values

None

Limitations

You must start an SMS retriever only after the previous registered retriever responds with a success or failure response. However, if you register a second SMS retriever before the first SMS retriever provides a response, the possible limitations are as follows:

- When two SMS retrievals without a user's consent run simultaneously, the SMS callbacks are triggered in the same sequence in which they are registered. This sequence might not be same as the sequence in which the user has sent the requests to the server. As a result, a mismatch in the provided OTP config might occur.
- When two SMS retrievals with a user's consent provide two unique sender details for two parallel SMS retrieval registrations, the SMS callbacks are triggered in the same sequence in which they are registered. The SMS details received at the native level does not contain the senders information. Consequently, a mismatch between the senders might occur.

Exceptions

Error Codes:

- 100 - Invalid number of arguments to API kony.phone.retrieveSMS()
- 101 - Invalid type of arguments to API kony.phone.retrieveSMS()

Platform Availability

Android

---

kony.phone.sendSMS

This API provides access to the Messaging service of the underlying platform. You can send a text message to a specified number. This API accesses the messaging application of the underlying platform and uses that application to send text message to the specified number.

> ***Important:*** You can send only text messages and multimedia messages are not supported.

> ***Note:*** When you send messages using this API, the charges as imposed by the Service Provider are applicable. On iPhone this API opens the Messages application with a prepopulated text and number.

You can use this API to send messages to other numbers without leaving the application.

Syntax

```
kony.phone.sendSMS(phonenumber, text)
```

Input Parameters

Parameter	Description
phonenumber [String] - Mandatory	Number to which the SMS must be sent.
text [String] - Mandatory	Content of the SMS.

Example

```
function sendSMS() {
    try {

        var phNo = "123456789";
        var msg = "hello world";
        kony.phone.sendSMS(phNo, msg);
    } catch (err) {
        alert("error in sending sms:: " + err);
    }

}
```

```
sendSMS: function() {
    if (kony.os.deviceInfo().model == "iPhone Simulator") {
        alert("Works only on device");
    } else {
        var phoneNo = "+91 40 12345678"; // This is a dummy number
        var text = "Hi Kony developer";
        kony.phone.sendSMS(phoneNo, text);
    }
}
```

Return Values

None

Exceptions

PhoneError

The following error codes are defined for Phone APIs

- 2100 - Unable to send the Message.
- 2101 - Insufficient Permissions.

Remarks

When you send messages using this API, the messaging application of the underlying platform is used to send the message with the number and text you specified i.e., the phone number and the message are pre-populated in the messaging application.

Platform Availability

Available on all platforms except Desktop Web,SPA, Mobile Web, Android Tablets, iPad, Kiosk.

> ***Note:*** This API allows you to send only text messages. It does not support multimedia messages.

---

kony.phone.startVibration

This API starts the vibration feature on a device, provided that device supports vibration. If the device does not support vibration, this API has no effect. If a vibration pattern is already in progress when this API is called, the previous pattern is halted and the new one begins. This API is available from V8 SP3 onwards.

> ***Important:*** For Android, you must define the VIBRATE permission under Manifest Properties.
> **<uses-permission android:name = "android.permission.VIBRATE" />**

Syntax

```
kony.phone.startVibration (vibrationConfig, repeat)
```

Input Parameters

Parameter	Description
vibrationConfig [Array] - Mandatory	Specifies the list of all vibration patterns as an array. Each vibration pattern object accepts the **duration**, **amplitude**, and **delay** keys. **delay [Integer] - Mandatory** Specifies that the vibration will be paused for the specified delay time in milliseconds. This parameter must be a positive number. The default value of delay is 0 milliseconds. Any invalid or negative values specified for delay reverts to the default value, i.e., 0 milliseconds. **duration [Integer] - Mandatory** The number of milliseconds for which the device vibrates. This parameter must be a positive number. The default value of

Parameter	Description
repeat [Boolean] - Optional	**true** : The repeat vibration pattern is repeated until the **kony.phone.cancelVibration** API is called. **false** : Stops the repeat vibration pattern. This is the default value.

Example

```
//Device will vibrate for 500 milliseconds.
kony.phone.startVibration([{
    "delay": 0,
    "duration": 500,
    "amplitude": 250
}]); //Device will vibrate for 100 milliseconds, then pause for 200
milliseconds, and the device will vibrate again for another 300 milliseconds.
var config = [{
    "delay": 0,
    "duration": 100,
    "amplitude": 250
}, {
    "delay": 200,
    "duration": 300,
    "amplitude": 150
}];
kony.phone.startVibration(config, false);
```

```
//Device will vibrate for 100 milliseconds.

startVibration: function()
  {
    if(kony.phone.hasVibratorSupport()===true)
      {
        var vibrationConfig=[{
    "delay": 0,
    "duration": 100,
    "amplitude": 250
}];
        kony.phone.startVibration(vibrationConfig, true);
      }
    else{
        alert("The device does not support vibration");
    }
  }
```

Return Values

None.

Behavior, Requirements, and Limitations

- iOS
    - The startVibration API generates only a single vibration effect.
    - iOS does not support config parameters, such as amplitude and duration, for startVibration API.
    - The startVibration API accepts arguments on iOS devices, but it ignores them. The API does not throw any error/exception.
- Windows
    - The device must have vibration hardware.
    - Device family (Windows OS build version) must be: Windows 10 Creators Update (introduced v10.0.15063.0) or later.
    - API contract must be: Windows.Foundation.UniversalApiContract (introduced v4) or later.

Remarks

- You can call this API with or without arguments for Android and Windows. If the arguments are not specified, the device vibrates for 100 milliseconds to align with the behavior for iOS.

Platform Availability

- Android
- iOS
- Windows

---

45. Request App Review API

45.1 Overview

Using the Request App review API, you can get feedback about your applications from users. Both Apple’s App Store and Google’s Play store allow you (the developer) to request feedback on apps. Users can rate your app on a scale of one to five stars and also write a review. These reviews will help you in gaining new users or make improvements in your applications.

App users provide ratings and reviews on the Apple App Store and Google Play Store to give feedback on their experience of using an app. Ratings and reviews influence how your app ranks in search results and can affect whether someone downloads your app. Users can rate your app on a scale of one to five stars and also write a review. Using the kony.application.requestReview function you can enable the review feature and invoke the a user’s rating and review for an app. You can then respond to the provided feedback to improve your apps discoverability, encourage downloads, and build a rapport with the app users.

Typically, app users provide ratings and reviews on Apple App Store and Google Play Store to give feedback on their experience of using an app. This helps other users ascertain which apps they would like to use. You, as an app developer, can ask the users of your app for ratings and written reviews. You can then respond to the provided feedback in order to improve your app’s discoverability, encourage downloads, and build a rapport with the app users.

Ratings and reviews influence how your app ranks in search results, and can affect whether someone downloads your app. Users can rate your app on a scale of one to five stars. They can also write a review.

The Request App Review API contains the following API element:

Function	Description
kony.application.requestReview Function	Requests users to provide a rating and to write a review of an app.

Using the`kony.application.requestReview Function`, you can enable the review feature and request the user to provide a review for the app.

45.2 How Request App Review Works

45.2.1 In iOS

You need to call the requestReview function. The requestReview function internally calls the requestReview function of the SKStoreReviewController API, which gives a prompt to app users to provide feedback on the App Store without leaving the app. The prompt is displayed for users up to three times a year, as determined by Apple. Users can submit a rating through this standardized prompt. When you call this method in your shipping app and a rating/review request view is displayed, the system handles the entire process for you. The Request App Review API is available only on iOS 10.3 and later.

The Request App Review API should not be called on any user action. Refer this link to understand when it is appropriate to ask users for ratings: https://developer.apple.com/ios/human-interface-guidelines/system-capabilities/ratings-and-reviews - system-rating-and-review-prompts.

Note: If you call the requestReview function while your app is still in development mode, a rating/review request view is always displayed so that you can test the user interface and experience. However, this method has no effect when you call that function in an app, which you can distribute by using TestFlight.

45.2.2 In Android

By default, Android does not have the Play Store Reviewable API with the corresponding UI element. Consequently, you can build your own business logic by using the existing kony.ui.Alert function. Developers can directly control the Cancel and Rate callbacks.

When you invoke the requestReview function, the following course of action occurs:

a. Initially, framework searches for the relevant Google Play Store market URL ("**market://details?id=**").

b. If the required URL is not available, framework searches for a web browser, using which it can launch Google Play Store ("**https://play.google.com/store/apps/details?id=**").

Error Codes

The following error codes have been identified for Android:

1. **Error code**: 801
   Message: Review Action Not Found
   This error is displayed if the requested Play Store or browser is not found, or if they are not installed.

2. **Error code**: 802
   Message: Review Error Unknown

> ***Note:*** The Request App Review API directly searches in Google Play Store. However, this search process does not work for Amazon Store and BlackBerry 10. This is because the Google Play Store page opens directly and an "Item Not Found" message is displayed.

To view the functionality of the Request App Review API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

45.3 requestReview Function

The details of the kony.application.requestReview function, which is part of the kony.application Namespace, are as follows.

kony.application.requestReview

This function requests users to provide a rating and to write a review for an app.

Syntax

```
kony.application.requestReview()
```

Input Parameters

None.

Example

```
kony.application.requestReview();
```

```
 //To request the user for his rating on your app, use the below API
 requestAppReview: function(){
   kony.application.requestReview();
 }
```

Return Values

None.

IDE/CodeGen Requirements

None.

Platform Availability

- iOS
- Android

Error Codes

The Android error codes are as follows:

- **Error code**: 801
  Message: Review Action Not Found
  This error code is displayed if the requested Play Store or browser is not found.
- **Error code**: 802
  Message: Review Error Unknown

46. Runtime Permission API

Permissions that are granted/denied at the app run time are known as runtime permissions. These permissions are not requested by the app during its install process. Apps require runtime permissions to access the resources of the device or end-user's data. The runtime permissions give users more control on the application functionality. Mostly, users grant these permissions. Without users' acknowledgment, the app cannot access the user's confidential data. This approach helps the users to secure their data. For example, a user may grant permission to an app to access the device's Camera, but not the contacts. The runtime permissions also help fasten the install process - as the end-users need not grant permissions at the installation of the app.

Kony provides a set of APIs that helps you check the status of the permission, request and obtain permission to access a particular resource. By checking the status of the permission, the app will know whether a permission is granted by default or need to request for it. If an app needs to request for a permission, you can set up a prompt at the app run time to make the user acknowledge. Based on the user's acknowledgment, the app can access a particular resource or may not. That is, if user grants permission, the app can access the requested API. If denies, the app cannot access the resource.

Permissions that are granted/denied at the app's run time are known as runtime permissions. These permissions are not requested by the app during its install process. Apps require runtime permissions to access the resources of the device or the end-user's data. The runtime permissions give users more control over the functionality of the application. Kony provides a set of APIs that helps you check the status of the permission, request and obtain permission to access a resource.

The Runtime Permissions API uses the `kony.application Namespace` and contains the following functions.

Function	Description
`kony.application.checkPermission`	Checks for and returns the permission status of one or more resources.

Function	Description
`kony.application.requestPermission`	Requests for the end-user's consent to access a particular resource.
`kony.application.requestPermissionSet`	Sends a request for a set of permissions. The status of the request is sent back to the user through a callback.
`kony.application.openApplicationSettings`	Opens the application-specific settings or the device-level application settings.

To check the status of a permission for a resource, use the `kony.application.checkPermission` function. You can then request the user’s consent to access a resource or set of resources by using the `kony.application.requestPermission` or `kony.application.requestPermissionSet` function.Use the `kony.application.openApplicationSettings` function to open the Application settings to grant permissions.

In the kony.application.requestPermission function, specify the following Resource constants to proceed with the API.

- RESOURCE_EXTERNAL_STORAGE = 1009
- RESOURCE_PHOTO_GALLERY = 1011;

For Android, make sure that the WRITE_EXTERNAL_STORAGE permission is defined under the Manifest Properties. Here are the steps to define:

1. Open the application, and click **Project Settings** icon. The **Project Settings** window opens.
2. Click the **Native** tab, and then click **Android** sub-tab.
3. Under the **Manifest Properties**, click the **Permissions** tab.
4. Select the WRITE_EXTERNAL_STORAGE options from the left pane and click **Add**. Make sure that the selected options is moved to the right pane.
5. Under the **Manifest Properties**, click the **Tags** tab.
6. Click **Finish**.

To view the functionality of the Runtime Permission API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

46.1 Permission Model in different platforms

Android

The Android versions prior to 6.0, the system permissions were granted by the Android system to the apps during its installation process without user intervention. But from the Android 6.0 onwards, users grant permissions to the apps. These permissions are granted at the app run time, but not during the app installation. In Android, the system permissions are categorized in to:

Normal: The permissions that do not directly cause any risk to user’s privacy are considered as normal permissions. If you add a normal permission to your app’s manifest, the system grants permission automatically without user intervention.

Dangerous: The permissions that provide access to the user’s confidential data are known as dangerous permissions. If any dangerous permission added to your app’s manifest, explicitly the user has to grant permission to your app to access. All dangerous permissions are associated to a group. If user grants permission to any one in the group, the other permissions in the group are also granted automatically. For more information and list of dangerous permissions, refer to Android System Permissions.

The apps developed for Android 6.0 and later versions need to check the permission status manually, and then request to obtain the permission to use a particular resource.

iOS

Similarly, from iOS 8.0 onwards, there are a set of sensitive resources protected by user consent. To access these resources, apps need to check the permission status and obtain the necessary permissions manually. Application-specific permissions can also be altered from the device global settings.

In iOS devices, if your app does not have permission to access a particular resource, the app can request for permission only for once at launching the app for the first time. If the user grants permission, your app can access the resource; if the user denies, the app cannot access the resource and cannot request for permission once again from the app as well. When the app request for the permission second time or later, the system prompts the cannot access pop-up.

The only way to make the app access the resource is user has to enable the access using the device's global settings.

Windows

All sensitive resources are handled by the OS automatically.

SPA

In SPA, only Geo location related APIs require the end-user permission and this is automatically handled by the underlying Brower applications.

46.2 Handling Permissions Automatically by Platforms

Apart from providing the APIs to check, request, and obtain permissions - Kony enables the platforms handle the permissions for the apps. Platforms handling the permissions helps avoiding occurrence of any issues while migrating to the latest versions of the platforms. For example, changing Android target SDK from previous versions to 6.0 or later versions.

Certain synchronous APIs throw the “PermissionError” exception when user denies permission. To handle these exception, place try-catch block around these APIs.

46.3 Example for usage of the Runtime Permission API Functions

The following example gives you brief idea about how runtime permission APIs work and can be utilized.

```
var options = {
isAccessModeAlways:true
};
var result = kony.application.checkPermission(kony.os.RESOURCE_
LOCATION,options);
if(result.status == kony.application.PERMISSION_DENIED){
//Indicates permission denied
   if(result.canRequestPermission){
     kony.application.requestPermission(resourceConfig,
permissionStatusCallback);
     }
   else{
     Var basicConfig = {
     alertType : constants.ALERT_TYPE_CONFIRMATION,
     message : “Please enable the permission in Device Settings to
proceed. Do you want to open settings?”,
     alertHandler : permissionAlertHandler
     }
     kony.ui.Alert(basicConfig);
   }
}
else if(result.status == kony.application.PERMISSION_GRANTED){
     kony.location.getCurrentPosition();
     }
else if(result.status == kony.application.PERMISSION_RESTRICTED){
     alert(“Resource Aceess Restricted for User”);
     }
```

```
function permissionAlertHandler(resp){
   if(resp)
     kony.application.openApplicationSettings();
    }
function permissionStatusCallback(response){
   if(response.status == kony.os.PERMISSION_GRANTED)
     kony.location.getCurrentPosition(…);
   else if(response.status == kony.os.PERMISSION_DENIED){
     Var basicConfig = {
     alertType : constants.ALERT_TYPE_CONFIRMATION,
     message : "Please enable the permission in Device Settings to
proceed. Do you want to open settings?",
     alertHandler : permissionAlertHandler
     }
     kony.ui.Alert(basicConfig);
  }
}
```

47. Permission Status

When apps request for the permissions to access the required resources, the underlying platform responds the permission status. For example, if any resource is not protected by a permission, the platform responds with granted status immediately. If the resource is not available, the platform responds with denied status.

Here is a list of permission statuses returned by platforms when apps requested for permissions. The platforms return these statuses based on the user's decision when the app requested for a permission. For example, if an app wants to access the user's location and for this, the app prompts a dialog box with the "Allow" and "Disallow" options. If user taps the "Allow" option, the permission is granted and if user taps the "Disallow" option, the permission is denied. And the platform returns the respective statuses to the app in the back end. From Android 6 onwards, two options “Allow” or “Deny and Never Ask Again” are displayed. When "Never Ask Again" option is chosen by the user, request permission dialog box will not be shown for that application and for the respective permissions unless it is changed manually from the phone settings.

kony.application.PERMISSION_DENIED

Indicates not enough permissions to access the resource. So, app should request for a permission. The status is responded when:

- end-user denies to access the resource
- the resource is not available.

kony.application.PERMISSION_GRANTED

Indicates app is authorized to access the resource. The status is responded when:

- end-user grants permission to access the resource
- resource is not protected by any permission.

kony.application.PERMISSION_RESTRICTED

Indicates device policy prohibits the app to use the resource. In this case, app cannot not access the resource.

kony.application.RESOURCE_NOT_SUPPORTED

Indicates given resource ID is not applicable in the platform. The status is responded when

- the resource ID is not supported in the underlying platform
- the resource ID is invalid.

kony.application.PERMISSION_NEVER_ASK_AGAIN

Applicable for Android only. Indicates not enough permissions to access the resource and the user will not see the request again. The status is responded when:

- user denies to access the resource and does not want to see the request again.

48. Resource IDs

As runtime permissions are required to access the resources - the resources are identified with the help of resource IDs exhibited by the platforms. You can also pass additional options to identify the exact resource to access. The additional options contain platform-specific keys based on the resource ID that needs permission. For example, if your app wants to access the user location, the resource ID is "LOCATION" and you can use the "isAccessModeAlways" property as an additional option, which is a platform-specific key for iOS devices.

For the resources protected by the permissions, the Android platform identifies the permission required based on the permission declared in the AndroidManifest.xml using the additional options provided along with the resource ID.

The following table lists the resource IDs along with the supported platform-specific additional options and associated Android permissions. The Android permissions listed for the resource IDs in the table below, need to be set true under Manifest Properties in the Project Settings>Native>Android tab.

Resource ID	Description	Platform-specific Option	Android Permission
kony.os.RESOURCE LOCATION	Access to the user's location.	**isAcessModeAlways**: Indicates the mode of geo location usage. *Type*: Boolean *Platform*: iOS	• ACCESS_ FINE_ LOCATION • ACCESS_ COARSE_ LOCATION

Resource ID	Description	Platform-specific Option	Android Permission
kony.os.RESOURCE_ CAMERA	Access to the device's camera.	**isVideoCapture**: Indicates permission required to access camera to capture a photo or a video. The default value is false. To access camera to record a video needs RECORD_AUDIO permission in the Android platform. *Type*: Boolean *Platform*: Android **isAccessModePublic**: Indicates whether a captured photo can be saved to external storage or not. The isAccessModePublic option needs WRITE_ EXTERNAL_STORAGE permission in the Android platform to access external storage location. *Type*: Boolean *Platform*: Android	• CAMERA • WRITE_ EXTERNAL_ STORAGE • RECORD_ AUDIO
kony.os.RESOURCE_ PHOTO_GALLERY	Access to the photo album or gallery.	-	Not protected by any permission.

Resource ID	Description	Platform-specific Option	Android Permission
kony.os.RESOURCE_CONTACTS	Access to the user's contacts.	-	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
kony.os.RESOURCE_CALENDAR	Access to the device's calendar.	-	• READ_CALENDAR • WRITE_CALENDAR
kony.os.RESOURCE_EXTERNAL_STORAGE	Access to the external storage such as SD card.	-	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

48.1 Functions

The Runtime Permission API contains the following functions, which are part of the kony.application Namespace.

kony.application.checkPermission

Checks and returns the permission status of one or more resources.

Syntax

```
kony.application.checkPermission(resourceId[constant/String], options
[JSObject])
```

Input Parameters

Parameter	Description
resourceId [constant/String] - Mandatory	Specify the ID of the resource or name of the permission (only for Android) for which you want to check the status. You can specify either a String (permission name) or an integer (resourceId) value. The feature to specify the name of the permission as a String is applicable only for Android. For instance, you can query a Native Android permission from the AndroidManifest.xml file by specifying the String directly: "android.permission.READ_PHONE_STATE".
options [JSObject] - Optional	Specify the additional option to identify the exact resource of which you want to know the status. This is a platform-specific key. For more information, refer to Resource ID.

Example 1

```
var options = {
    isAccessModeAlways: true
};
var result = kony.application.checkPermission(kony.os.RESOURCE_LOCATION,
options);
if (result.status = = kony.application.PERMISSION_DENIED) {
    kony.application.requestPermission();
} else if (result.status = kony.application.PERMISSION_GRANTED) {
    kony.location.getCurrentPosition();
}
```

Example 2

```
< uses - permission  android: name = "android.permission.READ_PHONE_STATE" / >
var result = kony.application.checkPermission("android.permission.READ_PHONE_
STATE");
if (result.status = = kony.application.PERMISSION_DENIED) {
    kony.application.requestPermission();
} else if (result.status = kony.application.PERMISSION_GRANTED) {
    kony.location.getCurrentPosition();
}
```

Return Values

JSObject

A JS Object contains the authorization status of the requested resource. The returned JSObject contains the following keys:

Return value	Description
status [constant]	Resource status constant which indicates the overall status of the resource authorization. For more information, refer to Permission Status.

Return value	Description
canRequestPermission [Boolean]	Indicates whether you can request for the permissions or not in case the value of the status is PERMISSION_DENIED. In the iOS platform, authorization for a resource can be requested only once. For more information, refer to Permission model in iOS. In the Android platform, the app can request for the permissions even though the status return value is PERMISSION_DENIED or direct the user to app settings to turn on or off the authorization.

Platform Availability

- Android
- iOS
- Windows 10
- SPA

kony.application.requestPermission

Sends a request to the end-user to provide the access to specific resource.

Syntax

```
kony.application.requestPermission(resourceId[constant/String],
statusCallback[Function], options[JSObject])
```

Input Parameters

Parameter	Description
resourceId [constant/String] - Mandatory	Specifies the ID of the resource or name of the permission (only for Android) that you want to access. You can specify either a String (permission name) or an integer (resourceId) value. The feature to specify the name of the permission as a String is applicable only for Android. For instance, you can query a Native Android permission from the AndroidManifest.xml file by specifying the String directly: "android.permission.READ_PHONE_STATE". The available **resourceId** constants are as follows: • kony.os.RESOURCE_CAMERA • kony.os.RESOURCE_LOCATION • kony.os.RESOURCE_PHOTO_GALERY • kony.os.RESOURCE_CONTACTS • kony.os.RESOURCE_CALENDAR • kony.os.RESOURCE_SIRI (iOS-specific) • kony.os.RESOURCE_AUDIO_RECORD • kony.os.RESOURCE_NOTIFICATION (iOS-specific)

Parameter	Description
statusCallback [Function] - Mandatory	A callback function receives the end-user's decision. The statusCallback function receives a JS Object, which contains overall status and permission-specific status that end-user responded on the permission dialog box. `function statusCallback(response);` Here, **response** is a hash map that contains the authorization status of the requested resource. This argument contains the following key: **status [constant]** Resource status constant that indicates the overall status of the resource authorization. The possible values for **status** are as follows: • kony.application.PERMISSION_GRANTED • kony.application.PERMISSION_DENIED • kony.application.PERMISSION_NEVER_ASK_AGAIN

options [JSObject] - Optional	Specifies the additional option to identify the resource for which you want permission. This key is applicable on android only.
	To obtain the kony.application.PERMISSION_NEVER_ASK_AGAIN status, you have to set the `getNeverAskAgainStatus` key to true and pass the key in the options object. If the key is not set, and the user selects either the Deny or Never Ask Again options, then the permission status is considered as Kony.application.PERMISSION_DENIED.
	`var options = {`
	`    "isVideoCapture": true,`
	`    "getNeverAskAgainStatus": true`
	`}`

Parameter	Description
options [Object] - For Notifications	This is a mandatory parameter for notifications. `{notificationtypes : constants}` The available constants are as follows: • kony.notificationsettings.BADGE • kony.notificationsettings.SOUND • kony.notificationsettings.ALERT

Example 1

```
//< uses - permission  android: name = "android.permission.READ_PHONE_
STATE" >
    kony.application.requestPermission("android.permission.READ_PHONE_
STATE", permissionStatusCallback);

function permissionStatusCallback(response) {
    if (response.status == kony.application.PERMISSION_GRANTED) {
        kony.location.getCurrentPosition();
    } else if (response.status == kony.application.PERMISSION_DENIED)
{
        //Display Application Settings alert by using
kony.application.openApplicationSettings()
    }
}
```

Example 2

```
function requestpermission() {

    var options = {
        "isVideoCapture": true,
        "getNeverAskAgainStatus": true
    }

    kony.application.requestPermission(kony.os.RESOURCE_LOCATION,
permissionStatusCallback, options);

}

function permissionStatusCallback(response) {

    alert("response ::" + JSON.stringify());

    if (response.status == kony.application.PERMISSION_GRANTED) {

        kony.location.getCurrentPosition();

    } else if (response.status == kony.application.PERMISSION_DENIED)
{


        Requestpermission(); /* To show the reason to users for
granting the permission to use the feature and then raise a request.
*/

    } else if (response.status == kony.application.PERMISSION_NEVER_
ASK_AGAIN) {


        kony.application.openApplicationSettings(); /* To show the
reason to users for granting the permission to use the feature and
then open application settings to grant the request. */

    }
```

```
)

}
```

Return Values

Function	Description
JSObject	A JSObject contains the authorization status of the requested resource. The returned JSObject contains the following key: **status [constant]** Resource status constant which indicates the overall status of the resource authorization. For more information, refer to Permission Status. ***Note:*** In the Android platform, the status remains PERMISSION_DENIED if at least one of the permissions associated with the resource is denied by the end-user.

Platform Availability

- Android
- iOS
- Windows 10
- SPA

Kony.application.requestPermissionSet

When invoked, this API sends a request for a set of permissions. The status of the request is sent back to the user through a callback.

Syntax

```
Kony.application.requestPermissionSet(permissions, callback)
```

Input Parameters

Parameter	Description
Permissions	Array of qualified android permission strings.

Parameter	Description
Callback	Function object result will invoke this function. The result is a JSobject where the key is permission string and the value is the permission status.

Example

```
function requestpermission() {

    kony.application.requestPermissionSet(["android.permission.CAMERA",
"android.permission.WRITE_CONTACTS"], permissionStatusCallback);

}

function permissionStatusCallback(response) {

    var camera = "android.permission.CAMERA";

    var contacts = "android.permission.WRITE_CONTACTS";

    for (var i in response) {

        /* iterating through permissionSet key value pair from response
jsObject where 'i' is permission key and result is permission status */
        var result = response[i];

        if (result == kony.application.PERMISSION_DENIED) {

            // show message  and raise request again

        } else if (result == kony.application.PERMISSION_NEVER_ASK_AGAIN) {

            // show message and open settings page

            kony.application.openApplicationSettings();

        }

    }

}
```

Return Values

None.

Platform Availability

Android

kony.application.openApplicationSettings

Opens the application-specific settings or device-level application settings.

You may need to direct the end-user to application settings to manually enable or disable a permission for the app to access a particular resource. This function is required when the end-user had denied the permission when the app prompted with a dialog box, and later wants the app to access the resource. For example, if your app wants to access the user's contacts - so the app displayed a dialog box with "Allow" and "Deny" options, asking end-user to grant permission for the first time. The end-user tapped the "Deny" option and the app cannot access the user's contacts. Later, after some point of time, if end-user wants the app access the user's contacts; at that time, you can call the openApplicationSettings API that allows the user to navigate to the application settings screen, and then grant the required permission to the app.

The behavior of the openApplicationSettings API in different platforms:

- **Windows 10**: There is no provision to open the application-level settings. The openApplicationSettings API accepts the resourceid as an optional parameter that helps open the resource-specific settings screen. If the resourceid is not provided, results in unexpected behavior.
- **iOS**: Opens the application-level settings screen showing the access status of the resource. The end-user can turn on or off the access to the resource from the app. The resourceid parameter is ignored in the iOS platform.
- **Android**: Opens the application-level settings screen showing the access status of the resource. The end-user can turn on or off the access to the resource from the app. The resourceid parameter is ignored in the iOS platform.

Syntax

```
kony.application.openApplicationSettings(resourceId[const])
```

Input Parameters

Function	Description
resourceId [constant] - Optional	Specify the resource ID of the resource that you want open its settings. The parameter works only for Windows 10. For more information, refer to Resource ID.

Example

```
kony.application.openApplicationSettings(kony.os.RESOURCE_CONTACTS);
```

Return Values

None

Platform Availability

- Android
- iOS
- Windows 10

49. Shared App Group Container API for iOS

Apps on an iOS device can share common data through the use of an app group container. The Kony Visualizer API provides you with the functions and objects that you need to create and manage app groups.

Each app group is associated with a data container that is located in persistent storage outside of the application sandbox of any app on the iOS device. Every app should have the right permissions and provisioning profiles that will enable them to access the app group. Each app group is specified by a unique identifier.

To provide app group functionality, the following functions in the kony.ds namespace (the Data Store API) now take an optional parameter called storeContext.

- read
- save
- delete

Apps on an iOS device can share common data using an app group container. The Shared App Group Container API provides functions and objects to create and manage app groups.

Each app group is associated with a data container present in a persistent storage outside of the application sandbox of any app on the iOS device. Every app must have the right permissions and provisioning profiles that enable it to access the app group. Each app group is specified by a unique identifier.

To provide the app group functionality, **read**, **save**, and **delete** functions in the **kony.ds** Namespace (the Offline Data Access API) now take an optional parameter called **storeContext**. The StoreContext parameter has the **storeAsUserPreference** key that is set to true to access the data in an app group, and **KonyAppGroupID** that is unique identifier specifying the app group to be accessed. To retrieve the root directory path for the app group, your app calls **kony.io.FileSystem.getAppGroupDirectoryPath** using the **konyAppGroupID** key.

The storeContext parameter is a dictionary that holds the following keys.

Key	Type	Description
storeAsUserPreference	Boolean	Contains true to access the data in an app group. Otherwise, set to false.
konyAppGroupID	string	Holds a unique identifier that specifies the app group to be accessed.

To retrieve the root directory path for the app group, your app can invoke the kony.io.FileSystem.getAppGroupDirectoryPath.

50. Standard Kony API

The Standard Kony API provides a set of functions as part of the kony Namespace that can retrieve and modify data from files, print data in the log files, and run tasks on worker threads. The Standard Kony API also contains functions and constants of the kony.nosql API to connect to a database and modify the data.

The Standard Kony API provides a variety of functions that are primarily for your convenience.

The Standard Kony APIs use the `kony Namespace` and the following API elements.

kony APIs

Function	Description
`kony.convertToBase64`	Converts rawbytes (returned by the camera or the encryption API) to a base64 encoded string.
`kony.convertToRawBytes`	Provides your app with the ability to read rawbytes from a base 64 encoded string.
`kony.evaluateJavaScriptInNativeContext`	Enables a web app JavaScript module, which is running in the Browser widget, to execute JavaScript code in the Kony native context.
`kony.getError`	Retrieves an error object from a handle to the error object.

Function	Description
`kony.props.getProperty`	Used to access parameters from an external properties file.
`kony.print`	Prints debugging output.
`kony.runOnMainThread`	Helps you run the JavaScript code on the main thread. It is an asynchronous API.
`kony.runOnWorkerThread`	Provides apps with multithreding capabilities.
`kony.type`	Retrieves the data type of the specified input.

Convert rawbytes returned by the camera or encryption APIs to a base64 encoded string using the `kony.convertToBase64` function. To provide your app with the ability to read rawbytes, use the `kony.convertToRawBytes` function. Use the `kony.getError` function to retrieve an error object from the try/catch blocks. Using the `kony.props.getProperty` function, you can access parameters from an external properties file. Print the debugging output using the `kony.print` function, and retrieve data types of inputs using the `kony.type` function. To run a JavaScript code on the Main Thread, use the `kony.runOnMainThread` function. To provide multithreading capabilities to an app, use the `kony.runOnWorkerThread` function.

kony.nosql APIs

Function	Description
`kony.nosql.addRecords`	Replaces a record if the primary key matches, else the record is added.

Function	Description
`kony.nosql.addOrReplaceRecords`	Replaces a record if the primary key matches, else the record is added.
`kony.nosql.clearTable`	Clears the existing data in a table in the callback of openTransaction API.
`kony.nosql.closeDatabase`	Closes the database connection.
`kony.nosql.createTable`	Creates a table (object store) in the indexed database.
`kony.nosql.databaseExists`	Specifies whether the required database exists or not.
`kony.nosql.deleteDatabase`	Deletes an existing database.
`kony.nosql.deleteRecords`	Deletes the rows that match the specified condition.
`kony.nosql.deleteTable`	Deletes a table in the database.
`kony.nosql.fetchRecords`	Reads the rows of a table.
`kony.nosql.getPrimaryKeys`	Fetches the primary key of the required table.
`kony.nosql.getTables`	Returns the list of tables in the selected database.
`kony.nosql.openDatabase`	Opens an existing database; if the database does not exist, however, the API creates the database and then opens it.

Function	Description
`kony.nosql.openTransaction`	Opens a transaction in which you can execute any of the insert, update, or delete operations.
`kony.nosql.replaceRecords`	Replaces the records that match the provided condition. The API also updates the full record with a new set of columns.
`kony.nosql.tableExists`	Returns the appropriate Boolean value depending if the required table exists or not.
`kony.nosql.updateRecords`	Updates rows and specific columns that match the provided condition.

Further, you can check whether a required database exists using the `kony.nosql.databaseExists` function. Use the `kony.nosql.openDatabase` function to open an existing database, or create a new database and open it. Once the database is open, use the `kony.nosql.openTransaction` function to open a transaction and execute operations. To close the connection to a database, use the `kony.nosql.closeDatabase` function. Delete existing databases using the `kony.nosql.deleteDatabase` function.

To check whether a table exists in the database, use the `kony.nosql.tableExists` function. Use the kony.nosql.createTable function to create a table in an indexed database. Retrieve the primary key of a table using the `kony.nosql.getPrimaryKeys` function. To retrieve a list of tables present in the database, use the `kony.nosql.getTables` function. To delete existing data in a table, use the `kony.nosql.clearTable` function, and delete a table in the database using the `kony.nosql.deleteTable` function.

Add a record in a table using the `kony.nosql.addRecords` function. To replace records matching a specific condition, use the `kony.nosql.replaceRecords` function. Replace a record matching the primary key using the `kony.nosql.addOrReplaceRecords` function. To view the rows of a table, use the `kony.nosql.fetchRecords` function. Update the rows and columns of a table using the `kony.nosql.updateRecords` function. Use the `kony.nosql.deleteRecords` function to delete the rows matching a specific condition.

Note: The **setAppHeaders**, **setAppFooters**, **appreset**, **readfrombase64** APIs have been deprecated and must not be used to develop new software. However, documentation for them is provided to help in the maintenance of legacy software.

To view the functionality of the Standard Kony API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

50.1 kony Namespace

The kony namespace provides the following API elements.

50.1.1 Functions

The kony namespace provides the following functions.

kony.convertToBase64

Converts rawbytes (returned by the camera or the encryption API) to a base64 encoded string.

Syntax

```
kony.convertToBase64 (rawbytes)
```

Input Parameters

Parameter	Description
rawbytes	The rawbytes that you want to convert to an encoded base64 string. These rawbytes can be those that are returned from a camera or the encryption API. The datatype for the rawbytes differs per platform

Example

```
var base64 = "MIICKTCCAZKgAwIBAgIESrI6bzANBgkqhkiG9w0BAQUFADBZMQswCQYDVQQGEwJp
bjELMAkGA1UECBMCYXAxDDAKBgNVBAcTA2h5ZDENMAsGA1UEChMEa29ueTENMAsGA1UECxMEa29ue
TERMA8GA1UEAxMIcGF0dGFiaGkwHhcNMDkwOTE3MTMzMjMxWhcNMDkxMjE2MTMzMjMxWjBZMQswCQ
YDVQQGEwJpbjELMAkGA1UECBMCYXAxDDAKBgNVBAcTA2h5ZDENMAsGA1UEChMEa29ueTENMAsGA1U
ECxMEa29ueTERMA8GA1UEAxMIcGF0dGFiaGkwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIGP
dqCQCtFgsy1sM494o1F07aN+UXgsilTuKsNRExOb03RGrg2WpAI8PqMXD1XzGZAg+qC9iQexpWHUj
XgCYCbYrETvB3wNAToOrRE6mhZ0iaJij/0tLZACocLiTnvmzZU1B/xowvlioD3zsEs5N5n0U0fIsv
W/22MZ6WtZuAZTAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAXfOB1Mhx2R9ameeZz0hvCzYYSGcWsWN
ZaM7lMdBHFPzASYcVrmLj7JlLpuMm679A2p2JwXBsfetmhKse1ixqVeWdbe/FUfESU+8Krdvkcknv
ZaDqXYFxQAaVjTwWOn+zcDHf7LjjDohgDsMOJWXHkVQj2jooXqiktrBrpccm864=";
var rawBytes = kony.convertToRawBytes(base64);
var newbase64 = kony.convertToBase64(rawBytes);
```

Return Values

The encoded base64 string that was converted from the provided rawbytes. This converted encoded base64 string can be displayed on the screen to the user whenever required as this string is readable. If the conversion could not be performed, this function returns `null`.

Exceptions

An error is thrown if the input type is invalid or follows an unexpected structure.

102-Invalid input error

Remarks

The rawbytes returned by the camera or the encryption API are a set of junk characters that are not readable. When you want to display this data in a readable format to the user, you can use this API.

Platform Availability

Available on all platforms except Desktop Web and Server Side Mobile Web.

kony.convertToRawBytes

Provides your app with the ability to read rawbytes from a base 64 encoded string.

Syntax

```
kony.convertToRawBytes(base64String);
```

Input Parameters

Parameter	Description
base64String	The base64 encoded string from which you want to read the rawbytes.

Example

```
var base64 =
 "MIICKTCCAZKgAwIBAgIESrI6bzANBgkqhkiG9w0BAQUFADBZMQswCQYDVQQGEwJpbjELMAkGA1UEC
BMCYXAxDDAKBgNVBAcTA2h5ZDENMAsGA1UEChMEa29ueTENMAsGA1UECxMEa29ueTERMA8GA1UEAxM
IcGF0dGFiaGkwHhcNMDkwOTE3MTMzMjMxWhcNMDkxMjE2MTMzMjMxWjBZMQswCQYDVQQGEwJpbjELM
AkGA1UECBMCYXAxDDAKBgNVBAcTA2h5ZDENMAsGA1UEChMEa29ueTENMAsGA1UECxMEa29ueTERMA8
GA1UEAxMIcGF0dGFiaGkwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIGPdqCQCtFgsy1sM494o
1F07aN+UXgsilTuKsNRExOb03RGrg2WpAI8PqMXD1XzGZAg+qC9iQexpWHUjXgCYCbYrETvB3wNATo
OrRE6mhZ0iaJij/0tLZACocLiTnvmzZU1B/xowvlioD3zsEs5N5n0U0fIsv

 W/22MZ6WtZuAZTAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAXfOB1Mhx2R9ameeZz0hvCzYYSGcWsWNZ
aM7lMdBHFPzASYcVrmLj7JlLpuMm679A2p2JwXBsfetmhKse1ixqVeWdbe/FUfESU+8Krdvkcknv
ZaDqXYFxQAaVjTwWOn+zcDHf7LjjDohgDsMOJWXHkVQj2jooXqiktrBrpccm864=";
kony.convertToBase64(myVal);
var rawBytes = kony.convertToRawBytes(base64);
```

Return Values

This API returns the modified rawbytes of the image. The data type of the rawbytes varies per platform. In JavaScript, there is no specific type for rawbytes and every platform represents the rawbytes in a unique way.

If the base64String parameter does not contain a valid value, this function returns `null`.

Exceptions

An error occurs if input type is invalid or does not follow the expected structure.

Platform Availability

Available on all platforms except Desktop Web and Server Side Mobile Web. On SPA, reading base64 from an image src is not supported, but you can read the base64 from an image which is displayed through base64.

kony.evaluateJavaScriptInNativeContext

The kony.evaluateJavaScriptInNativeContext API enables a web app JavaScript module, which is running in the Browser widget, to execute JavaScript code in the Kony native context. This API works only when the enableNativeCommunication property of the Browser / CordovaBrowser widget is set to true.

> ***Note:*** This API is applicable for the widgets Browser and CordovaBrowser.

Syntax

The syntax of the API varies depending on the application type.

The syntax for native platforms is as follows.

```
kony.evaluateJavaScriptInNativeContext(methodName, args)
```

The syntax for SPA and Desktop Web platforms is as follows.

```
kony.evaluateJavaScriptInNativeContext(methodName('args'))
```

Input Parameters

Parameter	Description
methodName	The name of the function executed in native context. The function name must be provided as a string. Further, the definition of the function must be provided in the **Modules** section of the application.
args	The array of arguments passed to the function given in the *methodName* parameter. The array must be converted to string using `JSON.stringify` function before passing it to the *args* parameter.

Example

Example 1

```
/*In this sample code, the evaluateJavaScriptInNativeContext API invokes the
noparamsfunction method in the native context without any parameters.*/
var arr = [];
kony.evaluateJavaScriptInNativeContext("noparamsfunction", JSON.stringify
(arr));

//function definition in modules
function noparamsfunction() {
    alert("noparamsfunction invoked");
}

//Output: Displays an alert with message "noparamsfunction invoked".
```

Example 2

```
/*In this sample code, the evaluateJavaScriptInNativeContext API invokes the
twoparamsfunction method in the native context with two parameters.*/
/*Here, the arr Array contains all the arguments required for invoking the
twoparamsfunction method.*/

var arr = [];
//First Argument
arr.push("hi");
//Second Argument
arr.push("helloworld");

kony.evaluateJavaScriptInNativeContext("twoparamsfunction", JSON.stringify
(arr));

//Function definition in the modules section of the project.
function twoparamsfunction (arg1, arg2){
alert("twoparamsfunction invoked with arg1: "+arg1+" and arg2: "+arg2);
}
```

```
/*Output: Displays an alert with the following message:
"twoparamsfunction invoked with arg1: hi and arg2: helloworld".*/
```

Example 3

```
/*Sample code for the implementation of the evaluateJavaScriptInNativeContext
API in the SPA and the Desktop Web platforms.*/

var arr = [];
arr.push("hello world");
kony.evaluateJavaScriptInNativeContext ('sampleMethod('+JSON.stringify(arr)+')
');

//Function method in modules
function sampleMethod(arg1){
alert("sampleMethod invoked with the parameter: "+arg1);
}
/*Output: Displays an alert with the following message:
"sampleMethod invoked with the parameter: hello world"*/
```

Return Values

None

Platform Availability

- iOS, Android, Windows, SPA, and Desktop Web

kony.getError

Retrieves an error object from a handle to the error object.

Syntax

```
kony.getError(error)
```

Input Parameters

Parameter	Description
error	An object that is either the Original Error object or a handle received in try/catch block.

Example

```
try {
    // Label to check UserData Type.
    var basicconf = {
        id: "lblWithUserData",
        text: "userdata",
        isVisible: true,
        skin: "sknlbl"
    };
    var layoutconf = {
        containerWeight: 100,
        hExpand: true,
        vExpand: true
    };
    var lbl1 = new kony.ui.Label(basicconf, layoutconf, {});

    hbox.add(lbl1);
} catch (e) {
    var err = kony.getError(e);

    if (err instanceof KonyError)

        alert("A Kony error");
    else if(err instanceof EvalError)
    alert("A JavaScript Eval Eror");
}
```

Return Values

Returns the JavaScript error object.

Exceptions

If the input type is invalid or there is an unexpected structure.

Remarks

Use *getError* APIs for cross platform applications to handle exceptions consistently.

Platform Availability

Available on all platforms except Windows 8 and Windows 7 / Kiosk.

kony.props.getProperty

This API is used to access parameters from an external properties file.

Syntax

```
kony.props.getProperty(
    group,
    key)
```

Input Parameters

Parameter	Description
group	Set this to *null*. Reserved for future use.
key	A string containing the key for the property value your app needs to retrieve.

Example

```
frmhome.label103982946332169.text = kony.props.getProperty(null,"key1");
```

Return Values

Returns a string containing the property value that matches the given key, or null if there is no value matching the given input key in the properties file.

Remarks

All the external properties files must be appended to the `<appID>.properties` file and be deployed at the following folder on the Kony Application Server:

```
<middlewarehome>\middleware\middleware-
bootconfig\tc\<appID>.properties
```

Platform Availability

Available on Mobile Web and SPA.

kony.print

Prints debugging output.

Syntax

```
kony.print(
    myString);
```

Input Parameters

Parameter	Description
myString	Specifies the string to be printed.

Example

```
kony.print("This is a test message.");
```

Return Values

None.

Remarks

If any other type of argument is passed a string representation of that input is passed. It prints the value to the Standard Output specific to the platform. It prints their values to *stdout*, using the `tostring` function to convert them to strings. print is not intended for formatted output, but only as a quick way to show a value, typically for debugging. For formatted output, use `string.format`.

Platform Availability

Available on all platforms.

kony.runOnMainThread

This API helps you run the JavaScript code on the main thread. It is an asynchronous API. It posts a message to the main thread to invoke a function `f` with parameters arguments.

Note: If `runonMainThread` is invoked in a JavaScript function that is already running on the main thread, then the function is executed in synchronous mode.

Syntax

```
kony.runOnMainThread (f, args)
```

Input Parameters

Parameters	Description
f [Function] - Mandatory	Specifies the callback function that must be executed.
args [Array] - Mandatory	Specifies the JavaScript array that holds the parameters to be passed to function `f`.

Example

```
kony.runOnMainThread(mainthread, []);
function mainthread () {
   kony.print ("Running on On Main Thread");
}
```

Return Values

None

Platform Availability

- Android
- iOS

kony.runOnWorkerThread

Provides apps with multithreading capabilities.

Syntax

```
kony.runOnWorkerThread(f,args)
```

Input Parameters

Parameters	Description
f	Specifies the callback function that must be executed.
args	Specifies an array that holds the parameters to be passed to the function indicated by the *f* parameter.

Example

```
kony.runOnWorkerThread(workermethod, []);
function workermethod () {
   kony.print ("Running on On Worker Thread");
}
```

Return Values

None.

Remarks

This function helps you run JavaScript code asynchronously on a worker thread. It posts a message to the worker thread that owns the current JavaScript context to invoke the function specified in the *f* parameter.

> ***Important:*** The assumption here is that main thread does not own any JavaScript context. The VM/closure thread and worker threads own the JavaScript context. When the `kony.runOnWorkerThread` is invoked from the main thread, a message is posted to the thread that originally invoked the `kony.runOnMainThread`. If the `runonWorkerThread` is invoked in a JavaScript function that is already running on the worker thread, then the function would be executed in synchronous mode.

Platform Availability

- Android
- iOS

---

kony.type

This API retrieves the data type of the specified input. The possible return values in JavaScript are:

- string
- number
- boolean
- function
- userdata - not JavaScript type but the platform specific type that platform returns and consumes like rawBytes.

- In case of kony custom objects, this API returns the fully qualified name of the class from which object is instantiated. for e.g *kony.ui.Form2*
- null

Use the *typeof* operator instead of *kony.type* to achieve the same result. If the *typeof* operator cannot be used, use *kony.type* instead.

Syntax

```
kony.type(variable)
```

Input Parameters

Parameter	Description
variable [String] or [Number] or [Boolean] or [Function] - or [userdata] or nil/null - Mandatory	Specifies the value for which you want to retrieve the data type.

Example

```
var myVal = 2000;
var varType = kony.type(myVal);//Returns a number
var myVal1 = "hello";
var varType = kony.type(myVal1);//Returns a string
```

Return Values

returnValue [String]

Returns a string that determines the data type of the specified input. Possible values are:

Return Value	Description
string	this value is returned when you pass a string as the input parameter.
number	this value is returned when you pass a number as the input parameter.
boolean	this value is returned when you pass a boolean value as the input parameter.
function	this value is returned when you pass a function as the input parameter.
userdata	this value is returned when the input parameter is not a string, number, boolean or function. ***Note:*** Any data type which is not a string, number, boolean, or function is treated as userdata.
null/nil	this value is returned when the variable is assigned null/nil

- In case of kony custom objects, this API returns the fully qualified name of the class from which object is instantiated. for e.g *kony.ui.Form2*. When you pass any custom / built in JSObject to this API, it should return the name of the custom JSObject.

Platform Availability

Available on all platforms.

kony.web.WebAuthenticationSession

This API helps you to manage the sharing of a one-time login credentials between the Safari web browser and an app. The one-time login credentials can also be used to automatically log on to associated apps. The kony.web.WebAuthenticationSession API is available from V8 SP4 onwards.

In addition, this API facilitates a single sign-on (SSO) experience when used with standards such as OAuth. It puts users in control of whether they want to use their existing logged-in session from Safari browser.

When users try to authenticate an URL from your application by using this API, the API displays an alert containing two buttons: **Continue** and **Cancel**. The following scenarios can occur when the Login screen of the web service is displayed:

- If the user taps **Continue**, one secure controller will be opened with authenticationURL. If the user has already authenticated the URL in Safari browser, it calls the redirectionURL (the URL scheme that points to this app), which is then passed in the API.
- If the user cancels the alert, the session will be canceled and the constants.WEB_AUTH_SESSION_ERROR_CANCELLED_LOGIN error message will be displayed.
- If the user taps **Cancel**, the session will be canceled and the constants.WEB_AUTH_SESSION_ERROR_CANCELLED_LOGIN error message will be displayed.

This will also dismiss the view controller that displays the login page of the web service.

Syntax

```
kony.web.WebAuthenticationSession(authenticationURL, redirectionURL,
callbackfun)
```

Input Parameters

Parameter	Description
authenticationURL	A String that points to the authentication web page. After the authentication process is complete, the server/service sends a callbackURL along with an authentication token to the completion handler. The authenticationURL parameter only supports URLs with http:// or https:// schemes.
redirectionURL	A String URL scheme. This is a redirection URL for the app to receive callbacks when the authentication process is complete.
callbackfun	A callback function with one argument. It is a dictionary with **callbackURL** and **errorCode** keys. The argument is invoked when the session is either completed successfully or is canceled by the user. If the session is successful, the errorCode value is null and the callbackURL is sent to the callback. If the session fails or is canceled, the callbackURL value is null and the respective errorCode value is sent to the callback.

Methods

Method	Description
session.start()	Returns boolean value (yes/no), based on whether the session starts successfully or if it fails to start.
session.cancel ()	Cancels the session.

Example

```
var session = new kony.web.WebAuthenticationSession(authenticationURL,
redirectionURL, callbackfun);
session.start();

function callbackfun(response) {
    if (constants.WEB_AUTH_SESSION_ERROR_CANCELLED_LOGIN ==
callbackURL.errorCode) {
        kony.print("Error occured during authentication ",
response.errorCode);
    } else {
        //user-defined flow
    }
    kony.print("resonse callbackURL ", response.callbackURL);
}
```

Return Values

None

Platform Availability

- iOS

50.1.2 kony.nosql APIs

50.1.2.1 Available Constants

Constant	Used in
kony.nosql.AND	condition.addRule or condition.addCondition API.
kony.nosql.OR	conditionInstance.addRule or conditionInstance.addCondition API.
kony.nosql.EQ	While instantiating kony.nosql.Rule class, represents "=" operator.
kony.nosql.NEQ	While instantiating kony.nosql.Rule class, represents "!=" operator.

kony.nosql.GT	While instantiating kony.nosql.Rule class, represents ">" operator.
kony.nosql.GTE	While instantiating kony.nosql.Rule class, represents ">=" operator.
kony.nosql.LT	While instantiating kony.nosql.Rule class, represents "<" operator.
kony.nosql.LTE	While instantiating kony.nosql.Rule class, represents "<=" operator.
kony.nosql.READ	kony.nosql.openTransaction API. The kony.nosql.READ constant specifies that the transaction is to be opened in read mode.
kony.nosql.READ_WRITE	kony.nosql.openTransaction API. The kony.nosql.READ_WRITE constant specifies that the transaction is to be opened in read/write mode.
kony.nosql.ASCENDING	result.sort API. The kony.nosql.ASCENDING constant sorts the records in ascending order.
kony.nosql.DESCENDING	result.sort API. The kony.nosql.DESCENDING constant sorts the records in descending order.

50.1.2.2 Functions

Note: Performing update, delete, or fetch operations on a table that contains a large number of records may have an affect on the performance when the condition object contains more than one rule/condition.

Note: Usage of Boolean values in the condition object may also cause a performance issue. You must use either 0/1 or 'false'/'true', instead of Boolean false/true.

The functions of the kony.nosql Namespace are as follows.

kony.nosql.addRecords

Replaces a record if the primary key matches, else the record is added.

Syntax

```
kony.nosql.addRecords(transactionObject, tableName, records)
```

Input Parameters

Parameter	Description
transactionObject	The object returned by kony.nosql.openTransaction API.
tableName	The table on which the insert operation is to be applied.
records [Array]: [{},{},{}]	Array of rows containing key-value pairs that are to be inserted in the table.

Example

```
kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE, function
(transactionObject)
{
    kony.nosql.addRecords(transactionObject, 'EMP', [{
        'Emp_id': 1234,
        'Emp_name': 'abc',
        'Emp_manager': 'def'
    }, {
        'Emp_id': 1235,
        'Emp_name': 'efg',
        'Emp_manager': 'xyz'
    }, {
        'Emp_id': 1236,
        'Emp_name': 'ijk',
        'Emp_manager': 'def'
    }]).then(function() {
        //add records success callback
    }).
    catch(function(errorObject) {
        //add records error callback
    });
}).then(function(transactionObject) {
    //transaction complete callback
});
```

Return Values

Promise.

- Resolve parameter: none
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.addOrReplaceRecords

Replaces a record if the primary key matches, else the record is added.

Syntax

```
kony.nosql.addOrReplaceRecords(transactionObject, tableName, records)
```

Input Parameters

Parameter	Description
transactionObject	The object returned by kony.nosql.openTransaction API.
tableName	The table on which the addOrReplace operation is to be applied.
records[Array]: [{},{},{}]	Array of rows containing key-value pairs that are to be added or replaced.

Example

```
kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE, function
(transactionObject) {
    kony.nosql.addOrReplaceRecords(transactionObject, 'EMP', {
        Emp_manager: 'updated manager name'
    }).then(function() {
        //addOrReplaceRecords records success callback
    }).
    catch (function(errorObject) {
        //addOrReplaceRecords records error callback
    });
}).then(function(transactionObject) {
    // transaction complete callback
});
```

Return Values

Promise.

- Resolve parameter: none
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.clearTable

Clears the existing data in a table in the callback of openTransaction API.

Syntax

```
kony.nosql.clearTable(transactionObject, tableName)
```

Input Parameters

Parameter	Description
transactionObject	Object returned in callback of openTransaction API.
tableName	The name of the table for which the data is to be cleared.

Example

```
kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE, function
(transactionObject) {
    kony.nosql.clearTable(transactionObject, 'EMP').then(function() {
        //clear table success callback
    }).
    catch(function(errorObject) {
        //clear table error callback
    });
}).then(function(transactionObject) {
    // transaction complete callback
});
```

Return Values

Promise.

Platform Availability

- SPA
- Desktop Web

kony.nosql.closeDatabase

Closes the database connection.

Syntax

```
kony.nosql.closeDatabase(dbObject)
```

Input Parameters

Parameter	Description
dbObject	The dbObject returned in successcallback of kony.nosql.openDatabase API

Example

```
kony.nosql.openDatabase('Company', 1, function(dbObject) {
    //upgrade callback
}).then(function(dbObject) {
    //success callback
    kony.nosql.closeDatabase(dbObject).then(function() {
        //close database success callback
    }).
    catch (function(errorObject) {
        //close database error callback
    });
}).catch(function(errorObject) {
    //error callback
});
```

Return Values

Promise.

- Resolve parameter: none
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.createTable

Creates a table (object store) in the indexed database. You can only create tables in the upgradedbcallback.

Syntax

```
kony.nosql.createTable(dbObject, tableName, config)
```

Input Parameters

Parameter	Description
dbObject	The object that is returned in the upgradecallback parameter of kony.nosql.openDatabase API.
tableName [String]	The name of the table to be created.
config [Object]	Stores the following optional properties: • primarykey: 　◦ [String]; If primarykey is of single value. 　◦ [Array]; If primarykey is of composite value. • indexes [Object]: Indexes to be created along with metadata. • autoIncrement [Boolean]: If you set this property as true and do not specify the primarykey value while inserting a record, the primarykey value is generated automatically.

Example

```
kony.nosql.openDatabase('Company', 1, function(dbObject) {
    kony.nosql.createTable(dbObject, 'EMP', {
        primaryKey: 'Emp_id',
        indexes: {
            'Emp_id': {},
            'Emp_name': {},
            'Emp_manager': {},
            'Emp_DOJ': {},
            'Emp_email': {
                unique: true
            }
        }
    });
}).then(function(dbObject) {
    //success callback
}).catch(function(errorObject) {
    //error callback
});
```

Return Values

None.

Remarks

- If there is any error in creating a table, you will be directed to the reject callback.
- If you successfully create a table, you will be directed to the success callback of openDatabase API.
- Error in the creation of the table results in any one of the following scenarios:
    - If the database already exists: roll-back to the previous version of database occurs.
    - If this is the first version of the database: no new database is created.
- The autoIncrement property is not applicable when the primarykey is of composite value.
- Unlike SQL, you do not need to specify all properties but only the one you wish to index.

- Do not index properties that contain images, movies, or large-sized strings. Store them in IndexedDB, but do not index them; it may affect the performance.

Platform Availability

- SPA
- Desktop Web

kony.nosql.databaseExists

Specifies whether the required database exists or not.

Syntax

```
kony.nosql.databaseExists(dbName)
```

Input Parameters

Parameter	Description
dbName	The name of the database that you want to verify if it exists.

Example

```
kony.nosql.databaseExists('dbName').then(function(isExists) {
    //successcallback
}).
catch(function(errorObject) {
    //error callback
});
```

Return Values

Promise.

- Resolve parameter: true if database exists; else, false
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.deleteDatabase

Deletes an existing database.

Syntax

```
kony.nosql.deleteDatabase(dbname)
```

Input Parameters

Parameter	Description
dbname	Name of the database to be deleted.

Example

```
kony.nosql.deleteDatabase('Company').then(function()
{
   //success callback
}).
catch(function(errorObject)
{
   //error callback
});
```

Return Values

Promise.

- Resolve parameter: none
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.deleteRecords

Deletes the rows that match the specified condition.

Syntax

```
kony.nosql.deleteRecords(transactionObject, tableName, condition)
```

Input Parameters

Parameter	Description
transactionObject	Object returned by openTransaction API.

Parameter	Description
tableName	The name of the table on which the delete operation is to be applied.
condition	It has two parameters: kony.nosql.Condition instance and null. • kony.nosql.Condition instance: Deletes the records based on the given condition. • null: Deletes all records in the table.

Example

```
kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE, function
(transactionObject)
{
    kony.nosql.deleteRecords(
        transactionObject,
        'EMP',
        condition
    ).then(function() {
        //delete records success callback
    }).
    catch(function(errorObject) {
        //delete records error callback
    });
}).then(function(transactionObject) {
    // transaction complete callback
});
```

Return Values

Promise.

- Resolve parameter: none
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Remarks

- If you do not pass any condition, the kony.nosql.deleteRecords API deletes all the records.
- If you want to delete all the records, use the kony.nosql.clearTable API for better performance efficiency.

Platform Availability

- SPA
- Desktop Web

kony.nosql.deleteTable

Deletes a table in the database. This can only be done in the upgradecallback parameter of openDatabase API.

Syntax

```
kony.nosql.deleteTable(dbObject, tableName)
```

Input Parameters

Parameter	Description
dbObject	dbObject returned in the upgradecallback parameter of openDatabase API.
tableName	The name of the table that is to be deleted.

Example

```
kony.nosql.openDatabase('Company', 1, function(dbObject) {
    kony.nosql.deleteTable(dbObject, 'EMP');
}).then(function(dbObject) {
    //success callback
}).
catch(function(errorObject) {
    //error callback
});
```

Return Values

None.

Platform Availability

- SPA
- Desktop Web

kony.nosql.fetchRecords

Reads the rows of a table.

Syntax

```
kony.nosql.fetchRecords(transactionObject, tableName, condition)
```

Input Parameters

Parameter	Description
transactionObject	Object returned by openTransaction API.
tableName	The name of the table on which the insert operation is to be applied.

Parameter	Description
condition	It has two parameters: kony.nosql.Condition instance and null. • kony.nosql.Condition instance: Filters the records based on the condition. • null: Returns all records in the table.

Example

```
kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE, function
(transactionObject) {
    /*
    e.g. How to form a condition like - "((Emp_name = 'Joe' || Emp_name =
'John') &amp;&amp; (Emp_manager = 'Albert' || Emp_manager = 'Bill'))"

        var rule1 = new kony.nosql.Rule('Emp_name', kony.nosql.EQ, 'Joe');
        var rule2 = new kony.nosql.Rule('Emp_name', kony.nosql.EQ, 'John');
        var rule3 = new kony.nosql.Rule('Emp_manager', kony.nosql.EQ,
'Albert');
        var rule4 = new kony.nosql.Rule('Emp_manager', kony.nosql.EQ, 'Bill');

        var cond1 = new kony.nosql.Condition(rule1);
        cond1.addRule(kony.nosql.OR, rule2);

        var cond2 = new kony.nosql.Condition(rule3);
        cond2.addRule(kony.nosql.OR, rule4);

        var condition = new kony.nosql.Condition(cond1);
        cond2.addCondition(kony.nosql.AND, cond2);

        condition.toString() will return the below line, so that one can
verify their condition.
        "((Emp_name = 'Joe' || Emp_name = 'John') &amp;&amp; (Emp_manager =
'Albert' || Emp_manager = 'Bill'))"
```

```
        //Now pass "condition" to kony.nosql.fetchRecords API
    */

    kony.nosql.fetchRecords(transactionObject, 'EMP', condition).then(function
(result) {
        //fecth records success callback

        /* How to use result object

        result object has the following properties
            1) result.data will return list of records (i.e. an array).
            2) result.length will return number of records fetched.
            3) result.next will move the internal array index by 1, and return
true/false depending on, if record exists at that index or not.
            4) result.record will return a single record, make sure
"result.next" is called before calling "result.record"

        e.g. How to iterate through a result object
            //Considering "result.next" was not called before
            //if one calls "result.record" here, it would return "null"
            while(result.next) {
                kony.print(result.record);
                kony.print(result.record); //This will print the same record
as that of above line.
            }
            //if one calls "result.record" here, it would return "null"


        result object has the following methods
            1) result.sort(columnName, sortType) will return list of sorted
records WRT columnName in ascending/descending order
            2) result.sort(compareFunction) will return list of sorted records
WRT passed compareFunction
            3) result.limit(startIndex, recordCount)

        The above APIs are chainable and mutate the output of result.data
```

```
list.
        e.g. result.sort(columnName, sortType).limit(0, 10).data;
             Will first sort all records and on the sorted records it will
pluck the first 10 records.

        e.g. result.limit(0, 10).sort(columnName, sortType).data;
             Will first pluck the first 10 records and then sort those 10
records.
        */
    }).
    catch (function(errorObject) {
        //fetch records error callback
    });
}).then(function(transactionObject) {
    // transaction complete callback
});
```

Return Values

Promise.

- Resolve parameter: resultset (a type of iterator)
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.getPrimaryKeys

Fetches the primary key of the required table.

Syntax

```
kony.nosql.getPrimaryKeys(dbObject, tableName, transactionObject)
```

Input Parameters

Parameter	Description
dbObject	The dbObject returned in successcallback of kony.nosql.openDatabase API
tableName	Name of the table for which the primary key is to be fetched.
transactionObject [Optional]	The object returned by openTransaction API. ***Note:*** You must open the transactionObject parameter for the tableName mentioned in the input.

Example

```
kony.nosql.openDatabase('Company', 1, function(dbObject) {
    //upgrade callback
}).then(function(dbObject) {
    //success callback
    kony.nosql.getPrimaryKeys(dbObject, tableName).then(function
(primayKeyList) {
        //getPrimaryKeys success callback
    }).
    catch(function(errorObject) {
        //getPrimaryKeys error callback
    });
}).
catch(function(errorObject) {
    //error callback
});
```

Return Values

Promise.

- Resolve parameter: [Array] list of primary keys
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.getTables

Returns the list of tables in the selected database.

Syntax

```
kony.nosql.getTables(dbObject)
```

Input Parameters

Parameter	Description
dbObject	The dbObject returned in successcallback of kony.nosql.openDatabase API

Example

```
kony.nosql.openDatabase('Company', 1, function(dbObject) {
    //upgrade callback
}).then(function(dbObject) {
    //success callback
    kony.nosql.getTables(dbObject).then(function(tableNames) {
        //getTables success callback
    }).
    catch(function(errorObject) {
        //getTables error callback
    });
}).
catch(function(errorObject) {
    //error callback
});
```

Return Values

Promise.

- Resolve parameter: tableNames[Array]
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.openDatabase

Opens an existing database; if the database does not exist, however, the API creates the database and then opens it.

Syntax

```
kony.nosql.openDatabase(dbname, version, upgradecallback)
```

Input Parameters

Parameter	Description
dbname [String]	Name of the database to be opened.
version [Integer]	Version of the database.
upgradecallback [function]	Called when you open a new database or when you upgrade an existing database. Upgrading an existing database involves adding new tables or removing existing tables.

Example

```
kony.nosql.openDatabase('Company', 1, function(dbObject) {
    //upgrade callback
}).then(function(dbObject) {
    //success callback
}).
catch (function(errorObject) {
    //error callback
});
```

Return Values

Promise.

- Resolve parameter: dbObject
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.openTransaction

Opens a transaction in which you can execute any of the insert, update, or delete operations.

Syntax

```
kony.nosql.openTransaction(dbObject, tableNames, mode, callback)
```

Input Parameters

Parameter	Description
dbObject	The dbObject returned in the success callback of openDatabase API.

Parameter	Description
tableNames [Array]	The tables that are to be part of this transaction.
mode	The mode in which you open the transaction, for example, kony.nosql.READ or kony.nosql.READ_WRITE).
callback	The openTransaction callback in which transaction-based operations are possible. Parameter of callback: transactionObject.

Example

```
kony.nosql.openDatabase('Company', 1).then(function(dbObject) {
    kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE,
function(transactionObject) {
        //transaction callback
    }).then(function(transactionObject) {
        //transaction complete callback
    }).
    catch(function(errorObject) {
        //transaction error callback
    });
}).
catch(function(errorObject) {
    //open database error callback
});
```

Return Values

Promise.

- Resolve parameter: transactionObject
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Remarks

- In case of nested transactions, ensure that the child and parent transactions are not opened in the same table.

Platform Availability

- SPA
- Desktop Web

kony.nosql.replaceRecords

Replaces the records that match the provided condition. The API also updates the full record with a new set of columns.

Syntax

```
kony.nosql.replaceRecords(transactionObject, tableName, columnValuePair,
condition)
```

Input Parameters

Parameter	Description
transactionObject	The object returned by kony.nosql.openTransaction API.
tableName	The name of the table on which the replace operation is to be applied.
columnValuePair	Values of the columns to be replaced. ***Note:*** The replace columnValuePair must not contain primarykey; otherwise columnValuePair will be ignored even if it is passed.
condition	It has two parameters: kony.nosql.Condition instance and null. • kony.nosql.Condition instance: Replaces the records that satisfy the given condition. • null: Replaces all records in the table.

Example

```
kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE, function
(transactionObject) {
    kony.nosql.replaceRecords(
        transactionObject,
        'EMP', {
            Emp_manager: 'updated manager name'
        },
        condition
    ).then(function() {
        //replace records success callback
    }).
    catch (function(errorObject) {
        //replace records error callback
    });
}).then(function(transactionObject) {
    // transaction complete callback
});
```

Return Values

Promise.

- Resolve parameter: none
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.tableExists

Returns the appropriate Boolean value depending if the required table exists or not.

Syntax

```
kony.nosql.tableExists(dbObject, tableName)
```

Input Parameters

Parameter	Description
dbObject	The dbObject returned in successcallback of kony.nosql.openDatabase API
tableName	Name of the table that you want to verify if it exists.

Example

```
kony.nosql.openDatabase('Company', 1, function(dbObject) {
    //upgrade callback
}).then(function(dbObject) {
    //success callback
    kony.nosql.tableExists(dbObject, tableName).then(function(isExists) {
        //tableExists success callback
    }).
    catch(function(errorObject) {
        //tableExists error callback
    });
}).
catch(function(errorObject) {
    //error callback
});
```

Return Values

Promise.

- Resolve parameter: true if table exists; else, false
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

kony.nosql.updateRecords

Updates rows and specific columns that match the provided condition.

Syntax

```
kony.nosql.updateRecords(transactionObject, tableName, columnValuePair,
condition)
```

Input Parameters

Parameter	Description
transactionObject	The object returned by kony.nosql.openTransaction API.
tableName	The name of the table on which the update operation is to be applied.
columnValuePair	Values of the columns to be updated.
condition	It has two parameters: kony.nosql.Condition instance and null. • kony.nosql.Condition instance: Updates the records that satisfy the given condition. • null: Updates all records in the table.

Example

```
kony.nosql.openTransaction(dbObject, 'EMP', kony.nosql.READ_WRITE, function
(transactionObject) {
    kony.nosql.updateRecords(
        transactionObject,
        'EMP', {
            Emp_manager: 'updated manager name'
        },
        condition
    ).then(function() {
        //update records success callback
    }).
    catch (function(errorObject) {
        //update records error callback
    });
}).then(function(transactionObject) {
    // transaction complete callback
});
```

Return Values

Promise.

- Resolve parameter: none
- Reject parameter: errorObject {errorMsg:" ", errorCode: " "}

Platform Availability

- SPA
- Desktop Web

51. Streaming API

Streaming is a feature that enables devices to listen to a continuous data stream and use the data. The data is available on streaming servers. The common use of the Streaming API is to monitor live stream data in text or JSON format , in your application.

The Streaming feature enables devices to capture a continuous data stream and use the data available on streaming servers. Use the Streaming API to monitor your application's live stream data in text or JSON format.

> ***Note:*** The JSON format is supported only if the streaming servers provide the valid JSON-formatted text in a single line.

Some real-world usages of the Streaming API are as follows:

- Live stock market prices
- Sensor data from artificial sensors on various parameters, such as Humidity and Temperature

Streaming data uses different protocols to fetch the data.

> ***Note:*** Kony currently supports only the HTTP 1.1 protocol.

The Streaming API comprises of the `kony.stream Namespace` and related API elements:

Function	Description
`kony.stream.deregisterDataStream`	Deregisters the application that was registered earlier with the Streaming Server
`kony.stream.registerDataStream`	Registers the application for streaming with the OS provider of the underlying platform.

Function	Description
kony.stream.setCallback	Specifies the callback function that needs to be executed for a registered data stream.

Register an application for streaming using the `kony.stream.registerDataStream` function, and specify the callback function to be executed for a registered data stream using the `kony.stream.setCallback` function. You can unregister an application from the Streaming server using the `kony.stream.deregisterDataStream` function.

You can test the Streaming APIs by configuring a streaming server locally using the StreamingServer.zip and the readme.txt artefacts.

To view the functionality of the Streaming API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

51.1 kony.stream Namespace

The kony.stream namespace forms the Streaming API and provides the following API elements.

51.1.1 Functions

The kony.stream namespace provides the following functions.

51.1.2 kony.stream.deregisterDataStream

This API allows you to deregister the application that was registered earlier with the Streaming Server.

Use Case

Use this API when you want to discontinue an application from streaming the data from a streaming server.

Syntax

```
kony.stream.deregisterDataStream(streamObject)
```

Input Parameters

Parameter	Description
streamObject [Object] - Mandatory	Streaming Object that is returned during the registration process.

Example

```
function deregisterDataStream() {
    //Define config parameters to the register stream.
    var configParams = {
        url: "http//www.mobilestreaming.kony.com",
        method: "post"
    };

    //Registers the stream.
    var streamObj = kony.stream.registerDataStream("http1.1",
configParams, "json", myCallbackFunction);


    //Deregisters the data stream with the specified identifier on
Windows Phone.
    var status = kony.stream.deregisterDataStream(streamObj);

    //Reading the status.
    alert("Status is ::" + status);
}
```

```
deregisterDataStream: function() {
    //Define config parameters to the register stream.
    var configParams = {
        url: "http://localhost:8081/listStreamData",
        method: "post",
        outputformat: "JSON"
    };

    //Registers the stream.
    var streamObj = kony.stream.registerDataStream("http1.1",
configParams, "json", this.myCallbackFunctionDeregister);
```

```


    //Deregisters the data stream with the specified identifier on
Windows Phone.
    var status = kony.stream.deregisterDataStream(streamObj);

    //Reading the status.
    alert("Status is ::" + status);
}
```

Return Values

None.

Exceptions

- StreamingError
- Error

Platform Availability

Available on all platforms except Desktop Web and Windows Kiosk*. *Dummy Implementation on Mobile Web.

51.1.3 kony.stream.registerDataStream

This API registers the application for streaming with the OS provider of the underlying platform.

Syntax

```
kony.stream.registerDataStream
(protocol,configparams,outputformat,callback)
```

Input Parameters

Parameter	Description
protocol [String] - Mandatory	Specifies the underlying streaming protocol to be used. Kony currently supports only HTTP1.1 protocol.

Parameter	Description

configparams [Table] - Mandatory	Specifies an Object with key-value pairs of configuration parameters required for the underlying protocol.
	
	The following are the configuration parameters for HTTP1.1 protocol:
	
	• **url [String]** - **Mandatory**: Specifies the URL of the streaming server of a web site that has data that can be streamed. For example: url="http://www.mobilestreaming.kony.com.
	
	• **cookie [Object] - Optional**:Specifies an Object with the following key-value pairs:
	  ◦ streamingid (String)
	  ◦ jessessionid (String)
	
	`cookie = {`
	`    streamingid: "12218181agdgdhjhjd",`
	`    jessessionid:`
	`"198181819919agagah1t1g1"`
	`};`
	
	• **headers [Object] - Optional**: Specifies the header parameters.
	
	`//For example: {authmode:basic}`
	
	• **params [Object] - Optional**: Specifies any additional parameters.
	
	For example: {quote:"goog"}
	
	• **method [String] - Mandatory**: Specifies the method. Possible values are:
	  ◦ **get**
	  ◦ **post**(Default value is post)
	
	For example: "post"

Parameter	Description
outputformat [String] - Mandatory.	Specifies the output format of the stream. Currently supported values are: • String (default) • JSON ***Note:*** If JSON is the output format specified, output from the streaming server is converted to an Object and is passed as an input parameter to the callback function.

Parameter	Description
callback [Function] - Mandatory.	Specifies the function that needs to be executed when the streaming data is available. ***Note:*** This function is also executed incase of any network error or system error in listening to the data stream. The callback function has the following signature: **callbackFunction(status, data, context)** • **status[Number]** - indicates the status of the streaming operation. Possible values are: ◦ 0 - indicates success ◦ 1011 - Device has no wifi or mobile connectivity. Please try the operation after establishing connectivity. ◦ 1012 - Request Failed .This is a generic error and is thrown if the platform is not able to differentiate between various errors. ◦ 1014 - Request timed out ◦ 1015 - Cannot find host. ◦ 1016 - Cannot connect to host. ◦ 1002 - Error reading from stream. ◦ 1019 - Error writing into stream. ◦ 1020 - End of stream with errors. ◦ 1021 - End of stream. • **data[Object or String]** - If the output format is JSON it is an Object and if the output format is string, it is a String. Incase of any error, data will be null/nil. • **context[Object]** - Contains additional information that needs to be passed to the underlying streaming API. For HTTP1.1 protocol, this table contains a value httpresponsecode.

Example

```
function myCallbackFunction(status, data, context) {
    //Execute the logic here
}

function registerDataStream() {
    //Define config parameters to register stream.
    var configParams = {
        url: "http//www.mobilestreaming.kony.com",
        method: "post"
    };

    //Registers the stream.
    var streamObj = kony.stream.registerDataStream("http1.1",
configParams, "json", myCallbackFunction);

    //Reading the stream Object.
    alert("Stream object is ::" + streamObj);
```

```
myCallbackFunctionRegister: function(status, data, context) {
    //Execute the logic here
   alert("You have successfully registered a streaming service");
},

    //Registers the stream.
    var streamObj = kony.stream.registerDataStream("http1.1",
configParams, "json", this.myCallbackFunctionDeregister);

    //Reading the status.
    alert("Status is ::" + status);
}
```

Return Values

- **streamObject [Object]**: Stream Object for this registration. This object needs to be used during the de-registration process. It is null when there is any input error for a JavaScript application.

Exceptions

- StreamingError: Error thrown by Streaming API.
- Error

Platform Availability

Available on all platforms except BlackBerry 10, Desktop Web and Windows Kiosk*. *Dummy Implementation on Mobile Web.

51.1.4 kony.stream.setCallback

This API specifies the callback function that needs to be executed for a registered data stream.

Syntax

```
kony.stream.setCallback(streamObject,callbackfunction)
```

Input Parameters

Function	Description
streamObject [String] - Mandatory	Specifies the streamObject that identifies the stream.
callbackfunction [Function] - Mandatory	Specifies the callback function that needs to be executed.

Example

```
//Callback function for setCallback
function myCallbackFunction1(status, data, context) {
    //Execute the logic here
}

function setCallback() {
    kony.stream.setCallback(streamObj, myCallbackFunction1);

}
```

```
//Callback function for setCallback
myCallbackFunctionRegister: function(status, data, context) {
    //Execute the logic here
   alert("You have successfully registered a streaming service");
}
function setCallback() {
    kony.stream.setCallback(streamObj, myCallbackFunction1);

}
```

Return Values

None

Exceptions

- StreamingError: Error thrown by Streaming API.
- Error

Platform Availability

Available on all platforms except Desktop Web and Windows Kiosk*. *Dummy Implementation on Mobile Web.

52. String API

The String API provides your app with functions for common string processing tasks.

The **kony.string** namespace provides static string functions that augment the functions in the `String` type that is available by default. For more information see, https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/string.

Note: JavaScript distinguishes between String objects and primitive string values.

String literals (denoted by double or single quotes) and strings returned from `String` calls in a non-constructor context (i.e., without using the new keyword) are primitive strings. JavaScript automatically converts primitive strings and `String` objects, making it possible to use `String` object methods for primitive strings. In cases where a method is to be invoked on a primitive string or a property lookup occurs, JavaScript automatically wraps the string primitive and calls the method or performs the property lookup.

The String API contains the `kony.string Namespace` and related API elements:

Function	Description
`kony.string.containsChars`	Verifies if any one of the specified set of characters is available in the given string and returns a boolean value.
`kony.string.containsOnlyGivenChars`	Verifies if only (and only) the specified set of characters is available in the given string and returns a boolean value.
`kony.string.containsNoGivenChars`	Verifies that the input string does not contain any of the specified characters and returns a boolean value.

Function	Description
kony.string.endsWith	Returns a boolean value indicating if the source string ends with the specified string.
kony.string.equalsIgnoreCase	Determines whether two strings contain the same data, ignoring the case of the letters in the String.
kony.string.isAsciiAlpha	Verifies if the input string contains only ASCII alphabet characters and returns a boolean value.
kony.string.isAsciiAlphaNumeric	Verifies if the input string contains only ASCII alphabet characters and numbers, and returns a boolean value.
kony.string.isNumeric	Verifies if the input string contains only numeric characters, and returns a boolean value.
kony.string.isValidEmail	Verifies if the input string is a valid email address and returns a boolean value.
kony.string.rep	Generates a string which is equivalent to ***n*** *copies of the source string concatenated together*.
kony.string.reverse	Reverses the characters in the source string.
kony.string.startsWith	Returns a boolean value indicating if the source string begins with the specified string.

Function	Description
`kony.string.trim`	Removes the leading and ending spaces from the source string.

Use the `kony.string.containsChars` function to check whether the specified set of characters are available in the given string. To verify that only a specified set of characters is available in the given string, use the `kony.string.containsOnlyGivenChars` function. If you want to verify that no given characters are available in the given string, use the `kony.string.containsNoGivenChars` function. You can check whether a string starts or ends with a specified string using the `kony.string.startsWith` or the `kony.string.endsWith` functions respectively. Use the `kony.string.equalsIgnoreCase` function to determine whether two strings contain the same data, ignoring the case of the letters in the String. To check whether the input string contains only ASCII alphabet characters, use the `kony.string.isAsciiAlpha` function that returns a boolean value. Use the `kony.string.isAsciiAlphaNumeric` function to verify if the input string contains only ASCII alphabet characters and numbers. The `kony.string.isNumeric` function verifies if the input string contains only numeric characters. Use the `kony.string.isValidEmail` function to verify if the specified string is a valid email address. The `kony.string.reverse` reverses the characters in the source string. Use the `kony.string.trim` to remove the leading and ending spaces from a source string.

All the functions in the Kony String API are in the **kony.string** namespace. The kony.string namespace contains functions that are deprecated and must therefore not be used in new software development. However, documentation for them is provided to help with the maintenance of legacy apps.

To view the functionality of the String API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

52.1 Overview

The **kony.string** namespace provides static string functions for your use. These static functions augment the functions in the JavaScript `String` type that is available by default. For more information see, https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/string

Note that JavaScript distinguishes between `String` objects and primitive string values.

52.2 kony.string Namespace

The kony.string namespace forms the String API and provides the following API elements.

52.3 Functions

The kony.string namespace provides the following functions.

52.3.1 kony.string.containsChars

This API verifies if any one of the specified set of characters is available in the given string and returns a boolean value.

Syntax

```
kony.string.containsChars (inputstring, characterArray)
```

Input Parameters

Parameter	Description
inputstring [String] - Mandatory	Specifies the input string that needs to be verified.
characterArray [Array] - Mandatory	Specifies the set of characters that need to be searched within the input string.

Example

In this example, *kony.string.containsChars* returns true if the input string consists the characters specified in the characterArray. For example, since the input string *"abcd|ADV"* consists the characters specified in the character array, *kony.string.containsChars* returns true.

```
var inputstring = "abdcdADV";
var charsArr = ["|", "-"];
kony.print(kony.string.containsChars(inputstring, charsArr));
// prints false as | and - are NOT contained in the input
var inputstring = "abdcd|ADV"
kony.print(kony.string.containsChars(inputstring, charsArr));
// prints true as | is contained in the input
```

Return Values

Return Value	Description
status [Boolean]	*true* - if the given input string contains any one of the characters in the specified list. *false* - if there is an error in the input or if the given input string does not contain any of the specified input characters.

Exceptions

An error is thrown if input type is invalid or follows an unexpected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.2 kony.string.containsOnlyGivenChars

This API verifies if only (and only) the specified set of characters is available in the given string and returns a boolean value.

Syntax

```
kony.string.containsOnlyGivenChars (inputstring, characterArray)
```

Input Parameters

Parameter	Description
inputstring [String] - Mandatory	Specifies the input string that needs to be verified.
characterArray [Array] - Mandatory	Specifies the set of characters that need to be searched within the input string.

Example

In this example, *kony.string.containsOnlyGivenChars* returns true only if the input string consists the characters specified in the characterArray. For example, since the input string *"abdcdADV"* consists only the characters specified in the character array, *kony.string.containsOnlyGivenChars* returns true.

```
//Test case where API returns true
var inputStr = "abdcdADV";
var charsArr = ["a", "b", "d", "c", "A", "D", "V"];
var resultantString = kony.string.containsOnlyGivenChars
(inputStr,charsArr);
frmContainsOnlyGivenChars.containsOnlyGivenCharsLabel.text =
resultantString;
//Test case where API returns false
var inputStr = "abdcdADH";
var charsArr = ["a", "b", "d", "c", "A", "D", "V"];
var resultantString = kony.string.containsOnlyGivenChars
(inputStr,charsArr);
frmContainsOnlyGivenChars.containsOnlyGivenCharsLabel.text =
resultantString;
```

Return Values

Return value	Description
status [Boolean]	*true* - if the given input string contains only the characters in the specified list. *false* - if there is an error in the input or if the given input string contains any other characters apart from the specified input characters.

Exceptions

An error is thrown if input type is invalid or follows an unexpected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.3 kony.string.containsNoGivenChars

This API verifies that the input string does not contain any of the specified characters and returns a boolean value.

Syntax

```
kony.string.containsNoGivenChars (inputstring, charsArray)
```

Input Parameters

Parameter	Description
inputstring [String] - Mandatory	Specifies the input string that needs to be verified.
charsArray [Array] - Mandatory	Specifies the set of characters that need to be searched within the input string.

Example

In this example, *kony.string.containsNoGivenChars* returns true only if the input string does not consist the characters specified in the characterArray. For example, since the input string *"abdcdADV"* consists none of the characters specified in the character array, *kony.string.containsNoGivenChars* returns true.

```
var inputstring = "abdcdADV";
var charsArr = ["|", "-"];
kony.print(kony.string.containsNoGivenChars(inputstring, charsArr));
//prints true as string does not contain the given characters (| and -
are NOT contained in the input)
var inputstring = "abdcd|ADV";
var charsArr = ["|", "-", "a" ];
kony.print(kony.string.containsNoGivenChars(inputstring, charsArr));
//prints false as string contains the given character "a".
```

Return Values

Return Value	Description
status [Boolean]	*true* - if the given input string contains none of the characters in the specified list. *false* - if there is an error in the input or if the given input string contains any of the specified input characters.

Exceptions

An error is thrown if input type is invalid or follows an unexpected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.4 kony.string.endsWith

This API returns a boolean value indicating if the source string ends with the specified string.

Syntax

```
kony.string.endsWith(sourcestring, comparestring, ignorecase)
```

Input Parameters

Parameter	Description
sourcestring [String] - Mandatory	Specifies the source string.
comparestring [String] - Mandatory	Specifies the string to be compared with the source string.
ignorecase [Boolean] - Optional	Default value is *true*, i.e, the comparison is not case sensitive. If you want the comparison to be case sensitive, you must specify the value as *false*.

Example

```
//Returns true since comparison is not case sensitive.The resultant is
assigned to a label text.

var resultantString = kony.string.endsWith("Kony Solutions",
"solutions", true);
frmEndsWith.endsWithLabel.text = resultantString;
```

Return Values

Return Value	Description
boolean	*true* - source string ends with compared string. *false* - source string does not end with compared string.

Platform Availability

Available on all platforms.

52.3.5 kony.string.equalsIgnoreCase

Determines whether two strings contain the same data, ignoring the case of the letters in the String.

Syntax

```
kony.string.equalsIgnoreCase(string1, string2)
```

Input Parameters

Parameter	Description
string1 [String] - Mandatory	Specifies the content of the first string.
string2 [String] - Mandatory	Specifies the content of the second string.

Example

Perform a *kony.string.equalsIgnoreCase* on the string "Hello" with another string "HEllo":

```
var ignorecase = kony.string.equalsIgnoreCase("Hello", "HEllo");
kony.print(ignorecase);
```

The *kony.string.equalsIgnoreCase* compares the strings "Hello" and "HEllo" without case sensitivity and returns a Boolean indicating if the strings are a match.

In this example, *true* is the value returned.

Return Values

Return Value	Description
boolean	*true* - both strings contain exactly the same data. *false* - both strings do not contain exactly the same data

Platform Availability

Available on all platforms.

52.3.6 kony.string.isAsciiAlpha

This API verifies if the input string contains only ASCII alphabet characters and returns a boolean value.

Syntax

```
kony.string.isAsciiAlpha (inputstring)
```

Input Parameters

Parameter	Description
inputstring [String] - Mandatory	Specifies the input string that needs to be verified.

Example

In this example, the *kony.string.isAsciiAlpha* API returns true or false based on the input string entered.

```
var inputstring = "abdcdADV";
kony.print(kony.string.isAsciiAlpha(inputstring));
//prints true
var inputstring = "123ab3dcdADV";
kony.print(kony.string.isAsciiAlpha(inputstring));
//prints false
```

Return Values

Return Value	Description
status [Boolean]	*true* - if the input string contains only alphabetic characters. *false* - if there is an error in the input or the input string contains characters other than alphabet.

Exceptions

An error is thrown if input type is invalid or follows an unexpected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.7 kony.string.isAsciiAlphaNumeric

This API verifies if the input string contains only ASCII alphabet characters and numbers, and returns a boolean value.

Syntax

```
kony.string.isAsciiAlphaNumeric (inputstring)
```

Input Parameters

Parameter	Description
inputstring [String] - Mandatory	Specifies the input string that needs to be verified.

Example

In this example, *kony.string.isAsciiAlphaNumeric* returns true or false based on the input string.

```
var inputstring = "abdcdADV";
kony.print(kony.string.isAsciiAlphaNumeric(inputstring));
//prints false because the string is not a combination of alphanumeric
characters
var inputstring = "abcd1234";
kony.print(kony.string.isAsciiAlphaNumeric(inputstring));
//prints true because the string is a combination of alphanumeric
characters
```

Return Values

Return Value	Description
status [Boolean]	*true* - if the input string contains only alphanumeric characters. i.e it can contain a string of numbers, characters or a string containing both numbers and characters. *false* - if there is an error in the input or the input string contains characters other than alphanumeric.

Exceptions

An error is thrown if input type is invalid or follows an unexpected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.8 kony.string.isNumeric

This API verifies if the input string contains only numeric characters, and returns a boolean value.

Syntax

```
kony.string.isNumeric (inputstring)
```

Input Parameters

Parameter	Description
inputstring [String] - Mandatory	Specifies the input string that needs to be verified.

Example

In this example, kony.string.isNumeric returns true if the input string has only numbers.

```
kony.print(kony.string.isNumeric("123ab3dcdADV"));
// prints false
kony.print(kony.string.isNumeric("12344")) ;
// prints true
```

Return Values

Return Value	Description
status [Boolean]	*true* - if the input string contains only numeric characters. *false* - if there is an error in the input or the input string contains characters other than numbers.

Exceptions

An error is thrown if input type is invalid or follows an unexpected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.9 kony.string.isValidEmail

This API verifies if the input string is a valid email address and returns a boolean value.

Syntax

```
kony.string.isValidEmail (inputstring)
```

Input Parameters

Parameter	Description
inputstring [String] - Mandatory	Specifies the input string that needs to be verified if it is a valid email address.

Example

In this example, *kony.string.isValidEmail* returns true only if the input string comprises a valid email address.

```
var inputstring = "abcd@";
kony.print(kony.string.IsValidEmail(inputstring));
//prints false as there are no chars after @
var inputstring = "abcd@tccc";
kony.print(kony.string.IsValidEmail(inputstring));
//prints false as the chars after @ does not have . followed by at
least 2 chars
```

Return Values

Return Value	Description
status [Boolean]	*true* - if the given input string satisfies the email rules and is a valid email address. *false* - if there is an error in the input or if the given input string does not satisfy email rules.

Rules and Restrictions

The API performs the following validations on the input string:

- has at least 6 characters.
- has @.
- has at least 4 characters after @.
- has **.** (dot) followed by at least 2 characters.
- has at least 1 character before @.

Exceptions

An error is thrown if input type is invalid or follows an unexpected structure.

Platform Availability

Available on all platforms.

52.3.10 kony.string.rep

This API generates a string which is equivalent to "*n* copies of the source string concatenated together".

Syntax

```
kony.string.rep(sourcestring,no)
```

Input Parameters

Parameter	Description
sourcestring [String] - Mandatory	Specifies the source string.
no [Number] - Mandatory	Specifies the number of times the source string must be repeated.

Example

Perform a *kony.string.rep* operation on the source string "kony":

```
var resultantString = kony.string.rep("kony",5);
frmRep.concatenatedStringLabel.text = resultantString;
```

The resultant string contains *konykonykonykonykony*.

Return Values

Return Value	Description
concatenatedstring [String]	A string containing *n* copies of the source string concatenated together is returned.

Exceptions

An error is thrown if input type is invalid or does not follow the expected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.11 kony.string.reverse

This API reverses the characters in the source string.

Syntax

```
kony.string.reverse(string)
```

Input Parameters

Parameter	Description
string [String] - Mandatory	Specifies the source string.

Example

Perform a *reverse* operation on the source string "kony":

```
var resultantString = kony.string.reverse("kony");
frmReverse.reversedStringLabel.text = resultantString;
```

The resultant string will contain *ynok*.

Return Values

Return Value	Description
reversedstring [String]	A string containing the characters of the source string in reverse is returned.

Exceptions

An error is thrown if input type is invalid or does not follow the expected structure.

102-Invalid input error

Platform Availability

Available on all platforms.

52.3.12 kony.string.startsWith

This API returns a boolean value indicating if the source string begins with the specified string.

Syntax

```
kony.string.startsWith(sourcestring, comparestring, ignorecase)
```

Input Parameters

Parameter	Description
sourcestring [String] - Mandatory	Specifies the source string.
comparestring [String] - Mandatory	Specifies the string to be compared with the source string.
ignorecase [Boolean] - Optional	Default value is *true*. If you do not specify the ignorecase parameter, the comparison of the string will be case insensitive. If you want the comparison to be case sensitive, you must specify the value as *false*.

Example

Perform a *kony.string.startsWith* on the *source string* "Kony Solutions" for the string "kony" specifying the ignorecase parameter:

```
var resultantString = kony.string.startsWith("Kony
Solutions","kony",true);
frmStartsWith.startsWithLabel.text = resultantString;
```

The *kony.string.startsWith* compares the source string "Kony Solutions" with the string "kony" without case sensitivity and returns a boolean indicating if the source string begins with the specified string.

In this example, *true* is the value returned.

Return Values

Return Value	Description
boolean	*true* - source string begins with the compared string. *false* - source string does not begin with the compared string.

Exceptions

An error is thrown if input type is invalid or does not follow the expected structure.

102-Invalid input error.

Platform Availability

Available on all platforms.

52.3.13 kony.string.trim

This API removes the leading and ending spaces from the source string.

Syntax

```
kony.string.trim(string)
```

Input Parameters

Parameter	Description
string [String] - Mandatory	Specifies the source string.

Example

Remove the leading and ending spaces in " kony solutions ":

```
var resultantString = kony.string.trim("      kony   solutions     ");
frmTrim.trimmedStringLabel.text = "\"" + resultantString + "\"";
```

In this example, *kony solutions* is the value returned.

Return Values

Return Value	Description
trimmedstring [String]	A string without the leading and ending spaces is returned.

Exceptions

An error is thrown if input type is invalid or does not follow the expected structure.

102-Invalid input error

Platform Availability

Available on all platforms.

53. Sync API

The Sync API is a hook that enables a client app on a user's device to access the functionality of the Kony Fabric Sync ORM API through a server-side app.

The Sync API contains the following API elements:

sync Namespace

Function	Description
`sync.getPendingAcknowledgement`	Fetches pending acknowledgment for all objects.
`sync.getPendingUpload`	Fetches all the rows for all objects those are pending for upload.
`sync.init`	Used to initialize the creation of device database for sync.
`sync.reset`	Resets the device database to initial state.
`sync.rollbackPendingLocalChanges`	Used to roll back the application level pending changes which are not synchronized.
`sync.startSession`	Used to start the sync process and download the data from the Enterprise Data Source to the device database.

Sync Object

Method	Description
`<syncObject>.create`	Enables you to create a record in a Sync object.
`<syncObject>.deleteByPK`	Enables you to delete a record using the object's primary key.
`<syncObject>.getAll`	Fetches all the records for a Sync object.
`<syncObject>.getAllDetailsByPK`	Fetches a record using primary key value for a Sync object.
`<syncObject>.getPendingAcknowledgement`	Enables you to fetch pending acknowledgment for a Sync object.
`<syncObject>.getPendingUpload`	Enables you to fetch all the rows for a Sync object which are pending for upload.

Method	Description
`<syncObject>.getXXX`	Retrieves all the records from the related object(XXX) corresponding to the current primary key.
`<syncObject>.remove`	Enables you to delete a record for a Sync object using the *where* clause.
`<syncObject>.rollbackPendingLocalChanges`	Enables you to rollback the object level pending changes which are not synchronized.
`<syncObject>.rollbackPendingLocalChangesByPK`	Enables you to fetch all the records for a Sync object.
`<syncObject>.update`	Enables you to update a record for a Sync object using a *where* clause
`<syncObject>.updateByPK`	Enables you to update a record using the object's primary key.

53.1 sync Namespace

The sync Namespace implements the Sync API. It contains the following API elements.

53.1.1 Functions

The sync Namespace contains the following functions.

53.1.2 sync.getPendingAcknowledgement

This API is used to fetch pending acknowledgment for all objects.

Syntax

```
sync.getPendingAcknowledgement (successCallback, errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function SyncGetPendingAcknowledgement()
sync.getPendingAcknowledgement(successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Sync Pending Acknowledgement Success", nil, "info",
"Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Sync Pending Acknowledgement Failed", nil, "error",
"Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.1.3 sync.getPendingUpload

This API fetches all the rows for all objects those are pending for upload.

Syntax

```
sync.getPendingUpload (successCallback, errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function SyncGetPendingUpload()
sync.getPendingUpload(successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Sync Pending Upload Success", nil, "info", "Ok", "", "",
nil)
end

function errorFailCallback(err)
window.Alert("Sync Pending Upload Failed", nil, "error", "Ok", "",
"Error", nil)
end
```

Platform Availability

Available on all platforms.

53.1.4 sync.init

This API is used to initialize the creation of device database for sync.

Syntax

```
sync.init (initSuccessCallback , initFailCallback)
```

Input Parameters

Parameter	Description
initSuccessCallback [function] - Optional	Specifies the function which will get invoked on success.
initFailCallback [function] - Optional	Specifies the function which will get invoked on failure.

Example

```
function SyncInit()
sync.init(initSuccessCallback ,initFailCallback)
end

function initSuccessCallback(res)
window.Alert("Sync Initialized", nil, "info", "Ok", "", "", nil)
end

function initFailCallback(err)
window.Alert("Sync Failed", nil, "error", "Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms

53.1.5 sync.reset

This API is used to reset the device database to initial state. This API will remove all data from the database.

Syntax

```
sync.reset (successCallback, errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function SyncReset()
sync.reset(successCallback ,errorFailCallback)
end

function successCallback(res)
window.Alert("Sync Reset Success", nil, "info", "Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Sync Reset Failed", nil, "error", "Ok", "", "Error",
nil)
end
```

Platform Availability

Available on all platforms.

53.1.6 sync.rollbackPendingLocalChanges

This API is used to roll back the application level pending changes which are not synchronized.

Syntax

```
sync.rollbackPendingLocalChanges (successCallback, errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function SyncRollbackPendingChanges()
sync.rollbackPendingLocalChanges(successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Sync Rollback Pending Changes Success", nil, "info",
"Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Sync Rollback Pending Changes Failed", nil, "error",
"Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.1.7 sync.startSession

This API is used to start the sync process and download the data from the Enterprise Data Source to the device database.

Syntax

```
sync.startSession(config)
```

Input Parameters

Parameter	Description
config [table] - Mandatory	Specifies the table to set the configuration parameters.

Example

```
function startSyncSession(config)
local config = {}
config.onsyncstart = onsyncstartDemo
config.onuploadstart = onuploadstartDemo
config.onuploadsuccess = onuploadsuccessDemo
config.ondownloadstart = ondowloadstartDemo
config.ondownloadsuccess = ondownloadsuccessDemo
config.onbatchstored = onbatchstoredDemo
config.onbatchprocessingstart = onbatchprocessingstartDemo
config.onbatchprocessingsuccess = onbatchprocessingsuccessDemo
config.onsyncsuccess = onsyncsuccessDemo
config.onsyncerror = onsyncerrorDemo
config.onscopestart = onscopestartDemo
config.onscopeerror = onscopeerrorDemo
config.onscopesuccess = onscopesuccessDemo
// where onXXXXDemo is function which will be called on these
callbacks
// the below values can be passed by the developer
config.userid = "syncadmin";
config.password = "SyncAdmin123";
config.appid = "APPIDNorthwind";
config.serverhost = "10.10.5.63";
config.serverport = 8080;
config.sessiontasks = {
    Scope1 = {
        doupload = true, dodownload = false
    }, Scope2 = {
        doupload = true, dodownload = false
    }
}
```

```
sync.startSession(config) //starting sync session
end
```

Platform Availability

Available on all platforms.

53.2 Sync Object

The Sync object implements the Sync API at the object level.

53.2.1 Methods

The Sync object contains the following methods.

53.2.2 <syncObject>.create

This API enables you to create a record in a Sync object.

Syntax

```
<syncObject>.create (object, successCallback, errorCallback)
```

Input Parameters

Parameter	Description
Object [object] - Mandatory	Specifies the object that needs to be created in the database.
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function CreateProduct()
<syncObject>.create(objectProduct, successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Get All Success", nil, "info", "Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Get All Failed", nil, "error", "Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.3 <syncObject>.deleteByPK

This API enables you to delete a record using the object's primary key.

Syntax

```
<syncObject>.deleteByPK (pk, successCallback, errorCallback)
```

Input Parameters

Parameters	Description
pk [Integer] - Mandatory	Specifies the object's primary key using which the respective row data has to be deleted in the database.
successCallback [function] - Optional	Specifies the function which will get invoked on success.

Parameters	Description
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function DeleteProductByPK()
<syncObject>.deleteByPK(123, successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Delete By Primary Key Success", nil, "info", "Ok", "",
"", nil)
end

function errorFailCallback(err)
window.Alert("Delete By Primary Key Failed", nil, "error", "Ok", "",
"Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.4 <syncObject>.getAll

This API fetches all the records for a Sync object.

Syntax

```
<syncObject>.getAll (successCallback, errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function SyncGetAll()
sync.getAll(successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Get All Success", nil, "info", "Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Get All Failed", nil, "error", "Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.5 <syncObject>.getAllDetailsByPK

This API fetches a record using primary key value for a Sync object.

Syntax

```
syncObject>.getAllDetailsByPK (pk, successCallback, errorCallback)
```

Input Parameters

Parameter	Description
pk [object] - Mandatory	Specify the object's primary key using which the respective row data needs to be fetched from the database on the particular object.
successCallback [function] - Optional	Specifies the function which will get invoked on success.

Parameter	Description
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function SyncGetAllDetailsByPK()
<syncObject>.getAllDetailsByPK(ProductId="123", successCallback,
errorFailCallback)
end

function successCallback(res)
window.Alert("Get All Details By Primary Key Success", nil, "info",
"Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Get All Details By Primary Key Failed", nil, "error",
"Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.6 <syncObject>.getPendingAcknowledgement

This API enables you to fetch pending acknowledgment for a Sync object.

Syntax

```
<syncObject>.getPendingAcknowledgement (successCallback,
errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function GetPendingAcknowledgement()
<syncObject>.getPendingAcknowledgement(successCallback,
errorFailCallback)
end

function successCallback(res)
window.Alert("Get Pending Acknowledgement Success", nil, "info", "Ok",
"", "", nil)
end

function errorFailCallback(err)
window.Alert("Get Pending Acknowledgement Failed", nil, "error", "Ok",
"", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.7 <syncObject>.getPendingUpload

This API enables you to fetch all the rows for a Sync object which are pending for upload.

Syntax

```
<syncObject>.getPendingUpload (successCallback, errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function GetPendingUpload()
<syncObject>.getPendingUpload(successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Get Pending Upload Success", nil, "info", "Ok", "", "",
nil)
end

function errorFailCallback(err)
window.Alert("Get Pending Upload Failed", nil, "error", "Ok", "",
"Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.8 <syncObject>.getXXX

This API retrieves all the records from the related object(XXX) corresponding to the current primary key.

Syntax

```
<syncObject>.getXXX (pk, successCallback, errorCallback)
```

Input Parameters

Parameter	Description
pk [Integer] - Mandatory	Specifies the object's primary key using which the respective row data needs to be fetched from the database.

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
E.g.: If there is a relationship from Order to OrderDetails the below
function will retrieve all the OrderDetails corresponding to that
order.

function GetOrderDetails()
<syncObject>.getOrderDetails(123, successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Get Order Details Success", nil, "info", "Ok", "", "",
nil)
end

function errorFailCallback(err)
window.Alert("Get Order Details Failed", nil, "error", "Ok", "",
"Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.9 <syncObject>.remove

This API enables you to delete a record for a Sync object using the *where* clause.

Syntax

```
<syncObject>.remove (whereclause, successCallback, errorCallback)
```

Input Parameters

Parameter	Description
whereclause [String] - Mandatory	Specifies the string using which the data from database has to be fetched.
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function DeleteProduct()
<syncObject>.remove(ProductId="123", successCallback,
errorFailCallback)
end

function successCallback(res)
window.Alert("Delete Success", nil, "info", "Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Delete Failed", nil, "error", "Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.10 <syncObject>.rollbackPendingLocalChanges

This API enables you to rollback the object level pending changes which are not synchronized.

Syntax

```
<syncObject>.rollbackPendingLocalChanges (successCallback,
errorCallback)
```

Input Parameters

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function RollbackPendingLocalChanges()
<syncObject>.rollbackPendingLocalChanges(successCallback,
errorFailCallback)
end

function successCallback(res)
window.Alert("Rollback Pending Local Changes Success", nil, "info",
"Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Rollback Pending Local Changes Failed", nil, "error",
"Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.11 <syncObject>.rollbackPendingLocalChangesByPK

This API enables you to fetch all the records for a Sync object.

Syntax

```
<syncObject>.rollbackPendingLocalChangesByPK (pk, successCallback,
errorCallback)
```

Input Parameters

Parameter	Description
pk [Integer] - Mandatory	Specify the object's primary key using which the respective row data needs to be rollback in the database
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function <syncObject>.RollbackPendingLocalChangesByPK()
<syncObject>.rollbackPendingLocalChangesByPK(123, successCallback,
errorFailCallback)
end

function successCallback(res)
window.Alert("Rollback Pending Local Changes Success", nil, "info",
"Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Rollback Pending Local Changes Failed", nil, "error",
"Ok", "", "Error", nil)
end
```

Platform Availability

Available on all platforms.

53.2.12 <syncObject>.update

This API enables you to update a record for a Sync object using a *where* clause

Syntax

```
<syncObject>.update (whereClause, successCallback, errorCallback)
```

Input Parameters

Parameter	Description
whereclause [String] - Mandatory	Specifies the string using which the data from database to be fetched.

Parameter	Description
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

Example

```
function UpdateProduct()
<syncObject>.update(ProductId="123", successCallback,
errorFailCallback)
end

function successCallback(res)
window.Alert("Update Product Success", nil, "info", "Ok", "", "", nil)
end

function errorFailCallback(err)
window.Alert("Update Product Failed", nil, "error", "Ok", "", "Error",
nil)
end
```

Platform Availability

Available on all platforms.

53.2.13 <syncObject>.updateByPK

This API enables you to update a record using the object's primary key.

Syntax

```
<syncObject>.updateByPK (pk, successCallback, errorCallback)
```

Input Parameters

Parameter	Description
pk [Integer] - Mandatory	Specifies the object's primary key using which the respective row data needs to be updated in the database
successCallback [function] - Optional	Specifies the function which will get invoked on success.
errorCallback [function] - Optional	Specifies the function which will get invoked on error.

© 2013 by Kony, Inc. All rights reserved

Example

```
function UpdateProductByPK()
<syncObject>.updateByPK("123", successCallback, errorFailCallback)
end

function successCallback(res)
window.Alert("Get Product by Primary Key Success", nil, "info", "Ok",
"", "", nil)
end

function errorFailCallback(err)
window.Alert("Get Product by Primary Key Failed", nil, "error", "Ok",
"", "Error", nil)
end
```

Platform Availability

Available on all platforms.

54. Theme API

The Theme API enables you to apply skins to your application.

A theme is a set of skins available at the application level that let you specify common skins for widgets in different states (normal, focus, and so on). You can define your own themes in Kony Visualizer or download themes from the network (a specific URL) and use them in the application.

The Theme API uses the `kony.theme Namespace` and related API elements:

Function	Description
`kony.theme.createTheme`	Enables you to create a theme.
`kony.theme.createThemeFromJSONString`	Enables you to create or replace a JSON string theme in the current session.
`kony.theme.deleteTheme`	Allows you to delete a specified theme in the application programmatically
`kony.theme.getAllThemes`	Returns all the themes available in the application.
`kony.theme.getCurrentTheme`	Returns the current theme that is applied to the application.
`kony.theme.getCurrentThemeData`	Returns the meta data of the current theme in the application.
`kony.theme.isThemePresent`	Allows you to check the existence of specific theme in the application.
`kony.theme.setCurrentTheme`	Allows you to apply a specified theme to the application at runtime.

A theme is represented as a file containing a JSON string. The JSON string has the key-value pairs in which the key is **skinid** and the value is JSON string that represents the skin.

54.1 Overview

A theme is a set of skins available at the application level. Themes let you specify common skins for widgets in different states (normal, focus, and so on).

You can define your own themes in Kony Visualizer or download themes from the network (a specific URL) and use them in the application.

A theme is represented as a file containing a JSON string. The JSON string has the key-value pairs in which the key is skinid and the value is JSON string that represents the skin.

Note: The themes or the skins must be pre-defined in the IDE or should have been downloaded over the network.

Note: Every application must have a "default" theme (stored in the in the default.theme file). Deleting or overriding the contents of this file may lead to undesired behavior. Kony Visualizer does not allow the developer to delete this theme. All other user defined themes can be deleted.

The Theme API enables you to customize, set, and delete themes in an application. If you already have all the attributes of a theme such as the font size, font colour and so on, listed in a JSON string, you can use a URL to point to the JSON String using the `kony.theme.createTheme` function. If you want to define the attributes while creating the theme, use the `kony.theme.createThemeFromJSONString` function. Use the `kony.theme.getCurrentTheme` function and the `kony.theme.getCurrentThemeData` function to return the current theme and the current theme's meta data respectively. Return all the themes available in your application using the `kony.theme.getAllThemes` function, and then use the `kony.theme.setCurrentTheme` function to select a theme to apply it in your application. You can also check if a specific theme is present in your application using the `kony.theme.isThemePresent` function, and delete the theme using the `kony.theme.deleteTheme` function, if required.

54.1.1 Use Cases

- Themes allow you to package the skins in a way that resemble a particular user interface or branding for a particular set of users. For example, all the *Gold* customers will see all the buttons in *Golden Yellow* color.
- Themes allow you to personalize the look and feel of the application for the end users. For example, the users can choose between a *green theme* or a *blue theme*.
- Themes, when downloaded over the network, allow to remotely control the branding of the native application. For example, during the *Christmas* season, the marketing or the branding team can push a *Christmas Theme* to the native application.

54.1.2 Capabilities

The theme APIs provided by Kony Platform have the following capabilities:

- Defining new themes.
- Applying the theme to the application or only for the current session.
- Switching the application from one theme to another theme dynamically.
- Deleting and loading the skins for memory efficiency.
- Provision for custom meta data.

54.1.3 Important Considerations

- The default theme is the base-point for new theme.
- If the referred skin does not exist both in the current theme and default theme, a runtime error occurs.
- When a skin is referred, the IDE searches for the skin in the order of precedence)
  - Current theme
  - Default theme.

54.1.3.1 Important Considerations for SPA, Desktop Web, and Mobile Web

- For the SPA, Desktop Web and Mobile Web platforms, the theme files should be CSS files. It cannot accept the theme files as JSON files. If the project is developed using IDE, the platform automatically generates CSS files for the above platforms based on the skins/themes defined by the developer in the IDE. The css files will be generated under "<ThemeID>" folder.

  Note: The names of the css files in the theme are same as the css files in the normal build.

- For those customers/developers, interested in developing applications without IDE, the platform will provide an offline tool to convert a given .theme JSON file into corresponding CSS file. This will be provided post 5.0.
- For the createTheme API, ensure that the URL mentioned in the createTheme API should point only to a CSS file. The platforms do not support a URL with a *.Theme* file and convert it into a CSS file at run time.
- getCurrentThemeData will always return null. As theme files are converted into CSS, the platforms cannot have custom structures/variables in CSS files as browsers fail to parse them.

- Do the above platforms support skin level fallback?

  The platforms expect each theme file i.e. *CSS* file to have all the skins. In the case of projects developed in IDE the platform takes care of skin conversion of this during generation. For customers, hand coding the application, the above mentioned tool should take care of that. But essentially at run time the platforms cannot look up into multiple CSS's and check for a class. Technically it is possible to do so but it requires the platform to load multiple CSS files and in a specific order. Adding new downloads is an always avoidable and we intend to handle this during the generation of the CSS file itself.

54.1.4 Limitations

You cannot dynamically modify the skin attributes.

To view the functionality of the Theme API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

54.2 kony.theme Namespace

The kony.theme namespace implements the Theme API. It contains the following API elements.

54.3 Functions

The kony.theme namespace provides the following functions.

54.3.1 kony.theme.createTheme

This API enables you to create a theme.

Syntax

```
kony.theme.createTheme(url, themeIdentifier, onsuccesscallback,
onerrorcallback)
```

Input Parameters

Parameter	Description
url [String] - Mandatory	Specifies a string (URL) from which the theme is to be downloaded. The theme is represented as a JSON object. ***Note:*** If the JSON object contains invalid skin attributes, the platforms use the default attributes (platform specific and may vary from platform to platform).
themeIdentifier [String] - Mandatory	Specifies a flag that indicates if the current theme must be replaced with the same identifier or use the theme only in the current session.
onsuccesscallback [Function]- Mandatory	Specifies the callback function that needs to be executed in case of success. This callback function is executed after the theme is created.
onerrorcallback [Function] - Mandatory	Specifies the callback function that needs to be executed in case of error. This callback function has the following signature: onerrorcallback (errorcode,errormessage) • *errorcode* - the error code thrown if there was a problem while creating the theme • *errormessage* - the error message that corresponds to the error code.

Example

```
function onsuccesscallback() {
    alert("successfully set the theme to app");
}

function onerrorcallback() {
    alert("Theme is not set to the app");
}

kony.theme.createTheme("", "Mytheme", onsuccesscreatecallback,
onerrorcreatecallback);
```

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on all platforms.

For SPA, Desktop Web, and Mobile Web ensure that the URL mentioned in the createTheme API should point only to a CSS file. The platforms do not support a URL with a *.Theme* file and convert it into a CSS file at run time.

54.3.2 kony.theme.createThemeFromJSONString

This API enables you to create or replace a JSON string theme in the current session.

Syntax

```
kony.theme.createThemeFromJSONString(jsonString, themeIdentifier,
onsuccesscallback, onerrorcallback)
```

Input Parameters

Parameter	Description
jsonString [String] - Mandatory	A well-defined theme JSON string with which a theme is created. The theme is represented as a JSON object. You can use the jsonString parameter to set the required skin attributes for various properties that are applicable for Kony widgets. For instance, you can set values for properties that are applicable for several Kony widgets. These properties include background color, font weight, font size, border color, shadow, text shadow, and so on. *Note:* For more information on what properties are applicable for each widget and what values you can specify for each property, refer to the **default.themes** file in the **build-> dist -> Project folder-> assets** path of Kony Visualizer. *Note:* If the JSON object contains invalid skin attributes, the platforms use the default attributes (platform specific and may vary from platform to platform).
themeIdentifier [String] - Mandatory	Specifies an identifier with which current theme must be created or replace if the theme is already exists and this theme will be available only in the current session. *Note:* The theme will not be created and an error callback is called, if null or undefined or non-string themeIdentifier is provided.
onsuccesscallback [Function]- Optional	Specifies the callback function that needs to be executed in case of success. This callback function is executed after the theme is created. *Note:* The theme will be created but the successcallback is not called, if null or undefined successcallback is provided.

Parameter	Description
onerrorcallback [Function] - Optional	Specifies the callback function that needs to be executed in case of error. This callback function has the following signature: onerrorcallback (errorcode,errormessage) • *errorcode* - the error code thrown if there was a problem while creating the theme • *errormessage* - the error message that corresponds to the error code.

As an example, here is a set of values that you can specify for the following applicable properties of a Button widget.

Property	Value
"wtype"	"Button"
"bg_type"	"one"
"background_color"	"ff000000"
"font_weight"	"bold"
"font_size"	120
"font_color"	"314e8900"
"font_name"	"Arial-BoldMT"
"border_color"	"9f9f9f00"
"border_width"	1
"border_style"	"rc"
"shadow"	{"x":0,"y":0,"br":0,"color":"00000000","inner":false}
"text_shadow"	{"x":0,"y":0,"br":0,"color":"00000000"}

Example

```
function testCreateThemeFromJSONString() {
    function onsuccesscallback() {
        kony.print("Successfully created theme.");
    }

    function onerrorcallback(errorcode, errormessage) {
        kony.print("Unable to create theme.");
    }

    var jsonString = '{"metadata":{"currTheme":"MyTheme1",
"themeState":"0"},
     "sknLbl1": {"background_color": "11111164", "bg_type": "one",
     "border_color": "42424216", "border_style": "plain",
     "border_type": 0, "border_width": 0,
     "font_color": "33000016", "font_name": "iphoneSystem",
     "font_size": 200, "font_style": "normal",
     "font_weight": "normal", "isDefaultSkin": false, "wType":
"Label"}}';
    kony.theme.createThemeFromJSONString(jsonString, "MyTheme1",
onsuccesscallback, onerrorcallback);
}
```

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on iOS, Android, and Windows platforms.

54.3.3 kony.theme.deleteTheme

This API allows you to delete a specified theme in the application programmatically.

Important:

- On all Platforms, pre-bundled themes in the application cannot be deleted, but only the themes created through createTheme API, which are in memory, can be deleted.
- On Windows Platforms, only the themes created using the createTheme API can be deleted. Pre-bundled themes and currently used theme cannot be deleted.

Syntax

```
kony.theme.deleteTheme(themeidentifier, onsuccesscallback,
onerrorcallback)
```

Input Parameters

Parameter	Description
themeidentifier [String] - Mandatory	Specifies a string that denotes the theme ID. The specified theme will be deleted from the application.
onsuccesscallback [Function] - Mandatory	Specifies the callback function that needs to be executed in case of success. This callback function is executed after the theme is deleted.
onerrorcallback [Function] - Mandatory	Specifies the callback function that needs to be executed in case of error. This callback function has the following signature: onerrorcallback (errorcode,errormessage) • *errorcode* - the error code thrown if there was a problem while creating the theme • *errormessage* - the error message that corresponds to the error code.

Example

```
kony.theme.deleteTheme ("green");
```

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on all platforms except Server side Mobile Web.

54.3.4 kony.theme.getAllThemes

This API returns all the themes available in the application.

Syntax

```
kony.theme.getAllThemes()
```

Input Parameters

None

Example

```
var themes = kony.theme.getAllThemes();
alert("No Of themes are " + themes.length);
```

Return Values

Return Value	Description
JavaScript: Array	Returns an array with a list of all theme Identifiers available in the application.

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on all platforms.

54.3.5 kony.theme.getCurrentTheme

This API returns the current theme that is applied to the application.

Syntax

```
kony.theme.getCurrentTheme()
```

Input Parameters

None

Example

```
var crntTheme = kony.theme.getCurrentTheme();
alert("current theme is:" + crntTheme+" And the type is " + typeof
(crntTheme));
```

Return Values

Return Value	Description
themeID[String]	Returns the identifier of the current theme that is applied to the application

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on all platforms.

54.3.6 kony.theme.getCurrentThemeData

This API returns the meta data of the current theme in the application.

Syntax

```
kony.theme.getCurrentThemeData()
```

Input Parameters

None

Example

```
function onsuccesscallback() {
    kony.theme.setCurrentTheme("MyTheme1", onsuccesscallbacktheme1,
onerrorcallbacktheme1);
    kony.print(kony.theme.getCurrentThemeData());
}

function onerrorcallback(errorcode, errormessage) {
    kony.print("Unable to create theme.");
}

function fun_createTheme_and_set() {
    var jsonString = '{"metadata":{"currTheme":"MyTheme1",
"themeState":"0"},
      "sknLbl1": {"background_color": "11111164", "bg_type": "one",
      "border_color": "42424216", "border_style": "plain",
      "border_type": 0, "border_width": 0,
      "font_color": "33000016", "font_name": "iphoneSystem",
      "font_size": 200, "font_style": "normal",
      "font_weight": "normal", "isDefaultSkin": false, "wType":
"Label"}}';
    kony.theme.createThemeFromJSONString(jsonString, "MyTheme1",
onsuccesscallback, onerrorcallback);
}
```

Return Values

Return Value	Description
metadata [Object]	Returns an object that contains the metadata of the current theme in the application. In each theme, you can store the metadata (additional key, values) relevant for the theme by using the standard "metadata" key and the same can be read programmatically by using kony.theme.getCurrentThemeData API.

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on all platforms except SPA and Mobile Web.

For SPA, Desktop Web, Windows 7 Kiosk, and Mobile Web, this API will always return null. As theme files are converted into CSS, the platforms cannot have custom structures/variables in CSS files as browsers will fail to parse them.

54.3.7 kony.theme.isThemePresent

This API allows you to check the existence of specific theme in the application.

Syntax

```
kony.theme.isThemePresent(themeidentifier)
```

Input Parameters

Parameter	Description
themeidentifier [String] - Mandatory	Specifies a string that represents a theme.

Example

```
var isThemePresent = kony.theme.isThemePresent("green");
alert("IS theme present ? True/False: " + isThemePresent);
```

Return Values

Return Value	Description
status [Boolean]	Returns the status of the execution of this API. • *true* - if the specified theme is present in the application. • *false* - if the specified theme does not exist in the application.

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on all platforms

54.3.8 kony.theme.setCurrentTheme

This API allows you to apply a specified theme to the application at runtime.

Syntax

```
kony.theme.setCurrentTheme (themeidentifier, onsuccesscallback,
onerrorcallback)
```

Input Parameters

Parameter	Description
themeidentifier [String] - Mandatory	Specifies a string that denotes the theme ID. The specified theme is applied to the application.
onsuccesscallback [Function] - Mandatory	Specifies the callback function that needs to be executed in case of success. This callback function is executed after applying the specified theme.

Parameter	Description
onerrorcallback [Function] - Mandatory	Specifies the callback function that needs to be executed in case of error. This callback function has the following parameters: • *errorcode* - the error code thrown if there was a problem while applying the specified theme • *errormessage* - the error message that corresponds to the error code. This callback function is executed if there is an error while applying the specified theme.

Example

```
function onsuccesscallback() {
    alert("successfully set the theme to app");
}

function onerrorcallback(1900, "Skin Error") {
    alert("Skin does not exist");
}
kony.theme.setCurrentTheme("red", onsuccesscallback, onerrorcallback);
```

Exceptions

- 1900- SkinError. This error occurs when there is an error related to skin.
- Error - This error is thrown when there is a generic error.

Platform Availability

Available on all platforms.

55. Threading API

A subset of native APIs runs only on the main or UI thread. In a Kony application, the bulk of JavaScript code runs on a VM thread (also called a closure thread). Threading APIs help JavaScript bindings work on main thread. When the code completes its run on the main thread, you need a mechanism to resume JavaScript execution on the VM thread.

Threading is a process where multiple threads run at the same time to increase the efficiency of the processor. Using the Threading API, you can run a main thread and a worker thread simultaneously in your JavaScript application.

The Threading API uses `kony Namespace` and the following API elements.

Function	Description
`kony.runOnMainThread`	Helps you run the JavaScript code on the main thread. It is an asynchronous API. It posts a message to the main thread to invoke a function `f` with parameters arguments.
`kony.runOnWorkerThread`	Provides apps with multithreading capabilities.

Create a Main thread or a UI thread using the`kony.runOnMainThread` function to run all the operations that involve interaction with the UI. The Worker thread can be created using the `kony.runOnWorkerThread` function to run all the background tasks in parallel with the main thread. For example, Garbage collection thread is a worker thread that runs in the background to clear all the unused data in an application.

55.1 Guidelines for using Threading API

To use Threading APIs in Kony Visualizer, follow the guidelines:

- Threading APIs natively support multi-threading environment.
- An application can be composed of multiple concurrent threads.
- The UI or main thread is responsible for dispatching events to the user interface widgets and drawing the elements of the UI.
- Do not block the UI thread. Performing long operations, like network access or database queries on the UI thread, will block the user interface.
- Do not access the UI components from outside the UI thread.
- JavaScript Thread:
    - Application logic written in JavaScript that does not require UI update executes in a different thread than the Main or UI thread.
    - Operations that update the UI are posted on to the Main or UI thread.
- Use the API `kony.runOnMainThread(function, args)` to execute JavaScript bindings and JavaScript logic on UI Thread.

55.2 Functions

Threading APIs consist of the following functions, which are a part of the kony Namespace.

55.2.1 kony.runOnMainThread

This API helps you run the JavaScript code on the main thread. It is an asynchronous API. It posts a message to the main thread to invoke a function `f` with parameters arguments.

> ***Note:*** If `runonMainThread` is invoked in a JavaScript function that is already running on the main thread, then the function is executed in synchronous mode.

Syntax

```
kony.runOnMainThread (f, args)
```

Input Parameters

Parameters	Description
f [Function] - Mandatory	Specifies the callback function that must be executed.
args [Array] - Mandatory	Specifies the JavaScript array that holds the parameters to be passed to function `f`.

Example

```
kony.runOnMainThread(mainthread, []);
function mainthread () {
   kony.print ("Running on On Main Thread");
}
```

Return Values

None

Platform Availability

- Android
- iOS

55.2.2 kony.runOnWorkerThread

Provides apps with multithreading capabilities.

Syntax

```
kony.runOnWorkerThread(f,args)
```

Input Parameters

Parameters	Description
f	Specifies the callback function that must be executed.
args	Specifies an array that holds the parameters to be passed to the function indicated by the *f* parameter.

Example

```
kony.runOnWorkerThread(workermethod, []);
function workermethod () {
   kony.print ("Running on On Worker Thread");
}
```

Return Values

None.

Remarks

This function helps you run JavaScript code asynchronously on a worker thread. It posts a message to the worker thread that owns the current JavaScript context to invoke the function specified in the *f* parameter.

Important: The assumption here is that main thread does not own any JavaScript context. The VM/closure thread and worker threads own the JavaScript context. When the `kony.runOnWorkerThread` is invoked from the main thread, a message is posted to the thread that originally invoked the `kony.runOnMainThread`. If the `runonWorkerThread` is invoked in a JavaScript function that is already running on the worker thread, then the function would be executed in synchronous mode.

Platform Availability

- Android
- iOS

56. Timer API

The Timer API helps you to schedule the execution of functions after configured time intervals.

The APIs in this library provide you the ability to schedule the execution of a function block at regular intervals.

The Timer API consists of the `kony.timer Namespace` and the following API elements:

Function	Description
`kony.timer.cancel`	Cancels the timer that has already been scheduled for execution.
`kony.timer.setCallBack`	Specifies the callback function that needs to be executed for a scheduled timer.
`kony.timer.schedule`	Executes the given function after a specified interval of time.

Schedule the execution of functions using the `kony.timer.schedule` function. You can override the callback function of the scheduled timer and set a new callback function using the `kony.timer.setCallBack` function. Use the `kony.timer.cancel` function to cancel the execution of a scheduled timer.

To view the functionality of the Timer API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

56.1 kony.timer Namespace

The kony.timer namespace implements the Timer API. It contains the following API elements.

56.1.1 Functions

The kony.timer namepace provides the following functions.

56.1.2 kony.timer.cancel

This API cancels the timer that has already been scheduled for execution.

Syntax

```
kony.timer.cancel(timerid)
```

Input Parameters

Parameter	Description
timerid [String] - Mandatory	Specifies the unique ID that identifies the timer that needs to be canceled.

Example

```
function buttoncallback2() {

    try {
        kony.timer.cancel("mytimer");
    } catch (err) {
        alert("error in second button onclick and err is:: " + err);
    }
}
```

```
 /*To cancel the scheduled timer use the below code snipppet. Use the
Timer ID of the
  scheduled timer to cancel the timer.
  */
  cancelTimer: function(){
    kony.timer.cancel("timer4");
    alert("You have successfully cancelled the timer");
  }
```

Return Values

None

Exceptions

Error

Error Codes

- 100 - invalid arguments
- 101 - internal error

Implementation Details

If the timer is waiting to be run, it will not be run the next time. If the timer is in the middle of execution, it will be allowed to complete and then canceled.

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

56.1.3 kony.timer.setCallBack

This API specifies the callback function that needs to be executed for a scheduled timer. The callback function handles the logic that needs to be executed after a scheduled timer was run successfully or the scheduled timer failed to execute. The setCallBack timer API is provided to override the existing time call back function which is defined in the "schedule" timer API.

Syntax

```
kony.timer.setCallBack(timerid, callbackfunction)
```

Input Parameters

Parameter	Description
timerid [String] - Mandatory	Specifies the unique ID that identifies the timer.
callbackfunction [Function] - Mandatory	Specifies the callback function that needs to be executed.

Example

```
//nested function

function timerFunc1() {
    alert("Timer invoked");
}
kony.timer.setCallBack("mytimer", timerFunc1);
```

```
//This function is called when you use the setCallBack Timer API
  callbackfunction: function(){
    alert("Hello World");
  },

  //To set a callback function for a scheduled timer, use the below
code snippet.
  setCallBackTimer: function(){
    kony.timer.setCallBack("timer4", this.callbackfunction);
  },
```

Return Values

None

Exceptions

Error

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

56.1.4 kony.timer.schedule

This API executes the given function after a specified interval of time.

Use Cases

You can use this API when you want to execute a function after a specified interval of time. i.e, for example to run a specific function for every 15 seconds.

Syntax

```
kony.timer.schedule(timerid, functionObj, interval, repeat)
```

Input Parameters

Parameter	Description
timerid [String] - Mandatory	Specifies the unique ID that identifies the timer.
functionObj [Function] - Mandatory	Specifies the function that needs to be executed.
interval [Number] - Mandatory	Specifies the time in seconds after which the function needs to be executed.
repeat [Boolean] - Mandatory	Specifies a flag that indicates if the function needs to executed once or repeated. • *true* - indicates that the execution of the function needs to repeated at regular intervals • *false* - indicates that the function needs to be executed just once

Example

```
function timerFunc() //nested function
{
       i=i+5;
       lbl1.text = i+" secs ";
}
kony.timer.schedule("mytimer12",timerFunc, 5, true);
//The function timerFunc will be executed after every 5 seconds.
```

```
  //This function is called when you schedule the timer
  generateAlert: function(){
    alert("You have generated an alert after 3 seconds ");
  },
  //To schedule a timer to display an alert after 3 seconds use the
below snippet.
  scheduleTimer: function(){
    kony.timer.schedule("timer4",this.generateAlert, 3, true);
  },
```

Return Values

None

Exceptions

Error

Platform Availability

Available on all platforms*. *Dummy Implementation on Mobile Web.

57. Toast API

A Toast is a small feedback message that appears on the screen for a short span of time. The Toast API enables your app to display the feedback messages to the user. These messages include warnings or success messages that appear on the user's screen. Users cannot interact with them and they fade away automatically after a short period of time. Users can continue to interact with the app while the toast is displayed.

A Toast is a small feedback message that appears for a short time on the screen. Toasts can be used to display informative messages, and they are non-interactive.

The Toast API uses the `kony.ui Namespace` and the following API elements.

kony.ui Namespace

Function	Description
`kony.ui.Toast`	Creates a Toast object.

Toast Object

Method	Description
`show`	Displays the toast message on screen.

Property	Description
alignConfig	Sets the alignment of the toast.
data	Provides the information that you want to display in the toast.
isVisible	Configures the visibility of a toast.
template	Holds a FlexContainer that is used as the template for the custom toast.
widgetDataMap	Maps the information between widget IDs and keys in the data.

Using the `kony.ui.Toast` function, you can create a toast object, define the text message that you want to display, and the duration of the toast on the screen.

Create a toast object using the kony.ui.Toast function. In the function, define the text message that you want to display, and the duration of the toast on the screen. Then, invoke the toast object's show method to display the toast on screen.

```
var toast = new kony.ui.Toast({"text": "This is the toast's text.",
"duration": constants.TOAST_LENGTH_SHORT});
toast.show();
```

In the Toast object, you can set the alignment of the toast using the alignConfig property, visibility of the toast using the isVisible property, the text that you want to display in different widgets of toast using the data property. Further, you can customize the toast using the template property and map the widget IDs with their keys using the widgetDataMap property.

Use the `alignConfig` property to set the alignment of the toast object. You can check the visibility of the toast using the `isVisible` property. To set the text to be displayed on the toast for different widgets, use the `data` property. Further, you can customize the toast using the `template` property. If you do not set a template, a default text style is applied to the toast. Configure the `widgetDataMap` property to update the toast each time there is a change in data. After you configure the properties of the toast, invoke the `Toast.show` function to display the toast on the screen.

> ***Important:*** The Toast API is only available for the Android platform.

57.0.1 Toast Templates

You can use templates to customize the appearance of toasts. To set a template for a toast, assign a FlexContainer widget to the toast object's template property. You can only use the following widgets in the template.

- Label widget
- Link widget
- RichText widget
- Button widget
- Image widget

> ***Important:*** Usage of widgets other than those that are mentioned earlier, would result in undefined behavior.
> An image widget with a dynamic URL and widget animations are not supported.

To view the functionality of the Toast API in action, download the sample application from the link below. Once the application is downloaded, build and preview the application using the Kony Quantum App.

DOWNLOAD THE APP

57.1 Function

The Toast API contains the following function.

kony.ui.Toast

Creates a `Toast` object.

> ***Important:*** The kony.ui.Toast function is only available for the Android platform.

Syntax

```
kony.ui.Toast(configParams)
```

Input Parameters

Parameter	Description
configParams	A JavaScript object that contains key-value pairs that provide the configuration of the toast to be created. The following keys are supported. • text: The text for the toast to display. • duration: The duration of time that the toast appears on the screen. This must be set to one of the Toast Duration Constants.

Example

```
var toast = new kony.ui.Toast({
  "text": "This is the toast's text.",
  "duration": constants.SHORT
  });
toast.show();
```

Return Values

Returns an instantiated Toast Object.

Platform Availability

- Android

57.2 Toast Object

The toast object contains a collection of methods and properties. You can configure various properties of the toast and invoke the show method to display the toast in your app.

57.2.1 Methods

The Toast object provides the show method.

show

When you invoke the show method, the toast message is displayed on the screen.

Syntax

```
show()
```

Example

```
var toast = new kony.ui.Toast({"
        text": "Hello World.", "duration":constants.TOAST_LENGTH_SHORT});
toast.show();
```

```
kony.ui.Toast({
    "text": "Hello World.",
    "duration": constants.TOAST_LENGTH_SHORT
});
toast.show();
```

Parameters

None

Return Values

None

Platform Availability

- Android

57.2.2 Properties

The Toast object provides the following properties.

alignConfig

Using the alignConfig property, you can configure the alignment of the toast message on the screen.

Syntax

```
alignConfig
```

Example

```
var offset = {
    gravity: constant.TOAST_POS_MIDDLE_CENTER,
    x: "100",
    y: "200"
};
myToast.alignConfig = offset;
```

Type

The alignConfig proeprty is a JavaScript object which contains key-value pairs to set the alignment of the toast. The following keys are supported.

Constant	Description
gravity	Specifies the anchor point for the toast. The value for this key must be a member of the Toast Position Constants. The default value for this key is `constants.TOAST_POS_MIDDLE_CENTER`.
x	Sets the x position of the toast relative to the middle center of the device's screen. The value of this key is only used when the `gravity` key is set to `constants.TOAST_POS_MIDDLE_CENTER`.
y	Sets the y position of the toast relative to the middle center of the device's screen. The value of this key is only used when the `gravity` key is set to `constants.TOAST_POS_MIDDLE_CENTER`.

Read/Write

Read + Write

Platform Availability

- Android

data

Using the data property, you can provide the information that you want to display in the toast.

Syntax

```
data
```

Example

```
myToast.template = Tempflex;
myToast.widgetDataMap = { //specifying the data item IDs and the widgets in a
template
    "Tempflex": "TempFlex",
    "img1": "img1",
    "lbl1": "Lbl1"
};
myToast.data = { //specifying the data item IDs and the data values directly
    "img1": "header.png",
    "lbl1": "Label Custom Toast"
};
```

Type

A JavaScript object that contains key-value pairs consisting of the IDs of each widget and the values for the individual widgets.

Read/Write

Read + Write

Remarks

There are two ways in which you can initialize the object in the `data` property. The first way is to specify the widget IDs and the data values directly. The second is to specify the widget IDs including the widgets in a template. These are demonstrated in the **Example** provided below.

Platform Availability

- Android

---

isVisible

Using the isVisible property, you can configure the visibility of a toast.

Syntax

```
isVisible
```

Example

```
myToast.isVisible = true;
```

Type

Boolean

Read/Write

Read + Write

Remarks

If you set the value of this property to `true`, the toast is displayed after the invocation of the **show** method.

If you set the value of this property to `false`, the toast is not displayed even after the invocation of the **show** method.

Platform Availability

- Android

template

Using the template property, you can set a FlexContainer widget as the template for a toast.

Syntax

```
template
```

Example

```
myToast.template = Tempflex;
myToast.widgetDataMap = {
    "Tempflex": "Tempflex",
    "img1": "img1",
    "lbl1": "lbl1"
};
myToast.data = {
    "img1": "header.png",
    "lbl1": "Label Custom Toast"
};
```

Type

A JavaScript object.

Read/Write

Read+Write

Remarks

If you do not set a template, it uses the default appearance for the toast. Only the following widgets can be used in the template of the toast.

- Label widget
- Link widget
- RichText widget

- Button widget
- Image widget

Note: An Image widget with a dynamic URL is not supported. Widget animations are also not supported.

Platform Availability

- Android

widgetDataMap

Using the widgetDataMap property, you can map information between the widget IDs and keys in the data.

Syntax

```
widgetDataMap
```

Example

```
myToast.template = Tempflex;
myToast.widgetDataMap = {
    "Tempflex": "Tempflex",
    "img1": "img1",
    "lbl1": "lbl1"
};
myToast.data = {
    "img1": "header.png",
    "lbl1": "Label Custom Toast"
};
```

Type

A JavaScript object that contains key-value pairs consisting of the IDs of each data-item and keys.

Read/Write

Read + Write

Remarks

Using this property you can create a mapping between widget IDs and specific items in your app's data. Ensure that the widgetDataMap property accommodates all widget IDs, including widgets referred to in dynamic templates.

After your app provides the data mapping, it updates the data in the toast whenever the data changes.

Platform Availability

- Android

---

58. Worker Thread API

Worker threads are a means to execute different tasks in multiple parallel contexts of execution in a concurrent manner, which can take advantage of multiprocessor and multithreaded environments as well as to keep UI Thread in Application responsive by delegating or offloading work which need not be handled in UI Main thread, to a different secondary thread.

Worker thread is a continuous parallel thread that runs and accepts messages until the time it is explicitly closed or terminated. Messages to a worker thread can be sent from the parent thread or its child worker threads. Through out this document, parent thread is referred as thread where a worker thread is spawned.

Worker can have logic that gets executed parallel for each of the messages that it receives. If worker thread is busy handling messages, incoming messages that it receives will be queued for processing.

Sharing data between parent thread and worker threads is done through message passing and by default variables, functions, or state is not shared.

The current specification is inspired from and based on HTML5 Web Worker threads standard, that use *Message Passing* model or mechanism for communication between threads.

Note: To use AWS in workerthreads, you must follow these steps.
1. Copy all AWS related files from **<workspace>/<appname>/cloudsdks** folder into the **<workspace>/<appname>/workerthreads** folder of the project. This is because while importing the AWS files using Visualizer, the Kony HttpRequest and DOMParser related code are added to the AWS files. These modified files are then saved in the **cloudsdks** folder by Visualizer.
2. After the files have been copied, use the `require(<aws files>)` code in the `workerthread.js` file to import all the AWS files into worker threads context.
AWS objects must be created in the worker thread. Any object created in the regular thread will not work in the worker thread.

The Worker Thread API contains the following API Elements:

kony.worker Namespace

Function	Description
`kony.worker.hasWorkerThreadSupport`	Determines whether the current platform environment has worker thread support.
`kony.worker.WorkerThread`	Creates a WorkerThread object and returns a handle to it. The worker object represents a worker thread.

WorkerThread Object

Method	Description
`addEventListener`	Event Handlers can be registered using *addEventListener()* method on the worker Objects and once registered messages and errors from a worker thread can be received in parent thread.
`close`	Worker thread can be terminated from inner scope of the worker by invoking close(). The worker thread is killed immediately without an opportunity to complete its operations or clean up.
`postMessage`	postMessage() sends a JSON object or String message to the Parent/worker's scope by invoking respective registered "message" event handlers.
`removeEventListener`	removeEventListener() is used to remove the previously registered *message* or *error* event listener that was registered using *addEventListener*().
`terminate`	When called from parent scope immediately terminates the worker. This does not offer the worker an opportunity to finish its operations. It is simply stopped at once.

58.1 Scope

1. Kony Platform version >= 5.6.2.
2. Support for JavaScript.
3. Supported mobile Platforms:
    a. iOS
    b. Android
    c. SPA: Supported List of browsers starting from versions.
4. Supported List of Browsers

SPA

iOS	Blackberry OS	Android Native	Android Chrome	Windows phone OS
5.0-5.1	10.0	4.4	33.0	8

DesktopWeb

IE	Firefox	Chrome	Safari
10.0	4.0	20	5.0

5. Windows:
    a. Windows Phone 8
    b. Windows Phone 8.1
    c. Windows Kiosk
    d. Windows 8.1

58.2 Introduction to Constructor - WorkerThread()

- The *WorkerThread()* constructor creates and returns the handle to the newly created worker thread. The new worker thread can be used by the Parent thread for any further communication with the worker thread.
- To create a worker thread, it requires a JavaScript file name or a functional module name. The *WorkerThread()* constructor is invoked with the JavaScript file or a functional module name as its only argument and a worker thread instance is then created and returned.
- Worker threads may in turn initiate new worker threads.
- If a Javascript file name with ".js" extension is passed as *WorkerThread()* constructor argument, it looks up only in the `workerthreads` directory in `modules/js` path and loads if the file is found. This holds good for functional modules based projects as well as non-functional modules based projects.
- If a function module name is provided as an argument for *WorkerThread()* constructor, in case of functional modules based project then the module will be loaded if found in the modules listing.

58.2.1 Worker Thread Scenarios

The scenarios of using *WorkerThread()* constructor are as follows:

- The *WorkerThread()* constructor creates a new worker thread and returns a handle to the new worker thread, which can be used by the parent thread for any further communication with the worker thread.

  Creating a worker thread requires a JavaScript file name or a functional module name. The *WorkerThread()* constructor is invoked with the JavaScript file or a functional module name as its only argument and a worker thread instance is then created and returned:

  ```
  var worker = new kony.worker.WorkerThread('helper.js');
  var worker = new kony.worker.WorkerThread
  ('functionModuleName');
  ```

- A *message* event handler can be registered with the worker by parent thread to receive messages from the worker thread.

  ```
  worker.addEventListener("message", function (event) {
  ... });
  ```

- To send data from parent to a worker, *postMessage()* method can be used from parent.

  ```
  worker.postMessage({ operation: 'find-edges', input:
  'buffer', threshold: 0.6 } );
  ```

- To send messages back from worker thread to parent thread scope, *postMessage()* can be used.

  ```
  postMessage({'msg':'Data'});
  ```

- To receive a messages inside the worker thread from parent thread, the *message* event handler can be registered using *addEventListener()* inside worker thread.

  ```
  self.addEventListener( "message", function (event) { ...
  });
  ```

58.2.2 Worker Thread Life Cycle

The following steps provide the work flow to use worker thread:

1. Call to Worker constructor will create a new Worker instance and a new parallel execution environment context is created, and immediately starts execution in the new parallel thread of control in an asynchronous manner. In this new thread, first the Worker will try to load the ‘workerjs’ script.

2. As a result of the asynchronous parallel nature of execution in worker thread context, invocation of Worker constructor call in Parent thread will return a new Worker instance handle and Parent proceeds with execution of next instructions.

3. Every Worker thread will have its own event loop which takes care of the execution of all the received message tasks which are queued for this worker in that order until ‘self.close()’ in worker scope or ‘worker.terminate()’ in parent worker scope are invoked.

4. From the moment of successful creation of worker thread and until ‘self.close()’ in worker scope or ‘worker.terminate()’ in parent worker scope are invoked, the worker thread will be alive and can receive and process messages which are sent to this worker form its parent or from its child workers if created, as well as it can send messages using postMessage() to its parent thread and its child worker threads if created.

58.3 message event handler

"message" event handler receives an "event" object which contains the JSON or string message that is passed to postMessage() during invocation and the same message can be accessed from its "event.data" field. The data passed to postMessage() should be a String or JSON object.

Adhering to the JSON standard, the JSON object passed to postMessage() API should be serializable JSON without opaque object handles or function object handles etc. The data which is passed between the parent thread and worker thread using postMessage() API are copied, not shared, so the end result is that a duplicate is created on each end.

Multiple "message" event handlers can also be registered in Parent scope and in workers inner scope and all the registered event handlers will be invoked in the registered order whenever a postMessage () is called.

58.3.1 Syntax

```
function(event) { });
```

58.3.2 Input Parameters

String / JSON Object

- "message" event handler receives an "event" object which contains the JSON or string message that is passed to postMessage() during invocation and the same message can be accessed from its "event.data" field.
- The data passed to postMessage() should be a String or JSON object.
- Adhering to the JSON standard, the JSON object passed to postMessage() API should be serializable JSON without opaque object handles or function object handles etc.
- The data which is passed between the parent thread and worker thread using postMessage() API are copied, not shared, so the end result is that a duplicate is created on each end.
- Multiple "message" event handlers can also be registered in Parent scope and in workers inner scope and all the registered event handlers will be invoked in the registered order whenever a postMessage() is called.

58.3.3 Example

```
var evtMessageHandler_1 = function(event) {
//In case of JSONkony.print ("Received message :" + event.data
["msg"]);"

//In case of string
kony.print ("Received message :" + event.data);
};
```

58.3.4 Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

58.4 error event handler

"error" event handler receives an "event" object which has the has the following three fields: message, filename, lineno. Registered ‘error’ event handler is invoked whenever an unhandled exception arises in worker’s scope. "error" event handler can be registered in parent thread scope on worker object and as well as in worker thread’s inner scope, where both event handlers will be invoked if present whenever an unhandled exception occurs in workers inner scope.

Multiple "error" event handlers can also be registered in Parent scope and in workers inner scope and all registered event handlers will be invoked in the registered order whenever an unhandled exception arises in worker’s scope.

58.4.1 Syntax

```
function(event) { });
```

58.4.2 Input Parameters

error event object (message, filename, lineno)

58.4.3 Example

```
        function(event) {
               kony.print ('ERROR: Line '+ event.lineno + ' in ' + event.filename
': ' + event.message);
  }
```

58.4.4 Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

58.5 Importing scripts

Worker threads can use importScripts() function to import external scripts their scope by providing the JS file name to import. This method takes one or more JavaScript file names to import.

This API is only available in worker thread scope and not in main parent thread scope.

In case of Functional modules based project " kony.modules.loadFunctionalModule()" API can be used to import a functional module into workers scope. Refer Functional Modules specification document for usage help on loadFunctionalModule() API.

importScripts() if invoked with .js file, looks up only in the "workerthreads" directory under "modules/js/" in Kony IDE Project structure to import scripts into workers scope. This holds good for both functional modules based projects and non-functional modules based projects.

In case of loading multiple files using importScripts(), if an error occurs while loading one of the script, then the remaining scripts are not loaded into context scope.

For more information on Functional Module APIs, refer Functional Modules APIs.

58.5.1 Syntax

```
importScripts(".js_file_name");
```

or

```
importScripts("functional_module_name");
```

58.5.2 Input Parameters

JSFileNames [Object]

or

Functional_Module_Name [Object]

- One or more comma separated list of JavaScript file names.

58.5.3 Example

```
importScripts("Utility.js"); // loads Util.js


importScripts("Utility1.js", "Utility2.js", "Utility3.js");
```

58.5.4 Return Values

None

58.5.5 Exceptions

Note: If no argument is given, no exception is raised and it does nothing.

When an error is encountered, the KonyError JS object is thrown with the following information:

Error Code	Name	Message	Reason
3002	WorkerThreadError	importScripts: InvalidParameter. Invalid script name	This exception occurs when the argument passed is not a string.

Error Code	Name	Message	Reason
3002	WorkerThreadError	importScripts: InvalidParameter. Unable to import script. <scriptname>	This exception occurs when it is unable to find and load the JS script.

- In worker scope, if these exceptions are not handled and if an *error* event handler is registered in worker's inner scope or/and in parent scope for this worker object, then it is invoked with an error event object and its message attribute is set as follows:

  Exception 1 - message: "importScripts: InvalidParameter. Invalid script name"

  Exception 2 - message: "importScripts: InvalidParameter. Invalid script name"

Differences in behavior of *importScripts*() and *kony.modules.loadFunctionalModule()* API with respect to Functional Modules:

Without Functional Modules	With Functional Modules
From inside Worker context if importScripts() is used to import external JS scripts the search criteria would be : only "workerthreads" directory.	From inside Worker context if importScripts() is used to import external JS scripts the search criteria would be : only "workerthreads" directory.
kony.modules.loadFunctionalModule() function cannot be used in workers scope to load any FunctionalModule.	kony.modules.loadFunctionalModule() function can be used in workers scope to load any JavaScript script which is part of some Functional Module.

58.5.6 Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

58.6 Using Worker Threads Feature

The following topics helps you to use the worker thread feature:

- Communicating and Data Processing Between Threads
- Nesting of Threads and Performing Parallel Tasks
- Scope Rules and Supported APIs
- FFI and Custom Widgets
- Guidelines and Limitations
- Debugger Support

58.6.1 Communicating and Data Processing Between Threads

Main.js

```
//create new worker
var worker = new kony.worker.WorkerThread('1_worker.js');

//invoked when worker calls postmessage() from its inner scope
worker.addEventListener("message", function (event) {
    kony.print('Parent Scope : onmessage : event.data : ' + event.data
["message"]);
});

kony.print('Parent Scope : Invoking worker.postmessage()');
//will invoke worker's inner scope onmessage()
worker.postMessage({
    'message': 'Hello World From Parent'
});
```

1_worker.js

```
//workers inner scope
//invoked when Parent calls worker.postmessage()
self.addEventListener("message", function (event) {
    kony.print('Worker Scope : onmessage : event.data : ' + event.data
["message"]);
    //call func
    do_something_in_worker();
});

function do_something_in_worker() {
    kony.print('Worker Scope : invoking postMessage()');
    //will invoke Parent worker.onmessage()
    postMessage({
        'message': "Hello World From Worker "
    });
};
```

Expected Output

```
"Parent Scope: Invoking worker.postmessage()"
"Worker Scope: onmessage : event.data : " "Hello World From Parent"
"Worker Scope: invoking postMessage()"
"Parent Scope: onmessage : event.data : " "Hello World From Worker "
```

Explanation

1. In Parent Scope: Creates new worker using new *kony.worker.WorkerThread* ()
2. In Parent Scope: Call to *worker.postMessage*() invokes *message* event handler registered using *addEventListener*() in worker threads inner scope.

3. In Worker Scope: Call to *postMessage*() invokes *message* event handler registered using *addEventListener*() in the parent thread scope.

58.6.2 Nesting of Threads and Performing Parallel Tasks

Main.js

```
try {
    kony.print("Parent Scope: Init test_case_parent_thread()");
    kony.print("Parent Scope: In try block");

    //create new kony.worker.WorkerThread
    var worker = new kony.worker.WorkerThread('WorkerThread.js');

    //invoked when worker calls postmessage() from its inner scope
    worker.addEventListener("message", function (event) {
        kony.print('Parent Scope : onmessage : event.data : ' +
event.data);
    });

    worker.postMessage("Hello from Parent");

} catch (err) {
    kony.print("Parent Scope: In Catch block");

}

//invoke a function
invoke_timer_task();

//
function invoke_timer_task() {
    kony.print("Parent Scope :- kony.timer.schedule - ");
    var timerId = "mytimer12111";
    var i = 0;

    function timerFunc() {

        i++;
        kony.print("Parent Scope :- kony.timer.schedule - In timerFunc
() : " + i);
        if (i > 20) {
            kony.print("Parent Scope :- kony.timer.schedule - Stopping
timer : ");
            kony.timer.cancel(timerId);
        }
    };

    //
    kony.timer.schedule(timerId, timerFunc, 1, true);
    kony.print("Parent Scope :- kony.timer.schedule - Done");
};
kony.print("Parent Scope: Exit test_case_parent_thread()");
```

WorkerThread.js

```
//worker
//workers inner scope

kony.print("Worker Scope: Init");

var worker = new kony.worker.WorkerThread('WorkerThread2.js');

//invoked when Parent calls worker.postmessage()
this.addEventListener("message", function(event) {
       kony.print('Worker Scope : onmessage : event.data : ' + event.data);
});
```

```

self.postMessage("Hello from Worker");
//
invoke_timer_task();

//
function invoke_timer_task() {
    kony.print("Worker Scope :- kony.timer.schedule - ");

    var timerId = "mytimer121";
    var i = 0;

    function timerFunc() {

        i++;
        kony.print("Worker Scope :- kony.timer.schedule - In timerFunc
() : " + i);


        if(i > 20) {
            kony.print("Worker Scope :- kony.timer.schedule - Stopping
timer : ");
            kony.timer.cancel(timerId);
        }
    };
    //
    kony.timer.schedule(timerId,timerFunc, 1, true);
    kony.print("Worker Scope :- kony.timer.schedule - Done");
};
kony.print("Worker Scope: Loading done");
```

WorkerThread2.js

```
//Grand child worker2 - nested worker
//workers inner scope

kony.print("Grand child: Worker2 Scope: Init");

//invoked when Parent calls worker.postmessage()
this.addEventListener("message", function(event) {
       kony.print('Grand child: Worker2 Scope : onmessage : event.data : ' +
event.data);
});

self.postMessage("Hello from Worker2");
//
invoke_timer_task();

//
function invoke_timer_task () {
    kony.print("Grand child: Worker2 Scope :- kony.timer.schedule -
");

    var timerId = "mytimer1211";
    var i = 0;

    function timerFunc()
    {
        i++;
        kony.print("Worker2 Scope :- kony.timer.schedule - In
timerFunc() : " + i + " : Grand child ");

        if(i > 20) {
            kony.print("Grand child: Worker2 Scope :-
kony.timer.schedule - Stopping timer : ");
```

```
            kony.timer.cancel(timerId);
        }
    };

    //
    kony.timer.schedule(timerId,timerFunc, 1, true);
    kony.print("Grand child: Worker2 Scope :- kony.timer.schedule -
Done");
};
kony.print("Grand child: Worker2 Scope : Loading done");
```

58.6.3 Scope Rules and Supported APIs

Global resources in the App context will not be available in the worker thread context as it can lead to Race conditions since no locking mechanisms are provided.

Every worker has its own context of execution, which is not shared between the parent and its worker. As a result the global variables in parent scope are not available in worker scope and vice versa.

Not Supported APIs	• Kony UI APIs and UI object handles • App Global variables

Supported APIs and Platform	String API	iOS	Android	Windows	SPA	Desktop Web
	Math API	iOS	Android	Windows	SPA	Desktop Web
	Table API	iOS	Android	Windows	SPA	Desktop Web
	Standard Kony API	iOS	Android	Windows	Limited Availability * For SPA and Desktop Web, the following APIs are supported: • kony.convertToBase64 • kony.convertToRawBytes • kony.type • kony.props.getProperty • kony.print() is not supported in Safari. • Kony.print() is supported in Mozilla from version 29.0.	
	Network API	iOS	Android	Windows	For SPA and DesktopWeb: Limited Availability ▪ The following API will not be available in SPA & Desktop Web. kony.net.setNetworkCallbacks	
	Offline	Lo	iO	Andr	Windows	

58.6.4 FFI and Custom Widgets

With and without Functional Module in Worker context:

	iOS	Android	Windows	SPA
FFI	Platform will load modules by default			
Custom Widgets	No need to load Custom Widgets in worker scope.			

58.6.5 Guidelines and Limitations

The following guidelines are recommended before using worker thread:

1. The objects "this" and "self" are available in worker thread inner scope that is referred as worker thread itself.

2. Event propagation cannot be stopped by using *event.stopPropagation*() as in HTML specification. Where *event.stopPropagation*() stops the DOM event to be propagated further by breaking the event chain.

3. Data passed between the main thread and workers are copied, but not shared. Objects are serialized as they are handed to the worker, and subsequently, de-serialized on the other end. The main thread and worker do not share the same instance. So the end result is that a duplicate is created on each end. HTML5 worker threads support transferable objects that allow transferring the objects from one thread to other without making a copy.

4. As explicit thread synchronization mechanisms like locking or mutexes are not available in JS environment, you must take required care in scenarios where multiple threads concurrently or simultaneously are trying to access and write/insert data into local database or local datastore using WebSQL or Local datastore APIs, as these are shared resources across the Application context.

    - The outcome of these simultaneous or concurrent accesses to database/datastore might push the database/datastore to inconsistent state and is dependent on the individual platforms WebSQL or DataStore implementations. It is always suggested to avoid such scenarios of multiple threads accessing the database/datastore simultaneously/concurrently.

Limitations in SPA and DesktopWeb:

1. Nested Workers support will be available only if they are supported by the underlying browser platforms.

2. Error event messages in error event handler sometimes might be appended with extra prefix or suffix string messages from underlying browsers (like Uncaught, Uncaught error etc).
3. Even though error event handler is invoked in case of unhandled exceptions, the same event messages might also be logged on browser console.
4. Line number, file name populated in error event object passed to error event handler might be different from user defined files.

> ***Note:*** For SPA and Desktop Web, nested workers are not supported in Google Chrome.

terminate API

As the worker threading model is mapped to underlying native threading models in native platforms, there can be some deviations from what specification says, this is due to technical limitations in the underlying platforms which include:

1. If terminate() is invoked in Parent thread on worker, and if worker is currently executing a large task, it might not immediately terminate, it will continue to execute the task to completion and terminate.
2. If terminate() is invoked in Parent thread on worker, and if there are pending tasks waiting in the message queues for this worker to perform, some platforms might discard all the pending tasks without accepting them for execution and terminate, and on some platform all the pending tasks are executed to completion and then the worker terminates.
3. It is to be noted that to achieve cross platform consistency, it is always advised that terminate() be invoked on the worker once all the tasks in message queues are completed.

close API

1. As the worker threading model is mapped to underlying native threading models in native platforms, there can be some deviations from what specification says, this is due to technical limitations in the underlying platforms which include:

2. When close() is invoked in worker scope, and if worker is currently executing a large task, it might not immediately terminate, it will continue to execute the task to completion and then terminate.
3. When close () is invoked in worker scope, and if there are pending tasks waiting in the message queues for this worker to perform, some platforms might discard all the pending tasks without accepting them for execution and terminate, and on some platform all the pending tasks are executed to completion and then the worker terminates.
4. Hence it is to be noted that to achieve cross platform consistency, it is always advised that close () be invoked in worker scope once all the tasks in message queues are completed.

58.6.6 Debugger Support

Debugger support for worker threads is not available in 5.6.2 release.

58.6.7 Worker Life Cycle Notes

Following are some worker life cycle notes:

- Call to Worker constructor will create a new Worker instance and a new parallel execution environment context is created, and immediately starts execution in the new parallel thread of control in an asynchronous manner. In this new thread, first the Worker will try to load the 'workerjs' script.
- As a result of the asynchronous parallel nature of execution in worker thread context, invocation of Worker constructor call in Parent thread will return a new Worker instance handle and Parent proceeds with execution of next instructions.
- Every Worker thread will have its own event loop which takes care of the execution of all the received message tasks which are queued for this worker in that order until 'self.close()' in worker scope or 'worker.terminate()' in parent worker scope are invoked.
- From the moment of successful creation of worker thread and until 'self.close()' in worker scope or 'worker.terminate()' in parent worker scope are invoked, the worker thread will be alive and

can receive and process messages which are sent to this worker form its parent or from its child workers if created, as well as it can send messages using postMessage() to its parent thread and its child worker threads if created.

- In worker thread scope when there is no "message" event handler registered which indicates that no tasks can be posted or queued by parent for this worker, and if no callbacks are registered in this worker scope for network, timer APIs etc, then after postMessage() call in Workers inner scope if any, worker can terminate itself without worker.terminate() being called in Parent scope or self.close() called in inner scope.
- In case of orphaned threads if no tasks are queued in message handler and if the worker is not waiting for any callbacks to be returned from network, timer APIs etc then the orphaned thread can be terminated.
- If Parent stops either due to unhandled exception raised in it scope, Worker threads still continues to exist and execute.

58.7 kony.worker Namespace

The kony.worker namespace implements the Worker Thread API. It contains the following API elements.

58.7.1 Functions

The kony.worker namespace provides the following functions.

kony.worker.hasWorkerThreadSupport

This API is used during runtime only in parent scope to determine whether the current platform environment has worker thread support.

This API is most useful in case of SPA and Desktop Web platforms where runtime query can be made to determine whether a browser environment supports Worker Threads or not.

Syntax

```
kony.worker.hasWorkerThreadSupport()
```

Input Parameters

None

Example

```
var hasWorkerSupport = kony.worker.hasWorkerThreadSupport();
if (hasWorkerSupport) {
    var worker = new kony.worker.WorkerThread("worker1.js");

    //do something
} else {
    //workers not supported
}
```

Return Values

Return Value	Description
Boolean	*true*: When worker thread support is present. *false*: When worker thread support is not present.

Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

kony.worker.WorkerThread

The kony.worker.WorkerThread function creates a WorkerThread object and returns a handle to it. The worker object represents a worker thread.

Syntax

```
kony.worker.WorkerThread(
    workerjs);
```

Input Parameters

Parameter	Description
workerjs	A string that contains either the name of javascript file name or functional module name which contains worker thread code.

Example

```
try {

    var worker = new kony.worker.WorkerThread("worker1.js");

} catch (e) {

    var err = kony.getError(e);

    if (err instanceof KonyError) {

        kony.print("KonyError Catch : " + JSON.stringify(err));

    }

}
```

Return Values

Returns a `WorkerThread` object.

Exceptions

Error Code	Name	Message	Reason
3001	WorkerThreadError	WorkerThread: MissingMandatoryParameter. Failed to construct WorkerThread	If no argument is given to this function, this exception is thrown and `workerThread` object is not created.

Error Code	Name	Message	Reason
3002	WorkerThreadError	WorkerThread: InvalidParameter. Invalid script name	If the `workerjs` argument is not a string, then this exception is thrown and `workerThread` object is not created.

- If the script or module specified by the `workerjs` argument is not found, and if an error event handler is registered for this `WorkerThread` object in the parent scope, then the error event handler is invoked with an error object and its `message` attribute set to following.

  message:"WorkerThread: InvalidParameter. WorkerThread script not found"

  - In this error scenario no ‘KonyError’ exception will be raised.
  - Also the `filename`, `lineno` properties in the error event object for this error scenario have undefined values.
- If multiple arguments are provided separated by a comma, the first argument is considered as the `workerjs` filename or module name and rest all are ignored.

Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web.

58.8 WorkerThread Object

The WorkerThread object is used to manage a worker thread. For more details, see Worker Thread API.

The WorkerThread Object contains the following API elements.

58.8.1 Methods

The WorkerThread object contains the following methods.

58.8.2 addEventListener

Event Handlers can be registered using *addEventListener()* method on the worker Objects and once registered messages and errors from a worker thread can be received in parent thread.

Due to the asynchronous parallel execution nature of the newly created worker thread, which starts executing the worker JS script upon creation. It is always advisable to immediately register the ‘message’ event handler as well as ‘error’ event handler on the worker handle in parent thread context after creation of the worker. This ensures that posted messages or errors in worker scope are handed over to the respective ‘message’ or ‘error’ event listeners in parent scope.

***message* event handler:**

Registered "message" event handler will be invoked in Parent or Worker thread whenever postMessage() is called from inner scope of Worker or Parent thread.

1. In parent scope, call to worker.*postmessage()* invokes *message* event handler registered using *addEventListener()* in Workers inner Scope.
2. In worker scope, call to *postmessage()* invokes *message* event handler registered using *addEventListener()* in Parents scope.

Multiple *message* event handlers can also be registered in parent scope and workers inner scope and all the registered event handlers will be invoked in the registered order whenever a postMessage() is called.

***error* event listener:**

The *error* handler can also be registered using *addEventListener* () in parent thread scope on worker object and as well as in worker thread’s inner scope, which will be invoked whenever an unhandled exception occurs in workers inner scope.

The error event object has the following three fields: message, filename, lineno.

If an *error* event handler is registered in worker inner scope and in parent scope on the worker object, then both the event handlers are invoked in case of an unhandled exception in worker's inner scope.

Multiple *error* event handlers can be registered on the worker handle in parent scope and also in worker threads inner scope. Registered *error* event handlers will be invoked in the order of registration in case of unhandled exception in worker thread.

If an *error* event handler is not registered for a worker in parent scope or in worker inner scope and if an unhandled exception arises in worker's inner scope then worker halts execution and no error info is propagated.

The *error* event handlers work for unhandled exceptions occurring only in worker scope but not in main parent scope.

Syntax

Parent Scope:

```
worker.addEventListener(key,listener)
```

Worker Scope:

```
addEventListener(key,listener)
self.addEventListener(key,listener)
this.addEventListener(key, listener)
```

Input Parameters

Parameter	Description
key	• *message*: When message event handler is being registered. • *error:* When error event handler is being registered.
listener	The listener parameter indicates the event listener function to be added.

Example

```
//Parent Scope
var worker = new kony.worker.WorkerThread("worker1.js");

var evtMessageHandler_1 = function(event) {
    kony.print("Parent Scope : In message handler 1");
};
var evtErrorHandler_1 = function(event) {
    kony.print("Parent Scope: In error handler 1");

    kony.print('ERROR: Line ' + event.lineno + ' in ' + event.filename
+ ': ' + event.message);
};
```

```

//"message" event handler:


//Parent Scope
var worker = new kony.worker.WorkerThread("worker1.js");

var evtMessageHandler_1 = function(event) {
    kony.print("Parent Scope : In message handler 1");
};

var evtMessageHandler_2 = function(event) {
    kony.print("Parent Scope : In message handler 2");
};

worker.addEventListener("message", evtMessageHandler_1);
worker.addEventListener("message", evtMessageHandler_2);
```

```
//Worker Scope
var evtMessageHandler_1 = function(event) {
    kony.print("Worker Scope : In message handler 1");
};

self.addEventListener("message", evtMessageHandler_1);
//or
this.addEventListener("message", evtMessageHandler_1);
//or
addEventListener("message", evtMessageHandler_1);
```

```
//"error" event listener:


//Parent Scope
var worker = new kony.worker.WorkerThread("worker1.js");

var evtErrorHandler_1 = function (event) {
     kony.print("Parent Scope: In error handler 1");

kony.print('ERROR: Line ' + event.lineno + ' in ' + event.filename +
': ' + event.message);

};

var evtErrorHandler_2 = function (event) {
     kony.print("Parent Scope: In error handler 2");
};

//
```

```
worker.addEventListener("error", evtErrorHandler_1);
worker.addEventListener("error", evtErrorHandler_2);



//Worker Scope
var evtErrorHandler_1 = function (event) {
     kony.print("Worker Scope: In error handler 1");
};


self.addEventListener("error", evtErrorHandler_1);

//or
this.addEventListener("error", evtErrorHandler_1);

//or
addEventListener("error", evtErrorHandler_1);
```

Return Values

None

Exceptions

1. If no argument is given or if the number of arguments are less than two, it raises an exception and throws a "KonyError" JS Object with the following attributes:

   ```
   errorCode: 3001.
   name: "WorkerThreadError".
   message: "addEventListener: MissingMandatoryParameter. Mandatory
   arguments missing"
   ```

2. If the first argument is not equal to "message" or "error" and if second argument is not a function object, it raises an exception and throws a "KonyError" JS Object with the following attributes:

```
errorCode: 3002.
name: "WorkerThreadError".
message: "addEventListener: InvalidParameter. Invalid arguments"
```

3. In Worker scope if these exceptions are not handled and if an "error" event handler is registered in worker's inner scope or/and in Parent scope for this worker object then it is invoked with an error event object and its message attribute set to following.

   In case of exception 1)

   ```
   message: "addEventListener: MissingMandatoryParameter . Mandatory
   arguments missing"
   ```

   In case of exception 2)

   ```
   message: "addEventListener: InvalidParameter. Invalid arguments"
   ```

4. In Parent(not a worker to some other parent) scope if these exceptions are not handled registered "error" event handler will not be invoked.

Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

58.8.3 close

Worker thread can be terminated from inner scope of the worker by invoking close(). The worker thread is killed immediately without an opportunity to complete its operations or clean up.

The tasks pending in the message queue and callbacks registered for network, timer APIs etc are discarded without wait until completion.

Syntax

Worker Scope:

```
close()
self.close()
this.close()
```

Input Parameters

None

Example

```

//worker init

//post a message to parent
self.postMessage("Hello from Worker");

//do some work

//terminate this worker from inner scope
self.close();
```

Return Values

None

Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

Limitations

As the worker threading model is mapped to underlying native threading models in native platforms, there can be some deviations from what specification says, this is due to technical limitations in the underlying platforms which include:

- When close() is invoked in worker scope, and if worker is currently executing a large task, it might not immediately terminate, it will continue to execute the task to completion and then terminate.
- When close () is invoked in worker scope, and if there are pending tasks waiting in the message queues for this worker to perform, some platforms might discard all the pending tasks without

accepting them for execution and terminate, and on some platform all the pending tasks are executed to completion and then the worker terminates.

- Hence it is to be noted that to achieve cross platform consistency, it is always advised that close () be invoked in worker scope once all the tasks in message queues are completed.

58.8.4 postMessage

postMessage() sends a JSON object or String message to the Parent/worker's scope by invoking respective registered "message" event handlers.

- In Parent thread scope, call to worker.postMessage() invokes "message" event handler which is registered using addEventListener() in Workers inner Scope.
- In Worker thread scope, call to postMessage() invokes "message" event handler which is registered using addEventListener() in Parents scope.

The JSON or string object passed to *postMessage()* is delivered to the registered "message" event handler. The "event" object "data" attribute will have the message passed: "event.data".

Adhering to the JSON standard, the JSON object passed to postMessage() API should be serializeble JSON without opaque object handles or function object handles.

Syntax

```
postMessage(<String> or <JSON>)
```

Parent Scope:

```
//JSON Object
worker.postMessage({‘msg’ : ‘hello’})
```

```
//String
worker.postMessage("messsage hello")
```

Worker Scope:

```
//JSON Object
postMessage({‘msg’ : ‘hello’})
```

```
//String
postMessage("messsage hello")
```

Input Parameters

Parameter	Description
String	For parent scope, here is an example: worker.`postMessage("messsage hello");` For worker scope, here is an example: `postMessage("messsage hello");`
JSON Object	For parent scope, here is an example: worker.`postMessage({'msg' : 'hello'});` For worker scope, here is an example: `postMessage({'msg' : 'hello'});`

Example

```

var worker = new kony.worker.WorkerThread("worker1.js");

// Parent scope:
//JSON Object
worker.postMessage({
    'msg': 'hello'
});

//String
worker.postMessage('message hello');

// Worker scope:

//JSON Object
postMessage({
    'msg': 'hello'
});
//or
this.postMessage({
    'msg': 'hello'
});
//or
self.postMessage({
    'msg': 'hello'
});



//String
postMessage('message hello');
//or
this.postMessage('message hello');
```

```
//or
self.postMessage('message hello');
```

Return Values

None

Exceptions

1. If no argument is given it raises an exception and throws a "KonyError" JS Object with the following attributes:

   ```
   errorCode: 3001.
   name: "WorkerThreadError".
   message: "postMessage: MissingMandatoryParameter. Message
   undefined"
   ```

2. If the message argument passed is not String or JSON object, it raises an exception and throws a "KonyError" JS Object with the following attributes:

   ```
   errorCode: 3002.
   name: "WorkerThreadError".
   message: "postMessage: InvalidParameter. Invalid Message"
   ```

3. In Worker scope if these exceptions are not handled and if an "error" event handler is registered in worker’s inner scope or/and in Parent scope for this worker object then it is invoked with an error event object and its message attribute set to following.

   In case of exception 1:

   ```
   message: "postMessage: MissingMandatoryParameter. Message
   undefined"
   ```

   In case of exception 2:

   ```
   message: "postMessage: InvalidParameter. Invalid Message"
   ```

4. In Parent (not a worker to some other parent) scope if these exceptions are not handled registered "error" event handler will not be invoked.

Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

58.8.5 removeEventListener

This API is used to remove the previously registered *message* or *error* event listener that was registered using *addEventListener*().

This API can be used in both parent and worker scope.

Syntax

Parent Scope:

```
worker.removeEventListener(key,listener)
```

Worker Scope:

```
removeEventListener(key,listener)
self.removeEventListener(key,listener)
this.removeEventListener(key, listener)
```

Input Parameters

key

Key	Description
message	In case of message event handler.
error	In case of error event handler.
listener	The listener parameter indicates the event listener function to be removed.

Example

```
"message" event handler:

//Parent Scopevar worker = new kony.worker.WorkerThread("worker1.js");

var evtMessageHandler_1 = function(event) {
    kony.print("Parent Scope : In message handler 1");
};

var evtMessageHandler_2 = function(event) {
    kony.print("Parent Scope : In message handler 2");
};

//Adding event listeners
worker.addEventListener("message", evtMessageHandler_1);
worker.addEventListener("message", evtMessageHandler_2);

//Removing event listener
worker.removeEventListener("message", evtMessageHandler_2);


//Worker Scope
var evtMessageHandler_1 = function(event) {
    kony.print("Worker Scope : In message handler 1");
};

var evtMessageHandler_2 = function(event) {
    kony.print("Worker Scope : In message handler 2");
};

//Adding event listeners
```

```
self.addEventListener("message", evtMessageHandler_1);
self.addEventListener("message", evtMessageHandler_2);

//Removing event listener
self.removeEventListener("message", evtMessageHandler_2);
```

```
"error" event listener:

//Parent Scope
var worker = new kony.worker.WorkerThread("worker1.js");

var evtErrorHandler_1 = function(event) {
    kony.print("Parent Scope: In error handler 1");
};

var evtErrorHandler_2 = function(event) {
    kony.print("Parent Scope: In error handler 2");
};

//adding event listeners
worker.addEventListener("error", evtErrorHandler_1);
worker.addEventListener("error", evtErrorHandler_2);

//removing event listener
worker.removeEventListener("error", evtErrorHandler_2);



//Worker Scope
var evtErrorHandler_1 = function(event) {
    kony.print("Worker Scope: In error handler 1");
};

var evtErrorHandler_2 = function(event) {
    kony.print("Worker Scope: In error handler 2");
};

//adding event listeners
self.addEventListener("error", evtErrorHandler_1);
self.addEventListener("error", evtErrorHandler_2);

//removing event listener
self.removeEventListener("error", evtErrorHandler_2);
```

Exceptions

1. If no argument is given or if the number of arguments are less than two, it raises an exception and throws a "KonyError" JS Object with the following attributes:

   ```
   errorCode: 3001.
   name: "WorkerThreadError".
   message: "removeEventListener: MissingMandatoryParameter.
   Mandatory arguments missing"
   ```

2. If the first argument is not equal to "message" or "error" and if second argument is not a function object, it raises an exception and throws a "KonyError" JS Object with the following attributes:

   ```
   errorCode: 3002.
   name: "WorkerThreadError".
   message: "removeEventListener: InvalidParameter. Invalid
   arguments"
   ```

3. In Worker scope if these exceptions are not handled and if an "error" event handler is registered in worker’s inner scope or/and in Parent scope for this worker object then it is invoked with an error event object and its message attribute set to following.

   In case of exception 1:

   ```
   message: "removeEventListener: MissingMandatoryParameter.
   Mandatory arguments missing"
   ```

In case of exception 2:

```
message: "removeEventListener: InvalidParameter. Invalid
arguments"
```

4. In Parent(not a worker to some other parent) scope if these exceptions are not handled registered "error" event handler will not be invoked.

Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

58.8.6 terminate

When called from parent scope immediately terminates the worker. This does not offer the worker an opportunity to finish its operations. It is simply stopped at once.

The tasks pending in the message queue and callbacks registered for network, timer APIs are discarded without wait until completion.

Syntax

```
worker.terminate()
```

Input Parameters

None

Example

```
var worker = new kony.worker.WorkerThread("worker1.js");

//Post a message to the worker
worker.postMessage("Hello from Parent");
```

```
//Terminate the worker
worker.terminate();
```

Return Values

None

Platform Availability

Available for iOS, Android, Windows, SPA, and Desktop Web. For more information, see Scope.

Limitations

As the worker threading model is mapped to underlying native threading models in native platforms, there can be some deviations from what specification says, this is due to technical limitations in the underlying platforms which include:

- When terminate() is invoked in Parent thread on worker, and if worker is currently executing a large task, it might not immediately terminate, it will continue to execute the task to completion and terminate.
- When terminate() is invoked in Parent thread on worker, and if there are pending tasks waiting in the message queues for this worker to perform, some platforms might discard all the pending tasks without accepting them for execution and terminate, and on some platform all the pending tasks are executed to completion and then the worker terminates.
- Hence it is to be noted that to achieve cross platform consistency, it is always advised that terminate() be invoked on the worker once all the tasks in message queues are completed.

59. Exceptions

This section covers the following topics:

59.1 JavaScript Native Exceptions:

JavaScript natively supports the following error objects as per the ECMA specification:

- Error
- EvalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Error - A generic error class which is a super class of all other specific error classes in JavaScript. This is the error class that programmers typically use to throw exceptions. All other subclass errors are thrown by the JavaScript engines.

EvalError - This error is thrown when there is illegal use of Eval() function.

RangeError - It is thrown when a numeric value is not in its valid range.

ReferenceError - It is thrown when you attempt to access a variable that does not exist.

SyntaxError - It is thrown when certain functions experience syntax errors like RegExp(), Function(), eval().

TypeError - It is thrown when the value is not of the type required for the context. Accessing a property of null or undefined values can cause this error. Accessing an undefined method for a class can also throw this error. It is also thrown when a function is called with too many arguments specified.

URIError - It is thrown when there is an error in decoding a URI.

As per ECMA spec, all the above error types have two properties:

name - The type of the exception. All subclasses, by default, populate this property with value as its Constructor name, which is TypeError class will populate name with "TypeError".

message - A string describing the exception.

The above native exceptions are thrown by the JavaScript engine in scenarios when trying to access a missing property, or trying to invoke an undefined method.

Throw Statement:

In JavaScript, a developer can throw an error using the "throw" statement and the same can be caught using the try/catch block.

For example, to throw an exception, the JavaScript statement is as follows:

```
throw new Error("You have an exception");
```

or you can throw any Primitive type like String, number and boolean:

```
throw "You have an exception";
```

or

```
throw 1001;
```

or even any JavaScript object can be thrown using throw statements:

```
throw new MyError("errorcode", "name", "message");
//MyError is your own custom JavaScript object, need not be one of the
native JavaScript error types.
```

59.2 Kony Error

As JavaScript *throw* statements support any JavaScript object which can be caught in the try/catch block, Kony APIs leverage this capability to throw the custom error object in case of errors.

Kony APIs throw this custom type of error object in different scenarios, in addition to the above native JavaScript error types.

Kony APIs throw a custom error object (JavaScript Object) called KonyError and will have the below properties:

- **errorCode** - the error code to represent a specific exception type. This is additional and you will not find this in JavaScript Native exceptions.
- **name** - the name of the error, similar property as in the native Error object. This will be predefined value from the platform with meaningful names representing the error / exception.

  For example, "MissingMandatoryParameter" for errorcode 121, "InvalidInputValue" for errorcode 122 etc.
- **message** - the error message, similar property as in native Error object.

As Kony APIs cannot throw any error other than **KonyError**, developers are expected to distinguish the errors based on the name and error code properties.

Please note that **KonyError** is not a replacement for the error callbacks that are available in the Kony APIs. Any error callbacks present in the API continue to be the same.

59.2.1 Error Names and Codes

The following are the Error Names with corresponding codes and descriptions:

Error - thrown when there is a generic error.

- 100 - invalid type of parameters.
- 101 - invalid number of arguments.
- 102 - invalid input - thrown when the input is invalid based on the context.
- 103 - invalid operation - thrown when the operation is invalid based on the context.
- 104 - not supported error - thrown when the method is not supported at all.
- 105 - index out of range.
- 106 - unknown error.

WidgetError - All widget related error codes are wrapped in this error object type. Various error conditions related to Widget creation will be covered through the following error codes:

- 1100 - Not a widget error.
- 1101 - Widget with duplicate ID error when widget is added to the hierarchy.
- 1102 - Widget cannot be created due to invalid input data.
- 1103 - Same widget instance is added to multiple hierarchies.
- 1104 - Widget cannot be created due to insufficient data (mandatory parameters missing)

NetworkError - All network related error codes are wrapped in this error object type. Various error conditions related to network will be covered through the following error codes:

- 1000

i18nError - All error codes thrown by the Internationalization APIs. Various error conditions related to i18n will be covered through the following error code:

- 1300

ContactsError - All error codes thrown by the Contact APIs. Various error conditions related to Contact will be covered through the following error code:

- 1400

DataStoreError - All error codes thrown by the DataStore APIs. Various error conditions related to DataStore will be covered through the following error code:

- 1500

LocationError - All error codes thrown by the location based service APIs. Various error conditions related to LocationError will be covered through the following error code:

- 1600

PushNotificationError - All error codes thrown by the Push Notification APIs. Various error conditions related to Push Notification will be covered through the following error code:

- 1700 - Unable to connect to push service - Push Notification Service (PNS) is not available.
- 1701 - Registration failed by PNS - Account related or device restrictions (thrown by kony.push.register() API during an onfailureregistration callback).
- 1702 - Deregistration failed - Unable to close channel or PNS internal error while deregistration (thrown by kony.push.deRegister() API during an onfailurederegistration callback).
- 1703 - Duplicate Registration (thrown by kony.push.register() API during an onfailureregistration callback).
- 1704 - Platform Specific issue. Complete details are available in the error message. JavaScript Example: Received payload but payload is in incorrect format etc.

PropertyAccessError - All error codes thrown by the Push Notification APIs. Various error conditions related to property access will be covered through the following error code:

- 1800 - No such property exists. Thrown when a non-existent property is accessed.
- 1801

SkinError - All error codes thrown when there is an error related to the skins.

- 1900

CryptoError - All error codes thrown by Crypto API. Various error conditions related to CryptoError will be covered through the following error code:

- 2001 - unsupported algorithm.
- 2002 - invalid key strength specified.
- 2003 - insufficient buffer provided for specified operation.
- 2004 - memory allocation failure.
- 2005 - input data did not encode or encrypt properly.
- 2006 - specified name already exists.
- 2007 - key with the specified unique ID is not found.

PhoneError - All error codes thrown by Phone API.

Various error conditions related to Phone Error will be covered through the following error code.

- 2100: Unable to send the Message.
- 2101 - Insufficient Permissions.

- 2102 - Cannot open mail, mail not configured.
- 2103 - Cannot open media gallery.

StreamingError - All error codes thrown by Streaming API. Various error conditions related to Stream Error will be covered through the following error code.

- 2201 - Streaming identifier does not exist.

Note: The actual streaming error is returned in the callback function.

Please refer to the API section for error details. Each API mentions the error name and the error code that it throws.

59.2.2 Accessing the Kony Error

While mapping the Native Exceptions that are thrown by underlying platforms to the JavaScript Exception, it was found that for some platforms like BlackBerry, it is not consistent with other implementations.

```
try
{
        ….
}
catch ( e )
{
         if (e instanceof KonyError)
         alert("A Kony error");
         else (e instanceof EvalError)
         alert("A JS Eval Eror");
}
```

The above code snippet could have been ideal from the developer's perspective, but due to limitations in BlackBerry in mapping the native Java exceptions to JavaScript, the recommended cross platform means of accessing the Kony Error objects is as follows:

```
try
{
….
}
catch ( e )
{
         var err = kony.getError(e);
         if (err instanceof KonyError)
         alert("A Kony error");
         else (err instanceof EvalError)
         alert("A JS Eval Eror");
}
```

kony.getError will give you access to the actual error that is thrown by the JavaScript Engine or by the underlying platforms.

Please refer the Kony APIs section for the respective **type of error** object, **error codes** and the defined **name** against each API.

60. Appendix: General FAQ

This appendix explains a few frequently asked questions (FAQs).

60.1 Network Calls

Can I cancel synchronous network calls?

Canceling synchronous network calls in not a cross platform feature. iPhone platform provides provide a means of canceling synchronous network calls, which is very specific to those platforms.

For example, on iPhone, the developer can choose to show the Cancellation popup. If this option is available, the cancel option pops up from the bottom of the screen allowing the developer to cancel the synchronous service call.

Can I handle the logic that needs to be executed when a synchronous network call is canceled?

Yes. When a network call is canceled, an error is thrown with *opstatus* 1021. Because cancellation of synchronous calls is not cross platform, ensure that any logic that is written to handle 1021 *opstatus* does not have any business logic but displays a proper message to the user.

Is blocking the UI still an option that I can choose while making a synchronous network call?

The *isBlocking* parameter is not considered by platforms (except for iPhone). Only way to block the user interface is by explicitly using `showLoadingScreen` API supported by Kony Platform.

What is the advantage of asynchronous network calls compared to the synchronous network calls?

Any network call goes through the following phases:

1. Phase 1: preparing network request
2. Phase 2: making a network connection and wait for response

3. Phase 3: handle the returned response.

In case of synchronous calls, user clicks and touches on any widget of the form are ignored until network call finishes all the three phases. In case of asynchronous network calls, users can execute any logic while the network is busy waiting for the response.

Phase 1 and Phase 2 are handled by Kony Platform and Phase 3 a callback function is executed (the function that is passed to the asynchronous network call).

What are the important considerations for using Asynchronous network calls?

Kony Platform runs all the functions attached to the widgets and the callback for asynchronous network call in a single thread except for some of the system events.

The callback function is queued if any of the functions are in execution. This may lead to the order of execution issues.

Because the user can execute any other functions while waiting for the response of the network call, this may lead to the order of execution issues.

For example, user is performing account deletion logic by invoking a network call asynchronously. During the same time, user may end up updating the same account by executing another network call or a function. This could lead to erroneous data.

For example, user is performing the retrieval of accounts through asynchronous network call and if the user moves to the functionality that depends on the availability of the accounts, then application should display the appropriate message or block the UI till account retrieval is completed.

When to use asynchronous network calls?

In general, asynchronous calls should be preferred over synchronous calls in all situations. However, user should be aware that order of execution issues may creep in and handle such situations.

When you want to make network calls as part of the following events, use asynchronous network calls:

- pre-appinit

- post-appinit
- transactionalDataLoad
- onIdleTimeout
- onBackground
- onForeground
- appmenuevents.

How to handle error codes thrown by the network calls?

Define handlers for all the error codes including the generic error code. If a particular platform cannot distinguish an error case specifically, the platform throws a generic error.

Can I click/touch the widgets on a form and trigger an action when a function is getting executed?

No. As explained earlier, because executing the logic corresponding to touches/gestures on the form is run in a single thread.

Are the asynchronous callbacks (associated with asynchronous network call) executed when a function is under execution?

Yes. All Asynchronous callbacks are queued and executed on the main thread. Hence the callbacks are always guaranteed for execution when the current thread completes its current job.

Can I control the timeout of a network call?

No. Kony Platform does not support changing the timeout period. The default timeout period is 3 minutes.

Can I make the network calls outside the Kony Application Server?

No. Kony Platform does not support making network calls outside the Kony Application Server.

Are the network calls authenticated against the Kony Application Server?

Yes. The basic form based authentication is used between devices and the Kony Application Server. The relevant authentication headers are added to the application code in a transparent way.

Can the platform support un-trusted certificates when making HTTPS network calls ?

Apple does not allow usage of the private APIs (used to support un-trusted certificates) in the applications as part of application certification process. In platforms like Android, though there are ways to allow un-trusted certificates it is not guaranteed across the family of Android devices. Using un-trusted certificates also leads to security issues. Hence, on Kony Platform use of valid and trusted certificates is mandatory for secure (HTTPS) communication.

What are the certificates that are treated as trusted and work cross-platform?

iPhone

http://support.apple.com/kb/HT3580

http://support.apple.com/kb/HT4415.

Android

Only the certificates issued by CA that are bundled with the application.

Windows Phone

- America Online Root Certification Authority 2
- America Online Root Certification Authority 1
- COMODO RSA Certification Authority
- USERTrust RSA Certification Authority
- COMODO ECC Certification Authority
- USERTrust ECC Certification Authority

- UTN-USERFirst-Hardware
- UTN - DATACorp SGC
- AddTrust External CA Root
- DigiCert Assured ID Root CA
- DigiCert High Assurance EV Root CA
- DigiCert Global Root CA.

For more certificates, see http://www.microsoft.com.

For uniform cross-platform behavior, select a trusted root certificate which works across the devices.

How do I disable the widgets (such as text box, text area) or the complete form while a network call is in progress ?

You have the following options:

- Use the window.showloadingscreen API to block the UI while the network call is in progress.
- Use widget.setenabled method to disable the widgets one after the other. But in this approach, you must remember to enable the widgets after the completion of the network call.

60.2 form.destroy API

When I invoke form.destroy API, will I lose the widgets and data associated with the widgets also?

Yes. When `form.destroy` is invoked, complete form along with its widgets and data associated with the widgets will be lost. It is the developer's responsibility to save any critical, easily non-reproducible data using `ds.save` before invoking `form.destroy` API.

Can I access the form that was destroyed?

Yes. If you access the form that was destroyed, either by calling form.show or by accessing the widgets on the form or by accessing the properties of the form, the form will undergo through the entire form lifecycle.

---

61. Appendix: JavaScript FAQ

Can I use the standard JavaScript features like prototypes, "this", function semantics, or do I have to use a "restricted JavaScript"?

Yes, the JavaScript runtime environment embedded by the Kony platform is ECMA 262 compliant. So all the standard JavaScript features are available out of the box. All the custom Functions and Objects defined by the developer have access to standard JavaScript features. Though the JavaScript Objects/Functions created/defined by the Kony APIs adhere to this specification there are some known limitations that one must be aware of. These limitations mostly arise from the limitations posed by the JavaScript runtime environment when invoking native SDK APIs (defined in a different language). These limitations have been captured under the sections "JavaScript Function Semantics" and "JavaScript Object Semantics".

What are the different JavaScript runtime environments that Kony platform bundles for each of the Mobile platforms?

Platform	Supported Version	JavaScript Engine
iOS	4.0 and above	JS Core
Android	2.1 and above	Google V8 (via NDK)
MobileWeb	All	---
SPA	iOS,Android,BlackBerry,Windows 7.5	JS engine embedded in the browser
Windows Phone	7.5 and above	Jurassic
Windows Desktop	Coming soon	Coming soon

Are there any JavaScript constraints or cross platform nuances that I should be aware of when developing on Kony platform ?

Yes, even though Kony platform bundles or uses widely used JavaScript runtime environment, there are some JavaScript APIs that don't work consistently across the JavaScript runtime environments. Below are some of the JavaScript operators and semantics that can result in inconsistent experience across runtimes and should be avoided:

- *typeof* operator for the Objects created and returned by the Kony APIs
- *instanceof* operator for the Objects created and returned by the Kony APIs

---

I like that Kony now supports JavaScript. Can I pick a JavaScript based 3rd party library and include it in my project codebase ?

Yes, you can integrate any Javascript library which is purely written in Javascript language without any native, browser dependencies. In case the JavaScript library depends upon a browser DOM then it would only work on the SPA channel.

I see some of the widgets like Form and Image have got a "2" as suffix when creating the widget. For instance, when creating the form we have to say com.kony.ui.Form2. Why do we need to have "2" as suffix for these widgets ?

When drafting the specifications for the JavaScript APIs there was a conscious effort to ensure common cross platform behavior of some of the widgets. We realized that unless we break some backward compatibility the widget behavior cannot be made consistent across platforms. Hence Kony came with new API definitions for these widgets with the intention to consolidate the cross platform behavior. A version "2" of these widgets was introduced.

So in general is it recommended to use kony.ui.Image2 or kony.ui.Image, should I use the version "2" of the widgets or their original version ?

For better cross platform behavior we recommend you to use the kony.ui.Image2 widget wherever possible. The original version of the widgets have been retained only for backward compatibility.

Can I use Octal literals in my JavaScript Code?

Octal literals should be avoided in JavaScript Code. Numbers prefixed with 0 will be treated as octal numbers by most of the Javascript Engines but some throw an exception. If you want to parse the octal number from a string variable, use *parseInt* with *radix* parameter.

62. Appendix: Preprocessor Directives

Preprocessor directives are handled by the compiler's preprocessor. before the program is actually compiled for the target platform. Preprocessor directives start with the pound sign (#), and all preprocessor directives are processed before anything else in the source file. After the substitutions are performed, the program is bundled with the native executable.

Preprocessor directives are stored in comments, so they all begin with the symbols //#. For example, you can define an identifier as follows.

```
//# define MY_IDENTIFIER
```

Because Kony Visualizer sees statements starting with the characters //# as preprocessor directives, it will report an error if your source code uses the characters //# as something other than a preprocessor directive.

Kony Visualizer supports the following preprocessor directives.

Preprocessor Directive	Description
//#define <identifier>	Defines an identifier.
//#else	Provides an alternative block of code to parse on #ifdef or #ifndef statements.
//#elseif	Conditionally provides an alternative block of code to parse on on #ifdef or #ifndef statements.
//#endif	Closes a code block on #ifdef or #ifndef statements.

Preprocessor Directive	Description
//#ifdef <identifier>	Tests whether an identifier is defined. If it is, the statements in the code block are parsed until either an #endif, #else, or #elseif directive is reached. If the identifier is not found and there is no #else or #elseif, no statements in the code block are parsed.
//#ifndef <identifier>	Tests whether an identifier is not defined. If it is not, the statements in the code block are parsed until either an #endif, #else, or #elseif directive is reached. If the identifier is found and there is no #else or #elseif, no statements in the code block are parsed.
//#undef <identifier>	Removes an identifier that was previously defined, If the identifier is not found, this statement does nothing.

63. Appendix: SQLite

SQLite is a database engine packaged along with the native sdk. This section comprises the following:

63.1 SQLite Characteristics

SQLite has the following characteristics:

- It is embedded with the native sdk.
- It is ACID-compliant (atomicity, consistency, isolation, durability).
- It is weakly-typed i.e, any object can be stored in any column, regardless of how that column was declared. For example you can insert a string into a database column of type integer
- It does not officially support foreign key constraints, although triggers can be used as a workaround.
- It does not support RIGHT OUTER JOINs.

63.2 SQLite Column Types

Most databases use strong, static column typing which means that the elements of a column can only hold values compatible with the defined type of a column. SQLite utilizes a dynamic typing technique known as manifest typing. For each row value, manifest typing records the value's type along with the value data. This allows nearly any element of any row to hold almost any type of value. In the strictest sense, SQLite supports only five concrete datatypes. These are known as storage classes, and represent the different ways SQLite stores data on disk.

Every value has one of these five native storage classes:

- **NULL**: NULL is considered its own distinct type. A NULL type does not hold a value. Literal NULLs are represented by the keyword NULL.
- **Integer**: A signed integer number. Integer values are variable length, being 1, 2, 3, 4, 6, or 8 bytes in length, depending on the minimum size required to hold the specific value. Integer have a range of 9,223,372,036,854,775,808 to +9,223,372, 036,854,775,807, or roughly 19 digits. Literal integers are represented by any bare series of numeric digits (without commas) that does not include a decimal point or exponent.
- **REAL**: The value is a floating-point number, stored as an 8-byte value in the processor's native format. For most modern processors, this is an IEEE 754 double-precision number. Literal floating-point numbers are represented by any bare series of numeric digits that include a decimal point or exponent.
- **Text**: A variable-length string, stored using the database encoding (UTF-8, UTF-16BE, or UTF-16LE). Literal text values are represented using character strings in single quotes.
- **BLOB**: A length of raw bytes, copied exactly as provided. Literal BLOBs are represented as hexadecimal text strings preceded by an x. For example, the notation x'1234ABCD' represents a 4-byte BLOB. BLOB stands for Binary Large OBject. SQLite text and BLOB values are always variable length. The maximum size of a text or BLOB value is limited by a compile-time directive. The default limit is exactly one billion bytes, or slightly less than a full gigabyte. The maximum value for this directive is two gigabytes.

63.3 Type Affinity

The elements of most columns can hold any value type, the "type" of a column could be misleading. Rather than being an absolute type,like most databases, an SQLite column type (as defined in CREATE TABLE) becomes more of a suggestion than a hard and fast rule. This is known as type affinity, and essentially represents a desired category of type. Each type affinity has specific rules about what types of values it can store, and how different values will be converted when stored in that column. Generally, type affinity will cause conversion or migration of types only if it can be done without losing data or precision.

Each table column must have one of five type affinities:

- **Text**: A column with a text affinity will only store values of type NULL, text, or BLOB. If you attempt to store a value with a numeric type (float or integer) it will be converted into a text representation before being stored as a text value type.
- **Numeric**: A column with a numeric affinity will store any of the five types. Values with integer and float types, along with NULL and BLOB types, are stored without conversion. Any time a value with a text type is stored, an attempt is made to convert the value to a numeric type (integer or float). Assuming the conversion works, the value is stored in an appropriate numeric type. If the conversion fails, the text value is stored without any type of conversion.
- **Integer**: A column with an integer affinity works essentially the same as a numeric affinity. The only difference is that any value with a float type that lacks a fractional component will be converted into an integer type.
- **Real**: A column with REAL affinity behaves like a column with NUMERIC affinity except that it forces integer values into floating point representation.
- **None**: A column with a none affinity has no preference over storage class. Each value is stored as the type provided, with no attempt to convert anything.

63.4 Determination Of Column Affinity

Since type affinities are not part of the SQL standard, SQLite has a series of rules that attempt to map traditional column types to the most logical type affinity. The type affinity of a column is determined by the declared type of the column, according to the following rules (substring matches are case-insensitive):

1. If no column type was given, then the column is given the none affinity.
2. If the column type contains the substring "INT," then the column is given the integer affinity.
3. If the column type contains any of the substrings "CHAR," "CLOB," or "TEXT," then the column is given the text affinity.
4. If the column type contains the substring "BLOB," then the column is given the none affinity.
5. If the column type contains any of the substrings "REAL," "FLOA," or "DOUB," then it is given the float real affinity.
6. If no match is found, the column is assigned the numeric affinity.

As implied by the first rule, the column type is completely Optional. SQLite will allow you to create a table by simply naming the columns, such as CREATE TABLE t (i, j,k);. You'll also notice that there isn't any specific list of column types that are recognized. You can use any column type you want, even making up your own names. This might sound a bit fast and loose for a typing system, but it works out quite well. By keying off specific substrings, rather than trying to define specific types, SQLite is able to handle SQL statements (and their database-specific types) from just about any database, all while doing a pretty good job of mapping the types to an appropriate affinity. Note that a declared type of "FLOATING POINT" would give INTEGER affinity, not REAL affinity, due to the "INT" at the end of "POINT". And the declared type of "STRING" has an affinity of NUMERIC, not TEXT.

63.5 Affinity Name Example

The following table shows how many common datatype names from more traditional SQL implementations are converted into affinities by the six rules of the previous section.

JavaScript Example Typenames From The CREATE TABLE Statement or CAST Expression	Resulting Affinity	Rule Used To Determine Affinity
INT, INTEGER, TINYINT, SMALLINT, MEDIUMINT, BIGINT, UNSIGNED BIG INT, INT2, INT8	INTEGER	2
CHARACTER(20), VARCHAR(255), VARYING CHARACTER (255), NCHAR(55), NATIVE CHARACTER(70), NVARCHAR (100), TEXT, CLOB	TEXT	3
BLOB *no datatype specified*	NONE	1,4
REAL, DOUBLE, DOUBLE PRECISION, FLOAT	REAL	5
NUMERIC, DECIMAL(10,5) , BOOLEAN, DATE, DATETIME	NUMERIC	6

63.6 Column Affinity Behavior Example

```
CREATE TABLE t1(
t TEXT, -- text affinity by rule 3
nu NUMERIC, -- numeric affinity by rule 6
i INTEGER, -- integer affinity by rule 2
r REAL, -- real affinity by rule 5
no BLOB -- no affinity by rule 1,4
);
-- Values stored as TEXT, INTEGER, INTEGER, REAL, TEXT.
INSERT INTO t1 VALUES('500.0', '500.0', '500.0', '500.0', '500.0');
SELECT typeof(t), typeof(nu), typeof(i), typeof(r), typeof(no) FROM
t1;
4
text|integer|integer|real|text
-- Values stored as TEXT, INTEGER, INTEGER, REAL, REAL.
DELETE FROM t1;
INSERT INTO t1 VALUES(500.0, 500.0, 500.0, 500.0, 500.0);
SELECT typeof(t), typeof(nu), typeof(i), typeof(r), typeof(no) FROM
t1;
text|integer|integer|real|real
-- Values stored as TEXT, INTEGER, INTEGER, REAL, INTEGER.
DELETE FROM t1;
INSERT INTO t1 VALUES(500, 500, 500, 500, 500);
SELECT typeof(t), typeof(nu), typeof(i), typeof(r), typeof(no) FROM
t1;
text|integer|integer|real|integer
-- BLOBs are always stored as BLOBs regardless of column affinity.
DELETE FROM t1;
INSERT INTO t1 VALUES(x'0500', x'0500', x'0500', x'0500', x'0500');
```

```
SELECT typeof(t), typeof(nu), typeof(i), typeof(r), typeof(no) FROM
t1;
blob|blob|blob|blob|blob
-- NULLs are also unaffected by affinity
DELETE FROM t1;
INSERT INTO t1 VALUES(NULL,NULL,NULL,NULL,NULL);
SELECT typeof(t), typeof(nu), typeof(i), typeof(r), typeof(no) FROM
t1;
null|null|null|null|null
```

Note: Columns of type INTEGER PRIMARY KEY may only hold a 64-bit signed integer. An error will result if you try to put anything other than an integer into an INTEGER PRIMARY KEY column.

64. Background Jobs

Background Jobs feature in Kony Platform includes Background Fetch and Background Transfers.

64.1 Background Fetch

You can allow the application to fetch data from a network on regular basis while it is in a background state, with the background fetch capability in iOS 7 and later. This data can be utilized to keep the content of the application up-to-date and present it to the user when the user launches or re-opens the application. Also, the capability to support long running HTTP/HTTPS tasks like downloading large files if the application is in background state is supported.

- In iOS version 7 and above, an app that retrieves content regularly from a network can ask the system for background execution time to check for new content.
- An application developer who wishes to perform a background fetch can enable this feature in Kony Visualizer IDE and register for the feature using `kony.backgroundjob.registerBackgroundFetch`. This API can be used to register predefined fetch job that can handle tasks like file download, upload, or a custom task.
- `kony.net.HttpRequest` API supports background transfers. Tasks like upload or download from the network (HTTP/HTTPS) that are active even when the app is in background or suspended state is available in iOS.

64.2 Background Transfers

The purpose of the background transfer service is to allow long running network downloads or uploads to take place even when the associated application is placed in the background. Background transfers are supported in HTTP and HTTPS protocols.

Following are the features in Background Transfer:

- Uploads and downloads are managed by iOS
- Unlike background fetch, background transfers have unlimited time.

- It can be put in the queue anytime (foreground and background).
- App is woken up to handle authentication challenges, errors if any, or completion of the job.
- Background transfer tasks can be paused and resumed.
- Only HTTP and HTTPS protocols are supported.

64.3 Background Fetch Workflow

iOS 7 and later contains the `Background Fetch` API which allows an app to get updated content when the app is in background. iOS schedules the background fetch events based on the app usage so that the content is always up-to-date when user opens the application. iOS provides 30 seconds time frame for the app to wake up, fetch new data, update its interface, and go back to sleep.

If the Background Fetch feature is enabled in an app, the system wakes the app in the background from time to time and lets it go online to fetch new data with a goal to refresh its content. In this way the app is always up-to-date and users do not have to wait any more when they launch it.

The system determines how frequent the application can perform a background fetch and depends on the following factors:

- If network connectivity is available at that particular time.
- Is the device awake.
- Data and time the application has taken in its previous attempt to perform a background fetch.

The iOS system gives power to an application to schedule Background Fetch. Background Fetch can also be used to perform tasks internally in the app. Only synchronous tasks must be performed in background fetch, other than long running Background Transfers that are asynchronous. After the task is done `setBackgroundFetchCompletionStatus` must be called, passing a result that indicates whether content is available.

Invoking `setBackgroundFetchCompletionStatus` tells the system that it can move the app back to the suspended state and evaluate its power usage.

Apps that download small amounts of content quickly and accurately reflect when they had content to download are more likely to receive execution time in the future than apps that take longer to download their content.

It is advisable to use Background Transfers supported by `kony.net.HttpRequest` in fetch task to optimally utilize the execution time provided in the Fetch task.

64.4 Background Fetch and Background Transfers APIs

Background Fetch APIs	Background Transfer APIs
Following are the APIs for Background Jobs: 1. kony.backgroundjob.registerBackgroundFetch 2. kony.backgroundjob.setBackgroundFetchCompletionStatus 3. kony.backgroundjob.setBackgroundFetchInterval	Following are the Background Transfer APIs : 1. kony.net.HttpRequest 2. suspend 3. resume 4. getTaskState

64.5 Supported Versions and Platforms

The Background Job feature:

- Available in Kony iOS platform versions above 5.6.2.
- Supports iOS7 and the platforms above.
- Supported only in JavaScript.

64.6 kony.backgroundjob.registerBackgroundFetch

registerBackgroundFetch can be used to register a background fetch job. A JavaScript function that can be invoked whenever the system schedules the fetch task. The app is woken up after the specified amount of fetchInterval.

> ***Note:*** At the end of the background fetch job, you must invoke the `kony.backgroundjob.setBackgroundFetchCompletionStatus` passing a result that indicates whether content is available or not.

64.6.1 Signature

JavaScript: kony.backgroundjob.registerBackgroundFetch (backgroundCallBack, fetchInterval)

64.6.2 Input Parameters

backgroundCallBack [Function] - Mandatory

This callback is to be executed when the system invokes the app for a fetch job when the app is in background.

fetchInterval [Number] / [Constant] - Optional

Indicates the minimum time interval in seconds which must elapse before system can invoke the app for next fetch job .This is an optional parameter and value that is specified is only indicative and might not have any real effects on the scheduling frequency, as system decides the actual interval.

Possible values are:

1. constants.BACKGROUND_TASK_FETCH_INTERVAL_MINIMUM

   * This is the default value. System decides the fetch interval depending on the usage prediction for the app.

2. constants.BACKGROUND_TASK_FETCH_INTERVAL_NEVER

   Disable background task. Set the time interval to constants.BACKGROUND_TASK_FETCH_INTERVAL_NEVER so that the job is no longer scheduled for running.

3. Time in seconds. (Negative values are not accepted).

64.6.3 Return Values

<N.A> or “NONE".

64.6.4 Exceptions and Error Handling

1. If no arguments or less number of arguments than the mandatory number of arguments are provided, it raises an exception and throws a “KonyError” JS Object with the following attributes:

```
errorCode: 4001.
name: "BackgroundJobError".
message: "registerBackgroundFetch: MissingMandatoryParameters."
```

2. If arguments type does not match the allowed type, it raises an exception and throws a *KonyError* JavaScript Object with the following attributes:

```
errorCode: 4002.
name: "BackgroundJobError".
message: "registerBackgroundFetch: InvalidParameters."
```

64.6.5 Platform Availability

Available in iOS Platform.

64.6.6 JavaScript Example

```
function backgroundFetchCallBack() {


//fetch handler invoked whenever the system invokes the App for a
fetch job
// do something here


// in the end of the handler setBackgroundFetchCompletionStatus() need
// to be called mandatorily other wise system might termitane the app.

// task completion status.
//1. constants.BACKGROUND_TASK_STATUS_NEW_DATA
//2. constants.BACKGROUND_TASK_STATUS_FAILED
//3. constants.BACKGROUND_TASK_STATUS_NO_NEW_DATA

var completionStatus = constants.BACKGROUND_TASK_STATUS_NEW_DATA;
        kony.backgroundjob.setBackgroundFetchCompletionStatus
(completionStatus);

};



       //Fetch interval
       //1. constants.BACKGROUND_TASK_FETCH_INTERVAL_MINIMUM
    //2. constants.BACKGROUND_TASK_FETCH_INTERVAL_NEVER
    //3. Time in seconds. (Negative values are not accepted).

    var fetchInterval = constants.BACKGROUND_TASK_FETCH_INTERVAL_
MINIMUM;

       //register background fetch job
    kony.backgroundjob.registerBackgroundFetch
(backgroundFecthCallBack, fetchInterval);
```

64.7 kony.backgroundjob.setBackgroundFetchCompletionStatus

One must call this method in the end of `backgroundCallBack` passing a result that indicates whether content was available or not. This method intimates the system of the completion status of the background fetch job that has been scheduled.

Executing this call tells the system that it can move the app back to the suspended state and evaluate its power usage. Apps that download small amounts of content quickly and accurately display when they have content to download are more likely to receive execution time in the future than apps that take longer to download their content

Failure to call this method results in undefined behavior and could cause the app to be terminated by the system. If an error arises during invoking this API, default value: `constants.BACKGROUND_TASK_STATUS_FAILED` will be set.

64.7.1 Signature

JavaScript: kony.backgroundjob.setBackgroundFetchCompletionStatus(completionStatus)

64.7.2 Input Parameters

completionStatus [constant] - Mandatory

This is a mandatory parameter. Possible values are as follows

1. **constants.BACKGROUND_TASK_STATUS_NEW_DATA**

   This tells the system that the fetch was successful and internally, the system then updates the apps UI (if the fetch resulted in UI change) in the background. This new UI is presented to the user once the app is brought to fore ground. The snapshot of the new UI is also presented when the user tries to switch apps using the App switcher.

2. **constants.BACKGROUND_TASK_STATUS_FAILED**

   This tells the system that the fetch was unsuccessful. UI is not updated and the task will be run later based on the available system resources.

3. **constants.BACKGROUND_TASK_STATUS_NO_NEW_DATA**

   This intimates the system the fetch did not result in any new data, UI is not updated and the job is run after any point later in time but not before fetchInterval.

64.7.3 Return Values

<N.A> or “NONE".

64.7.4 Exceptions

1. If no arguments or less number of arguments than the mandatory number of arguments are provided, it raises an exception and throws a “KonyError” JS Object with the following attributes:

   ```
   errorCode: 4001.
   name: "BackgroundJobError".
   message: "setBackgroundFetchCompletionStatus:
   MissingMandatoryParameters."
   ```

2. If arguments type does not match the allowed type, it raises an exception and throws a “KonyError” JS Object with the following attributes:

```
errorCode: 4002.
name: "BackgroundJobError".
message: "setBackgroundFetchCompletionStatus: InvalidParameters."
```

3. If an error arises during invoking of this API, default value: constants.BACKGROUND_TASK_STATUS_FAILED will be set.

64.7.5 Platform Availability

Available in iOS Platform.

64.7.6 JavaScript Example

```
function backgroundFecthCallBack() {


//fetch handler invoked whenever the system invokes the App for a
fetch job
// do something here


// in the end of the handler setBackgroundFetchCompletionStatus() need
// to be called mandatorily other wise system might termitane the app.

// task completion status.
//1. constants.BACKGROUND_TASK_STATUS_NEW_DATA
//2. constants.BACKGROUND_TASK_STATUS_FAILED
//3. constants.BACKGROUND_TASK_STATUS_NO_NEW_DATA

var completionStatus = constants.BACKGROUND_TASK_STATUS_NEW_DATA;
        kony.backgroundjob.setBackgroundFetchCompletionStatus
(completionStatus);
};



       //Fetch interval
       //1. constants.BACKGROUND_TASK_FETCH_INTERVAL_MINIMUM
    //2. constants.BACKGROUND_TASK_FETCH_INTERVAL_NEVER
    //3. Time in seconds. (Negative values are not accepted).

    var fetchInterval = constants.BACKGROUND_TASK_FETCH_INTERVAL_
MINIMUM;

       //register background fetch job
    kony.backgroundjob.registerBackgroundFetch
(backgroundFecthCallBack, fetchInterval);
```

64.8 kony.backgroundjob.setBackgroundFetchInterval

This method sets the fetch interval for the background fetch job scheduled.

64.8.1 Signature

JavaScript: kony.backgroundjob.setBackgroundFetchInterval(fetchInterval)

64.8.2 Input Parameters

fetchInterval [Number]/[Constant] - Mandatory

Indicative minimum time interval in seconds that should elapse before system can invoke the app for next fetch job. This is only indicative and may not have any real effects on the scheduling frequency, as system decides the actual interval.

Possible values are:

1. **constants.BACKGROUND_TASK_FETCH_INTERVAL_MINIMUM**

   This is the default value. System decides the fetch interval depending on the usage prediction for the app.

2. **constants.BACKGROUND_TASK_FETCH_INTERVAL_NEVER**

   Disable background task. Set the time interval to constants.BACKGROUND_TASK_FETCH_INTERVAL_NEVER so that the job is no longer scheduled for running.

3. Time in seconds. (Negative values are not accepted).

64.8.3 Return Values

<N.A> or "NONE".

64.8.4 Exceptions

1. If no arguments or less number of arguments than the mandatory number of arguments are provided, it raises an exception and throws a "KonyError" JS Object with the following attributes:

   ```
   errorCode: 4001.
   name: "BackgroundJobError".
   message: "setBackgroundFetchInterval:
   MissingMandatoryParameters."
   ```

2. If arguments type does not match the allowed type, it raises an exception and throws a "KonyError" JS Object with the following attributes:

   ```
   errorCode: 4002.
   name: "BackgroundJobError".
   message: "setBackgroundFetchInterval: InvalidParameters."
   ```

64.8.5 Platform Availability

Available in iOS Platform.

64.8.6 JavaScript Example

```
//Fetch interval
       //1. constants.BACKGROUND_TASK_FETCH_INTERVAL_MINIMUM
    //2. constants.BACKGROUND_TASK_FETCH_INTERVAL_NEVER
    //3. Time in seconds. (Negative values are not accepted).

    var fetchInterval = constants.BACKGROUND_TASK_FETCH_INTERVAL_
MINIMUM;

       kony.backgroundjob.setBackgroundFetchInterval(fetchInterval);
```

64.9 Long Running Network Tasks

`kony.net.HttpRequest` now supports long running network tasks which run to completion even when the app is pushed to Background or Suspended state, these are also known as background transfers and are available from iOS7.

Background transfers can be scheduled while the app is in foreground or while the app is in background (with the help of background fetch jobs in background state).

Background transfers can be enabled by setting the property ‘backgroundTransfer’ to ‘true’ on the HTTP request object returned by kony.net.HttpRequest.

“<HttpRequestObject>.backgroundTransfer’ has to be set to ‘true’ before open() API is invoked on the HTTP request after its creation.

It is always advisable to set “<HttpRequestObject>.backgroundTransfer” to ‘true’ immediately after creation of HttpRequest object.

64.9.0.1 Signature

```
var httpclient = new kony.net.HttpRequest(
  {
      "timeoutIntervalForRequest": 60,
      "timeoutIntervalForResource": 600
  });
httpclient.backgroundTransfer = true;
```

64.9.0.2 Input Parameters

Http request object returned by kony.net.HttpRequest supports the following additional properties to enable running network tasks to run even when the App is in background state.

backgroundTransfer [Boolean]

Lets the HTTP request to know that Background transfers need to be supported. The default value of this property is false.

> ***Note:*** This property is only available for the iOS platform.

timeoutIntervalForRequest [Number]

Define a time-interval for request how long a task (upload or download) should wait for new data to arrive. If the new data arrives within the specified interval, the timer associated with this property resets. If the new data does not arrive within the specified interval, it triggers timeout.

> ***Note:*** This parameter is available for the iOS and Android platforms.

The default value of this property is 60 (in seconds).

> ***Note:*** Any upload or download tasks created by a background session are automatically retried if the original request fails due to a timeout. To configure how long an upload or download task should be allowed to be retried or transferred, use the timeoutIntervalForResource property.

timeoutIntervalForResource [Number]

Define a time-interval for resource how long to wait for entire resource to arrive. The resource timer starts when the request is initiated and counts until either the request completes or this timeout interval is reached, whichever comes first.

The default value of this property is 1 week (7 days).

Note: This parameter is available for the iOS and Android platforms.

64.9.0.3 JavaScript Example

```
var logPrefix = "bg_transfer_test_case_1 :- ";
    send_request();

    function send_request() {
        var request = new kony.net.HttpRequest();
        request.onReadyStateChange = callbackHandler;

        //BGtransfer
        request.backgroundTransfer = true;

        //session
        var req_session = request.getSession();
        kony.print(logPrefix + " request.getSession() : " + req_
session);
        //url
        var url =
"https://developer.apple.com/library/ios/documentation/iphone/concept
ual/iphoneosprogrammingguide/iphoneappprogrammingguide.pdf";
        kony.print(logPrefix + "URL : " + url);
        request.open(constants.HTTP_METHOD_GET, url);
        request.send();
    };
    function callbackHandler(request) {
        kony.print(logPrefix + " :
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++");
        kony.print(logPrefix + " : Scope :- request.status : " +
request.status);
        kony.print(logPrefix + " : Scope :- request.statusText : " +
request.statusText);
        kony.print(logPrefix + " : Scope :- request.responseType : " +
request.responseType);
        kony.print(logPrefix + " : Scope :- request.response : " +
request.response);
        kony.print(logPrefix + " : Scope :-
request.getAllResponseHeaders() : ");
        kony.print(request.getAllResponseHeaders());
        kony.print(logPrefix + " :
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++");
    };
```

64.9.0.4 Platform Availability

Available for the iOS and Android platforms.

64.9.1 suspend

Temporarily suspends a background transfer task, while suspended, it produces no network traffic and is not subject to timeouts.

A suspended background transfer task if resumable can be resumed using resume() API. If the task cannot be resumed the transfer task will start afresh.

This API is applicable only if `<HttpRequestObject>.backgroundTransfer = true` for a HTTPrequest.

64.9.1.1 Signature

<<HttpRequestObject>>.suspend

64.9.1.2 Input Parameters

None

64.9.1.3 Return Values

None

64.9.1.4 JavaScript Example

```
var request = new kony.net.HttpRequest();

        //BGtransfer
        request.backgroundTransfer = true;

               //url
        var url =
 "https://developer.apple.com/library/ios/documentation/iphone/concept
ual/iphoneosprogrammingguide/iphoneappprogrammingguide.pdf";
        kony.print(logPrefix + "URL : " + url);

        request.open(constants.HTTP_METHOD_GET, url);
        request.send();

               //suspend task
       request.suspend();
```

64.9.1.5 Platform Availability

Available in iOS Platform.

64.9.2 resume

- A resumable suspended task can be resumed using this API. If the task cannot be resumed the transfer task will start afresh.
- This API is applicable only if `<HttpRequestObject>.backgroundTransfer = true` for a HTTPrequest.

64.9.2.1 Signature

<<HttpRequestObject>>.resume()

64.9.2.2 Input Parameters

None

64.9.2.3 Return Values

None

64.9.2.4 JavaScript Example

```
var request = new kony.net.HttpRequest();

        //BGtransfer
        request.backgroundTransfer = true;

               //url
        var url =
"https://developer.apple.com/library/ios/documentation/iphone/concept
ual/iphoneosprogrammingguide/iphoneappprogrammingguide.pdf";
        kony.print(logPrefix + "URL : " + url);

        request.open(constants.HTTP_METHOD_GET, url);
        request.send();
               //suspend task
       request.suspend();

       //resume task
       request.resume();
```

64.9.2.5 Platform Availability

Available on iOS Platform.

64.9.3 getTaskState

- Returns the current state of the transfer task: active, suspended, in the process of being canceled, or completed
- This API is applicable only if `<HttpRequestObject>.backgroundTransfer = true` for a HTTPrequest.

64.9.3.1 Signature

var taskState = <<HttpRequestObject>>. getTaskState()

64.9.3.2 Input Parameters

None

64.9.3.3 Return Values

taskState: <constant>

- constants.SESSION_TASK_STATE_RUNNING
- constants.SESSION_TASK_STATE_SUSPENDED
- constants.SESSION_TASK_STATE_CANCELING
- constants.SESSION_TASK_STATE_COMPLETED

64.9.3.4 JavaScript Example

```
var request = new kony.net.HttpRequest();

        //BGtransfer
        request.backgroundTransfer = true;

               //url
        var url =
"https://developer.apple.com/library/ios/documentation/iphone/concept
ual/iphoneosprogrammingguide/iphoneappprogrammingguide.pdf";
        kony.print(logPrefix + "URL : " + url);

        request.open(constants.HTTP_METHOD_GET, url);
        request.send();



       //getTaskStat()
            var tastState = request_object.getTaskState();
            if (tastState == constants.SESSION_TASK_STATE_RUNNING) {
                kony.print("tastState == constants.SESSION_TASK_STATE_
RUNNING");
            } else if (tastState == constants.SESSION_TASK_STATE_
SUSPENDED) {
                kony.print("tastState == constants.SESSION_TASK_STATE_
SUSPENDED");
            } else if (tastState == constants.SESSION_TASK_STATE_
CANCELING) {
                kony.print("tastState == constants.SESSION_TASK_STATE_
CANCELING");
            } else if (tastState == constants.SESSION_TASK_STATE_
COMPLETED) {
                kony.print("tastState == constants.SESSION_TASK_STATE_
COMPLETED");
                }
```

64.9.3.5 Platform Availability

Available in iOS Platform.

64.10 Session Configuration API

64.10.1 getSession

- By default, if a new background transfer task is created using `kony.net.HttpRequest` ['<HttpRequestObject>.backgroundTransfer = true'] a new session is created and this new task is associated with it.
- <HttpRequestObject> now exposes a new method 'getSession()' using which the Session identifier associated with the HttpRequest can be obtained.
- Multiple background tasks can be associated with the same Session by passing the Session identifier returned by `<HttpRequestObject>.getSession()` to any new 'kony.net.HttpRequest' created there after.
- getSession() API is applicable only if `<HttpRequestObject>.backgroundTransfer = true` for a HTTPrequest.
- When ever a new `kony.net.HttpRequest` is created by passing a session identifier then by default backgroundTransfer property will be enabled ['<HttpRequestObject>.backgroundTransfer = true'].

> ***Note:*** A session object life is dependent on the life of any of the request objects or request object it is associated with. Which means to be certain that a session is alive, any of the request objects or request object with which it is associated should be alive.

64.10.1.1 Signature

var session_identifier = <<HttpRequestObject>>.getSession()
//To associate new HTTP Requests created with the same session
<New HttpRequestObject2> = new kony.net.HttpRequest(<session_identifier >);

64.10.1.2 Input Parameters

None

64.10.1.3 Return Values

Session identifier for the Session associated with the HttpRequest instance.

64.10.1.4 JavaScript Example

```
var request = new kony.net.HttpRequest();

        //BGtransfer
        request.backgroundTransfer = true;

               //url
        var url =
"https://developer.apple.com/library/ios/documentation/iphone/concept
ual/iphoneosprogrammingguide/iphoneappprogrammingguide.pdf";
        kony.print(logPrefix + "URL : " + url);

        request.open(constants.HTTP_METHOD_GET, url);
        request.send();

               //request session identifier
            var request_session = request.getSession();


       //To associate new HTTP Requests created with the same session
```

```
//request 2
        var request_2 = new kony.net.HttpRequest(request_session);

//request 3
        var request_3 = new kony.net.HttpRequest(request_session);
```