



Kony Application Design and Development Guidelines

Release 8.2

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.

Copyright © 2017 Kony, Inc.

All rights reserved.

August, 2018

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 Modifications/Release
09/19/2017	1.0	Document updated for V8 release.

Table of Contents

1. Preface	9
1.1 Purpose	9
1.2 Intended Audience	9
1.3 Formatting conventions used in this guide	9
1.4 Contact Us	10
2. Application Design Guidelines	11
2.1 User Interface (UI) Guidelines	12
2.2 Guidelines for App Menu	26
2.3 Guidelines for Fonts	30
2.4 Guidelines for Fonts	31
2.5 Guidelines for Title Bar or Navigation Controls	32
2.6 Guidelines for Progress Indicators (Rich Clients)	35
2.7 Guidelines for Pop-ups or Alerts	39
2.8 Guidelines for Alignments (Paddings)	40
2.9 Guidelines while using Images	42
2.10 Guidelines for using Segment Widget	43
2.11 Guidelines while using Map Widget	45
2.12 Guidelines while using Browser Widget	47
2.13 Guidelines for Calendar Widget	48
2.14 Landscape vs. Portrait View	51

2.15 Mobile Web Apps Only	52
2.16 Responsive Web Apps	52
2.17 General User Interface (UI) Guidelines	56
2.18 Naming Convention for Forms and Widgets	65
2.19 Components	67
2.20 Px vs Percentage vs Dp	68
2.21 Coding Guidelines	70
2.22 Performance Management Guidelines	77
2.23 Memory Management Guidelines	80
2.24 Device-side Security Guidelines	82
2.25 Pre/Post Processors and URL Provider Guidelines	82
3. Application Development Guidelines	91
3.1 Coding Standards	91
3.2 Usage of Global and Local Variables	100
3.3 File Names	102
3.4 Basics of JavaScript	110
4. Accessibility (508 Compliance)	117
4.1 Define accessibilityConfig	121
4.2 Widget Behavior	124
4.3 Container Widgets	125
4.4 Basic Widgets	135

4.5 Advanced Widgets	151
4.6 Accessibility Best Practices	160
4.7 Accessibility: Platform Specific Limitations	160
5. Flex Layout Guidelines	162
5.1 Animation and Flex Layout Limitations	162
5.2 iOS Limitations	165
5.3 Android Limitations	168
5.4 Windows Limitations	171
5.5 SPA Limitations	175
5.6 Flex Container Backward Compatibility	180
5.7 FlexContainer inside a Box Container	180
5.8 Box Container inside a FlexContainer	184
5.9 Flex Container Pseudocode Examples	184
5.10 Widget Level Animation Using Flex Forms	198
5.11 Key Frame Animations	198
5.12 References	200
5.13 Animation Properties Events and Methods	200
5.14 Animation Configurations	223
5.15 Applying Animations	227
5.16 Sequential and Parallel Animations	227
5.17 Querying Widget Properties	228

5.18	Layout Callbacks during Animation	229
5.19	Flex Container and Child Widgets	229
5.20	Multiple Parallel Animations	230
5.21	Interactions on the Widget during Animation	230
6.	Platform Specific Limitations	231
6.1	Desktop Web Limitations	231
6.2	SPA Limitations	233
6.3	Windows Kiosk	234
6.4	BlackBerry 10	235
7.	App Submission Guidelines	239
7.1	iOS 11	239
7.2	iOS 10	239
8.	DBX App Design Guidelines	241
8.1	Color Theme	241
8.2	Fonts	247
8.3	Text Field/Button Styles	251
8.4	Components and Iconography	255
8.5	Page Size and Grid - Online Banking	261
8.6	Header and Footer - Online Banking	265
8.7	Sign In - Online Banking	266
8.8	Account List	270

8.9 Account Details	280
8.10 Add View	292
8.11 PopUp Type - Online Banking	301
8.12 Confirmation - Online Banking	305
8.13 Acknowledgment - Online Banking	308
8.14 Toast Messages/ Notifications - Retail Banking	312
8.15 Animation/Transition - Retail Banking	316
8.16 Visual Indication and Button behavior - Retail Banking	320
8.17 Menu - Retail Banking	325

1. Preface

Designing a mobile application is more than putting various elements together to achieve the required functionality. Aspects like functionality, design, and performance are considered while creating an application. This document covers user interface, coding, pre/post processors, URL provider guidelines, performance management, memory management, device-side security guidelines, Coding standards, Basics of JavaScript, Flex Layout Guidelines and Platform specific limitations.

1.1 Purpose

This document describes what goes into application design and also emphasizes on the importance of aesthetics, design and development visualization. This document does not talk about how to implement individual controls to achieve a specific purpose.

1.2 Intended Audience

This document is intended for developers involved in designing and developing a mobile application. This document helps developers build applications that match the native look of the underlying platform with an enhanced performance.

1.3 Formatting conventions used in this guide

Following are the formatting conventions used throughout the document:

Conventions	Explanation
Monospace	<ul style="list-style-type: none">• User input text, system prompts, and responses• File path• Commands• Program code• File names

Conventions	Explanation
<i>Italic</i>	<ul style="list-style-type: none">• Emphasis• Names of books and documents• New terminology
Bold	<ul style="list-style-type: none">• Windows• Menus• Buttons• Icons• Fields• Tabs• Folders
<u>URL</u>	Active link to a URL.
<i>Note</i>	Provides helpful hints or additional information.
<i>Important</i>	Highlights actions or information that might cause problems to systems or data.

1.4 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

2. Application Design Guidelines

Designing a mobile application is more than putting various elements together to achieve the required functionality. Aspects like functionality, design, and performance are considered while creating an application.

In this section, you will learn:

1. [UI Guidelines](#)
2. [Coding Guidelines](#)
3. [Performance Management Guidelines](#)
4. [Memory Management Guidelines](#)
5. [Device-side Security Guidelines](#)
6. [Pre/Post Processors and URL Provider Guidelines](#)

2.1 User Interface (UI) Guidelines

User Interface and User Experience are critical elements of mobile application development. A good user interface design must make the user choices easy and intuitive without compromising on the performance. All major platforms provide specific guidelines to make the application look and feel appealing to the user.

While developing applications using a cross-platform tool like Kony Visualizer, one must strike a balance between design and performance. Also, the application must provide native flavor on every platform. The customer also may provide UI guidelines / Style guidelines to be followed on each platform.

The guidelines outlined in this document are targeted to achieve an optimum user interface across platforms and consistency within the application. We have listed the most commonly used UI components and defined for each component the guidelines to be followed for an application to achieve the native look and feel on respective platforms.

2.1.1 List of UI Components

At a high-level, the following UI components are present in every application:

- Application Icon
- Splash Screen
- Menu
- Fonts
- Title bar / Navigation Controls
- Progress indicator / Blocking UI / Interstitial screens
- Pop-ups
- Alignments (Margins & Paddings)

- Resources (Images)
- Container widgets, Basic widgets, and Advanced widgets. Few of the advanced widgets are:
 - Segment Widget
 - Map (Map Widget)
 - Browser (Browser Widget)

2.1.2 Guidelines for Application Icons

Following are the guidelines for creating an application icon:

- Home (Application) icon at the phone operating system should not be blurred.
- Home (Application) icon should not be a default phone OS icon. Should be an application specific icon provided by client.
- Home (Application) icon should be of the same size when compared to default phone applications icons.

2.1.3 Guidelines for Configuring Application Icon for iOS Applications

For more information, see

<https://developer.apple.com/ios/human-interface-guidelines/graphics/app-icon/>

Icon	Idiom	Size	Image Name
App icon (required)	iPhone / iPod	57 x 57 pixels	Icon.png
		114 x 114 pixels (@2x)	Icon@2x.png

Icon	Idiom	Size	Image Name
App icon (required)	iPad	72 x 72 pixels	Icon-72.png
		144 x 144 pixels (@2x)	Icon-72@2x.png
App icon for the App Store (required)	iPhone / iPod / iPad	512 x 512	iTunesArtwork
		1024 x 1024 (@2x)	iTunesArtwork@2x

Important: iTunesArtwork icon images should be in PNG format. Name the images without the .png extension after importing them into the Xcode project. For more information, see https://developer.apple.com/library/content/qa/qa1686/_index.html

2.1.4 Guidelines for Configuring Application Icon for Android Applications

For more information, see

http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher_archive.html

Icon	ldpi (120 dpi)	mdpi (160 dpi)	hdpi (240 dpi)	xhdpi (320 dpi)	xxhdpi (480 dpi)	xxxhdpi (640 dpi)
App Icon Size	36 x 36 pixels	48 x 48 pixels	72 x 72 pixels	96 x 96 pixels	144 x 144 pixels	192 x 192 pixels

2.1.5 Guidelines for Configuring Application Icon for Windows Applications

Following are the application icon resolutions to be used in Windows apps:

- Application Icon should be 62 x 62 pixels.
- Startup pin images should be 173 x 173 pixels.

2.1.6 Configure an Application Icon from Kony Visualizer

Project Settings
Edit project settings

Application | MobileFabric Details | App Settings | **Native** | Mobile Web | Desktop Web | Metrics | APM

Common | iPhone/iPad/Watch | Android | Windows Phone 8 & 8.1 | Windows Tablet

Displays on Application Menu

Name: TRAIL1

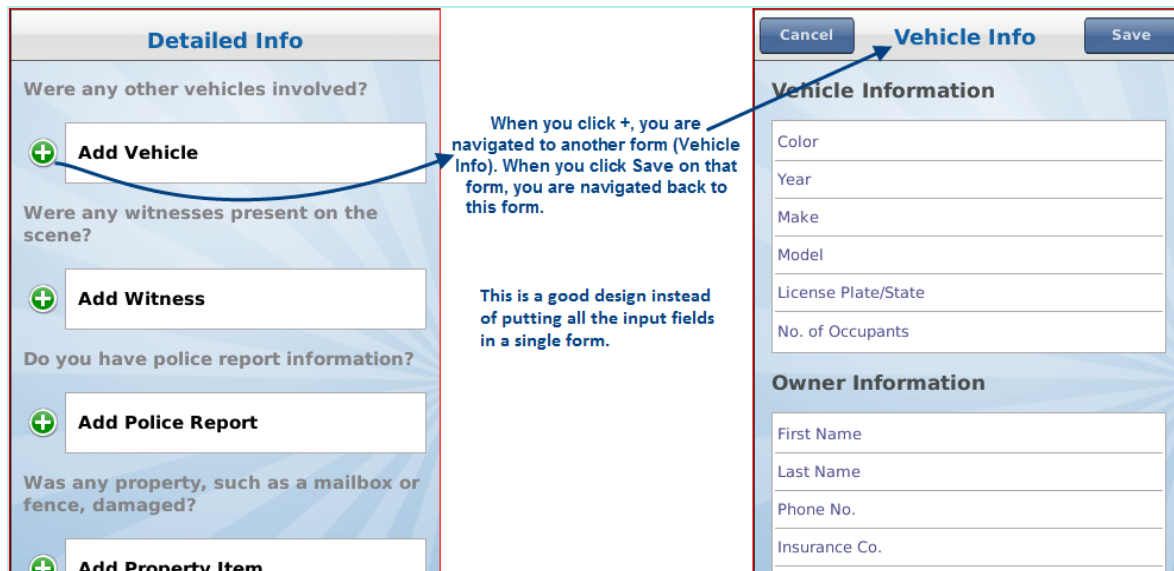
Logo:

2.1.7 General Guidelines

Following are the General guidelines:

- Use native widget paradigms for Navigation, Application Menu, and Group Widgets. Generally overriding these native behaviors are not advisable unless there is a very good reason. For example, iPhone applications typically have a back button on the top-left corner while Android applications use the device back key to achieve the same behavior. We recommend using the default behavior instead of providing a back key on the top left hand corner for Android applications. In this specific scenario, in case the UI designer mandates the use of back button for Android, the developer must still implement the platform specific behavior for the back key (if required, override the `onDeviceback` or `onEscape` form level events).
- Consider internationalization while designing the application even though it may not be required in the current application version. This will ensure that the application design is future proof.

- Avoid hard-coding user displayable content, instead use i18n keys to externalize this content. In case the application supports only English, use en_US as the default locale and define i18n keys for this locale.
- Follow proper naming convention for the widgets and images in the application. For more information, see *Kony Visualizer User Guide*. Do not name the widgets with keyword or namespaces used by Kony APIs.
- Avoid forms where the virtual height of the form is twice the height of the screen. Exception to this guideline is a Segmented UI as a screen level widget. In case, the form has more number of input fields, we recommend breaking the form into sections. Tapping on each section should lead to a new form. See the screen below for reference.



- Setting the screen level property to True also avoids the double scrolling problem in iPhone.
- Use the Container widgets only when required. For example, placing a single label in a FlexContainer or FlexScrollContainer is not an optimal usage of the container widgets.
- To set margins or paddings to zero, provide explicit values in the margins or paddings screen.

- Set margins for the containers first and then for the child widgets within.
- Use either top margins or bottom margins for creating space between components but not both when it can be achieved with only one of these two.
- Keep the data packet size between the device and the Kony Server as low as possible. This can be achieved by retrieving only those elements from the server that need to be displayed to the user or which drive the user navigation (UI Logic). In general, the elements which are never utilized on the client side, should be marked with Session scope in service request/response.
- Do not create too many skins. Use the existing skins as much as possible.
- Use skins with gradients (vertical and split) instead of images to achieve the same look and feel. If you use images, you must provide images for different form factors. Usage of images increases the size of the binary file.
- Remove all unused images and skins from the project, as it increases the binary size.
- Make sure you set the field-type for all user input fields. For example, the field-type (keyboard type) should be numeric if you have a field that expects the ZIP code to be entered.
- Ensure you have enough space (margins and paddings) between widgets when developing applications for touch devices. If you have too many widgets, it becomes difficult for the user to make selections, leading to a bad user experience.
- Use images that are big enough for clickable widgets. This enables you to easily click on touch phones.
- Avoid setting paddings for buttons with an image background. And set hexpand and vexpand to false.
- Use similar fonts, display, and focus patterns throughout the application for a uniform look and feel. Use themes wherever possible.
- Use font sizes that are large enough for the user to read.

- Make sure that the headers and footers do not exceed 30 percent of the screen height. This ensures that there is enough area for scrollable content in the page. If you severely restrict the data area for the form, the user will have to scroll a lot more to see the content.
- Avoid including text in images for the following reasons:
 - Supporting internationalized content would require multiple skins.
 - It increases application size.
 - It creates maintenance issues if the application supports internationalized content.
- Be aware that margins and paddings are *% properties* (dependent on the available width of the container individual platforms except iPhone).
- Follow a standard margins/paddings pattern for consistent representation within the application. For example, use 10 px (and 2%) for all left and right margins and 5px (1%) for all the top and bottom margins.
- Use progress indicators (`window.showloadingscreen` for Native Applications and `BlockedUI` for Mobile Web) to indicate the user that a long activity is running and is in progress. For example, fetching data from a network call or rendering data on a Segmented UI. If you do not use the progress indicators, it may lead to a bad user experience as the user may try to access other widgets or perform other actions on a form.
- Use the `pngout/pngcrusher` or other similar tools, to shrink the images for optimizing the memory size consumed by the image.
- Use keyboard codes instead of images to support internationalization of special characters. For example, use `numock+Alt+0169` for the copyright symbol (©).
- For TextBoxes, use placeholder property appropriately.
- For lines, create a Label with 1 dp height.

2.1.8 Browser

Following are the Browser Widget guidelines:

- Use RichText widget only when there is a need to display text with different fonts or HTML tags (for example, bold, italics, and so on). Use this widget only for small content (2-3 lines of text). If the content is large, use a Browser widget instead.
- Do not use more than two Browser widgets in an application as a Browser widget is heavy in terms of memory (especially on Android platform).
- Avoid placing other widgets on the forms that have either browser or Map widget as they are more optimized if they are screen level widgets. If absolutely essential, for such forms create a custom header and footer, in which other widgets can be placed (below or above the Map or Browser widgets). This is an important aspect to be considered for non-touch devices.
- When you set the ScreenLevel widget Platform Specific Property (PSP) to true for a Browser, make sure that other widgets like Map or Segment have this property set to false. If multiple widgets have this property set to true, the information in these widgets may not display with appropriate scrolling behavior.

2.1.9 Map

Following are the Map Widget guidelines:

- Avoid using multiple Map widgets in an application. Use a single Map widget and write the code in such a way that data is redirected to the same widget.
- Ensure that the Google Map Widget or any other such external interface has its branding associated with it. If you do not do this, the application may be rejected when submitted for certification.
- When you set the ScreenLevel widget PSP to true for a Map, make sure that other widgets like Segment or Browser have this property set to false. If multiple widgets have this property set to true, the information in these widgets may not display with appropriate scrolling behavior.
- For development convenience, ensure that all developers in the project use the same `Android debug key` and `Android Map key`.

2.1.10 Segmented UI

Following are the Segmented UI guidelines:

- Use the *orientation* property for a Segmented UI to align the child widgets horizontally or vertically instead of using an HBox or VBox to achieve the same.
- Avoid complex view hierarchy while designing Segmented UI. A view hierarchy is defined as the arrangement of boxes and widgets within them. Nesting of boxes in multiples of 4-5 and having 10-15 widgets in each of these boxes can lead to memory and performance issues.
- For a simple segment design, the number of rows in a segment can be as high as 100, while for complex segment design, it can be as low as 20.
- Use pagination if there are more than [15-20 records](#)¹ to be displayed in a Segmented UI.
- In case, the Segment has more than 15-20 records or has a complex view hierarchy, setting the ScreenLevel widget PSP to true for iPhone and Android leads to better scrolling performance. This is due to the fact that with this property set to true, the Segmented UI widget re-uses the boxes as the user scrolls. The downfall of this property is that the Segmented UI is the only scrollable widget on the form. The other widgets should be the part of header or footer.
- The `segui.setdata` method must not be invoked repeatedly or in a loop. Use the `setDataAt` or `addData` methods instead. Invoke the `segui.setdata` method only when all the data is ready.
- **Example of bad code:**

```
local segdata = {}
for i,v in ipairs(somedataset)
...segdata[i] = v;
...segui.setdata(formid.seguid, segdata);
```

¹The number 15-20 has been arrived at by looking at popular applications and taking into account the user experience.

```
end
```

Example of good code:

```
local segdata = {}  
for i,v in ipairs(somedataset)  
...segdata[i] = v;  
end  
segui.setdata(formid.seguid, segdata);
```

- Use the *widgetdatamap* property for Segmented UI to define the structure of the data and then invoke the *setdata* method.
 - Make sure that all the required widgets are added in the *widgetdatamap* property.
 - Do not create excessive labels in a Segment for mapping hidden fields. Use the *Hidden Columns* property to set hidden fields.
- For good scrolling behavior on non-touch devices, avoid SegmentedUI design that results in a focusable segment whose height is more than the available displayable height.

Note: Here the segment height is the height of the individual segment and not the combined height of all the segments in a SegmentedUI.

2.1.11 Guidelines for Splash Screen

When the system launches an app, it temporarily displays a static launch image on the screen. Your app should provide this image, with the image contents usually containing a pre-rendered version of your app's default user interface.

- The purpose of splash screen is to give the user immediate indication that the app is being launched. It also gives your application, time to initialize itself and prepare its initial set of views for display. When your app is ready to run, the system removes the image and displays your application's home screen.
- Splash screen should occupy the complete device screen. No Black or white Borders should be present at the top/bottom/sides of the splash screen.
- No Header Box should be present at the Splash Screen unless specified otherwise.
- Should be docked (Splash Screen should not move when application is in loading state).
- Should be consistent with the requirements in terms of Application Name Logo, Version Number and a Trade Mark Symbol.
- A video can be configured as a splash screen, which will be played while the application is getting initialized.

2.1.12 Guidelines for Configuring Splash Screen for iOS Applications

Image	Device	Resolution	Image Name
Splash Screen image (required)	iPhone 3 (Portrait)	320 x 480 pixels	Default_Portrait_ iPhone.png
	iPhone 3 (Landscape)	480 x 320 pixels	Default_Portrait_ iPhone@2x.png

Image	Device	Resolution	Image Name
Splash Screen image (required)	iPhone 4 (Portrait)	640 x 960 pixels (@2x)	Default.png
	iPhone 4 (Landscape)	960 x 640 pixels (@2x)	Default@2x.png
Splash Screen image (required)	iPhone 5 (Portrait)	640 x 1136 pixels (@2x)	Default_568h@2x.png
	iPhone 5 (Landscape)	1136 x 640 pixels (@2x)	Default_568h.png
Splash Screen image (required)	iPhone6 (Portrait)	750 x 1334 pixels (@2x)	Default_667h@2x.png
	iPhone6 (Landscape)	1334 x 750 pixels (@2x)	Default-Portrait_736h@3x.png
Splash Screen image (required)	iPhone 6 plus (Portrait)	1242 x 2208 pixels (@3x)	Default_667h.png
	iPhone 6 plus (Landscape)	2208 x 1242 pixels (@3x)	Default_Landscape_736h@3x.png
Splash Screen image (required)	iPad (portrait)	768 x 1004 pixels	Default_Portrait.png
		1536 x 2008 pixels (@2x)	Default_Portrait@2x.png
Splash Screen image (required)	9.7 inch iPad Pro, iPod Air 4, iPod Mini 2	1536 x 2048	

Image	Device	Resolution	Image Name
		2048 x 1536	
Splash Screen image (required)	iPad (Landscape)	1024 x 748 pixels	Default_Landscape.png
		2048 x 1496 pixels (@2x)	Default_Landscape@2x.png

For more information see:

http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Icons/Images/IconImages.html#//apple_ref/doc/uid/TP40006556-CH14-SW1.

2.1.13 Guidelines for Configuring Splash Screen for Android Applications

- Use a background color around the image so the image appears stretched. (for example, if the image is white near the edges, use a white background). It looks better if there is not an obvious border around the image and it fades into a background.
- Always include portrait and landscape versions of the images.

Display	Orientation	Resolution	Image Name
LDPI (low) ~120dpi	Portrait	200 x 320 pixels	ldpi_portrait.png
	Landscape	320 x 200 pixels	ldpi_landscape.png
MDPI (medium) ~160dpi	Portrait	320 x 480 pixels	mdpi_portrait.png
	Landscape	480 x 320 pixels	mdpi_landscape.png
HDPI (high) ~240dpi	Portrait	480 x 720 pixels	hdpi_portrait.png
	Landscape	720 x 480 pixels	hdpi_landscape.png

Display	Orientation	Resolution	Image Name
XHDPI (extra-high) ~320dpi	Portrait	640 x 960 pixels	xhdpi_portrait.png
	Landscape	960 x 640 pixels	xhdpi_landscape.png
XXHDPI (extra-extra-high) ~480dpi	Portrait	960 x 1440 pixels	xxhdpi_portrait.png
	Landscape	1440 x 960 pixels	xxhdpi_landscape.png
XXXHDPI (extra-extra-extra-high) ~640dpi	Portrait	1280 x 1920 pixels	xxxhdpi_portrait.png
	Landscape	1920 x 1280 pixels	xxxhdpi_landscape.png

For more information see: http://developer.android.com/guide/practices/screens_support.html

2.1.14 Guidelines for Configuring Splash Screen for Windows Applications

For splash screen, image resolutions are provided in the following table. In Kony Visualizer, splash screen images must be placed in `resources` folder.

- There is no support for Windows splash screen in landscape mode.

Image format	Resolution	Image Name
WVGA	480 × 800	SplashScreenImage.screen_WVGA.png
WXGA	768 × 1280	SplashScreenImage.screen_WXGA.png
720p	720 x 1280	SplashScreenImage.screen_720p.png

For more information see:

[https://msdn.microsoft.com/en-us/library/windows/apps/ff769511\(v=vs.105\).aspx](https://msdn.microsoft.com/en-us/library/windows/apps/ff769511(v=vs.105).aspx)

2.2 Guidelines for App Menu

2.2.1 iOS

Guidelines for the image resolutions to be used.

Image	Size
Tab bar image (normal)	30 x 30 pixels
Tab bar image (@2x)	60 x 60 pixels

- A maximum of five app menu items can displayed on the screen.
- Be sure to avoid using images that replicate Apple products in your designs. These symbols are copyrighted and product designs can change frequently.
- If we have 4 menu options at the bottom of the screen. Once the user logs out from the application and login's again, the first menu button should be in Focus.

Reference: [Guidelines for iOS](#)

2.2.2 Android

Guidelines for the image resolutions to be used:

Image	Size
Square Icon (ldpi)	22 x 22 pixels
Square Icon (mdpi)	30 x 30 pixels
Square Icon (hdpi)	44 x 44 pixels

- Do not provide an Exit button in the application unless it is absolutely required. Android is a multi-tasking platform and the application can exist in the background. This allows the users to get back to the application quickly.
- Be sure to avoid using images that replicate Android products in your designs. These symbols are copyrighted and product designs can change frequently.

For more information see: http://developer.android.com/guide/practices/ui_guidelines/icon_design_menu.htm

2.2.3 Windows

Guidelines for Windows App menu:

- App menu items should have an icon and should have a text hint.
- App menu items should be 48 x 48 pixels and have a white foreground on a transparent background using an alpha channel.
- The number of items displayed in an app menu is limited to five.
- Do not provide an Exit button for the application.

2.2.4 General Guidelines for App Menu

General guidelines for App menu:

- In iOS, you must provide an image or a skin for the focus icon of each menu item.
- Active menu page should be in Focus
 - If user is at Home Screen, Home menu should be in focus.
 - If user is at Insurance/ travel page, the Insurance / travel menu button should be in focus.
- Clicking the Application Menu should only show a form and not include time consuming logic like invoking services. This behavior is imposed by the underlying SDK and non-adherence to this guideline will lead to degraded user experience. Ideally as soon as an Application Menu is

clicked, the user should be able to view the target form without any perceived delay. In case, the form requires the data to be populated from a network.

2.2.5 Configure an App Menu

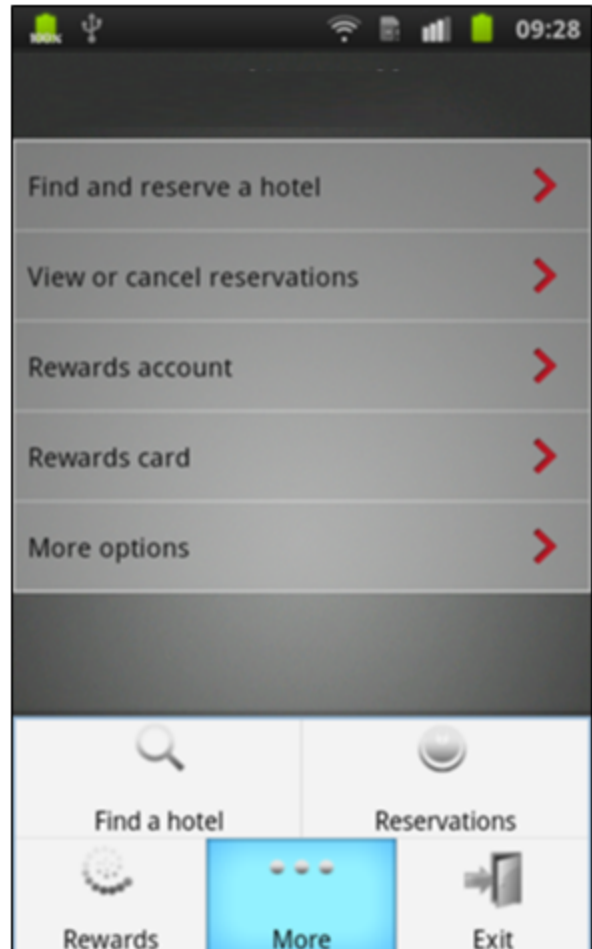
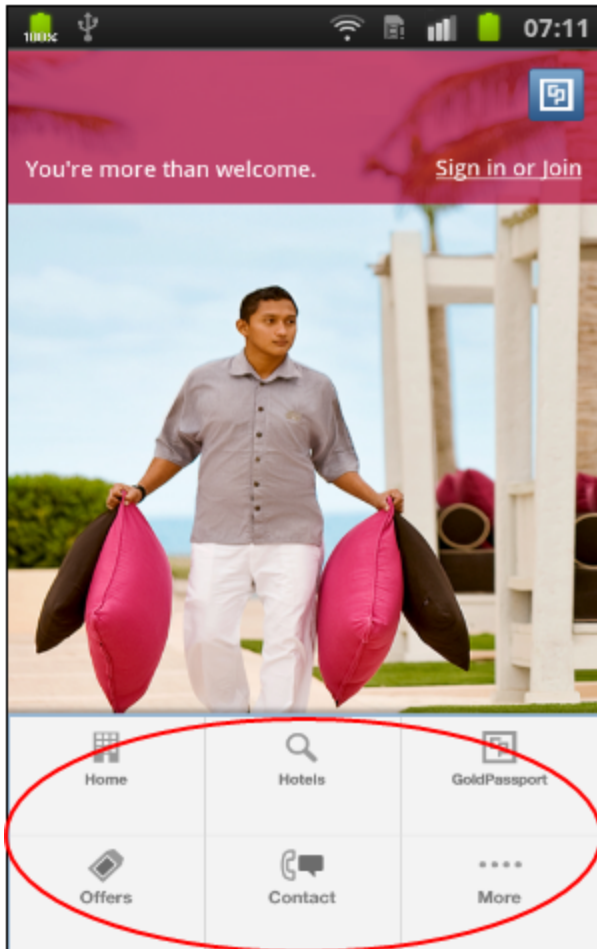
App-Menu Master Data ✕

Skin :

Focus Skin :

ID	Title	Icon	onClick	Selected Icon (iOS7)
appmenuitemid1	Item 1	option1.png <input type="text" value="..."/>	Not Defined <input type="text" value="..."/>	<input type="text" value="..."/>
appmenuitemid2	Item 2	option2.png <input type="text" value="..."/>	Not Defined <input type="text" value="..."/>	<input type="text" value="..."/>
appmenuitemid3	Item 3	option3.png <input type="text" value="..."/>	Not Defined <input type="text" value="..."/>	<input type="text" value="..."/>
appmenuitemid4	Item 4	option4.png <input type="text" value="..."/>	Not Defined <input type="text" value="..."/>	<input type="text" value="..."/>

2.2.6 App Menu Examples



2.3 Guidelines for Fonts

Fonts (size or type) should be consistent across the screens and as per client requirements.

2.3.1 Guidelines for Configuring Fonts for Rich Client Applications

Platform	Font Family
iOS	Helvetica
Android	Roboto
Windows	Segoe

Note: The table above lists the recommended fonts when not specified in the style guide. This recommendation depends on the version of platform being used.

2.3.2 Guidelines for Configuring Fonts for Mobile Web Applications

Mobile Web Category	Font Sizes (in Pixels)		
	Form Factor (240)	Form Factor (320)	Form Factor (480)
iOS	NA	15	NA
Android	12	15	18

Note: This guideline is applicable only if we assume that the font family is Arial. Also, these base font sizes (in pixels) are equivalent to 100%.

2.4 Guidelines for Fonts

Fonts (size or type) should be consistent across the screens and as per client requirements.

2.4.1 Guidelines for Configuring Fonts for Rich Client Applications

Platform	Font Family
iOS	Helvetica
Android	Roboto
Windows	Segoe

Note: The table above lists the recommended fonts when not specified in the style guide. This recommendation depends on the version of platform being used.

2.4.2 Guidelines for Configuring Fonts for Mobile Web Applications

Mobile Web Category	Font Sizes (in Pixels)		
	Form Factor (240)	Form Factor (320)	Form Factor (480)
iOS	NA	15	NA
Android	12	15	18

Note: This guideline is applicable only if we assume that the font family is Arial. Also, these base font sizes (in pixels) are equivalent to 100%.

2.5 Guidelines for Title Bar or Navigation Controls

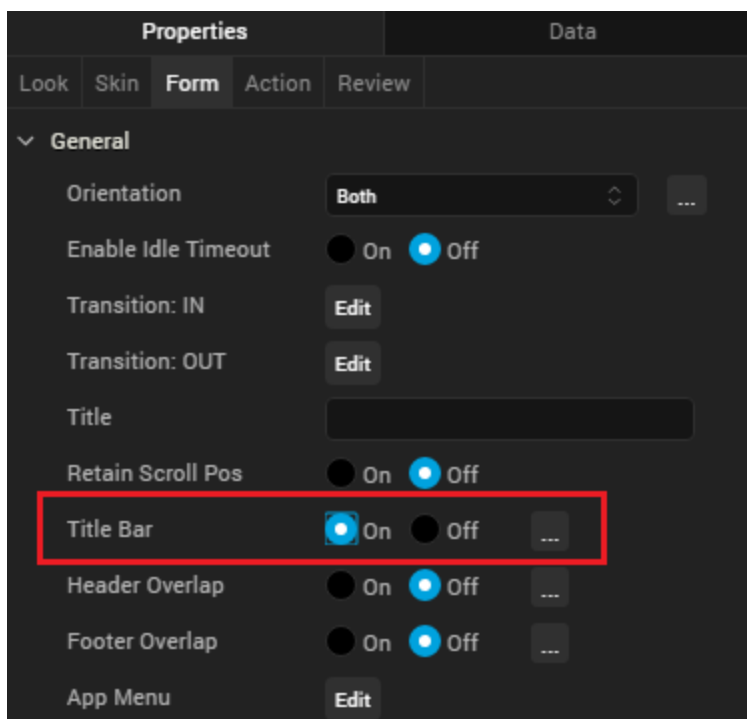
2.5.1 iOS

Typically, all iPhone applications have a title bar with a back button on the top-left corner which facilitates the navigation in the application. It is recommended to use this same native behavior.

2.5.2 Android

In Android native behavior, navigation in the application is through the use of the device back key. It is recommended to use this same native behavior.

2.5.3 Configure a Title Bar on a Form (iPhone)

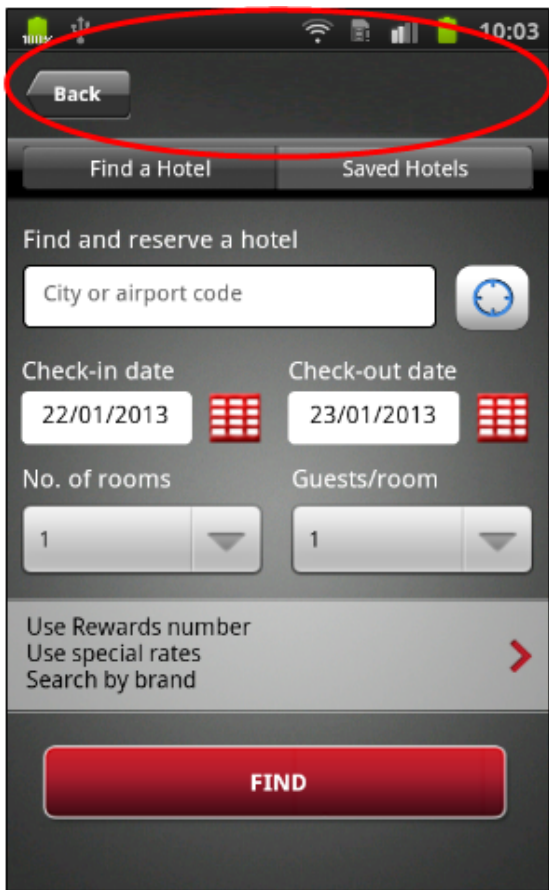


2.5.4 Configure Device Back on a Form (Android)

Look	Skin	Form	Action	Review		
			postShow	Click Edit to add actions	Edit	
			onHide	Click Edit to add actions	Edit	
			onDestroy	Click Edit to add actions	Edit	
			onNavigate	Click Edit to add actions	Edit	
			onTouchStart	Click Edit to add actions	Edit	
			onTouchMove	Click Edit to add actions	Edit	
			onTouchEnd	Click Edit to add actions	Edit	
			iPhone			
			onOrientationChange	Click Edit to add actions	Edit	
			widgetToZoom	Click Edit to add actions	Edit	
			onZoomStart	Click Edit to add actions	Edit	
			onZooming	Click Edit to add actions	Edit	
			onZoomEnd	Click Edit to add actions	Edit	
			Android			
			onDeviceMenu	Click Edit to add actions	Edit	
			onDeviceBack	Click Edit to add actions	Edit	
			onOrientationChange	Click Edit to add actions	Edit	

2.5.5 Title Bar Examples

Page Title Header will be horizontally center-aligned and vertically middle-aligned on the navigation bar by default.



2.6 Guidelines for Progress Indicators (Rich Clients)

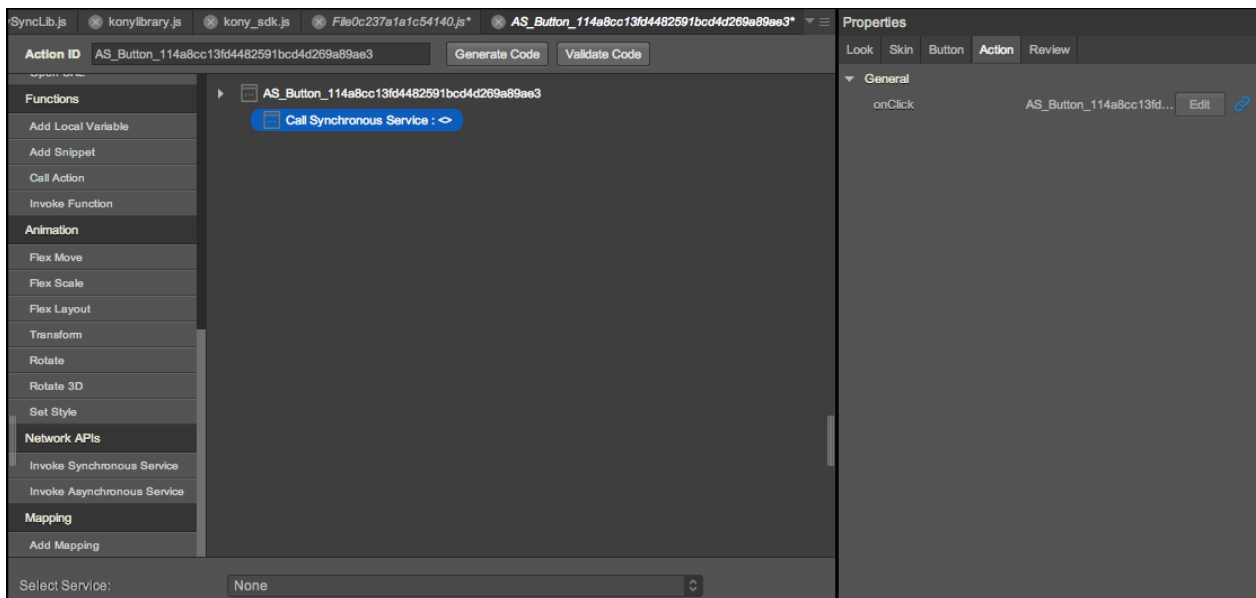
Following are the guidelines for progress indicators(Rich clients):

- It is recommended to use progress indicators to indicate to the user that a long running activity is in progress.
- If you do not use the progress indicators, it may lead to a bad user experience as the user may try to access other widgets or perform other actions on a form.
- Below are few scenarios in which a progress indicator is a must.
 - Fetching data from a network call.
 - Rendering data on a Segmented UI.
 - Any “time consuming” activity.
 - Make sure to dismiss the created progress indicator.
- We can design a form with static content on it and use it as an alternative for progress indicators as per the use case.
- Use the BlockedUI skin for Mobile Web.
- Use the BlockedUI skin whenever there is a use case to block the user interaction while a background activity is in progress, ex: a payments / transfers service call, login / logout service call etc.
- BlockedUI skin must get activated whenever the data starts Loading / Service gets invoked. (i.e. when you select an option in a screen, application starts loading the data, here the Block UI should get activated in that screen so that user should not be able to access other options in the screen).

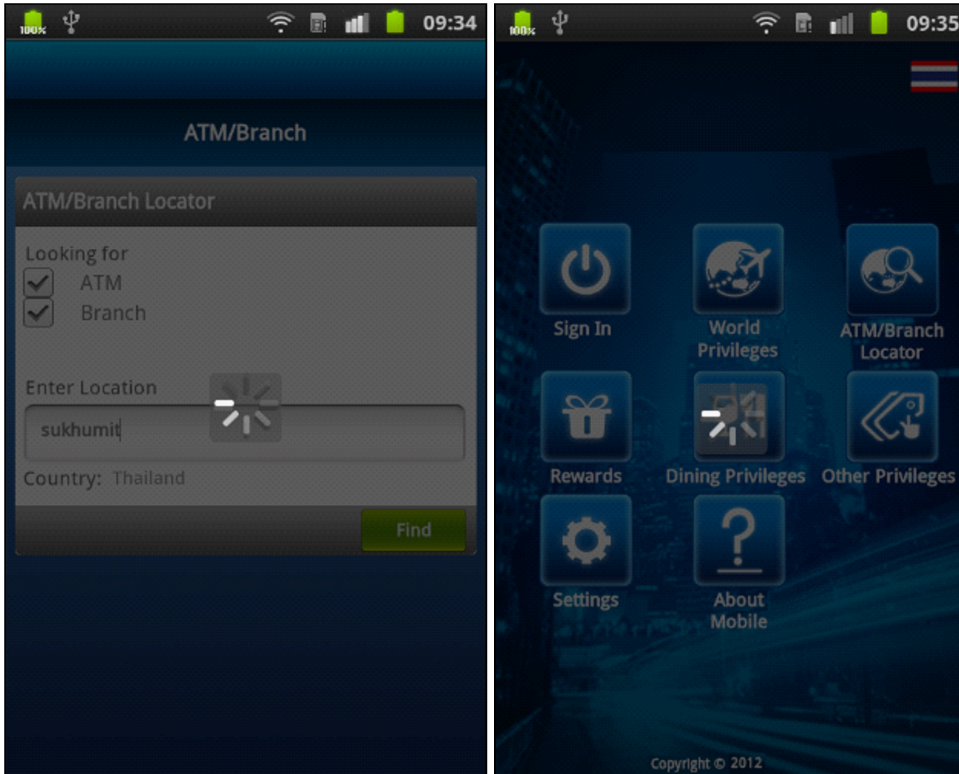
2.6.1 Create a Progress Indicator

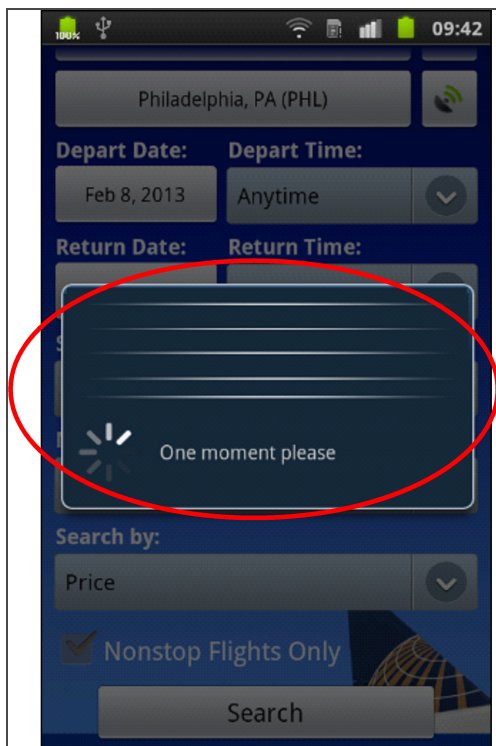
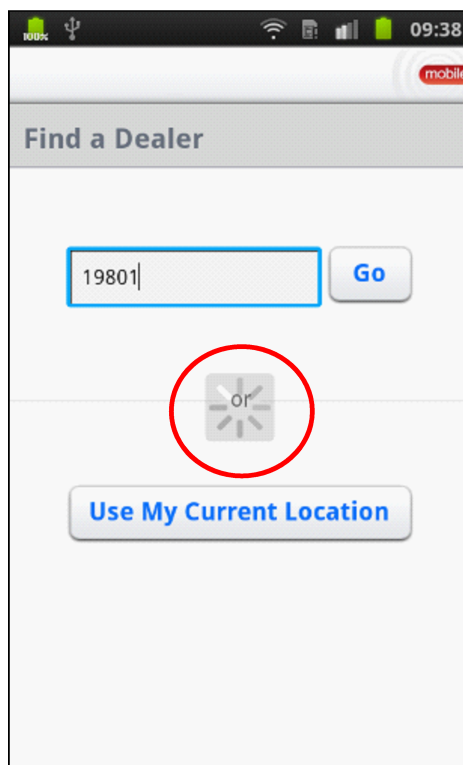
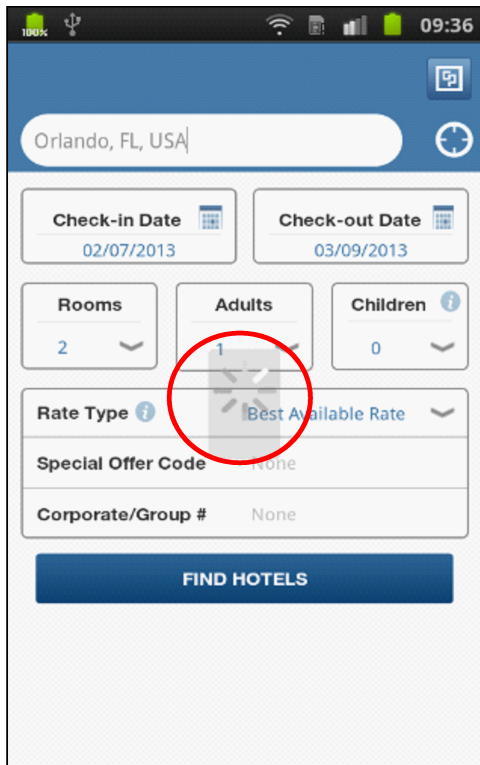
- Use the below API to create a progress indicator.
 - `window.showloadingscreen.`
- Use the below API to dismiss a progress indicator.
 - `window.dismissloadingscreen.`

2.6.2 Create a Blocking Indicator



2.6.3 Progress Indicator / Blocking UI Examples





2.7 Guidelines for Pop-ups or Alerts

Following are the guidelines for pop-ups or alerts.

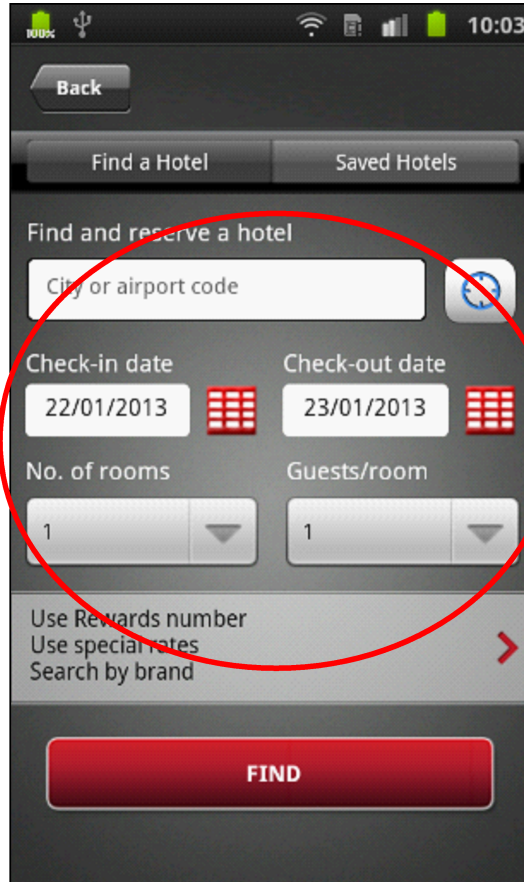
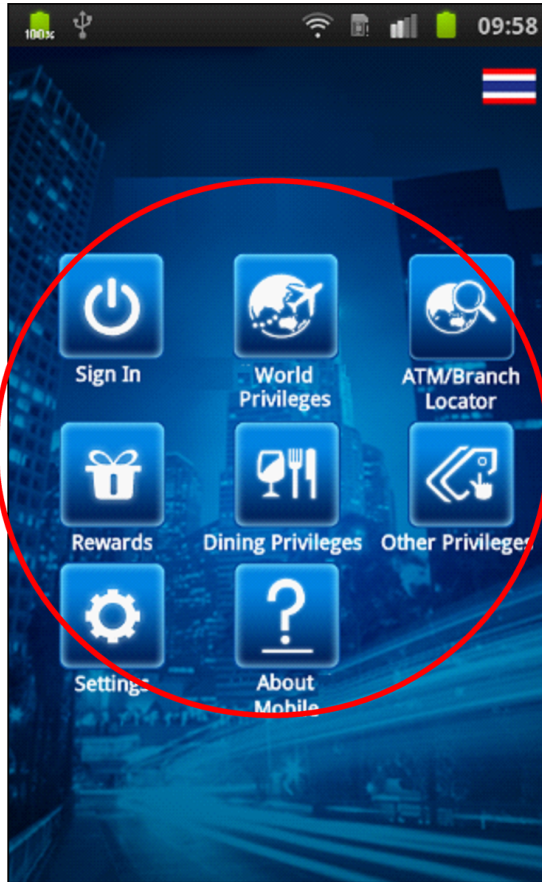
- Use native alerts to the extent possible (as the color scheme of native alert is dependent on device theme and will vary when the user changes the device theme), to give a more consistent (native) behavior of the application.
- While designing a custom pop-up, make sure that
 - It is aligned properly at the center of the screen.
 - Define the pop-up as a modal or non-modal, based on the use case. Use the modal pop-up when it is required to block the user, until the user responds to the pop-up.
- It is not recommended to open a new alert from within the handler of one alert.
- It is recommended to configure the “inTransition”, “outTransition”, and “transparencyBehindPopup” properties of a pop-up accordingly to the use case.

2.8 Guidelines for Alignments (Paddings)

Following are the guidelines for Alignment (paddings).

- It is recommended to use paddings as percentage based instead of pixel based. Using percentage based paddings will arrange the widgets dependent on the available width of the container.
- Follow a standard paddings pattern for consistent representation within the application.
- To set paddings to zero, specify the value as zero in the paddings screen.
- Ensure you have enough space (paddings) between widgets when developing applications for touch devices. If you have too many widgets, it becomes difficult for the user to make selections, leading to a bad user experience.
- Make sure that the widgets are aligned properly to the center of the screen from left to right.

2.8.1 Proper Margins and Padding Examples



2.9 Guidelines while using Images

Following are the guidelines while using images.

- During the design time, after placing the image, there is a property to set horizontal and vertical expansion either True or False. Set these properties to false to make sure that the image will not get stretched.
- Ensure that platform specific images are provided in the respective platform specific folders of the “resources” folder.
- While targeting the application for iOS platform, make sure to provide retina (@2x) images.

2.10 Guidelines for using Segment Widget

2.10.1 Avoid using unnecessary Container Widgets

Following are the guidelines to avoid using unnecessary Container widgets.

- Use the *orientation* property for a Segmented UI to align the child widgets horizontally or vertically instead of using unnecessary container widgets (HBox or VBox) to achieve the desired UI look and feel.
- The memory space occupied by a segment widget will be based on the number of widgets that are placed inside the segment widget, and the number of records that you populate into the segment widget.

2.10.2 Avoid Complex View Hierarchy

Avoid complex view hierarchy while designing Segmented UI. A view hierarchy is defined as the arrangement of container widgets and child widgets within them. Nesting of container widgets in multiples of 4-5 and having 10-15 widgets in each of these boxes can lead to memory and performance issues.

2.10.3 Standards around Maximum Number of Rows in a Segment

Following are standards around maximum number of rows in a Segment.

- For a simple segment design (segment with less than 5 widgets) - the maximum number of rows that we can populate into the segment is 100.
- For a complex segment design (segment with multiple nesting and having more than 5 widgets) - the maximum number of rows that we can populate into segment is 20.

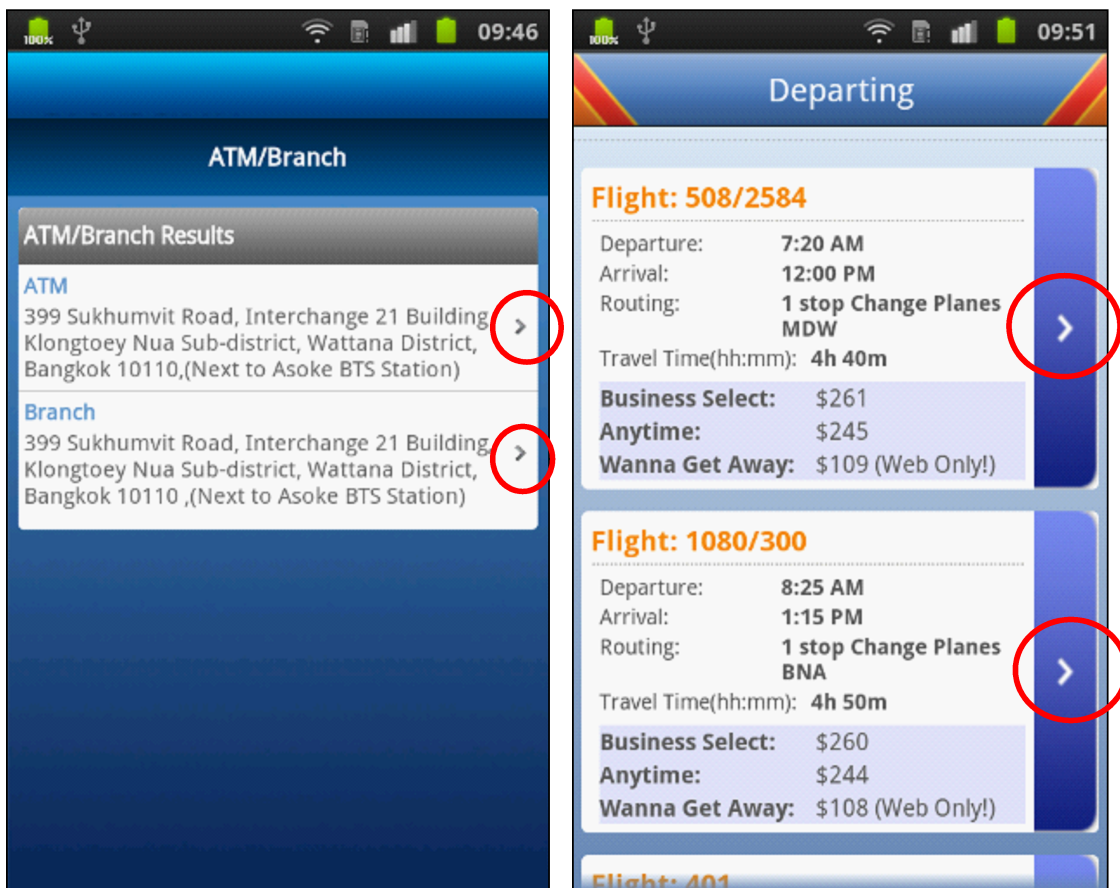
2.10.4 Use Pagination

It is recommended to use pagination whenever there is a use case to populate more than 20 records.

2.10.5 Do not use Hidden Widgets

- Use the Hidden Columns property to set hidden fields for each record in a segment. Do not use hidden widgets.
- The `segui.setData` method must not be invoked repeatedly or in a loop. Use the `setDataAt` or `addData` methods instead. Invoke the `segui.setData` method only when all the data is ready.

2.10.6 Segment Widget Examples

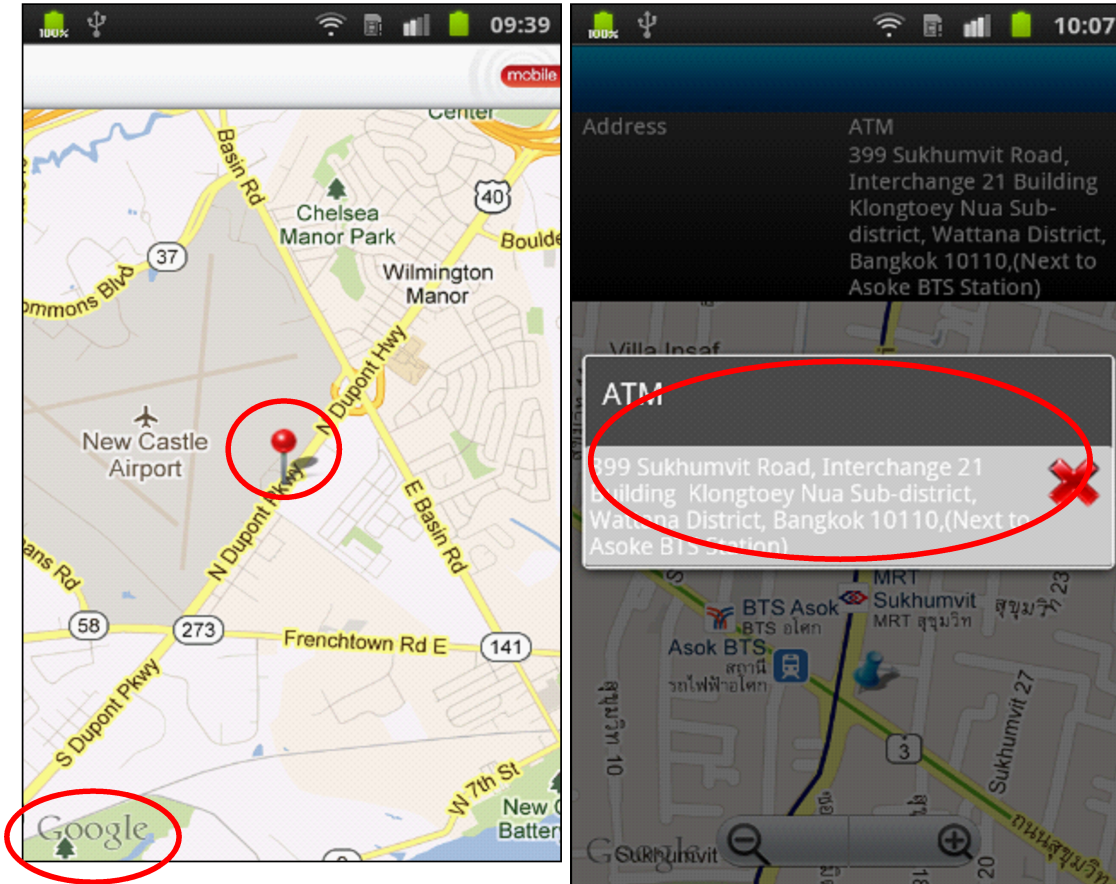


2.11 Guidelines while using Map Widget

Following are the guidelines while using Map widget.

- Understand that Map widget is a heavy widget which occupies lot of memory.
- Avoid using multiple Map widgets
 - Use a single Map widget and write the code in such a way that data is redirected to the same widget.
- Make sure to use proper branding while using external interface like Google Maps.
 - Ensure that the Google Map Widget or any other such external interface has its branding associated with it. If you do not do this, the application may be rejected when you submit the application for approval at respective App stores.
- The Pins on the Map should be easily clickable.
- If an application displays 10 results for a search result in a Map, then 10 Pins should get displayed in the map representing each Result.

2.11.1 Map Widget Examples



2.12 Guidelines while using Browser Widget

Following are the guidelines while using Browser widget.

- The Browser widget is a heavy widget and it consumes a lot of memory.
- Avoid using multiple Browser widgets in an application. Do not use more than two Browser widgets in an application as it is heavy in terms of memory (especially on Android platform).

2.13 Guidelines for Calendar Widget

Following are the guidelines for Calendar widget.

- Calendar widget - current date
 - Current Date in the Calendar widget of the application should be in Focus/Bold font, rest that can be selected should be in another normal color font while those that cannot be selected should be grayed out.

- Calendar skins can be configured using the viewConfig property as follows.

The screenshot shows the 'Properties' panel in Kony Studio, specifically the 'Skin' tab for a 'Calendar' component. The panel is divided into 'Properties' and 'Data' sections. The 'Properties' section includes a 'Normal' tab and a 'Data' section with a table of skin states. Below the table are 'Copy', 'Paste', 'Assign', and 'Duplicate' buttons. The 'General' section shows the 'Name' as 'slCalendar' and 'Platform' as 'Common'. The 'BackGround' section has 'Type' set to 'Single Color', 'Color' as black, and 'Opacity' at 0%. The 'Border' section has 'Size' at 0px, 'Type' as 'Single Color', 'Color' as black, 'Opacity' at 100%, and 'Style' as 'Plain'.

Normal	Focus	Grid	Cell
Cell - Selected	Cell - Focus	Cell - Today	Cell - Weekend
Cell - Inactive	Done Button	Cancel Button	

Copy **Paste** **Assign** **Duplicate**

▼ **General**

Name: 🔍

Platform: ... 🔗

▼ **BackGround**

Type: ⌵

Color:

Opacity: 0 %

▼ **Border**

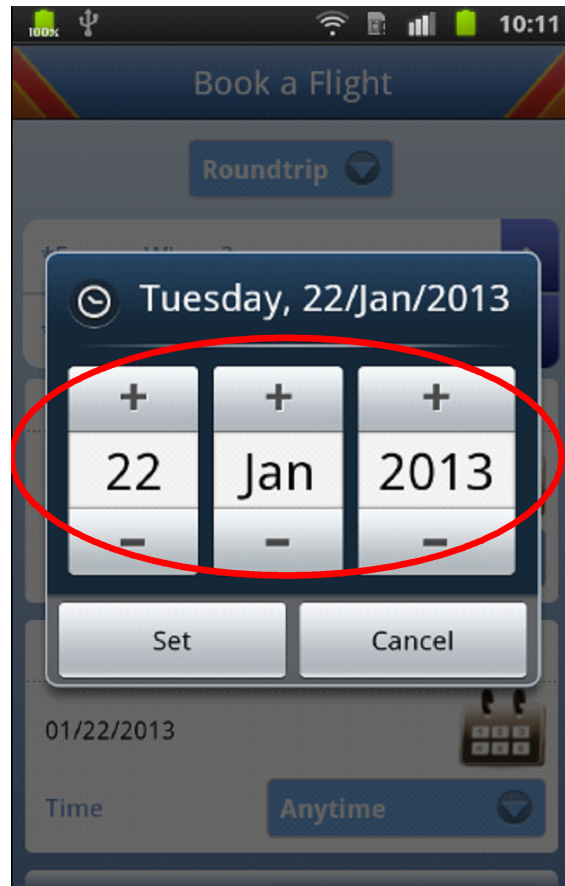
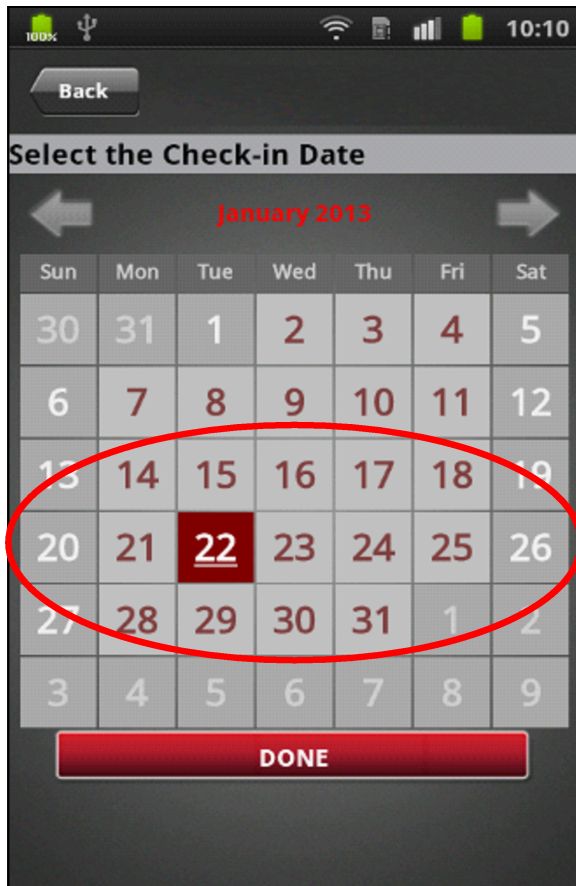
Size: 0 Px

Type: ⌵

Color:

Opacity: 100 %

Style: ⌵



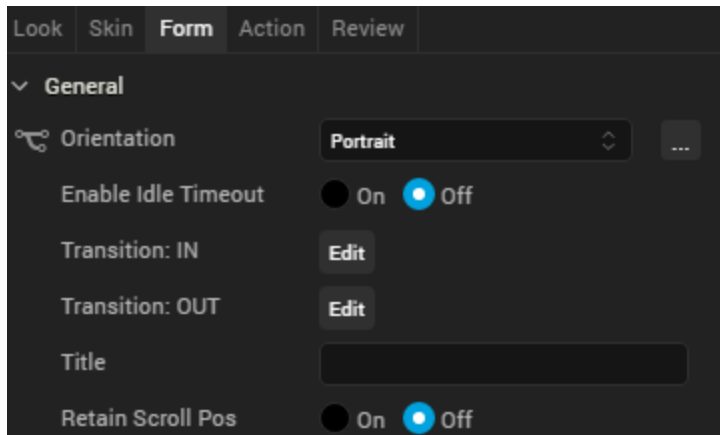
2.14 Landscape vs. Portrait View

In Landscape View, the user interface should not get distorted when compared to Application in Portrait View.

2.14.1 Achieve a Uniform UI in both Orientations

Following are the guidelines to achieve a uniform UI in both orientations.

- Create skins for both orientations for all the applicable widgets.
- Use percentage based paddings
- Use form forking, if the requirement is to have different forms designs for respective platforms.
- Define the “OnOrientationChange” event to dynamically change the form to be viewed / skins to be applied depending on the requirement.



2.15 Mobile Web Apps Only

Following are the guidelines for Mobile web apps.

- The color, text on the buttons should be same as in wireframes received from the customer.
- Need to have customized pages for unknown errors.
- iPhone - Application Name with icon (except color on the icon) should be added to Home Screen (Launch the Mobile web application in the safari, select adds to home screen by clicking on Bookmark menu). Application Name should get displayed on the icon.
- There should be a lock symbol on the browser top for secured applications (when redirects to https from http).
- The title of the screen on the top of the browser window should be in green color, if the application is having secured pages.
- The Page Title (Browser Top) should be constant or should change accordingly as per navigation, throughout the application.

2.16 Responsive Web Apps

Ensure that you follow these guidelines while designing Responsive Web applications.

- Assign tooltips to widgets at UI development stage.
- The new initial flex layout should be as mentioned in the diagrams attached.
 - The Layout Type of the Form must be Free Form.
 - The Layout Type of the content in the Form must be Flow Vertical.
 - Invoke the following code snippet in the postShow event of the form:


```

this.view.flxMain.minHeight=kony.os.deviceInfo
().screenHeight-300+"dp";
          
```

 Here, 300 is the total height of header and footer.

- Invoke the above code snippet each time you resize the browser vertically and on breakpoint change.

Pros of following this layout type:

- The Layout will be uniform and easy to understand.
 - You need not invoke multiple `adjustScreen()` calls.
 - No need to set the heights of the background flexes of popup.
- Create a component only if it satisfies one or both of the following criteria:
 - The UI can be reused in at least 3 places.
 - Some of the UI logic can be encapsulated within the component controller to make the form controllers cleaner.
 - Ensure that there are no instances of UI properties set in the form controllers. For example, do not set layout properties in the form controller.
 - Every major component must have a break point change function. This function contains the changes in the layout property for the respective breakpoints.
 - All the actions (other than the form init event) must be mentioned in the form controller.
 - Ensure that you maintain the same name for the methods across all the form controllers. For example, ensure that you define the `onBreakpointChange` method with the same name in all the form controllers.
 - Create a form for each view. This reduces clutter and improves the readability of the code. For example, ensure that you have less than 150 widgets in a component.
 - Maintain a file for the constants used in skins, images, fonts, and icons. Ensure that you use the constants to refer to them, so that any changes you make get reflected in all the places.
 - Maintain proper widget hierarchy and order for better accessibility.

- For responsive design, make sure that the width property of the widgets are in relative scale (%) units whenever possible, and the height property of the widgets are in fixed scale (dp, px, etc.) units.
- Use the strategy of designing the mobile application first to design the flow of Flex Containers (free form, vertical, or horizontal).
- Design Segment templates in a **self-contained way** to reconfigure themselves in different breakpoints. This avoids the need to change the template and repopulate segments when a breakpoint change occurs in form controllers.
- While using custom widgets, make sure that they are designed in a way that allows easy reconfiguration of themes and texts.
- Implement Animations in the Action tab. Avoid using the form controller to implement animations whenever possible to improve the readability and maintenance of the code.
- Use the commonUtilities function wherever possible for common tasks such as enabling or disabling widgets.
- Extract out common functionalities which we put in form controllers (more than two) and see if they can be placed in Parent Controller or CommonUtilities.
- Add comments in the code wherever you are applying a workaround or hack, so that it is easier to track.
- It is recommended to use Margins instead of using the Width property for mobile and tablet breakpoints.
- Ensure that the scale mode property of Images is set as `Maintain aspect ratio` to avoid distortions.
- Avoid the creation of duplicate skins by following proper naming conventions and determining the set of skins from the style guide.
- **Single responsibility principle**

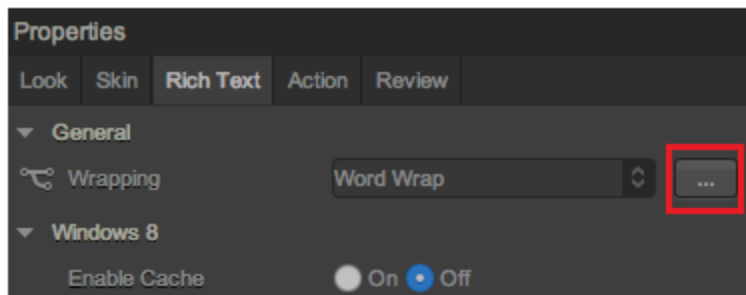
- Assign the Layout Type as Flow Horizontal or Flow Vertical only when it is required.
- Do not create a component if is not reused in at least 3 places.
- Do not create heavy components. It is recommended that you have less than 30 widgets in a component.
- Do not use nested components.
- Do not use static values (hardcoded values) in controllers.
- Do not use multiple force layouts. Restrict the use of force layout to the lowest widget level instead of at the form level.
- Do not commit changes to controller action files.
- Do not miss files (Studio Action JSON, skins) when you commit the changes to Git. Review the git status during a commit.

2.17 General User Interface (UI) Guidelines

2.17.1 Guidelines for Text and Grammar across the Application

Following are the guidelines for text and grammar across the application.

- Verify spelling and grammar on the screen.
- Capitalization of letters in the middle of a sentence should not be there unless required specifically
- Verify text does not get wrapped when it should not, for example, when displaying phone numbers in RichText or Label (only available on iPhone). Wrapping can be either char based or word based.



- Check for white spaces in between text.

2.17.2 Guidelines for Container Widgets

Use the container widgets only when required. For example, placing a single label in a FlexContainer or FlexScrollContainer is not an optimal usage of the container widgets.

2.17.3 Guidelines for Skins

Following are guidelines for skins.

- Do not create too many skins
 - Remove all unused images and skins from the project, as it unnecessarily increases the binary size.
- Use skins with gradients
 - Use skins with gradients (vertical and split) instead of images to achieve the same look and feel. If you use images, you should provide images for different form factors.
 - Usage of images increases the size of the binary file.
- Skin (background/controls/images) should be consistent across the screens.
- Ensure that you do not set layout properties for widgets in the formController to avoid the properties from being overridden at runtime. Kony recommends that you set the layout properties from the Properties panel of Quantum Visualizer.

2.17.4 Guidelines for Orientation Support

Building the application in both orientations

- If the application is targeted for both the orientations, then it is recommended to either fork the forms or to use the percentage (%) based margins and paddings, so that the user interface is not distorted between the orientations.

2.17.5 Guidelines for Headers and Footers

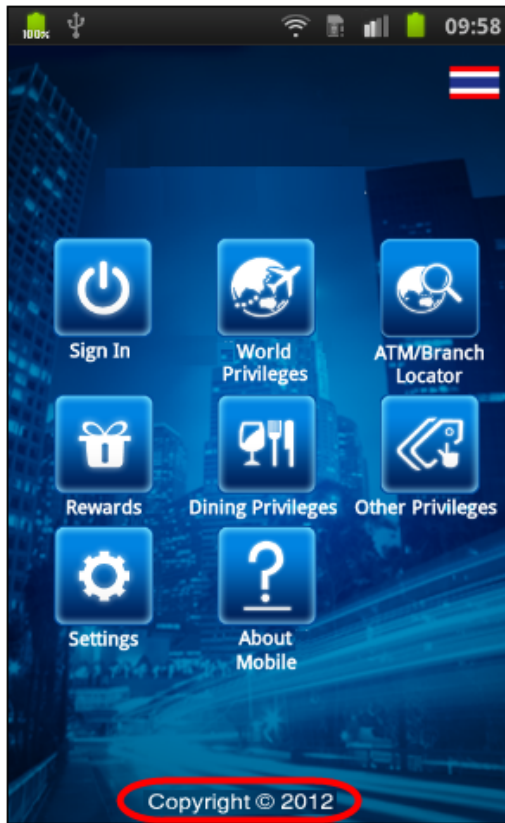
Following are the guidelines for Headers and Footers.

- Headers and Footers should not exceed 30% of the screen height:
 - Make sure that the headers and footers do not exceed 30% of the screen height.
 - This is to ensure that there is enough area for scrollable content in the page.
 - If you severely restrict the data area for the form, the user will have to scroll a lot more to see the content.
- The screen header and footers should be docked unless specified otherwise.
- Headers and footers should be consistent across screens.

2.17.6 Guidelines for Copyright

Following are the guidelines for Copyright.

- Make sure that the copyright year should be Current year.
- Copyright year image - © should be center aligned with the text.

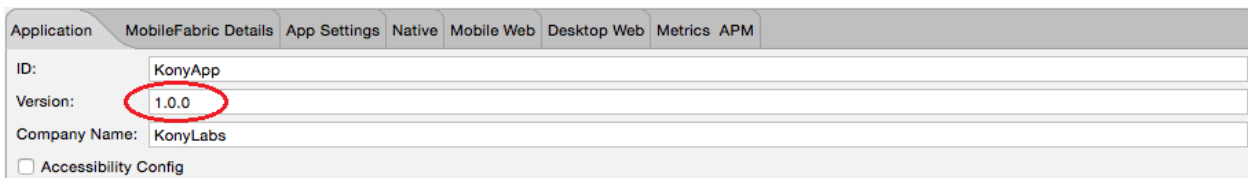


2.17.7 Guidelines for Application Info

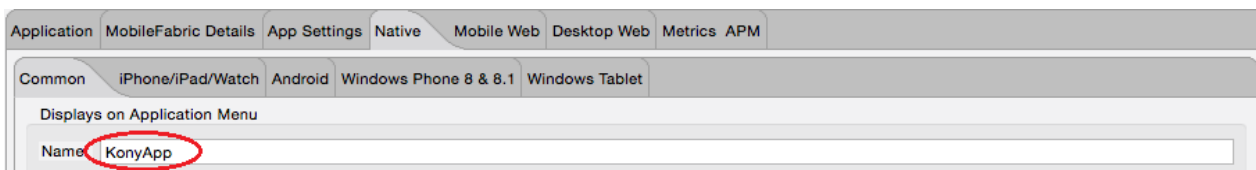
About the app should have the version of the application and this should be present in all apps.



- Version can be configured from the application properties window as follows.

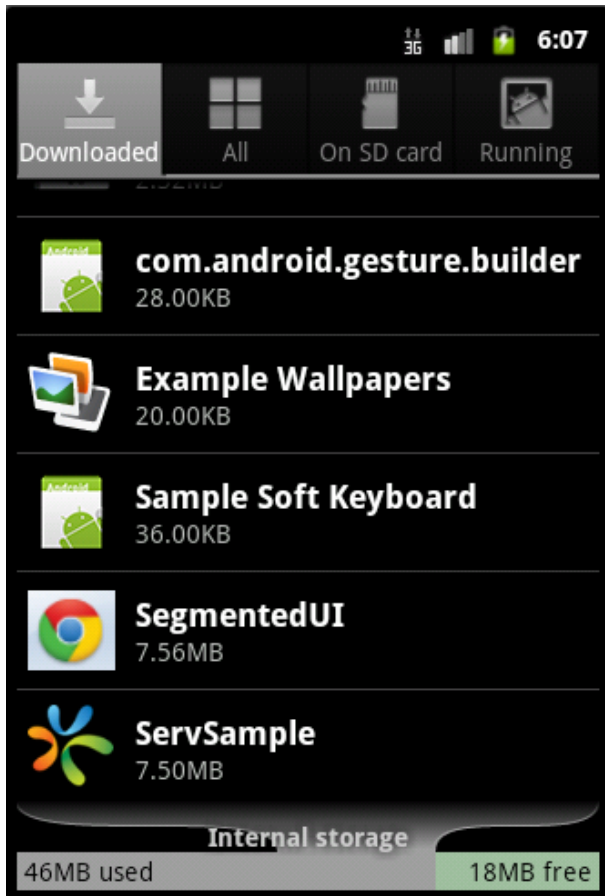


- Application display name can be configured from application properties as follow.



2.17.8 Guidelines for Application Icon in Phone Settings

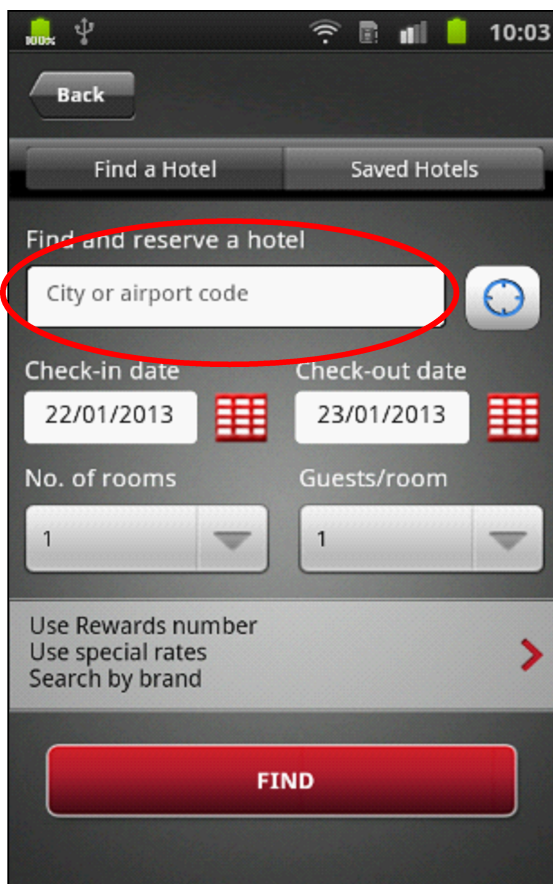
Application icon should be displayed in Installed/Uninstall Applications in the Phone Settings.



2.17.9 Other Guidelines

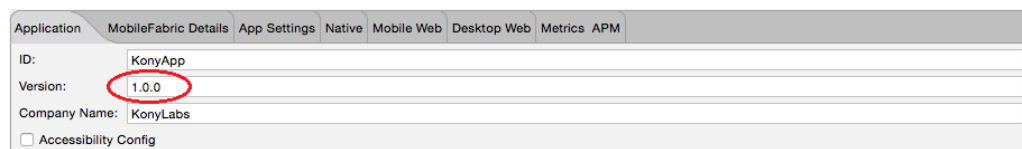
- Button size / shapes should be consistent across the screens and as per client requirements.
- Set the field-type for all the input fields:
 - Make sure to set the field-type for all user input fields. For example, the field-type (keyboard type) should be numeric if you have a field that expects the ZIP code to be entered.

- Avoid including text in images as:
 - Supporting internationalized content would require multiple skins.
 - It increases application size.
 - It creates maintenance issues if the application supports internationalized content.
- For Textboxes, use placeholder property appropriately.

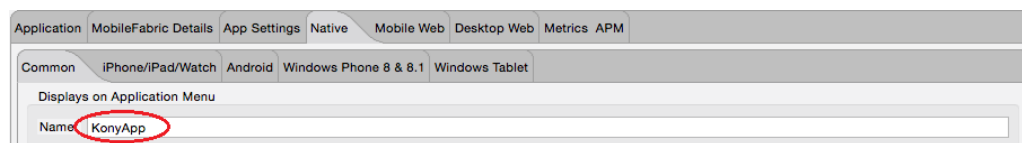


- Trademark Symbol should be super scripted to Normal Text.
- Mark Symbol should be present beside the Application Logo throughout the application unless specified otherwise.
- The Application logo will be center aligned within the screen header unless specified otherwise.

- Size of Vertical scroll bar while scrolling should not distort the application UI.
- Clickable indicator for rows in a segment (iOS):
 - On iOS segment record that has the details should consist an arrow “ > ” image unless specified otherwise. This indicates the row is clickable.
 - The arrow image for the segment to be shown only if the record has further information or data to display (e.g., list of flights followed by the flight details in the subsequent screen).
 - On iOS platform the arrow appears by default unless indicator defined as `SEGUI_NONE`. However, to achieve this look on other platforms do the following.
 - Add an extra image widget to the segment.
 - Configure (arrow) image as source to the extra image field in the segment.
 - Configure the correct version in the application properties window as follows.



- Configure the application display name from application properties windows as follow



- Only one tab in accordion view should be displayed with content at a time and shown as selected.
- Only one Radio button should be selectable at a time.
- Portrait/Landscape orientation should be verified wherever applicable.

- Selected Radio button should be retained after navigating back from other screens.
- The font size in the drop-down boxes to be same.
- The Text alignment within the Text field should be center aligned.
- Widgets in the Application should be docked. Should scroll with the screen, should not move if screen is not moving.
- If Any search option is present in the application- If application displays “ Showing / displaying 25 results “ , in that page total 25 results should get displayed. In Case if application Displaying 10 results in a screen , then message should be “ Showing 10 Results out of 25 “ with a Next button .
- When there are Next (>) and Previous (<) buttons (in the case of search results, photo galleries), in the first screen, Previous (<) should be disabled and in the last screen, Next (>) should be disabled.
- Non pagination scenario - if there are a few more results to be shown in the same screen after clicking on some button, then, new results should append to the existing. Here the focus should not go up/down. Should exactly be at the pixel where we clicked to view more results.
- Scrolling down to the application - data refresh should be seamless (Dependent on Network coverage)
- Phone Numbers in the application
 - i. On focus on the Skin should be Off after user cancels or performs a call, .i.e (when user Performs a call or exits the call, the particular Phone Number should have not have focus).

2.18 Naming Convention for Forms and Widgets

Vertical Applications are mobile/tablet solutions (products) that are built using the Kony App Foundation platform in a generic fashion for each of the domains (functional area Ex:- CRM) encompassing all the common functionality required by most of the customers in that domain. The idea behind building vertical applications is to reduce the time to deliver the product by re-using the out-of-the-box functionality provided by Apps as much as possible, with a little configuration and customization to suit the customer needs. This configuration and customization(if any) would be done by customer.

Naming convention would be needed for the following reasons:

- Avoid the usage of auto-generated ID (For example, CopyflexAccountsDetailsHeader0243d36df72834d) for a widget. The usage of auto-generated IDs reduces the maintenance and readability of referencing a widget from code.
- Understand the nature of the widget from the widget name.

Also, as customizations will be made on the vertical applications, it is recommended to use a suffix(KA) to the widget name so as to differentiate between widgets present in the apps and the newly added ones.

Following is the recommended naming conventions for UI development:

Component	Naming Convention
Form	frmFormNameKA
Pop Up Form	frmpopupnameKA
Skin	sknSkinNameKA
Resources	Resources.png
Global Variables	gblVariableNameKA

Component	Naming Convention
Internationalization Key	i18n. For example, i18n.lead.frmLeadListKA.lblLeadListHeaderKA
Flex Container	flxContainerNameKA
Label	lblLabelNameKA
Button	btnButtonNameKA
Image	imgImageNameKA
Segment	segSegmentNameKA
Calendar	calCalendarNameKA
CheckBoxGroup	cbxCheckBoxNameKA
ListBox	lbxListBoxNameKA
RadioButtonGroup	rbtnRadioButtonNameKA
RichText	rtxRichTextNameKA
TextArea	txtTextareaNameKA
TextBox	tbxTextboxNameKA
PickerView	pkcPickerNameKA
Map	mapMapNameKA
Camera	camCameraNameKA

Component	Naming Convention
Browser	bsrNameKA
DataGrid	grdNameKA
Tab	hdrNameTplKA
Header -Template	hdrNameTplKA
Footer -Template	fttrNameTplKA
Segment -Template	tmpNameTplKA
Map -Template	mapNameTplKA
GridCalendar -Template	grdNameTplKA

2.19 Components

Create a component for elements that have a similar design. A component can be reused across the different forms of an app.

Following are the guidelines for designing components:

- Analyze the designs of the app and find elements that use similar designs.
- If the same design is used in different forms in the app, create a component. This component can be dragged onto the required forms.
- Using components reduces the effort that is required to create similar designs. It also ensures consistency of design in the app.
- Using components also reduces the time that is taken to update the design across the app.

- The properties of the widgets (such as text or actions) in a component can vary for different forms. The properties can be overridden at the form level, which creates more scenarios in which a component can be reused.
- If a widget in a component requires the same action across the app, then the action can be configured at the component level.
- Some common use cases for components are: Headers, Footers, Breadcrumbs, Hamburger Menu, and Confirmation Dialog Boxes.

2.20 Px vs Percentage vs Dp

Follow these Guidelines while you set the Flex properties of a widget. You can set the Flex properties of a widget in the **Look** tab of the **Properties** panel.

UI Element properties	Px / Dp / percentage
Header Height	DP
Header Width	Percentage
Container Height	DP
Container Width	Percentage
Page Height	DP
Page Width	Percentage
Button Height	DP
Button Width	DP or Percentage
Icons	DP

UI Element properties	Px / Dp / percentage
Left Margin	DP
Right Margin	DP
Top Margin	DP
Bottom Margin	DP
Spacing between UI elements- Height	DP
Spacing between UI elements- Width	DP
Line separator- (Used as Labels)	PX
Text	Percentage
Text Height	Do not set to Preferred
Text Width	Preferred or Percentage
Background Image - Full page Image	Percentage
Background Image - Fixed size Image	DP

2.21 Coding Guidelines

This section explains a few guidelines to be followed while writing JavaScript code in an application:

- Use unique names for different elements (forms, widgets, code modules, images, and skins) within an application. For example, do not use the skin names as any form or variable names in the application. Using the same name in variable names would cause overwriting the skin with the variable. If you follow proper naming convention for different widgets this problem does not arise.
- Make `kony.application.showLoadingScreen` as blocking unless it is specifically required that the user must be able to select the user interface elements while the network call is in progress.
- The `blockUI` parameter in the service calls is applicable only for Mobile Web. For native applications, use `kony.application.showLoadingScreen` API to block the user interface.
- Define a function or global sequence for error handling of the service calls and handle all the basic error codes returned by Kony Application Server.
- Ensure that the widgets that are accessible from code follow a proper naming convention. Retaining the auto generated names makes it difficult to keep track of the widgets and their parent.
- Perform checks for null and empty string before doing any operation on any user input data. This ensures that unexpected results are avoided.
- Use try or catch blocks wherever possible to handle the errors in a proper way
- Define all the global variables at one place so that it is easier to keep a track of them and invalidate them when required.
- Do not declare local variables outside the functions. This guideline must be adhered for native code generation.
- Use different names for global and local variables.

- Write logically related functions (functions related to a specific functionality) within a single module.
- Do the necessary validations before making network calls. For example, check if the text box has data before invoking a service.
- Ensure to write proper code to handle the errors returned by network calls.
- Ensure that the widget ID counter is different for different developers working in the same project as the default ids created by the IDE depend on these and may lead to problems like duplicate IDs.
- Avoid code duplication by writing reusable functions which can be shared across modules or applications.
- Write functions with proper indentation for better debugging and readability (use tab indentation for logical blocks).
- Write smaller functions (maximum with 100 lines of code) instead of huge monolithic spaghetti code.
- Use APIs provided by the Kony Platform wherever possible instead of writing code to achieve the same behavior. The APIs are much faster than the user-written code as the code needs to be interpreted.
- Re-use the skins created by other developers while doing parallel development.
- If a widget is never shown on a specific platform, set the Render flag to false for that platform.
- Do not store the raw bytes captured by the camera (that need to be used globally in the application) in a global variable. Instead, store them in the datastore and retrieve them later.
- Avoid multiple timers, instead use multiple flags with a single timer ticking and different action being performed on the basis of state of various flags.
- Write generalized functions for the tasks that are repetitive across the application.

- Try to avoid form forking unless you require a completely different form for another platform. When you fork the forms, make sure all the widgets in the forked form are present in the parent form in a hidden state. This is because if they are not available in the parent form, you cannot map data to widgets in the forked form using Mapping Editor.
- `kony.ui.alert` must be the last statement in an execution flow as it is not a non-blocking call.
- Prefix the form names with `frm` and pop-ups with `popup`.

2.21.1 Foreign Function Interface (FFI) Guidelines

2.21.1.1 iPhone

- Ensure that the library uses the Object versions of primitive types while writing the wrappers for FFI compatibility. For example, if a number is mentioned in the IDE, the wrapper should accept `NSNumber`.
- Ensure that namespace for FFI does not clash with the names of the header files in the imported zip file. For example, if the namespace is `zxling` then there should not be any header file with the name `zxling.h`.
- Ensure that any dependent frameworks are added as part of the XCode project configuration before building the XCode project.
- Try to include the `.m` and `.h` files (Objective C source code for the third-party library) instead of `.a` files (binary formats) to avoid build time issues in XCode. `.a` files can still be integrated but the developer might run into build issues which might be hard to debug.

2.21.1.2 Android



- The Java Class on which the static method is invoked should have a *public* default constructor:

```
public class Foo
{
    ...public Foo()
    ...{
        ....//default constructor
    ...}
}
```

- Ensure that you pass Activity Context Object as the first parameter to the method. A few Android native APIs require access to the *Activity* class present in the `konywidgets.jar` file. Make sure that this JAR file is located at

<workspace>\temp\<appName>\build\luaandroid\dist\<appName>\libs\ folder.


- Use the static method `com.konylabs.KonyMain.getActContext()` to access the current *Activity* class.

Library :  

Package :

Class:

Constructor*:

Return type* : 

Parameters:

Name	JavaScript Type	Native Type

- You can reference any of the APIs that are already present on the device (including `konywidgets.jar`) in the Java projects.
- Ensure that you do not import these JAR files into Kony Visualizer.

2.21.1.3 BlackBerry

- The Java Class on which the static method is invoked should have a *public* default constructor:

```
public class Foo
{
...public Foo()
...{
.....//default constructor
...}
```

```
}
```

- The library should not have a *main* method. The *main* method is the entry point into the BlackBerry application. Kony Visualizer defines a *main* method during code generation. Another *main* method leads to erroneous behavior.

```
// Below method should not be part of any class in the library
public static void main(String[] args)
```

- Ensure that the Java class files are compiled using the following JAVAC Flags:

```
javac -target "1.1" -source "1.2"
```

- Ensure that resultant JAR file is pre-verified using:

```
C:\Program Files\Research In Motion\

```

- Do not have a class that extends *UiApplication* interface. If you use a class that extends this interface, the class is not integrated with the IDE and the IDE throws a *NoClassDefFoundError*. The possible reasons for this error are:
 - JAR file was not pre-verified
 - Native Wrapper library refers to a class that extends *UiApplication*.
- Do not import the `net_rim_api.jar` file into Kony Visualizer. This JAR file has been provided by BlackBerry only for compiling the BlackBerry Desktop Applications.

Note: All the methods in all the classes of this JAR file are marked as external (dummy).

- Check the BlackBerry API documentation available at `C:\Program Files\Research In Motion\ to check if the API used in the library is supported.`

- You can reference any of the APIs that are already present on the device (including `konywidgets.jar`) in the Java projects. But ensure that you do not import these JAR files into Kony Visualizer.
- Check the `*rapc.log` file (end of the file) located under `<workspace>\temp\<appName>\build\lua2me\blackberry` and `<workspace>\temp\<appName>\build\lua2me\blackberry47` if you get the `exec returned error 97` error while building an application for BlackBerry in the IDE. This error indicates that the generation of COD file was not successful from the application JAR file.

2.22 Performance Management Guidelines

The overall performance of any application is made up of the *actual* and *perceived* performance of application.

Actual performance of the application is driven by the following factors:

- Performance of the dependent services.
- Number of services involved to retrieve the required data.

Perceived performance of the application is driven by the following factors:

- Amount of time it takes to load the first form
- Does the application display "busy/progress indicator" for long running network calls.
- How quickly the application responds to user touches and gestures.

The guidelines in this section should be adhered to improve the overall performance of the application.

- Adopt good design principles to ensure that only UI logic is implemented on the client-side while other data processing and transformation logic should be implemented on the server-side.
- Avoid time consuming logic in the preshow event. Before we delve into the details of why this should be avoided, we need to be aware of some facts.

The preshow event is called on the main thread. Main thread is responsible for:

- Rendering and laying out forms and executing its events, except the post show event. Post show event gets called asynchronously (not on main thread) after presenting the user the desired form.
- Handles all system events such as background, foreground, idle time, `pre-appinit` and `post-appinit` events completely.

Executing any time taking activities in preshow such as network calls etc means, main thread is blocked from doing its operations. This leads to user experience issues as the UI is completely blocked during this execution.

As per mobile user interface guidelines, visually indicate the user for all time taking activities. If `preshow` has time taking activities, there is no means of indicating the user that some activities are happening. Any form level visual clues like popup or using the loading screen API will be effective only after showing the form, but not while executing the `preshow`.

- Any network service call should return within ~4 seconds (6 seconds max). Anything more will make the user feel that application is sluggish.
- Every network call adds up to ~2 seconds to the application response to a user action.
- Use device cache effectively. Attributes which do not change very often (like city list etc) can be a candidate for device storage.
- Avoid multiple network calls (one after another) from the device, use composite services where ever possible.
- Redesign forms which require data to be aggregated from more than 4 network services (at least let the customer know about the performance impact).
- Ask the services provider (the customer) to provide an aggregated service.
- Application launch (the time between the user's click on the application icon and the time the user lands on the home screen of the application) within ~5-6 seconds (needless to say lesser is better but anything above this limit may trigger comments in application feedback on app store).
- If the home screen requires a mandatory set of data that needs to be retrieved, take the user to another form and show the "Loading indicator" (instead of showing him the splash screen till the service call completes)
- Including long running call (network or local device store) as part of `preshow`, `onhide`, `onbackground`, and `onforeground` events can delay the rendering of the new form giving a "jerky" feeling to the end user.

- Avoid function calls or expressions (which give the same value in the loops) as these will slow down the loop. For example

```
a = 10;
b = 20;
i = 0;
while(true)
...{
.....if (sum(a,b) < i)
.....break;
.....i++;
...}
```

//The right way is

```
a = 10;
b = 20;
c = sum(a,b)
i = 0;
while(true)
...{
.....if (c < i)
.....break;
.....i++;
...}
```

2.23 Memory Management Guidelines

The memory available for the application depends on the device specification and other applications running in the background. Global variables and UI widgets consume the application memory. UI widgets include metadata (padding, margin, layout info, and so on), skin information, the data to be shown to the user, function definitions, and resources (Images , i18n text).

2.23.1 `form.destroy`

- Use `form.destroy` effectively.
- `form.destroy` destroys the following:
 - Data and views associated with the form.
 - Data and views associated with the child widgets hierarchy in the form.
 - After a form is destroyed, it returns to the pre-initialized state (a state similar to that when the application was first loaded).
 - Accessing properties, data, and child widgets on a destroyed form will initiate the form life cycle and puts the form in initialized state again. So, as a guideline, the developer should access the form's data or its widgets as required (Lazy access).
 - Module can be any logical group of forms that represents one of the application features.
- When user navigates to one application module to other application module, all the heavy forms of the old module can be destroyed.
- The forms like Terms & Condition, Maps, Help can be good candidates for `form.destroy()`.
- Delete the current form only when user is moving back from the current form. Do not delete the current form in the forward flow.

- For example, in a flow F1->F2->F3->F4->F5, when moving from F4 to F3, F4 can be destroyed if F4 is heavy in terms of the data and widgets. But do not remove F4 when moving to F5. Deleting the current form in forward flow makes the behavior undefined when user navigates back (at least in platforms like iPhone).
- Override the `onback` or `onescape` events to destroy the form when moving away from it.
- Avoid using placeholder widgets (like empty labels, boxes, or transparent images) for achieving spacing between the widgets. Instead, use paddings/margins and alignment attributes to achieve the same.

2.23.2 Global Variables

Global variables occupy device runtime memory for the lifetime of the application. They are released as soon as the application exits. Global variables should be used to store the application state.

- Do not use `ds.save`/`ds.read` or *hidden* widgets as replacement for global variables.
 - Widgets have lot of metadata associated with them and consume a lot more memory to hold the same amount of data than a global variable. `ds.save` and `ds.read` result in disk I/O and definitely has much more performance impact than accessing an in memory variable.
- Nullify the global variables as soon as the data they are holding is not needed.

```
tempglobavariabale = nil;
```

2.24 Device-side Security Guidelines

This section explains a few guidelines to be followed for Device-side security.

- For financial applications, all service calls should be marked as *secure* unless there is a specific requirement to use non-secure calls (by selecting *Secure* checkbox in service call invoker for Native Applications or the *Secure* property in the form for Mobile Web applications).
- Avoid storing personal information or sensitive data in the device datastore. In case, the application demands such information to be stored in the device, appropriate (TripleDes, RSA PKI, and other cryptography APIs) encryption mechanism should be adopted.
- Enable basic authentication on the Kony Middleware Server environment to avoid unauthorized access. An option is provided by Kony Platform for the service calls between device and Kony Application Server.
- Enable *SecureSubmit* property for all the forms where the user needs to enter sensitive data.

2.25 Pre/Post Processors and URL Provider Guidelines

This section explains a few guidelines to be followed while using Pre/Post processors and URL provider.

- Use `log4j` debugging facilities instead of `System.out.println` or other custom framework.
- Trim all user inputs before passing to services to remove any inadvertent trailing spaces. This can be done using a pre-processor.
- Write most of the processing and business logic at the server side either in pre/post processors.
- Check for `opstatus` (success or failure) value in the post-processor before processing any data.
- Perform all the formatting, conversions, sorting, and so on in the post-processor to keep the UI code simple.

2.25.1 Logging (Performance)

When using log4j in the pre/post-processors and URL Provider, check for the debug level before calling `logger.debug` or `logger.info`. See the code given below for more clarity.

Sample Usage

```
package com.kony.custom;

public class CustomPostProcessor implements PostProcessor {

    //Define logger as static final to improve performance
    private static final Logger logger= Logger.getLogger
    (com.kony.custom.CustomPostProcessor .class);

    //Declare the check for debug or info enabled as static final to improve
    performance
    private static final boolean isDebugEnabled =
    logger.isDebugEnabled();
    private static final boolean isInfoEnabled = logger.isInfoEnabled
    ();

    public Result execute (Result result, DataControllerRequest
    Request) {

        //Wrap logger.debug with isDebugEnabled to improve performance
        if (isDebugEnabled) {
            logger.debug("This is a debug statement");
        }

        //Wrap logger.info with isInfoEnabled to improve performance
        if (isInfoEnabled) {
            logger.info("This is an info statement");
        }
    }
}
```

```

    }
}

```

If the package name of your custom class does not start with `com.kony` or `com.konylabs` then add the package entry (`log4j.category.custompackagename`) in `middleware-log4j.properties`.

2.25.2 Class Variables

Pre-processor, post-processor, and URLProvider are singleton classes. Any user specific data should not be stored in class variables, otherwise it will result in abnormal behavior. Common Data can be stored in class variables.

2.25.3 Input Validation

Pre-processor must be used to validate all the inputs to overcome Denial-of-Service attacks. Even though the client side validations are performed, the server side validations are mandatory.

For example, user input must be validated at client and server also. Any data selection from collection widgets like ComboBox, ListBox, Segment, RadioButton, and CheckBoxGroup, and so on must have server validation.

Sample Usage

```

package com.kony.custom;

public class CustomPreProcessor implements PreProcessor {
    public boolean execute (HashMap map, DataControllerRequest
request, Result result) {
        if (customValidation()) {
        } else {
            result.addParam(new Param("opstatus", "23", "int" ));
            result.addParam(new Param("errormsg", "Invalid Input",
"string" ));
            return false;
        }
    }
}

```

```

        return true;
    }
}

```

2.25.4 Session Recycle

If user navigates from non-authenticated to authenticated state or authenticated to non-authenticated state, then the session needs to be recycled to avoid Man-in-Middle Attack. Session recycle will change JSESSIONID or cacheid value.

Sample Usage

```

package com.kony.custom;

public class CustomPostProcessor implements PostProcessor {
    public Result execute (Result result, DataControllerRequest
Request) {
        //get Session Object
        Session session = request.getSession(false);
        HashMap datamap = new HashMap(); //create temp data map
        //populate session data into temp data map
        if (session != null) {
            try {
                Enumeration enumeration = session.getAttributeNames();
                while (enumeration.hasMoreElements()) { String sessionKey = (String)
enumeration.nextElement(); Object sessionValue = session.getAttribute
(sessionKey); datamap.put(sessionKey, sessionValue);
            } catch (Exception e) {
                //log error
            }
        }
        //invalidate session
        session.invalidate();
    }
    //create new session
}

```

```

    session = request.getSession(true);
    // populate the data to new session
    if (session != null) {
        for (Iterator itr = datamap.keySet().iterator(); itr.hasNext(); ) {
            String sessionKey = (String) itr.next();
            Object sessionValue = datamap.get(sessionKey);
            session.setAttribute(sessionKey, sessionValue);
        }
    }

    //clear the temp data map
    datamap.clear();
    datamap = null;
}
}

```

2.25.5 Sensitive Data Logging

Do not log any sensitive data like userid, password etc in debug, info or error mode.

Sample Usage

```

package com.kony.custom;

public class CustomPreProcessor implements PreProcessor {

    public boolean execute (HashMap map, DataControllerRequest request, Result result) {

        //Do not log sensitive information
        logger.debug("userid:" + map.get("userid"));
        logger.debug("password:" + map.get("password"));
    }
}

```

2.25.6 Duping Authentication

There is a possibility from the device or man-in-middle to dupe the server that authentication is successful even though authentication has failed. If proper validations are not added, user can access the sensitive information. To overcome this problem, maintain a flag in session for successful authentication at server and also clear the flag in session after successful logout or sign-out.

Sample Usage

```
package com.kony.custom;

public class LoginPostProcessor implements PostProcessor {
    public Result execute (Result result, DataControllerRequest
Request) {
        //get Session Object
        Session session = request.getSession(false);
        //check for login is successful and then add a flag in
session
        session.setAttribute("login", true);
    }
}

package com.kony.custom;

public class AccountDetailPreProcessor implements PreProcessor {
    public boolean execute (HashMap map, DataControllerRequest
request, Result result) {
        //get Session Object
        Session session = request.getSession(false);
        // get the Flag
        boolean flag = (Boolean)session.getAttribute("login");
        if (! flag) {
            //sent error response to client
            result.addParam(new Param("opstatus", "23", "int" ));
        }
    }
}
```

```

        result.addParam(new Param("errmsg", "User is not logged
in", "string" ));
        return false;
    }
    return true;
}
}

package com.kony.custom;
public class LogoutPostProcessor implements PostProcessor {
    public Result execute (Result result, DataControllerRequest
Request) {
        //get Session Object
        Session session = request.getSession(false);
        // check for logout is successful and then remove the flag
from session
        session.removeAttribute("login");
    }
}
}

```

2.25.7 Sensitive Data Encryption

Asymmetric keys or Public-private keys (PKI) must be used to encrypt the sensitive data. For Symmetric keys, changing the seed will be a maintenance issue.

2.25.8 Invoking Services

In some cases, order of invoking services is must. For example, service-2 can be invoked after successful invocation of service-1. Validation must be captured in service-2 pr processor to ensure service-1 is invoked.

2.25.9 Post-Processor Execution in Case of Exception

If an exception occurs while executing a service, you do not have control to handle the exception. To handle the exception, a new feature `@OnException` is added to update the result object with the help of post-processor. Also provided the flexibility to write your own business logic based on the requirement like a positive scenario.

Note: The method signature is similar to `execute()`. Subsequently, all existing positive scenario cases can also be implemented for the exception case.

To implement this feature, add a new method with `@OnException` annotation as shown below.

```
@OnException
public Result yourMethodName (@KonyContext Result result,
    @KonyContext DataControllerRequest request, @KonyContext
    DataControllerResponse response,
    @KonyException Exception exception) {
    If (exception instanceof ConnectorException) {
        result.addParam(new Param("opstatus", "2323", "int" ));
        result.addParam(new Param("errmsg", "Socket timeout while connecting
        to endpoint", "string" ));
    }
    return result;
}
```

`@OnException` - annotation to identify the exception API in post-processor

`@KonyContext` - annotation for the parameters in case of exception

`@KonyException` - annotation for an exception parameter in case of exception

Example

For a JSON service, if the response is not JSON type, the middleware sends a 5001 error code . For an XML service, if the response is invalid, the middleware sends an 8006 error code. If the HTML comes from a service, the response can be parsed using `@OnException`. Based on the code, you have to write the business logic to parse the HTML or modify the result object.

The sample code to parse the HTML using `@OnException` follows:

```
@OnException
public Result processHTMLResponse (@KonyContextwe Result result,
@KonyContext
DataControllerRequest request, @KonyContext DataControllerResponse
response,
@KonyException Exception exception) {
    if (exception instanceof ConnectorException){
        String errorCode =
((ConnectorException)exception).getErrorCode();
        if("5001".equals(errorCode) || "777777".equals(errorCode)){
            result.setParam(new Param("RELOGIN", "true", "string"));
            String htmlString = response.getResponse();
            Document doc = Jsoup.parse(htmlString);
            String smPostPreserveValue =
            doc.select("INPUT[NAME=SMPostPreserve]").get(0).attr("VALUE");
            if(smPostPreserveValue != null &&
            !"".equalsIgnoreCase(smPostPreserveValue.trim())){
                result.setParam(new Param("SMPostPreserve",
smPostPreserveValue, "string"));
            }
        }
    }
    return result;
}
```

For the preceding HTML parsing, we used `jsoup-1.8.1.jar`.

3. Application Development Guidelines

This document describes the Application Development Guidelines within the Kony Visualizer. This topic will describe Coding standards and basics of JavaScript. This document is targeted at designers and developers who want to have an understanding of the JavaScript coding and are familiar with the Kony Visualizer.

In this section you will learn:

1. [Coding Standards](#)
2. [Basics of JavaScript](#)

3.1 Coding Standards

- JavaScript is case sensitive.
- Do not use JavaScript reserved keywords for variable names.
- Properties and methods:
 - Private properties, variables, and methods (in files or classes) should be named with a trailing underscore.
 - Protected properties, variables, and methods should be named without a trailing underscore (like public ones).
- Method and function parameters:
 - Optional function arguments start with `opt_`.
 - Functions that take a variable number of arguments should have the last argument named `var_args`. You may not refer to `var_args` in the code; use the arguments array.
 - Optional and variable arguments can also be specified in `@param` annotations. Although either convention is acceptable to the compiler, using both together is preferred.

- Getters and Setters

- Getters and Setters for properties are discouraged. However, if they are used, then getters must not change observable state.

```
/**
 * WRONG -- Do NOT do this.
 */
var foo = { get next() { return this.nextId++; } };
};
```

- Accessory functions

- Getters and setters methods for properties are not required. However, if they are used, then getters must be named `getFoo()` and setters must be named `setFoo(value)`. (For Boolean getters, `isFoo()` is also acceptable, and often sounds more natural.)

- Namespaces

- JavaScript has no inherent packaging or name-spacing support.
- Global name conflicts are difficult to debug, and can cause intractable problems when two projects try to integrate. In order to make it possible to share common JavaScript code, we've adopted conventions to prevent collisions.

- Use namespaces for global code

- Always prefix identifiers in the global scope with a unique pseudo namespace related to the project or library. If you are working on "Project Sloth", a reasonable pseudo namespace would be `sloth.*`.

```
var sloth = {};
sloth.sleep = function() {
  ...
};
```

Many JavaScript libraries, including the Closure library and Dojo toolkit give you high-

level functions for declaring your namespaces. Be consistent about how you declare your namespaces.

```
goog.provide('sloth');
sloth.sleep = function() {
  ...
};
```

- Respect namespace ownership
 - When choosing a child-namespace, make sure that the owners of the parent namespace know what you are doing. If you start a project that creates hats for sloths, make sure that the Sloth team knows that you're using `sloth.hats`.
- Use different namespaces for external code and internal code
 - External code is the code that comes from outside your codebase, and is compiled independently.
 - Internal and external names should be kept strictly separate. If you're using an external library that makes things available in `foo.hats.*`, your internal code should not define all its symbols in `foo.hats.*`.

```
foo.require('foo.hats');
/**
 * WRONG -- Do NOT do this.
 * @constructor
 * @extend {foo.hats.RoundHat}
 */
foo.hats.BowlerHat = function() {
};
```

- If you need to define new APIs on an external namespace, you should explicitly export the public API functions, and only those functions.

- Your internal code should call the internal APIs by their internal names, for consistency and so that the compiler can optimize them better.

```
foo.provide('googleyhats.BowlerHat');
foo.require('foo.hats');
/**
 * @constructor
 * @extend {foo.hats.RoundHat}
 */
googleyhats.BowlerHat = function() {
  ...
};
goog.exportSymbol('foo.hats.BowlerHat',
googleyhats.BowlerHat)
```

- Use different namespaces for external code and internal code
 - External code is the code that comes from outside your codebase, and is compiled independently.
 - Internal and external names should be kept strictly separate. If you're using an external library that makes things available in `foo.hats.*`, your internal code should not define all its symbols in `foo.hats.*`.

```
foo.require('foo.hats');
/**
 * WRONG -- Do NOT do this.
 * @constructor
 * @extend {foo.hats.RoundHat}
 */
foo.hats.BowlerHat = function() {
  };
```

- If you need to define new APIs on an external namespace, you should explicitly export the public API functions, and only those functions.
- Your internal code should call the internal APIs by their internal names, for consistency and so that the compiler can optimize them better.

```
foo.provide('googleyhats.BowlerHat');
foo.require('foo.hats');
/**
 * @constructor
 * @extend {foo.hats.RoundHat}
 */
googleyhats.BowlerHat = function() {
  ...
};
goog.exportSymbol('foo.hats.BowlerHat',
googleyhats.BowlerHat)
```

- Alias long type names to improve readability
 - Use local aliases for fully-qualified types if doing so improves readability. The name of a local alias should match the last part of the type.

```
/**
 * @constructor
 */
some.long.namespace.MyClass = function() {
};
/**
 * @param {some.long.namespace.MyClass} a
 */
some.long.namespace.MyClass.staticHelper = function(a) {
  ...
}
```

```
};  
myapp.main = function() {  
  var MyClass = some.long.namespace.MyClass;  
  var staticHelper =  
  some.long.namespace.MyClass.staticHelper;  
  staticHelper(new MyClass());  
};
```

- **Do not alias namespaces.**

```
myapp.main = function() {  
  var namespace = some.long.namespace;  
  namespace.MyClass.staticHelper(new namespace.MyClass());  
};
```

Avoid accessing properties of an aliased type, unless it is an enum.

```
/** @enum {string} */  
some.long.namespace.Fruit = {  
  APPLE: 'a',  
  BANANA: 'b'  
};  
myapp.main = function() {  
  var Fruit = some.long.namespace.Fruit;  
  switch (fruit) {  
  case Fruit.APPLE:  
    ...  
  case Fruit.BANANA:  
    ...  
  }  
};
```



```
myapp.main = function() {  
  var MyClass = some.long.namespace.MyClass;  
  MyClass.staticHelper(null);  
};
```

- Never create aliases in the global scope. Use them only in function blocks.
- Alias long type names to improve readability
 - Use local aliases for fully-qualified types if doing so improves readability. The name of a local alias should match the last part of the type.

```
/**  
 * @constructor  
 */  
some.long.namespace.MyClass = function() {  
};  
/**  
 * @param {some.long.namespace.MyClass} a  
 */  
some.long.namespace.MyClass.staticHelper = function(a) {  
  ...  
};  
myapp.main = function() {  
  var MyClass = some.long.namespace.MyClass;  
  var staticHelper =  
  some.long.namespace.MyClass.staticHelper;  
  staticHelper(new MyClass());  
};
```

- Do not alias namespaces.

```
myapp.main = function() {
  var namespace = some.long.namespace;
  namespace.MyClass.staticHelper(new namespace.MyClass());
};
```

Avoid accessing properties of an aliased type, unless it is an enum.

```
/** @enum {string} */
some.long.namespace.Fruit = {
  APPLE: 'a',
  BANANA: 'b'
};
```

```
myapp.main = function() {
  var Fruit = some.long.namespace.Fruit;
  switch (fruit) {
  case Fruit.APPLE:
    ...
  case Fruit.BANANA:
    ...
  }
};
```

```
myapp.main = function() {
  var MyClass = some.long.namespace.MyClass;
  MyClass.staticHelper(null);
};
```

- Never create aliases in the global scope. Use them only in function blocks.

- Alias long type names to improve readability
 - Use local aliases for fully-qualified types if doing so improves readability. The name of a local alias should match the last part of the type.

```
/**
 * @constructor
 */
some.long.namespace.MyClass = function() {
};
/**
 * @param {some.long.namespace.MyClass} a
 */
some.long.namespace.MyClass.staticHelper = function(a) {
...
};
myapp.main = function() {
var MyClass = some.long.namespace.MyClass;
var staticHelper =
some.long.namespace.MyClass.staticHelper;
staticHelper(new MyClass());
};
```

- Do not alias namespaces.

```
myapp.main = function() {
var namespace = some.long.namespace;
namespace.MyClass.staticHelper(new namespace.MyClass());
};
Avoid accessing properties of an aliased type, unless it is
an enum.
/** @enum {string} */
some.long.namespace.Fruit = {
```

```
APPLE: 'a',  
BANANA: 'b'  
};  
myapp.main = function() {  
  var Fruit = some.long.namespace.Fruit;  
  switch (fruit) {  
    case Fruit.APPLE:  
      ...  
    case Fruit.BANANA:  
      ...  
  }  
};
```

```
myapp.main = function() {  
  var MyClass = some.long.namespace.MyClass;  
  MyClass.staticHelper(null);  
};
```

- Never create aliases in the global scope. Use them only in function blocks.

3.2 Usage of Global and Local Variables

- A variable declared (using var) within a JavaScript function becomes local and can only be accessed from within that function.
- If you fail to specify “var”, the variable gets placed in the global context, potentially clobbering existing values. Also, if there is no declaration, it is hard to tell in what scope a variable lives (for example: It could be in the Document or Window just as easily as in the local scope). So always declare with “var”.
- Use locals rather than globals whenever possible.

```
//var x = 0 - do not declare variables outside functions
//unless required.
function count(){
var x = 0
x = x + 1
kony.print(x)
}
```

- Global variables have application scope and they are not garbage collected unless explicitly cleared or application exits. Set global variables to null after the usage is completed (equivalent to “null” in Java)

```
gblTempVar = null;
```

- Widgets have a lot of metadata associated with them and consume a lot more memory to hold the same amount of data than a global variable.
 - For example, `kony.store.setItem` and `kony.store.getItem` result in disk I/O and definitely has much more performance impact than accessing an in memory variable.
 - Do not use `setItem` / `getItem` or hidden widgets as replacement for global variables.
- Define all the global variables using the global variables section of services tab so that it is easier to keep a track of them.
 - If you want to read a key from a JSON object more than once, it is better to read that key in a local variable then use it.
 - Have a look at the following code snippet:

```
function validate(){
if (v["headerdesc2"] ~= ""){
innerTable["desc2"] = v["headerdesc2"]
}else{
innerTable["desc2"] = ""
}
}
```

```

}
Better Approach -
function validate(){
var headerValue = v["headerdesc2"]
if (headerValue ~= null and headerValue ~= "" ){
innerTable["desc2"] = headerValue
}else{
innerTable["desc2"] = ""
}
}
}

```

3.3 File Names

The following file naming conventions must be followed as best practices:

- File names should be all lowercase in order to avoid confusion on case-sensitive platforms.
- File names should end in .js, and should contain no punctuation except for - or _ (prefer - to _).

3.3.1 Readability

- Use Javadoc-like style for commenting.

Example 1:

```
--[[
```

```
*****
```

```

*   Name      : validateAddress
*   Name      : validateAddress
*   Author    : Author name
*   Date      : 27/05/2010
*   Purpose   : Deletes the session
*   Input     : @param id Session identification.

```

```
*****  
-]]  
  
function delete (id)  
    remove (filename (id))  
end
```

- Provide a comment whenever a construct ends.

```
for i,v in ipairs(t) do  
    if type(v) == "string" then  
        ...lots of code here...  
    end -- if  
end -- for
```

- Use proper indentation for debugging.
- Use tab indentation for logical blocks.

```
function makeARemoteDeposit()  
    local rdcStore = globalStore[ "RDC_STORE" ]  
    local amount = frmRDCRecap.txtrecapdepamtvalue.text  
    if(string.len(amount) ==0) then  
        widget.setvisibility  
(frmRDCRecap.hboxErrorlist,true)  
        return false  
    end  
end
```

- Write proper documentation and comments for each function/module and for typical logic in functions

- All wrapped lines must be indented either left to the expression or above or indented four spaces, except for array and object initializations and passing anonymous functions.

Note: Two space indentation must not be used.

```
function sample {
  someWonderfulHtml = '' +
  getEvenMoreHtml(val1, val2,
  evenMoreParams, 'a duck',
  true, 72,
  slightlyMoreMonkeys(0xff)) +
  '';
  if (searchableColl(allYourStuff).contains(theStuffYouWant) &&
  !ambientNotification.isActive() &&
  (client.isAmbientSupported() ||
  client.alwaysTryAmbientAnyways())) {
    ambientNotification.activate();
  }
}
```

3.3.2 Common mistakes

Following is a list of common mistakes to take care of:

- Do not have cyclic references between JSON objects (Two JSON objects having reference to each other) as it makes garbage collection difficult.
- Perform checks for null (for example: variables, mapping segment data etc) and empty string always before doing any operation on any user input data. This ensures that unexpected results / app crashes are avoided.


```
var myData = kony.store.getItem("friends")
    if(myData ~= null) {
        // do something
    }
```

```
local myData = ds.read("friends")
    if(myData ~= nil) then
        -- do something
    end
```

- Do not just define strings to null. Initialize strings to either "" or ". If you initialize strings to null, JavaScript will treat it as string value "null" and all the operations performed on this variable will get effected.

```
var name;
var x = null;

x = x + " Bentley Azure"
document.write(x);          //null Bentley Azure
document.write(name);      // undefined

results in
null Bentley Azure
undefined
```

- When validating the input, be careful with blank spaces and trim the string.

```
var holdername = string.trim(frmmCrdtCard.txtcardname.text)
    if holdername == null or holdername.length == 0 then
        displayAlertMsg("", "Please enter card holder
name.", frmmCrdtCard)
        clearCardNoAndSecCode(frmmCrdtCard)
```

```
        return false
    end
```

Better approach: *kony.string.trim()* will delete the leading and trailing blank spaces of the string.

```
var holdername = string.trim(frmmCrdtCard.txtcardname.text)
if holdername == null or holdername.length == 0 then
    showAlertMsg("", "Please enter card holder
name.", frmmCrdtCard) clearCardNoAndSecCode(frmmCrdtCard)
return false
end
```

- Have a look at this code snippet:

```
var termsDetails = {}
    for(v in resulttable["termsset"]){
        var innerTable = { termhdr =
            resulttable["termsset"][v].heading,
resulttable["termsset"][v].p1}
        if(resulttable["termsset"][v].p2 ~= ""){
            innerTable["para2"] =
resulttable["termsset"][v].p2
        }else{
            innerTable["para2"] = ""
        }
        .
        .
        if (resulttable["termsset"][v].p16 ~= "" ){
            innerTable["para16"] =
resulttable["termsset"][v].p16
        }else{
```

```
        innerTable["para16"] = ""
    }
    termsDetails.push(innerTable)
}
frmTktTermsCond. segTerms .setdata(termsDetails)
```

Better Approach: For improving the readability and code compactness, you can write the above code as a lengthy one. Following is an example of lengthy code:

```
var termsDetails = {};
var termSetList = resulttable["termsset"];
for (v in termSetList) {
    var termSet = termSetList[v];
    var innerTable = {termhdr = termSet.heading}
    for (var idx=1; idx <= 16; idx++){
        var rsKeyVal = termSet["p" .. idx]
        var seguiKey = "para" .. idx
        if(rsKeyVal ~= null and
rsKeyVal.length ~= 0) {
            innerTable[seguKey] = rsKeyVal
        }else{
            innerTable[seguKey] = ""
        }
    }
    termsDetails.insert(innerTable)
}
frmTktTermsCond.segTerms.setdata(termsDetails)
```

- If the function returns a value, do not have multiple return statements. Instead, build logic in such a way that you have only one return at the end of the function.

```
function getDeepLink(table){
    var finalForm = frmLanding
    if (table ~= null and table.length ~= 0){
    if (table.showscreen ~= null and table.showscreen ~= ""){
        for (v in table) {
            if (v == "cars" ) {
                finalForm = frmCarSearch
            }else{
                if (v == "hotels" )
                    finalForm = frmHotelSearch
            }
        }
    }
    }
    return finalForm
}
```

- If you are calling *kony.os.toNumber()*, ensure that the string you are passing to the function is a number string that is "5", "3443". Else you will get run-time exceptions.

```
Example: kony.os.toNumber("34B") --> runtime exception
kony.os.toNumber("55") --> 55 (number is obtained!)
```

- **Best practice:** To avoid run-time exceptions, always use the logic as explained in the following code:

```
var inputstring = "34B"
if(kony.string.isNumeric(inputstring){
    var inputNum = kony.os.toNumber(inputstring);
    -- now proceed with functionality
}
```

- Avoid writing code outside function blocks. Since Kony platform does not guarantee a specific order of loading JavaScript files, code written outside function blocks can lead to undefined errors.
 - Write code in function blocks and invoke the same through one of the application initialization callbacks (preappinit , appinit etc).
- The name of the form and the module file should not be the same. Since all the files are flattened when packed into a deployable binary, name clashes should be avoided.

3.3.3 General Guidelines

The following general guidelines must be followed as best practices:

- Write logically related functions (functions related to a specific functionality) within a single module. For example, write all the account summary related functions within accSummary.lua or accSummary.js module.
- Do the necessary validations in Lua JavaScript before making network calls. For example, check if the text box has data before invoking a service.
- Avoid code duplication by writing reusable functions which can be shared across modules/applications.
- Write smaller functions (maximum with hundred lines of code) instead of huge monolithic spaghetti code.
- Write generalized functions for the tasks that are repetitive across the application.
- Avoid function calls or expressions (which give the same value in the loops) as these will slow down the loop.

For example:

```
a = 10;
```

```
b = 20;
i = 0;
while(true)
...{
.....if (sum(a,b) < i)
.....break;
.....i++;
...}
//The right way is
a = 10;
b = 20;
c = sum(a,b)
i = 0;
while(true)
...{
.....if (c < i)
.....break;
.....i++;
...}
```

- As a good practice, keep the name of input and output variables in either lowercase or uppercase. Do not mix both the cases. As Lua JavaScript are case sensitive, this helps in making the development easy and also minimizes the errors.

3.4 Basics of JavaScript

Following are the Basics of JavaScript.

3.4.1 Function declaration within blocks

While most of the script engines support Function Declarations within blocks, it is not part of ECMAScript. Worse implementations are inconsistent with each other and with future ECMAScript proposals. ECMAScript only allows for Function Declarations in the root statement list of a script or function. Instead use a variable initialized with a Function Expression to define a function within a block.

Example:

```
if (x) {  
    var foo = function() {}  
}  
Do not do as below:  
if (x) {  
    function foo() {}  
}
```

If a widget is never shown on a specific platform, set the Render flag to false for that specific platform. Also, do not access that particular widget in that specific platform as it results in a null pointer exception.

3.4.2 Standard features

For maximum portability and compatibility, always prefer standards features over non-standard features (for example: *string.charAt(3)* over *string[3]* and element access with DOM functions instead of using an application-specific shorthand).

3.4.3 Closures

ECMAScript (JavaScript) allows inner functions; function definitions and function expressions that are inside the function bodies of other functions. And those inner functions are allowed access to all of the local variables, parameters and declared inner functions within their outer functions.

A closure is formed when one of those inner functions is made accessible outside of the function in which it was contained, so that it may be executed after the outer function has returned. At which point it still has access to the local variables, parameters and inner function declarations of its outer function. Those local variables, parameters and function declarations (initially) have the values that they had when the outer function returned and may be interacted with by the inner function.

One thing to keep in mind, however, is that a closure keeps a pointer to its enclosing scope. As a result, attaching a closure to a DOM element can create a circular reference and thus, a memory leak. For example, observe the following code:

```
function foo(element, a, b) {  
  element.onclick = function() { /* uses a and b */ };  
}
```

The function closure keeps a reference to element, a, and b even if it never uses element. Since element also keeps a reference to the closure, we have a cycle that would not be cleaned up by garbage collection. In these situations, the code can be structured as follows:

```
function foo(element, a, b) {  
  element.onclick = bar(a, b);  
}  
  
function bar(a, b) {  
  return function() { /* uses a and b */ }  
}
```

3.4.4 with() {}

Using “with” clouds the semantics of your program. Because the object of the “with” can have properties that collide with local variables, it can drastically change the meaning of your program. For example, check the following piece of code:

```
with (foo) {  
  var x = 3;
```



```
return x;
}
```

The local variable `x` could be clobbered by a property of `foo` and perhaps it even has a setter, in which case assigning `3` could cause lots of other code to execute. Hence don't use `with`.

3.4.5 for-in loop

Recommended only for iterating over keys in an object/map/hash.

The 'for-in' loops are often incorrectly used to loop over the elements in an Array. This is however very error prone because it does not loop from 0 to length - 1 but overall the present keys in the object and its prototype chain. Here are a few cases where it fails:

```
function printArray(arr) {
  for (var key in arr) {
    print(arr[key]);
  }
}

printArray([0,1,2,3]); // This works.
var a = new Array(10);
printArray(a); // This is wrong.
a = document.getElementsByTagName('*');
printArray(a); // This is wrong.
a = [0,1,2,3];
a.buhu = 'wine';
printArray(a); // This is wrong again.
a = new Array;
a[3] = 3;
printArray(a); // This is wrong again.
```

Always use normal for loops when using arrays.

```
function printArray(arr) {  
  var l = arr.length;  
  for (var i = 0; i < l; i++) {  
    print(arr[i]);  
  }  
}
```

3.4.6 Associative Arrays

Never use Array as a map/hash/associative array.

Associative Arrays are not allowed... more precisely you are not allowed to use non-number indexes for arrays. If you need a map/hash use Object instead of Array in these cases because the features that you want are actually features of Object and not of Array. Array just happens to extend Object (like any other object in JAVASCRIPT and therefore you might as well have used Date, RegExp or String).

3.4.7 Multiline string literals

The following code explains bad string literals.

```
var myString = 'A rather long string of English text, an error  
message \  
actually that just keeps going and going -- an error \  
message to make the Energizer bunny blush (right through \  
those Schwarzenegger shades)! Where was I? Oh yes, \  
you\'ve got an error and all the extraneous whitespace is \  
just gravy.  Have a nice day.';
```

The white space at the beginning of each line cannot be safely stripped at compile time; white space after the slash will result in tricky errors; and while most script engines support this, it is not part of ECMAScript.

Use string concatenation instead:

```
var myString = 'A rather long string of English text, an error
message ' +
'actually that just keeps going and going -- an error ' +
'message to make the Energizer bunny blush (right through ' +
'those Schwarzenegger shades)! Where was I? Oh yes, ' +
'you\'ve got an error and all the extraneous whitespace is ' +
'just gravy. Have a nice day.';
```

3.4.8 Array and Object literals

Use Array and Object literals instead of Array and Object constructors.

Array constructors are error-prone due to their arguments.

```
// Length is 3.
var a1 = new Array(x1, x2, x3);
// Length is 2.
var a2 = new Array(x1, x2);
// If x1 is a number and it is a natural number the length will be
x1.
// If x1 is a number but not a natural number this will throw an
exception.
// Otherwise the array will have one element with x1 as its value.
var a3 = new Array(x1);
// Length is 0.
var a4 = new Array();
```

Because of this, if someone changes the code to pass one argument instead of two arguments, the array might not have the expected length. To avoid these kinds of cases, always use the more readable array literal.

```
var a = [x1, x2, x3];
var a2 = [x1, x2];
```

```
var a3 = [x1];  
var a4 = [];
```

Object constructors don't have the same problems, but for readability and consistency object literals should be used.

```
var o = new Object();  
var o2 = new Object();  
o2.a = 0;  
o2.b = 1;  
o2.c = 2;  
o2['strange key'] = 3;
```

Should be written as:

```
var o = {};  
var o2 = {  
  a: 0,  
  b: 1,  
  c: 2,  
  'strange key': 3  
};
```

3.4.9 Eval()

Do not use eval() function as it can lead to security issues in the form of code injections.

4. Accessibility (508 Compliance)

Accessibility on the Web refers to the best practice of providing equal access and equal opportunity to people with diverse disabilities. This chapter provides the concepts of accessibility and implementation it through Kony Visualizer. This chapter explains the following topics:

- [Navigation Keys](#)
- [Achieving 508 Compliance Using Kony Visualizer](#)
- [Widget Navigation Model and Tab/Focus Order](#)

What is 508 Compliance?

In software application development, 508 Compliance is also referred to as accessibility. For example, people with impaired vision must be able to use software with the help of touch gestures, that is designed to run on a system that has a keyboard. The result of performing a function is read out using the screen reading technology.

The assistive technology (AT) for iPhone and Android platforms has built-in programs that support accessibility when enabled on devices. Browser-based platforms use the Web Accessibility Initiative - Accessible Rich Internet Applications (WAI-ARIA) framework. The WAI-ARIA framework enables you to add attributes to identify features for user interaction, map controls, events, and APIs used in a rich Internet application.

The table below shows different platforms and their assistive technology:

Platform	Assistive Technology
iOS	VoiceOver
Android	TalkBack
SPA	WAI-ARIA

Note: The screen-reading capabilities of the different assistive technologies may vary and may not produce the same results.

Navigation Keys

When a device enables assistive technology, visually impaired users typically navigate through the UI controls such as tab/enter/arrow keys/page up/down keys. On various touch-only devices, a few of these key actions are mapped to touchscreen finger gestures.

The chart below shows how keyboard-based navigation keys are mapped to gestures on mobile platforms:

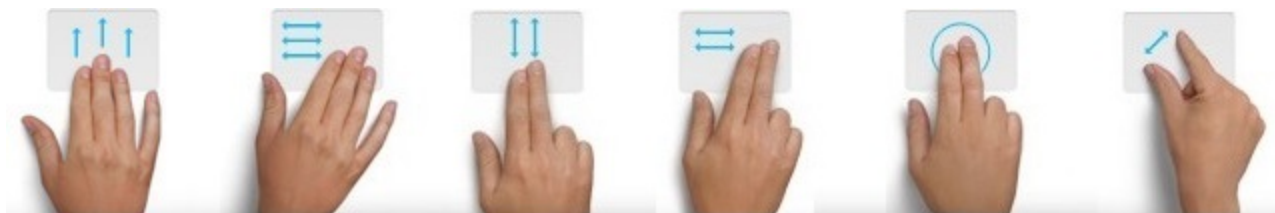
Note: A gesture may function differently in different assistive technologies. Refer to the [respective assistive technology](#) documentation for more information on gestures.

Keyboard based devices (desktop)	Purpose	Android touch (Android 4.1 and higher) (talkback)	iOS touch
Tab	To move focus in forward direction	One finger right/down flick gesture	One finger right flick gesture
Shift+Tab	To move focus in reverse direction	One finger left/up flick gesture	One finger left flick gesture
Enter /Space	To take action on the focused widget	One finger double tap	One finger double tap
Right Arrow/Up Arrow	To increase the value selection on specific widgets like slider/picker		One finger up flick gesture

Keyboard based devices (desktop)	Purpose	Android touch (Android 4.1 and higher) (talkback)	iOS touch
Left Arrow/Up Arrow	To decrease the value selection on specific widgets like slider/picker		One finger up flick gesture
Page Up	To scroll up/left of the content in a scroll container.	Two/three finger up/left flick gesture	Three fingers up/right flick gesture
Page Down	To scroll down/right the content in a scroll container	Two/three finger down/right flick gesture	Three fingers down/right flick gesture
	Starts reading from the beginning of the page		Two fingers up flick gesture
	Starts reading from the current focused item		Two fingers down flick gesture

iOS Gestures

Below are some of the gestures explained in the above table:



For more information on accessibility gestures, refer to:

- Android: <https://support.google.com/nexus/answer/2926960?hl=en>
- iOS: <https://www.apple.com/in/accessibility/osx/voiceover/>
- SPA and Desktop Web: <http://www.w3.org/WAI/mobile/>

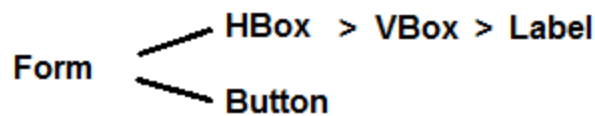
Achieving 508 Compliance Using Kony Visualizer

The built-in assistive technology in the iOS and Android platforms reads some basic widgets of Kony Visualizer, such as Button, Label, and TextBox. The assistive technology in iOS and Android platforms read the information differently on other widgets.

Every application that is created using Kony Visualizer is accessible to iOS and Android platforms built-in assistive technology. The way the assistive technology interprets the widget information is left to its individual capability. Developers can enhance the behavior of assistive technology with the configuration property (explained in the next page) available for each accessibility supported widget.

Widget Navigation Model and Tab/Focus Order

The general navigation model is for a user to tab/swipe to reach a widget, interact with the control in that widget, and then tab/swipe to move focus to the next widget in the tab order. When a container widget contains a widget, the tab/swipe gesture brings the focus to the container widget because it is the next item in the tab order. This continues down the layers of widgets until the last widget is reached. For example, we have two widgets ' HBox ' and ' Button ' on a page. The widget ' HBox ' contains a ' VBox ' widget. Within the ' VBox ' widget there is a ' Label ' widget.



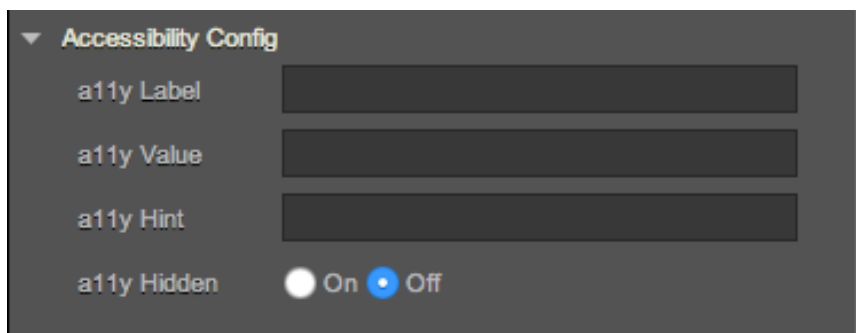
While tabbing, the focus lands on HBox, then another tab will move the focus to VBox, and then another tab will move the focus to Label. This is because Label is the last widget in VBox focus will not come directly to the Label. One more tab will move the focus to Button. The order in which the focus is passed from one widget to another widget is based on the nearest neighboring widget in a given direction.

Note: Some operating systems allow you to change the systems and enlarge all text displayed on the screen.

4.1 Define accessibilityConfig

To define accessibilityConfig on a widget from Kony Visualizer, follow the below steps:

- From Kony Visualizer, drag a widget where you want it. For example, a Button widget.
- Select the Button widget and locate the accessibilityConfig property from the **Properties Window**.
- The **Accessibility Config** appears under property editor (You need to enable Accessibility Config under Project Settings).



- Enter the following values in the respective fields:
 - **a11yLabel**: Specifies an alternate text to identify the widget. Generally the label should be the text that is displayed on the screen.

- **a11yValue**: Specifies the current state/value associated with the widget so that the user can perform an action. For example, a checkbox is in selected state or unselected state.
- **a11yHint**: Specifies the descriptive text that explains the action about the widget.
- **a11yHidden**: Specifies if the widget must be ignored by assistive technology.

For Camera widget, when the capture mode is set to video, you will get additional options to record the video. The below keys are used to configure accessibility for the additional options:

- **accessibilityConfigCaptureControl**: Provides accessibility support to video capture button.
- **accessibilityConfigTimerControl**: Provides accessibility support to video timer button.
- **accessibilityConfigSettingsControl**: Provides accessibility support to video settings button.
- **accessibilityConfigCancelControl**: Provides accessibility support to video cancel button.
- **accessibilityConfigVideoStartButton**: Provides accessibility support to video start button.
- **accessibilityConfigVideoStopButton**: Provides accessibility support to video stop button.

Define accessibilityConfig for a Widget Dynamically

The property accessibilityConfig enables you to specify the accessibility configuration property for a widget. Following are the predefined keys:

a11yLabel [String]

Optional. Specifies an alternate text to identify the widget. Generally the label should be the text that is displayed on the screen.

a11yValue [String]

Optional. Specifies the current state/value associated with the widget so that the user can perform an action. For example, a checkbox is in selected state or unselected state.

Note: On the Android platform, the text specified for a11yLabel is prefixed to the a11yValue.

a11yHint [String]

Optional. Specifies the descriptive text that explains the action about the widget.

Note: On the Android platform, the text specified for a11yValue is prefixed to the a11yHint.

a11yHidden [Boolean]

Optional. Specifies if the widget must be ignored by assistive technology. The default option is set to *false*.

Note: This option is supported on iOS 5.0 and above, Android 4.1 and above, and SPA.

Limitations

Android:

- If the results of the concatenation of a11y fields result in an empty string, then the accessibilityConfig is ignored and the text that is on widget is read out.
- The soft keypad does not gain accessibility focus with right/left swipe gesture when the keypad appears.

SPA: The a11y fields are mapped to the ARIA tags. The results may vary among browsers because not all browsers recognize all the ARIA tags.

Permissions

Read + Write

Example

```
//Defining the properties for a button.
var btnBasic={id:"button1", isVisible:true, skin:"btnSkin",
focusSkin:"btnFSkin", text:"Click Here", "accessibilityConfig":
  {
```

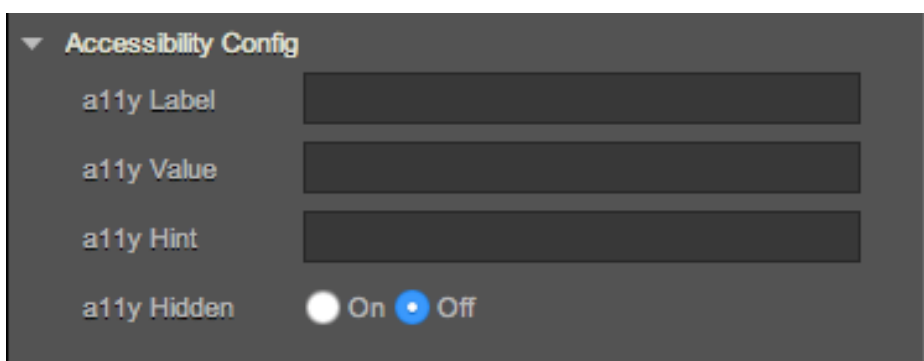
```
        "allyHidden": true,  
        "allyValue": "Your text goes here",  
        "allyLabel": "Your text goes here",  
        "allyHint": "Your text goes here"  
    }  
};  
var btnLayout={containerWeight:100, padding:[5,5,5,5], margin:  
[5,5,5,5], hExpand:true, vExpand:false, displayText:true};  
var btnPSP={};  
  
//Creating the button.  
var button1 = new kony.ui.Button(btnBasic, btnLayout, btnPSP);
```

Availability

- iOS
- Android
- SPA (iPhone and Android)

4.2 Widget Behavior

Here is a sample representation of accessibility on iPhone and Android devices. For example, you have a Confirm button on form frm1, and the accessibilityConfig is defined as below:



Then the assistive technology in the respective platforms will read as follows:



Note: In the above snapshot the highlighted text is role, generated by native platforms. The iPhone appended the text **button** to the value, and Android appended the text **button** to the hint automatically. Kony Visualizer has no control on this behavior. Developers should test the text that is provided for accessibilityConfig.

4.3 Container Widgets

Below are the platforms behaviors for the container widgets when the accessibility feature is enabled.

- [FlexForm](#)
- [Form](#)
- [HBox](#)

- [VBox](#)
- [ScrollBox](#)
- [Popup](#)

4.3.1 FlexForm

Keyboard/Gesture-based Interactions	<ul style="list-style-type: none"> • Tab key or equivalent touch gesture moves the focus to the first focusable child widget of the form. • Multiple tabs move the focus to the interactive child widgets of the form. • The title of the form is accessible in the platforms that support native form widget titleBar property using the tab key or equivalent touch gesture along tab order.
Default Behavior	<ul style="list-style-type: none"> • The a11yLabel overrides the text of the title property. • The a11yValue, a11yHint, and a11yHidden fields are not applicable to form and are ignored. • accessibilityConfig property is supported in iPhone, Android, and SPA-iPhone platforms.

Limitations**iOS:**

- When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form.
- The iOS VoiceOver, focus the first accessible widget available on the form.

Android:

- onTap gesture on a form, when there are no focusable widgets, the assistive technology reads all non-focusable widgets text available in the form.
- In Android OS versions less than 4.2, form does not scroll, although it has content to scroll. You have to enable an option in Android OS versions 4.2 and above in system accessibility settings to auto scroll the content on swipe gesture.
- Accessibility capability of the ActionBar is left to the device default behavior.

SPA:

- **SPA-iPhone:** When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form.
- **SPA-Android:** accessibilityconfig is not supported

4.3.2 Form

Keyboard/Gesture-based Interactions	<ul style="list-style-type: none">• Tab key or equivalent touch gesture moves the focus to the first focusable child widget of the form.• Multiple tabs move the focus to the interactive child widgets of the form.• The title of the form is accessible in the platforms that support native form widget titleBar property using the tab key or equivalent touch gesture along tab order.
Default Behavior	<ul style="list-style-type: none">• The a11yLabel overrides the text of the title property.• The a11yValue, a11yHint, and a11yHidden fields are not applicable to form and are ignored.• accessibilityConfig property is supported in iPhone, Android, and SPA-iPhone platforms.

Limitations**iOS:**

- When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form.
- The iOS VoiceOver, focus the first accessible widget available on the form.

Android:

- onTap gesture on a form, when there are no focusable widgets, the assistive technology reads all non-focusable widgets text available in the form.
- In Android OS versions less than 4.2, form does not scroll, although it has content to scroll. You have to enable an option in Android OS versions 4.2 and above in system accessibility settings to auto scroll the content on swipe gesture.
- Accessibility capability of the ActionBar is left to the device default behavior.

SPA:

- **SPA-iPhone:** When the VoiceOver focus reaches the bottom of the form, it does not cycle to the top of the form again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the form when you have reached the top of the form.
- **SPA-Android:** accessibilityconfig is not supported

4.3.3 HBox

Keyboard/Gesture based Interactions	<ul style="list-style-type: none">• Tab key or equivalent touch gesture moves the focus along the tab order when:<ul style="list-style-type: none">a. Box is clickableb. accessibilityConfig is defined.• With a focus on the HBox, press Spacebar or Enter or equivalent gesture action to select the HBox when it is clickable.• Multiple tabs or navigation keys help in navigating focus to the child widgets that are actionable.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

<p>Limitations</p>	<p>iOS:</p> <ul style="list-style-type: none"> If the accessibilityConfig is set for an HBox, then the focus never goes to its child widgets and other widgets within the HBox are not accessible to the user. <p>Android: None</p> <p>SPA:</p> <ul style="list-style-type: none"> SPA-iPhone: If the accessibilityConfig is set for an HBox, then the focus never goes to its child widgets, and other widgets within the HBox are not accessible to the user. SPA-Android: If the accessibilityConfig is set for an HBox, then widgets within the HBox are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.
---------------------------	---

4.3.4 VBox

<p>Keyboard/Gesture based Interactions</p>	<ul style="list-style-type: none"> Tab key or equivalent touch gesture moves the focus along the tab order when: <ol style="list-style-type: none"> Box is clickable. accessibilityConfig is defined. With a focus on the VBox, press Space or Enter or equivalent gesture action to select the VBox when it is clickable. Multiple tabs or navigation keys help in navigating focus to the child widgets that are actionable.
---	--

Default Behavior	<p>accessibilityConfig property is supported in iPhone, Android, and SPA platforms.</p>
Limitations	<p>iOS:</p> <ul style="list-style-type: none"> • If the accessibilityConfig is set for a VBox, then the focus never goes to its child widgets and other widgets within the VBox. <p>Android: None</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: If the accessibilityConfig is set for a VBox, then the focus will never go to its child widgets, and other widgets within the VBox are not accessible to the user. • SPA-Android: If the accessibilityConfig is set for a VBox, then widgets within the VBox are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.

4.3.5 ScrollBox

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Tab key or equivalent touch gesture moves the focus along the tab order when accessibilityConfig is defined. • Multiple tabs or navigation keys help in navigating focus to the child widgets that are actionable. • Page Up / Page Down or equivalent key/gesture allows you to scroll the content of the ScrollBox.
--	---

<p>Default Behavior</p>	<p>accessibilityConfig property is supported in iPhone, Android, and SPA platforms.</p>
<p>Limitations</p>	<p>iOS:</p> <ul style="list-style-type: none"> • If the accessibilityConfig is set for a ScrollBox, then the focus never goes to its child widgets, and other widgets within the ScrollBox are not accessible to the user. <p>Android: In Android OS versions less than 4.2, the form does not scroll although it has content to scroll. It depends on the capability of the built-in Accessibility service. You have to enable an option in Android OS versions 4.2 and above in the system accessibility settings to auto-scroll the content on swipe gesture. Similar behavior is observed with native applications as well.</p> <p>When the scrollDirection property is set to SCROLLBOX_SCROLL_BOTH, the behavior is undefined.</p> <p>SPA: When a user scrolls through the Scrollbox, it does not scroll and the widgets are not displayed in the view port, even if you set accessibility. This is due to the inability of the browsers to detect the touch gestures. But the widgets within Scrollbox are navigated and accessibility of the widget is read out.</p> <ul style="list-style-type: none"> • SPA-iPhone: If the accessibilityConfig is set for a ScrollBox, then widgets within the ScrollBox are not accessible to the user. • SPA-Android: If the accessibilityConfig is set for a ScrollBox, then widgets within the ScrollBox are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.

4.3.6 Popup

Keyboard/Gesture based Interactions	<ul style="list-style-type: none">• Tab key or equivalent touch gesture moves the focus to the first focusable child widget of the Popup.• Multiple tabs move the focus to the interactive child widgets of the Popup.
Default Behavior	<ul style="list-style-type: none">• The <code>a11yLabel</code> overrides the text of the title property.• The <code>a11yValue</code>, <code>a11yHint</code>, and <code>a11yHidden</code> fields are not applicable to Popup and are ignored.• <code>accessibilityConfig</code> property is supported in iPhone, Android, and SPA-iPhone platforms.

Limitations	<p>iOS:</p> <ul style="list-style-type: none">• When the VoiceOver focus reaches the bottom of the Popup, it does not cycle to the top of the Popup again, with a right swipe gesture. Similarly with a left swipe gesture, the focus does not cycle to the bottom of the Popup when you have reached the top of the Popup. <p>Android:</p> <ul style="list-style-type: none">• When there are no focusable widgets in a Popup, the assistive technology reads all non-focusable widgets text available in the Popup.• In Android OS versions less than 4.3, the Popup does not scroll although it has more content to scroll. It is the capability of the built-in Accessibility service (TalkBack) that is lacking in versions less than 4.3 OS versions. You have to enable an option in Accessibility settings in 4.3 and 4.4 Android OS versions to auto-scroll the content on a swipe gesture. <p>SPA:</p> <ul style="list-style-type: none">• SPA-iPhone: When the VoiceOver focus reaches the bottom of the Popup, it does not cycle to the top of the Popup again, with a right swipe gesture. Similarly, with a left swipe gesture, the focus does not cycle to the bottom of the Popup when you have reached the top of the Popup.• SPA-Android: accessibilityConfig is not supported.
--------------------	--

4.4 Basic Widgets

Below are the platforms behaviors of the basic widgets when accessibility feature is enabled.

- [Button](#)
- [Calendar](#)

- [CheckBox](#)
- [ComboBox](#)
- [Image](#)
- [Label](#)
- [Line](#)
- [Link](#)
- [ListBox](#)
- [RadioButton](#)
- [RichText](#)
- [Slider](#)
- [TextArea](#)
- [TextBox](#)

4.4.1 Button

Keyboard/Gesture based Interactions

- With a focus on the Button, press the Spacebar or Enter key or equivalent gesture action to select the Button.
- Single finger double tap to execute the action.
- Accessible by the tab key or equivalent touch gesture along tab order.

Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	<p>iOS: None</p> <p>Android: None</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: None • SPA-Android: The option a1yHint is not supported.

4.4.2 Calendar

Description	<p>A Calendar widget accepts dates from the user. Following are the view types that support accessibility in respective platforms:</p> <ul style="list-style-type: none"> • CALENDAR_VIEW_TYPE_DEFAULT (Android) • CALENDAR_VIEW_TYPE_WHEEL_POPUP (iPhone) • CALENDAR_VIEW_TYPE_GRID_POPUP (iPhone and SPA)
Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the Calendar, press the Spacebar or Enter key or equivalent gesture action to launch the date selector. • You can reach to each available day, month, and year in a calendar through one/some of the keys or touch gestures.

Default Behavior	<ul style="list-style-type: none"> • accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	<ul style="list-style-type: none"> • a11yValue is not applicable. • It is recommended to provide accessibility text to the assistive technology to read the date when the tab focus/gesture makes a selection. • accessibilityConfig is not supported for the viewTypees that are not focused as a whole.

4.4.3 CheckBox

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Accessible by the tab key or equivalent touch gesture along tab order. • Multiple tabs or navigation keys help in navigating the focus to each check button in the group. • With a focus on the CheckBox, press the Spacebar or Enter key or equivalent gesture to change the selection state of the focused check button.
Default Behavior	<ul style="list-style-type: none"> • accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

Limitations

iOS: Following are the limitations of iOS platform based on the selected viewType:

viewType	accessibilityConfig	
	Widget Level	Items within widget
CHECKBOX_VIEW_TYPE_LISTVIEW	Respected	Ignored *
CHECKBOX_VIEW_TYPE_TOGGLEVIEW	Ignored	Respected
CHECKBOX_VIEW_TYPE_ONSCREENWHEEL	Ignored	Ignored
CHECKBOX_VIEW_TYPE_TABLEVIEW	Ignored	Respected

* accessibilityConfig is ignored when set through *masterData* or *masterDataMap* to the items within the widget that pops up as pickerview from the bottom.

Android: None

SPA:

- **SPA-iPhone:** accessibilityConfig for CheckBox as a whole is not respected, but the items within the widget are respected. The CheckBox state and the item text gets the focus separately.
- **SPA-Android:** accessibilityConfig for CheckBox as a whole is not respected, but the items within the widget are respected. The CheckBox state and the item text get the focus separately. The option a1yHint is not supported.

4.4.4 ComboBox

Keyboard/Gesture based Interactions	<ul style="list-style-type: none">• Tab key or equivalent touch gesture along tab order.• With a focus on the ComboBox, press the Spacebar or Enter key or equivalent gesture action to open the drop-down list.• With drop-down list in an expanded state, press the Spacebar or Enter key or equivalent gesture to select the focused item.• Multiple tabs or navigation keys help in navigating the focus to each item in the ComboBox.
Default Behavior	<ul style="list-style-type: none">• accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

Limitations

iOS: Following are the limitations of iOS platform based on the selected viewType:

viewType	accessibilityConfig	
	Widget Level	Items Within Widget
COMBOBOX_VIEW_TYPE_LISTVIEW	Respected	Ignored *
COMBOBOX_VIEW_TYPE_TABLEVIEW	Ignored	Respected
COMBOBOX_VIEW_TYPE_TOGGLEVIEW	Ignored	Respected
COMBOBOX_VIEW_TYPE_ONSCREENWHEEL	Ignored	Ignored

* accessibilityConfig is ignored when set through *masterData* or *masterDataMap* to the items within the widget that pops up as pickerview from the bottom.

Android: If the accessibilityConfig is set, it will override the selected item.

SPA:

- **SPA-iPhone:** The ComboBox widget is mapped to the HTML ComboBox, and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ComboBox widget.
- **SPA-Android:** The ComboBox widget is mapped to the HTML ComboBox, and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ComboBox widget.

4.4.5 Image

Keyboard/Gesture based Interactions	Tab key or equivalent touch gesture along tab order.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	<p>On all platforms, except SPA, if the accessibility is not configured, the image widget is not accessible to the user.</p> <p>SPA-Android: The option a11yHint is not supported.</p>

4.4.6 Label

Description	A Label widget is used to display non-editable text to the user.
Keyboard/Gesture based Interactions	Tab key or equivalent touch gesture along tab order.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA-Android platforms.

Limitations	<p>iOS: None</p> <p>Android: None</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: accessibilityConfig property is not supported. • SPA-Android: The option a11yHint is not supported
--------------------	---

4.4.7 Line

Keyboard/Gesture based Interactions	Not accessible by the tab key or equivalent touch gesture.
Default Behavior	accessibilityConfig property is not supported.
Limitations	None

4.4.8 Link

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • With a focus on the link, press the Spacebar or Enter key or equivalent gesture action to select the link. • Single finger double tap to execute the action. • Tab key or equivalent touch gesture along tab order.
--	---

Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	<p>iOS: None</p> <p>Android: None</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.

4.4.9 ListBox

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the ListBox, press the Spacebar or Enter key or equivalent gesture to open the drop-down list. • With drop-down list in an expanded state, press the Spacebar or Enter key or equivalent gesture to select the focused item. • Multiple tabs or navigation keys help in navigating the focus to each item in the ListBox.
Default Behavior	<ul style="list-style-type: none"> • accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

Limitations

iOS: Following are the limitations of iOS platform based on the selected viewType:

viewType	accessibilityConfig	
	Widget Level	Items Within Widget
LISTBOX_VIEW_TYPE_LISTVIEW	Supported	Ignored *
LISTBOX_VIEW_TYPE_TABLEVIEW	Ignored	Supported
LISTBOX_VIEW_TYPE_TOGGLEVIEW	Ignored	Supported
LISTBOX_VIEW_TYPE_ONSCREENWHEEL	Ignored	Ignored

* accessibilityConfig is ignored when set through *masterData* or *masterDataMap* to the items within the widget that pops up as pickerview from the bottom.

Android: If the accessibilityConfig is set, it will override the selected item.

SPA:

- **SPA-iPhone:** The ListBox widget is mapped to the HTML ListBox, and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ListBox widget.
- **SPA-Android:** The ListBox widget is mapped to the HTML ListBox and browsers launch the list items as native popup. Accessibility configuration working for the list items cannot be controlled by Kony Platform. Accessibility is not supported for the items of the ListBox widget. The option `a11yHint` is not supported.

4.4.10 RadioButton

Keyboard/Gesture based Interactions	<ul style="list-style-type: none">• Accessible by the tab key or equivalent touch gesture along tab order.• With a focus on the RadioButton, press the Spacebar or Enter key or equivalent gesture to change the selection state of the focused item.• Multiple tabs or navigation keys help in navigating the focus to each item in the RadioButton.
Default Behavior	<ul style="list-style-type: none">• accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

Limitations

iOS: Following are the limitations of the iOS platform based on the selected viewType:

viewType	accessibilityConfig	
	Widget Level	Items Within Widget
RADIOGROUP_VIEW_TYPE_LISTVIEW	Supported	Ignored *
RADIOGROUP_VIEW_TYPE_TABLEVIEW	Ignored	Supported
RADIOGROUP_VIEW_TYPE_TOGGLEVIEW	Ignored	Supported
RADIOGROUP_VIEW_TYPE_ONSCREENWHEEL	Ignored	Ignored

* accessibilityConfig is ignored when set through *masterData* or *masterDataMap* to the items within the widget that pops up as pickerview from the bottom is ignored.

Android: None

SPA:

- **SPA-iPhone:** accessibilityConfig for RadioButton as a whole is not supported, but the items within the widget are supported.
- **SPA-Android:** accessibilityConfig for RadioButton as a whole is not supported, but the items within the widget are supported. The option a11yHint is not supported.

4.4.11 RichText

Keyboard/Gesture based Interactions	Accessible by the tab key or equivalent touch gesture along tab order.
Default Behavior	accessibilityConfig property is supported in iPhone, Android, and SPA platforms.
Limitations	On all platforms, links inside a RichText widget are not accessible. SPA-Android: The option a1yHint is not supported.

4.4.12 Slider

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the Slider, press the Right / Up key or equivalent gesture to increase the value of the slider. Press the Left / Down key or equivalent gesture to decrease the value of the slider.
Default Behavior	<ul style="list-style-type: none"> • accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

Limitations	<p>Android: Android OS cannot change the slider value when accessibility is set.</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: Browsers cannot change the slider value when accessibility is set. • SPA-Android: Browsers cannot change the slider value when accessibility is set.
--------------------	--

4.4.13 TextArea

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • With a focus on the TextArea, press the Spacebar or equivalent gesture to open the soft keypad for touch devices. • Single finger double tap to execute the action. • Accessible by the tab key or equivalent touch gesture along tab order. • Soft keypad gains focus on explicit touch on the soft keypad.
Default Behavior	<p>accessibilityConfig property is supported in iPhone, Android, and SPA platforms.</p>

Limitations	<p>iOS: None</p> <p>Android: When the accessibilityConfig is defined for placeholder or entered text, then behavior is left to the device.</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.
--------------------	--

4.4.14 TextBox

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • With a focus on the TextBox, press the Spacebar or equivalent gesture to open the soft keypad for touch devices. • Single finger double tap to execute the action. • Accessible by the tab key or equivalent touch gesture along tab order. • Soft keypad gains focus on explicit touch on the soft keypad.
Default Behavior	<p>accessibilityConfig property is supported in iPhone, Android, and SPA platforms.</p> <div data-bbox="565 1434 1386 1591" style="border: 1px solid #008000; background-color: #e6f2ff; padding: 5px;"> <p>Note: To configure the clear text button, use the property accessibilityConfigForClearButton and the keys are same as that of accessibilityConfig. This is applicable to iOS platform only</p> </div>

Limitations	<p>iOS: None</p> <p>Android: When the accessibilityConfig is defined for placeholder or entered text, then behavior is left to the device.</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: None • SPA-Android: The option a11yHint is not supported.
--------------------	--

4.5 Advanced Widgets

Below are the behaviors of the advanced widgets when the accessibility feature is enabled.

- [Alert](#)
- [Camera](#)
- [Hz Image Strip](#)
- [PickerView](#)
- [SegmentedUI - TABLEVIEW](#)
- [SegmentedUI - PAGEVIEW](#)
- [Switch](#)

4.5.1 Alert

Description	An Alert is a dialog displayed to show an alert message.
--------------------	--

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture to navigate and focus on the buttons and messages of the Alert dialog box. • With a focus on the Alert button, press the Enter or Spacebar or equivalent gesture to select the button. • By default, Alerts should gain focus as Alert displays.
Default Behavior	accessibilityConfig is not supported.
Limitations	On all platforms, the buttons within the Alert dialog are not configurable.

4.5.2 Camera

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the camera, press the Enter or Spacebar or equivalent gesture to launch the camera.
Default Behavior	<ul style="list-style-type: none"> • accessibilityConfig property is supported in iPhone and Android platforms.

Limitations	accessibilityConfig property is not supported in SPA platform
--------------------	---

4.5.3 Hz Image Strip

Keyboard/Gesture based Interactions	<ul style="list-style-type: none">• Touch gesture: Single finger double tap.• Accessible by the tab key or equivalent touch gesture along tab order.• On multiple tabs or equivalent touch gesture, move the focus to the images where accessibility is configured and visible on the screen.• Images that are not visible are scrolled automatically to a visible region on a tab or equivalent gesture.
Default Behavior	<ul style="list-style-type: none">• accessibilityConfig property is supported in iPhone, Android, and SPA platforms.

Limitations

If the entire image strip widget is not focused as a whole, then the accessibilityConfig is not respected in any of the platform. The accessibilityConfig is supported only when the viewType is set to HORIZONTAL_IMAGESTRIP_VIEW_TYPE_STRIPVIEW.

iOS:For the viewType when set to HORIZONTAL_IMAGESTRIP_VIEW_TYPE_STRIPVIEW, accessibility is ignored. But the accessibility configured for each image is supported. Accessibility is not available for all other viewtypes.

Android: In Android OS versions less than 4.2, Horizontal Image Strip does not scroll though it has content to scroll. It depends on the capability of the built-in accessibility service. You must enable an option in Android OS versions greater than equal to 4.2 in system accessibility settings to auto scroll the content on swipe gesture. You will observe similar behavior with native applications as well.

SPA:

- **SPA-iPhone:** If the accessibilityConfig is set for a Horizontal Image Strip, then widgets within the Horizontal Image Strip are not accessible to the user.
- **SPA-Android:** If the accessibilityConfig is set for a Horizontal Image Strip, then widgets within the Horizontal Image Strip are not accessible to the user with a swipe gesture. But when touched explicitly the widgets gain focus. The option a11yHint is not supported.

4.5.4 pickerView

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Touch gesture: Single finger double tap. • Accessible by the tab key or equivalent touch gesture along tab order. • Every column in the pickerView is reachable by the tab key or equivalent touch gesture. • With a focus on the pickerView, press the Right / Up key or Left / Down key or equivalent gesture to allow navigation between items in the focused column. • With a focus on the pickerView, press the Enter or Spacebar or equivalent gesture to select the focused item in the pickerView.
Default Behavior	<ul style="list-style-type: none"> • accessibilityConfig property is supported in Android platforms.
Limitations	<p>iOS: accessibilityConfig is not supported</p> <p>Android: In Android OS versions less than 4.2, SegmentedUI does not scroll though it has content to scroll. It depends on the capability of the built-in accessibility service. You must enable an option in Android OS versions greater than equal to 4.2 in system accessibility settings to auto-scroll the content on a swipe gesture. You will observe similar behavior with native applications as well.</p>

4.5.5 SegmentedUI - TABLEVIEW

Description	<p>A SegmentedUI is a container widget to display multiple rows of information in vertical order.</p>
--------------------	---

Keyboard/Gesture based Interactions	<ul style="list-style-type: none">• Accessible by the tab key or equivalent touch gesture along the tab order moves the focus to the first row.• If the first row is a section header, then the subsequent tabs move the focus to the interactive child widgets. If there are no child widgets or interactive widgets, or all child widgets are reached, the tab moves the focus to the next row until it reaches the last visible row.• In SINGLE_SELECT_MODE or MULTI_SELECT_MODE, as the row gets the focus through the tab, underlying accessibility technology conveys the user as either selected/unselected.
Default Behavior	<ul style="list-style-type: none">• At widget level, accessibilityConfig property is not supported in iPhone, Android, and SPA platforms because it is not focusable completely. Accessibility is supported only for the individual rows because they are focused completely.• accessibilityConfig is applied to the row template, and then accessibility is applied to each row, unless overridden by the row data.• Row template's accessibility configurations can be modified before setting the row data to the SegmentedUI and should not be modified after data are set.

<p>Limitations</p>	<p>iOS: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible.</p> <p>Android: In Android OS versions less than 4.2, SegmentedUI does not scroll though it has content to scroll. You must enable an option in Android OS versions 4.2 and above system accessibility settings to auto-scroll the content on a swipe gesture. You will observe similar behavior with native applications.</p> <p>SPA:</p> <ul style="list-style-type: none"> • SPA-iPhone: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. • SPA-Android: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. But when touched explicitly the widgets gain focus with a swipe gesture. The option a11yHint is not supported.
---------------------------	---

4.5.6 SegmentedUI - PAGEVIEW

<p>Description</p>	<p>A SegmentedUI is a container widget to display multiple rows of information in horizontal layout with a single row appearing on the widget.</p>
---------------------------	--

Keyboard/Gesture based Interactions	<ul style="list-style-type: none">• Touch gesture: Single finger double tap.• Accessible by the tab key or equivalent touch gesture along tab order.• If the first row is a section header, then the subsequent tabs move the focus to the interactive child widgets. If there are no child widgets or interactive widgets, or all child widgets are reached, the tab moves the focus out of the widget.• If there are page indicators at the bottom of the PAGEVIEW and the page indicators are interactive, the tab focus each page and pass the index of the total page information to the assistive technology.• In SINGLE_SELECT_MODE or MULTI_SELECT_MODE, as the row gets the focus through the tab, underlying assistive technology conveys the user as either selected/unselected.
Default Behavior	<ul style="list-style-type: none">• At widget level accessibilityConfig property is not respected. It is respected only for the page level in iPhone, Android, and SPA platforms.• accessibilityConfig applied to the row template and its internal widgets are applied to each row, unless overridden by the row data.• Row template's accessibility configurations can be modified before setting the row data to the SegmentedUI and should not be modified after data are set.

Limitations	<p>On all platforms, accessibilityConfig is not supported for page indicators.</p> <p>iOS: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible.</p> <p>Android: In Android OS versions less than 4.2, SegmentedUI does not scroll though it has content to scroll. It depends on the capability of the built-in accessibility service. You must enable an option in Android OS versions 4.2 and above, in system accessibility settings to auto-scroll the content on a swipe gesture. You will observe similar behavior with native applications as well.</p> <p>SPA: The event onRowClick is fired when any child widget is explicitly selected or clicked.</p> <ul style="list-style-type: none"> • SPA-iPhone: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. • SPA-Android: If the accessibilityConfig is set to a row of a SegmentedUI, then actionable child widgets of the row become inaccessible. But when touched explicitly the widget's gain focus. The option a11yHint is not supported.
--------------------	---

4.5.7 Switch

Keyboard/Gesture based Interactions	<ul style="list-style-type: none"> • Accessible by the tab key or equivalent touch gesture along tab order. • With a focus on the Switch, press the Enter key or Spacebar or equivalent gesture to toggle the state and initiate the action.
Default Behavior	<ul style="list-style-type: none"> • accessibilityConfig property is supported in iPhone and SPA platforms.

Limitations	<p>iOS: The option <code>a11yValue</code> is not supported.</p> <p>SPA:</p> <ul style="list-style-type: none">• SPA-iPhone: None• SPA-Android: The option <code>a11yHint</code> is not supported.
--------------------	--

4.6 Accessibility Best Practices

To read the best practices of accessibility, refer to [WebAccessibility](#).

4.7 Accessibility: Platform Specific Limitations

This section lists the accessibility limitations of SPA platforms.

4.7.1 SPA

This section lists the accessibility limitations of SPA platforms.

- Scrolling a form and SegmentedUI through a swipe with three fingers and tab gestures (custom scroll) is not supported in SPA platform.
- When a Popup is loaded, the default focus goes to the form on which the Popup is loaded, and then it comes to Popup.
- A Label widget without any text is not focusable with tab gestures.
- When a form is loaded, the default focus can be any where in the form.
- For the widgets such as ScrollBox, Horizontal Image Strip, Slider, and Segment (PAGEVIEW), a swipe or tab gesture will not bring the focused item into a view area.
- On the SPA Android platform, `accessibilityConfig` for form and Popup is not supported.
- On the SPA Android platform, before loading a form, some random text is read by assistive technology. The random text is read from the script tag that may be present in JavaScript.

- For container widgets, if only a11yHint is configured, then accessibility first reads the text of the child widgets and then a11yHint text that is configured for a container widget.
- When the accessibilityConfig is set for any container widget (HBox, VBox, Scrollbox, FlexContainer and FlexScrollContainer), the widgets inside the container will not get focused while navigating in iOS Safari.
- When the accessibilityConfig is set for any container widget (HBox, VBox, Scrollbox, FlexContainer and FlexScrollContainer), the container widget will not get focused while navigating in Android browsers.
- Accessibility in SPA/Desktop web platforms is achieved using HTML ARIA tags. To add any ARIA property specified in W3C (refer the following link for information on [ARIA in W3C](#)), use the attribute **a11yARIA**.

For Example:

```
Form1.Widget1.accessibilityConfig = { "a11yLabel": "Label",  
  "a11yValue": "Value", "a11yHint": "Hint", "a11yARIA": { "aria-  
labelledby" : "Example" } };
```

- Accessibility in SPA/Desktop web depends on Web browser, Web browser version, Voice Over Assistant, and Voice Over Assistant version.
-

5. Flex Layout Guidelines

A Flex Container Widget gives you the ability to position and size the widgets anywhere on a form. You can have multiple widgets in a Flex Container Widget. All the widget layout properties can be used to position and size the widgets, and flex container respects the values set. Visualizer helps us to develop Kony Applications using Flex Layout.

Flex Layout Guidelines describes some of the best practices that are to be followed during the application development using Flex Layout. The developers use different implementations of the widgets available in the Integrated Development Environment (IDE) to build Graphical User Interfaces (GUI).

In this section you will learn:

1. [Animation & Flex Layout Limitations](#)
2. [Flex Backwards Compatibility](#)
3. [Flex Pseudocode Examples](#)
4. [Flex Layout Animation](#)

5.1 Animation and Flex Layout Limitations

Following are the limitations applicable to iOS, Android, SPA, and Windows platforms using widget level animations:

- If width, height related parameters are explicitly given as negative values or resulted in negative values due to implicit calculations, then consistent cross platform behavior cannot be assured.
- The dimensional properties only indicate the values that developers set, but the layout engine will determine final frames and the animation happens between the final computed values.
- When widgets overlap, z-index specified along with the order in which the widgets added will determine the overlapping order.

- Animation properties in the animation steps must not be skipped in the middle for cross platform consistent animations.
 - For example, property 'x' is present in step 0 and not present in step 50 and again present in step 100 may lead to inconsistent results. However, property 'x' present in step 0, 50 and not being present in step 100 is perfectly fine.
- Calling rotate(), scale(), and translate() multiple times will overwrite the previous values but not club or sum. For example, rotate(45); rotate(50) is not equal to rotate(105), instead, it will be rotate(50).
- If scale, rotate, and translate are in a transform object, irrespective of the invocation order of operations, the method `kony.ui.makeAffineTransform` applies the transform in the following order:
 - Scale
 - Rotate
 - Translate
- Layout parameter animations:
 - Properties in the animation definition are interpolated between steps for smooth animation. If a property is not present in the intermediate steps then the property is not carried forward but interpolated between steps where the property is present. This rule of interpolation is applicable iOS, SPA, and Windows platforms for all properties except for layout parameters.
 - In iOS, SPA, and Windows platforms for every step, resultant frame is calculated from the layout properties mentioned for the step. If layout parameters are not mentioned for the step then layout parameters are carried forward from the previous steps.
 - This essentially mean in case of iOS, SPA, and Windows platforms, if layout parameters are not mentioned at a given step or layout parameters are mentioned in such a way that it does not yield any frame changes compared to the previous step then there will not be any visible layout related animations occurring between the steps.

- Android platform follows property based animation where there is no carried forward of layout parameters. If a layout property is not mentioned in the intermediate steps, then the property is interpolated between the steps where ever the layout property is present.
- Background color animation:
 - The animation on background color works only when the backgroundColor property is set before applying an animation definition.
 - Initial value of backgroundColor has to be specified explicitly. If not, platform will not deduce the values from the existing skin and will lead to undefined behavior.
 - For the background color animations to work, the skin configured for the widget should not have background type as multi step gradient or image. It has to be single color background.
- In iOS and Android platforms, matrix multiplication is used between the steps to arrive at the resultant transformation animation. This means in the final effect you may not see the effects of individual transform operations such as scale, rotate, and transform.
- In iOS and Android platforms, the animation effect applied to the widget during the steps would be a cumulative effect of all the transformations applied together. The cumulative effect would be similar to matrix multiplication applied as per the defined order.

For example, for any widget when 0-90 degrees rotate animation is applied along with scale and translate animations, the 0-90 degrees rotation of the widget is not seen individually because of the cumulative animation properties applied.

For any widget, when scale (0, 0) animation is applied along with combination of translate and rotate animations will result in matrix multiplication as '0'. In this case, the rotation is not seen initially because of the multiplication results in zero.
- In Desktop and SPA platforms, height and width transformation animation is not supported on the Switch widget. Also, Min/Max Height/Width animations cannot be applied on the Switch widget.
- When FlexContainer is added dynamically to Box, Form or Popup with layoutType as `kony.flex.VBOX_LAYOUT`, the height of the FlexContainer is calculated based on parent's

width (i.e. Box, Form, and Popup).

For example, if FlexContainer is added to Popup and its height is defined as 100 percent, then FlexContainer Height is calculated based on Popup width. In this case FlexContainer height will be 100% of Popup's width.

- 3D transformations will not result into any kind of layout changes after the animation is complete.
- When both 3D and 2D transforms are used together, which ever is applied latest will override the older transform.

Following are the limitations for Animations, Gestures, and Flex Layout on the respective platforms.

- [iOS Limitations](#)
- [Android Limitations](#)
- [Windows Limitations](#)
- [SPA Limitations](#)

5.2 iOS Limitations

Following are the limitations of iOS platform:

5.2.1 Flex Layout

- Touch events will not work for the part of the widget that is outside the parent's boundaries when clipBounds property of the parent is set to false.
- Dimensions of Switch and pickerView widgets are dictated by underlying SDK and might differ from the values specified through width and height properties.

- Widget shadows won't work as of now. We recommend you to place the widget inside a FlexContainer and configure shadows for the FlexContainer instead. Also clipBounds property of the flex container should be set to false.
- Scrolling events of FlexScrollContainer will not work if a scrollable widgets like SegmentedUI or Map with scrolling, in the same direction as FlexScrollContainer is present.
- For contentSize property, both width and height must be specified. Otherwise it will lead into undefined behaviors.
- Touch events will not work if the touch is started on the part of the widget, which is outside the parent's boundaries when the clipBounds is false.
- If two widgets are overlapping, then if the widget with higher zIndex has not registered for any touch/click events then below widget will not get the events even in the overlapped region.
- Switch widget will have constant width and height.

5.2.2 Animation Limitations Using Flex Layout

- 6.0 specific animation APIs are applicable only to flex widgets.
- The animations on the CheckBox and PickerView width and height will not work.
- If the widget has multi colored focus or non-focus skin (gradients, multi-step gradients, Vertical split), then backgroundColor animations will not work.
- In Horizontal/Vertical flow flex layout no two widgets can have explicit animations. They can be mutually exclusive. If they are explicitly called on all/multiple widgets, the behavior is undefined.
 - For example, say there are two buttons button1 and button2 inside a flex layout container with layoutType as Horizontal flow.
 - In this scenario, the user cannot call button1.animate(...) followed by button2.animate(...). You have to call animate API only on either button1 or button2. The rest of the widgets will get implicit animations.

- If an animation duration is zero (0), then a flicker may appear.
- If an animation has layout properties combined with non-layout properties, then a small flicker may appear before the animations.
- Browser widget with non integer width or height values, may result in a black line appearing at the corners.
- API Level 6000 flag is generated by Kony Visualizer is mandatory for the animations to work in Kony Visualizer 6.0.
- The keyframe animation is not supported for headers and footers.
- 3D animations are not supported for Header and Footer in the iOS platform.
- For Switch Widget, height and width animations are not supported on iOS platform.

5.2.3 Gesture Limitations Using Flex Layout

- Badge APIs on the widgets will not work when widgets are placed in FlexForm. Badge functionality can be achieved using zIndex property.
- The events onTouchStart, onTouchEnd, and onTouchMove may not work or yield desired results when registered on scrollable widgets such as FlexScrollContainer, Map, SegmentedUI, TabPane, TextArea, and Browser. These widgets internally uses the lower level touch events to get the scrolling behavior and may conflict with the externally registered touch events. On other non scrollable widgets these lower level events get fired along with the existing events as per the widget behavior. For example, onClick on Button widget get fired along with touch events if touch events are registered with the Button.

5.2.4 Segment Animation Limitations

- For the AutoGrow Segment, animations will get applied even for the rows which are not in the visible area. This is applicable for all methods.
- For the onRowDisplay method, VISIBLE animations will be called only when a row is brought to visible region due to the ADD/REMOVE operation on the Segment widget.

- Calling the `removeAt/removeSection At` methods multiple times simultaneously unexpected behavior results. Because of this, the `onRowDisplay` method may be called in inconsistent order.
- Z-index is low for the row added earlier and high for the row added later during the animation.

5.3 Android Limitations

Following are the limitations of Android platform:

5.3.1 Flex Layout

- If the height is given to a segment in the constructor itself, then `groupCells` property will not work for segment inside `FlexContainer`.
- `clipBounds` property for `Map` and `Browser` widgets will not work during animation, because the rendering happens in native `openGL` using `GPU`.
- `zIndex` property will not work for `TabPane` and `DataGrid` widgets.
- Android OS native theme skinning comes with some transparent pixels in the background image. Due to which even if `left` and `top` are specified as zero (0) without applying any skin, you will observe some gap from the parent containers `left` and `top` for the below widgets:
 - `Button`
 - `Check box`
 - `Label`
 - `Radio Group`
 - `RichText`
 - `Slider`
 - `TextBox`

- Camera
- Phone
- Slider widget placed inside horizontal scroll container (FlexScrollContainer or ScrollBox), then you will not be able to slide because of double scrolling issue.
- skin set to group widgets is applied to each individual item. Thus it appears as if the widget has not occupied the given height though it actually occupies. This can be verified by placing any other widget below it.
- Whenever a Text-Area/Text-Box widget gets focus, Android OS tries to bring Text-Area/Text-Box into visible region. Bringing the widget into the visible region depends on the container scroll direction. That is, if the scroll direction is horizontal, the Text-Area/Text-Box is brought into visible region by scrolling in horizontal direction, similarly with vertical direction also.

For instance, there is an HBox container widget with 150 percent width. An Image Widget is placed in this container widget, which occupies 80 percent of container's width. A TextBox Widget is added in the container widget, beside the Image Widget. As the Image Widget occupies 80 percent of the container, the TextBox Widget goes beyond the screen width horizontally and is not visible. When this TextBox is in focus, Android OS brings the TextBox into visible region by scrolling in horizontal direction.

- By default, TextBox/TextArea widget added to the Flex Form gets focus when this form gets rendered.

5.3.2 Gesture Limitations Using Flex Layout

- onTouchStart, onTouchMove, and onTouchEnd events will not work on Map, Browser, and group widgets.
- For contentSize property both width and height must be specified. Otherwise, it will lead into undefined behaviors.
- Scroll container can be scrolled only till the end of the content and over scrolling is not possible. Thus, if the x and y values are given to setContentOffset exceed the actual scrollable content,

then it will scroll only till the end of the content. For example, if the total content width is 150% and the content off set given to x is 100% then it will scroll only 50% to get the remaining content into the view port.

- Touch events will not work for the part of the widget which is outside the parent's boundaries when the clipBounds is false.
- In devices with Android OS Version less than 4.2, zIndex is respected in drawing order only. But touch events are propagated in the order of widget indices present in the parent. Thus, even though a widget has a higher zIndex it will not receive touch events if it is not the last widget among the siblings.
- If two widgets are overlapping, then if the widget with higher zIndex has not registered for any touch/click events then the widgets present below it, will get the events in the overlapped region also.
- During horizontal scroll, if the direction is changed to vertical (in more than 45 Degrees) without raising the finger then the events will be consumed by any other widget in the parent hierarchy which can scroll vertically.
- If the Form has enableScrolling as true, then if any child widgets registered for touch events or gestures, will not receive the desired events as the Form consumes events in vertical direction. But same will work in horizontal direction.
- The events onTouchStart, onTouchEnd, and onTouchMove may not work or yield desired results when registered on scrollable widgets such as FlexScrollContainer, Map, SegmentedUI, TabPane, TextArea, and Browser. These widgets internally uses the lower level touch events to get the scrolling behavior and may conflict with the externally registered touch events. On other non scrollable widgets these lower level events get fired along with the existing events as per the widget behavior. For example, onClick on Button widget get fired along with touch events if touch events are registered with the Button.

5.3.3 Animation Limitations Using Flex Layout

- 6.0 specific animation APIs are applicable only to flex widgets.
- Animations will not be smooth if high resolution images are used as part of the skin or source to image widgets.
- It is suggested to use 9 patch images (drawables) wherever images are being used. I.e. as part of skins or source to image widget.
- Animations will not be smooth if there are a hierarchy of views.
- For achieving best animations results, refer to the below link:
<http://developer.android.com/guide/topics/graphics/hardware-accel.html#tips>
- In the Android platform, you cannot apply both 2D and 3D transformations to the widgets simultaneously.

5.3.4 Segment Animation Limitations

- For AutoGrow Segment, animations will get applied even for the rows which are not in the visible area. This is applicable for all methods.
- Orientation change when an animation is playing would end the animation and operation immediately.
- Animation and operations may end immediately if setting or calling any method, which leads to a complete refresh of Segment. For example, changing `widgetdatamap` while animation is playing cancels the animation.

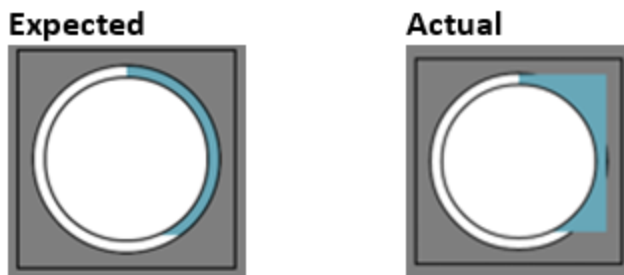
5.4 Windows Limitations

Following are the limitations of Windows platform:

5.4.1 Flex Layout

- Setting the `enableScrolling` property to `false` on the `FlexScrollContainer` will not retain the scroll position. The vertical and horizontal offset will be reset to 0,0 respectively. This is an underlying OS limitation.
- `ScrollToWidget` and `ScrollToOffset` APIs will not work when `enableScrolling` is set to `false` or `scrollDirection` is set to `None`. This is an underlying OS limitation.
- Setting the `clipBounds` property to `false` on flexcontainer with Free form layout may not work if flex container has `borderwidth`. It automatically clips.
- `convertPointFromWidget` and `convertPointToWidget` APIs are not supposed to be called in `doLayout` of any widget since this might cause application hang due to thread conflicts.
- `OnScrolling` event will be triggered with huge delays. Hence any UI updates in `onScrolling` may not yield to smooth UI changes.
- Calling `forceLayout()` on a container triggers `doLayout()` on grandchild widgets which has a change in location or size but not on all grandchildren widgets.
- Though `opacity` is set to `complete transparent`, all events are consumed on the same widget and not the widget which is behind the original widget. `Complete opaque` doesn't mean that events to pass through it.
- Any widget goes out of Flex container which has `clipBounds` set to `true`, may appear with its text but not with skin. This is an OS limitation.
- When complex forms have to be designed, it is suggested to turn off visibility for the widgets which are outside the view port or not required. Enable the visibility when required. This causes application to load faster.
- Switch widget will have constant width and height.
- Windows platform does not support `FlexContainer`'s `clipBounds` property for `Browser` and `Map Widgets`.

- Segment widget with `autogrowMode` is not supported inside the `FlexContainer` with `autogrowMode`.
- If the scrolling direction for `ScrollContainer` is defined as both horizontal and vertical, the `ContentOffset` property is respected to both X and Y coordinates. If the scroll direction is defined as vertical, it is respected only to Y coordinate but not to X coordinate, and vice versa.
- In Tablet with Windows 8.1 platform, when `Clip Bounds` is enabled for a `FlexContainer` with corner radius, the area after the radius is not clipped. The child widget of the `FlexContainer` is still shown in rectangle shape.



5.4.2 Gesture Limitations Using Flex Layout

- Gesture conflicts should be avoided in application code such as `Scroll Container` in another scroll container with the same scroll direction enabled. For example, `Slider`, `TabPane` (`Pivot` and `Panorama`), `Switch` like scrollable widgets inside horizontal flex scroll container.
- Touch events will not work on `ScrollBox` and `Segmented UI` (table view with scrolling enabled).
- When both `Tap` and `Double Tap` gestures are set, both will trigger for each double tap. `Tap` is not ignored.
- Do not assign low level touch events on clickable widgets to achieve click behavior.
- Do not assign both low level touch events and gestures (which may conflict with touch events) on same widget.
- `Map` and `Browser` widgets are not supposed to be kept in `Scroll container` as it may lead to gesture conflict. Any such gesture conflict may cause application rejection from Microsoft Store.

- onTouchMove event on any widget is executed asynchronously. Hence drag may not be smooth.
- AddgestureRecognizer will add the event every time it is called, hence same event may trigger several times if event is registered multiple times.
- The events onTouchStart, onTouchEnd, and onTouchMove may not work or yield desired results when registered on scrollable widgets such as FlexScrollContainer, Map, SegmentedUI, TabPane, TextArea, and Browser. These widgets internally uses the lower level touch events to get the scrolling behavior and may conflict with the externally registered touch events. On other non scrollable widgets these lower level events get fired along with the existing events as per the widget behavior. For example, onClick on Button widget get fired along with touch events if touch events are registered with the Button.

5.4.3 Animation Limitations Using Flex Layout

- % values for translate() in transform will not work. The Values passed as “100dp”, “100px”, “50%”, “90” will not work. The valid value will be like 90 etc.
- Calling rotate(), scale() and translate() multiple times will overwrite the previous values but not club or sum. Ex: rotate(45); rotate(50) is not equal to rotate(105), instead, it will be rotate(50).
- Skew effect observed in iOS and SPA will not happen for some animations (ex: scale in combination with rotate) in Windows 8.1 and Windows Phone 8/8.1.
- If scale, rotate, and translate are in transform object, the sequence applied in Windows Phone 8/8.1 and Windows 8.1 will be as follows irrespective of the order mentioned in transform.
 - Scale
 - Rotate
 - Translate
- Animation results may not be matching with other platforms when 2 dimensional or 2 positional properties are mentioned in step configuration for one widget when the values are given in percentage. Same is the case when parent and child are animated.

- Animation may start little slower in all windows channels when more steps are provided with more percentage values.
- Slight flickering of child widgets may happen when a child has % relationship with parent container and parent container is animating.
- Animation smoothness of any container depends on:
 - Number of direct children
 - Number of children with more levels of hierarchy
 - Number of children with % relationship
 - Images and their size.
- Windows always suggests to use transformation for animations instead of animating layout properties (Positional and dimensional).

5.4.4 Segment Animation Limitations

- For AutoGrow Segment, animations will get applied even for the rows which are not in the visible area. This is applicable for all methods.
- onRowDisplay method may not be supported on Windows 10.

5.5 SPA Limitations

Following are the limitations of SPA platform:

5.5.1 Flex Layout

- Setting enableScrolling property to false on Flex scroll container will not retain the scroll position and vertical and horizontal offset will be reset to 0,0 respectively. But if APILevel is 6000 then this will not reset to 0,0 respectively.
- onZoomStart, onZoomEnd, onZooming, and onDecelerationStarted are not supported in SPA.

- `zoomToRect` and `setZoomScale` APIs are not supported in SPA.
- `doLayout` is synchronous event in SPA unlike richclients.
- In Windows SPA `onScrolling` events will be triggered with more gaps in touch points, because of this scrolling may not be smooth.
- Though opacity is set to complete transparent, all events are consumed on the same widget and not the widget which is behind the original widget. Complete opaque doesn't mean that events to pass through it.
- Calling `forceLayout()` on a container triggers `doLayout()` on grandchild widgets which has a change in location or size but not on all grandchildren widgets.
- When complex forms have to be designed, it is suggested to turn off visibility for the widgets which are outside the view port or not required widgets. Enable the visibility when required. This causes application to load faster.
- Scroll events will not propagate in segmented UI, `TabPane` and other scrollable widgets which has `containerHeight` property set.
- `DataGrid` height is not fixed in SPA, it goes by its content.
- After text/data change in any widgets, widget dimension will get updated only after `forceLayout()` api on parent container or in next layout cycle.
- In SPA, implicit `forceLayout` will not happen at the end of the closure, developer has to call `forceLayout` to get changes reflected.
- Implicit `forceLayout` will happen in case of visibility change or changes in widget hierarchy (add/remove of widgets).
- Scrolling events are not fired after the touch release till `FlexScrollContainer` is decelerating. `onScrollEnd` event will be fired only after deceleration is complete.
- Shadow in focus skin for a widget will not work if widget is placed in `FlexContainer` or `FlexScrollContainer`.

- When you set `contentOffset` that is less than `containerWidth/containerHeight`, then offset point will come to view port, but will not add any extra space to get the offset point to top right corner of container.

5.5.2 Gesture Limitations Using Flex Layout

- Gesture conflicts should be avoided in application code such as Scroll Container in another scroll container with same scroll direction enabled.
- Touch events will not propagate in segmented UI, datagrid and other widgets which has `containerHeight` property set, as this will conflict with touch events to handle scroll in these widgets.
- Pan, pinch & zoom are supported in iPhone SPA (IOS 7 and above) only, but not in other SPA platforms.
- When both Tap and Double Tap gestures are set, both will trigger for each double tap. Tap is fired twice (one for each tap) on double tap.
- It is not advised to assign both low level touch events and gestures (which may conflict with touch events) on same widget.
- Map is not supposed to be kept in Scroll container since it might lead to gesture conflict.
- `onTouchMove` event on any widget is executed asynchronously. If there is complex operations in `onTouchMove` event then drag may not be smooth.
- `addGestureRecognizer` will add the event every time it is called hence same event may trigger several times if event is registered multiple times.
- In SPA Windows long press event is not supported.
- For Switch widget on safari browser will have constant width and height. In case of SPA Android and SPA Windows Switch will have constant height, but width may vary. Any additional width and height, which is more than required may leave the empty space on the right, bottom, and or left, top side of the switch widget.

- The events `onTouchStart`, `onTouchEnd`, and `onTouchMove` may not work or yield desired results when registered on scrollable widgets such as `FlexScrollContainer`, `Map`, `SegmentedUI`, `TabPane`, `TextArea`, and `Browser`. These widgets internally uses the lower level touch events to get the scrolling behavior and may conflict with the externally registered touch events. On other non scrollable widgets these lower level events get fired along with the existing events as per the widget behavior. For example, `onClick` on `Button` widget get fired along with touch events if touch events are registered with the `Button`.

5.5.3 Animation Limitations Using Flex Layout

- Default unit for input values in `translate()` API is `dp`. Other units are not supported.
- Animation may start little slower in all windows channels when more steps are provided with more percentage values.
- Slight flickering of child widgets may happen when child has % relationship with parent container and parent container is animating.
- Animation smoothness of any container depends on:
 - Number of direct children
 - Number of children with more levels of hierarchy
 - Number of children with % relations ship
 - Images and their size.
- If the widget skin contain gradient/multi gradient/image background then single color animation is not happening in SPA iPhone and SPA windows.
- In SPA windows, if opacity is changed from 1 to 0 in animation, at the end of animation, widget may blink and start the next iteration. This is native Internet Explorer limitation.
- If we apply animation on widget which is already animating then it cancels the previous animation, but widget model may not get updated with the final step configuration, even if fill

mode is forward/both.

- In the middle of the animation if you do the browser back, animation end event will not fire and widget model not update based on the animation fill mode.
- Animations on Image widget may not be accurate. It depend on image size, image load time.
- Positional and Dimensional properties are frame based animations. Whereas transform, background, opacity are Definition based animations. We are not deducing these properties from widget.
- For a Map widget, if you perform animation with width or height related properties, then map widget may not get the new or updated map from google.
- when you apply animation with combination of Rotate and Scale with Ease-In time function, the widget would flicker in IE browser.
- In SPA, when rotate animation is more than 180 degrees for all widgets, the direction may vary with different combinations of Scale and/or Translate values in IE browser.
- In Mac computers, full-page animation on Safari browser may not be as smooth as Chrome browser and there may be some flickering effect.
- In the SPA platform, the rotate3D animation may differ in IE(10/11) browser when compared to Chrome, Firefox, and Safari browsers for few scenarios & axis-combinations. Start and end frames remain same across all browsers but the rotate animation path differ for IE browsers. Few such cases are rotate3D(150, 0,1,0), rotate3D(150,1,1,1), and rotate3D(210,1,0,1).

5.5.4 Segment Animation Limitations

- For AutoGrow Segment, animations will get applied even for the rows which are not in the visible area. This is applicable for all methods.
- Calling the removeAt/removeSection At methods multiple times simultaneously unexpected behavior results. Because of this, the onRowDisplay method may be called in inconsistent order.

5.6 Flex Container Backward Compatibility

Note: You can add FlexContainer or FlexScrollContainer inside an HBox or a VBox or a Form through coding.

5.7 FlexContainer inside a Box Container

FlexContainer or FlexScrollContainer widgets support properties (containerWeight, hExpand, vExpand, widgetAlignment, and margin) required for box layout. FlexContainer or FlexScrollContainer widgets can be placed inside an HBox, a VBox or a Form with layoutType configured as `kony.flex.VBOX_LAYOUT`.

5.7.1 Positional Properties

Positional properties are not applicable to the FlexContainer, when a FlexContainer is placed inside a % HBox.

The following are the positional properties:

- left
- right
- top
- bottom
- centerX
- centerY

5.7.2 Dimensional Properties

Dimensional properties, when specified in % for a FlexContainer, will be treated as % relative to the parents (HBox) width.

You will notice the behavior of the dimensional properties when FlexContainer is placed inside a % HBox whose height is unknown.

The following are the dimensional properties:

- width
- height
- minHeight
- maxHeight
- minWidth
- maxWidth

5.7.3 FlexContainer in % HBox

FlexContainer and FlexScrollContainer widgets support the following % HBox related properties.

- containerWeight
- hExpand
- vExpand
- widgetAlignment
- margin

5.7.4 containerWeight vs Width

To understand how containerWeight and width properties work and their corresponding priorities, see the table below:

width and height of FlexContainer is specified as	If % HBox	Behavior
<p>containerWeight is more than the width</p> <div data-bbox="191 548 699 720" style="border: 1px solid #ccc; padding: 5px; background-color: #e6f2ff;"> <p>Note: If the width is not specified, then the preferredWidth of the FlexContainer will be considered for calculations.</p> </div>	hExpand = true	FlexContainer widget will expand to the given containerWeight in horizontal direction.
	hExpand=false	FlexContainer widget will retain the specified or derived width and will be placed horizontally inside the given containerWeight adhering to the widgetAlignment specified.
	vExpand = true	FlexContainer widget will expand to the height of the % HBox.
	vExpand = false	FlexContainer widget will be placed vertically inside the % HBox adhering to the widgetAlignment specified.

width and height of FlexContainer is specified as	If % HBox	Behavior
containerWidth is less than the width <div style="border: 1px solid #ccc; padding: 5px; background-color: #e6f2ff;"> <p>Note: If the width is not specified, then the preferredWidth of the FlexContainer will be considered for calculations.</p> </div>	hExpand = true	FlexContainer widget width is reduced to fit within the given containerWeight. The widgetAlignment property in the horizontal direction will not have any effect.
	hExpand=false	FlexContainer widget width is reduced to fit within the given containerWeight. The widgetAlignment property in the horizontal direction will not have any effect.
	vExpand = true	FlexContainer widget will expand to the height of the % HBox.
	vExpand = false	FlexContainer widget will be placed vertically inside the % HBox adhering to the widgetAlignment specified.

5.7.5 FlexContainer in non % HBox

The width and height specified are considered as the width and height of the FlexContainer widget. If the values are not specified, it will fall back to preferredWidth and preferredHeight.

It is similar to any other widget placed in non % HBox. FlexContainer widget when placed inside non % HBox will not have any effect of hExpand, vExpand, and widgetAlignment properties.

5.7.6 FlexContainer in VBox or Form with layoutType configured as kony.flex.VBOX_LAYOUT

The width and height specified are considered as the width and height of the FlexContainer widget. If the values are not specified, it will fall back to preferredWidth and preferredHeight.

The width of the FlexContainer widget stretches to the width of the parent, if hExpand property is configured as true. There is no effect on a FlexContainer widget, if vExpand property is configured as true.

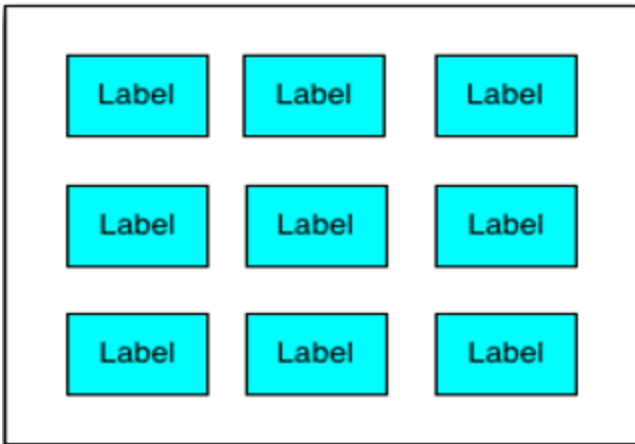
5.8 Box Container inside a FlexContainer

The widgets HBox and VBox are not supported inside the FlexContainer. We recommend you to use FlexContainer with layoutType configured as kony.flex.FLOW_VERTICAL instead of VBox and HBox.

5.9 Flex Container Pseudocode Examples

Below are some of the Pseudocode examples and their images:

5.9.1 Building a grid of widgets



```
//Sample code to build a grid of widgets.
var c = new kony.ui.FlexContainer();
c.setDefaultUnit(dp);
c.width = 170;
c.height = 170;

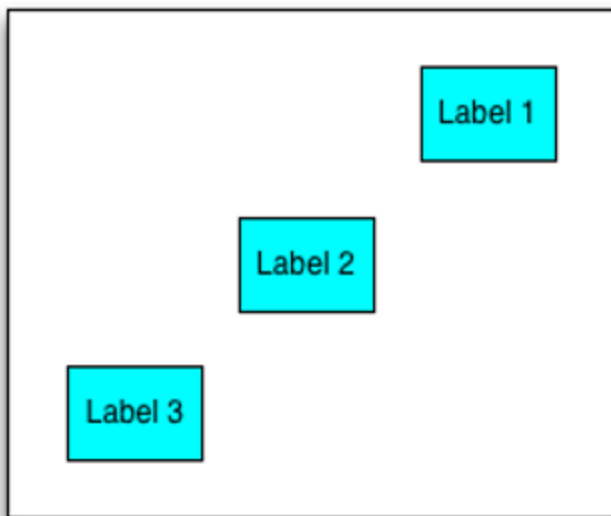
wWidth = 30;
wHeight = 20;
wSpacing = 20;

for (int I = 0; I < 3; I++ )
{
    for (int J = 0; J < 3 ; J++)
    {
        var label = new Label("id", "Label")

        label.width = wWidth;
        label.height = wHeight;
```

```
        label.top = (I + 1)* spacing + I* wHeight;  
        label.left = (J + 1)* spacing + J * wWidth;  
        c.add(label);  
    }  
}
```

5.9.2 Position the widgets diagonally



```
//Sample code to position the widgets diagonally.  
var c = new kony.ui.FlexContainer();  
c.setDefaultUnit(dp);  
  
var label1 = new Label("id", "Label1")  
c.add(label1);  
var label2 = new Label("id", "Label2")  
c.add(label2);  
var label3 = new Label("id", "Label3")  
c.add(label3);
```

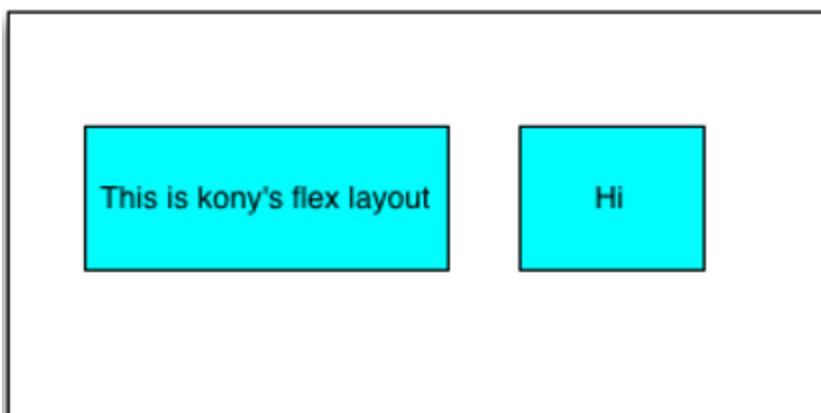
```
c.doLayout = function()
{
  wWidth = 30;
  wHeight = 20;
  wSpacing = 20;

  label1.width = label2.width = label3.width = wWidth;
  label1.height = label2.height = label3.height = wHeight;
  label1.right = 20
  label1.top = 20;

  label2.left = c.frame.width/2 - wWidth/2
  label2.top = c.frame.height/2 - wHeight/2

  label3.left = 20;
  label3.bottom = 20;
}
```

5.9.3 Position the widgets relative to siblings



```
//Sample code to position the widgets relative to siblings
var c = new kony.ui.FlexContainer();

var label1 = new Label("Hi")
c.add(label1);

var label2 = new Label("Hi")
c.add(label2);

label1.left = 20;
label1.top = 30
label1.width = kony.flex.USE_PREFERRED_SIZE
label1.height = 40

label1.doLayout = function(){

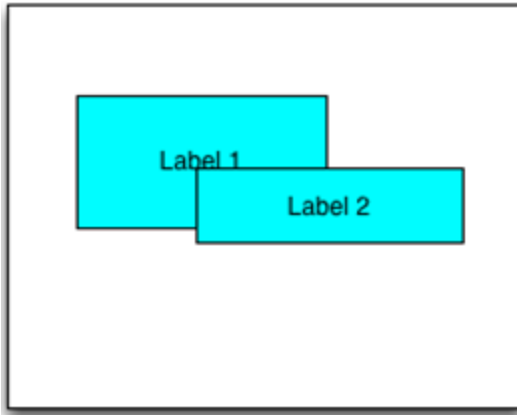
var lab = this.parent.label2; // accessing the child's through
parent

lab.left = 20 + label1.frame.width /2;
lab.left.top = 30;

label2.width = label1.frame.width /2
label2. height = 40

}
```

5.9.4 Overlapping the widgets using zIndex



```
//Sample code to overlap the widgets using zIndex
var label1 = new Label("id", "Label1")
label1.zIndex = 1;
c.add(label1);

var label2 = new Label("id", "Label2")
label2.zIndex = 2;
c.add(label2);

label1.left = 20;
label1.top = 30

label1.width = 60
label1.height = 50

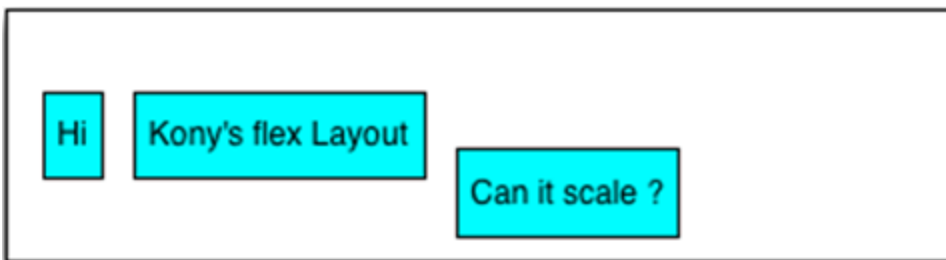
label1.doLayout = function(){

label2.left = 20 + label1.frame.width /2;
label2.top = 30 + label1.frame.height/2
```

```
label2.width    = 80
label2.height  = 25

}
```

5.9.5 Positioning the widgets horizontally



```
//Sample code to position the widgets horizontally
var c = new kony.ui.FlexContainer();
//set the container layout type as Kony.Flex.FLOW_HORIZONTAL;

var label1 = new Label("id", "Hi")
c.add(label1);

var label2 = new Label("id", "Kony's Flex Layout")
c.add(label2);

var label3 = new Label("id", "Can it scale ?")
c.add(label3);

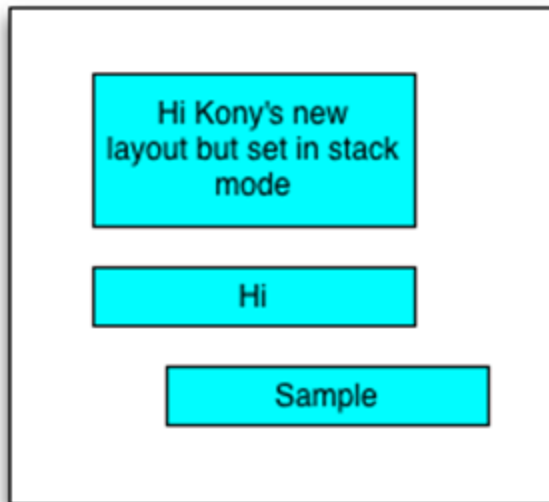
label1.top = label2.top = 20;
label3.top = 30

label1.left = label2.left = label3.left = 5;
```

```
label1.width = label2.width = label3.width = kony.flex.USE_  
PREFERRED_SIZE
```

```
label1.height = label2.height = label3.height = 50
```

5.9.6 Stacking the widgets vertically



```
//Sample code to stack the widgets vertically  
var c = new kony.ui.FlexContainer();  
//set the container layout type as Kony.Flex.FLOW_VERTICAL;  
  
var label1 = new Label("id", "Hi Kony's new layout but set in stack  
mode")  
c.add(label1);  
  
var label2 = new Label("id", "Hi")  
c.add(label2);
```

```
var label3 = new Label("id", "Sample")
c.add(label2);

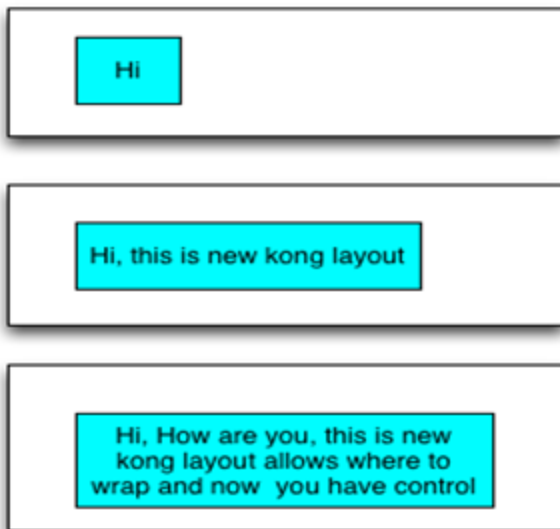
label1.left = label2.left = 30;
label3.left = 40

label1.top = label2.top = label3.top = 5;

label1.width = label2.width = label3.width = 60

label1.height = label2.height = label3.height = kony.flex.USE_
PREFERRED_SIZE
```

5.9.7 Wrapping the text when it reaches the specified width



```
//Sample code to wrap the text when it reaches the specified width
var c = new kony.ui.FlexContainer();
```

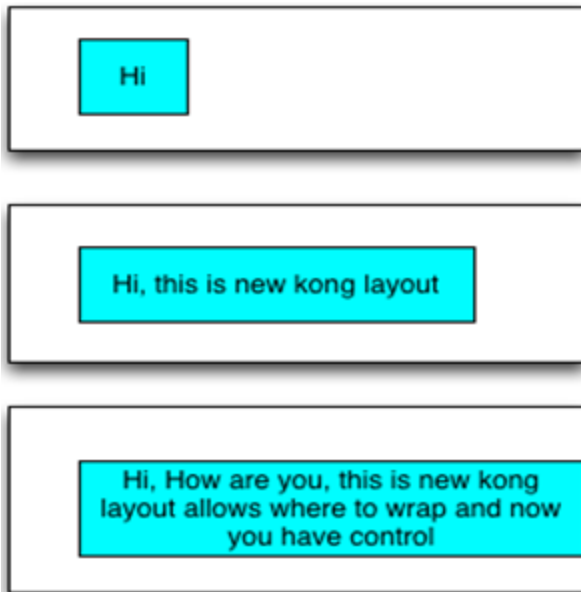


```
var label1 = new Label("id", "text")
c.add(label1);

label1.text = // Data is fed from net work

label1.left = 20;
label1.top = 20;
label1.maxWidth = 80
label1.height = kony.flex.USE_PREFERRED_SIZE
```

5.9.8 Wrapping the text when it reaches the parent boundaries



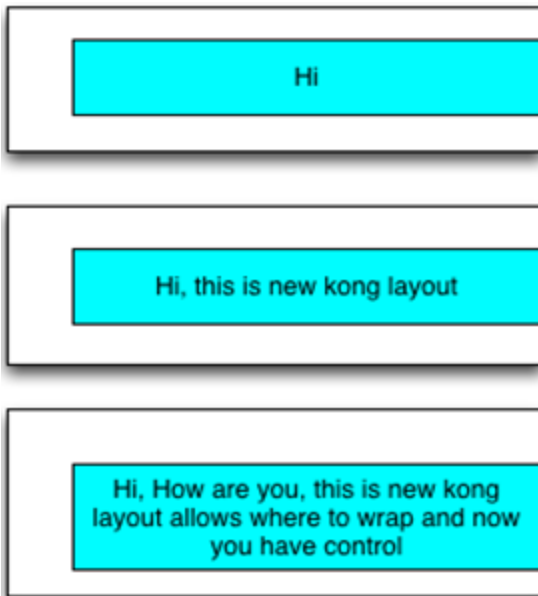
```
//Sample code to wrap the text when it reaches the parent boundaries
var c = new kony.ui.FlexContainer();

var label1 = new Label("id", "text")
c.add(label1);
```

```
label1.text = // Data is fed from net work

c.doLayout = function()
{
label1.left = 20;
label1.top = 20;
label1.maxWidth = c.frame.width - 20
label1.height = kony.fLex.USE_PREFERRED_SIZE;
}
```

5.9.9 Widget occupying the available horizontal space



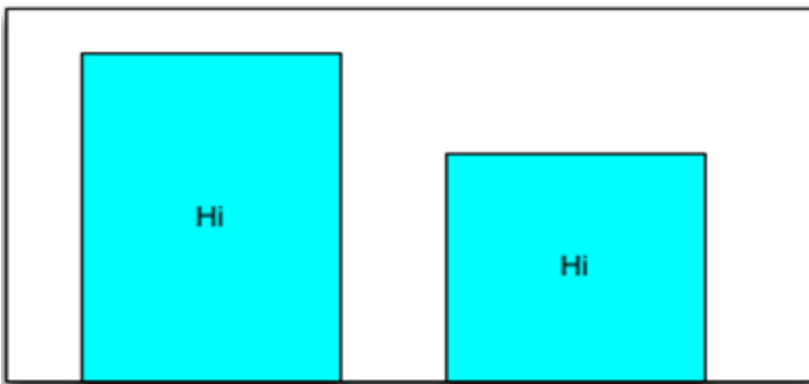
```
//Sample code of widget occupying the available horizontal space
var c = new kony.ui.FlexContainer();

var label1 = new Label("id", "text")
c.add(label1);
```

```
label1.text = // Data is fed from net work

label1.left = 20;
label1.top = 20;
label1.right = 0;
label1.height = kony.flex.USE_PREFERRED_SIZE
```

5.9.10 Widget occupying the available vertical space



```
//Sample code of widget occupying the available vertical space
var c = new kony.ui.FlexContainer();

var label1 = new Label("id", "Hi")
c.add(label1);

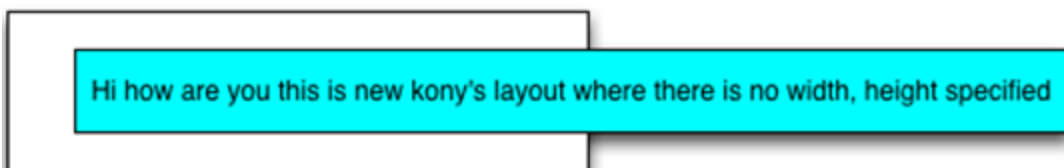
var label2 = new Label("id", "Hi")
c.add(label2);

label1.left = 20;
label1.top = 20;
```

```
label2.right = 30;
label2.top = 50;

label1.bottom = 0;
label1.width = 50;
```

5.9.11 widget to occupy its preferred size without any given height or width



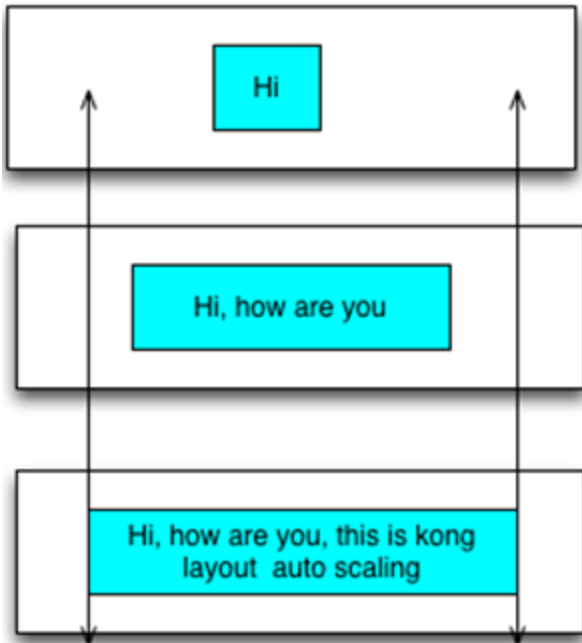
```
//Sample code of a widget to occupy its preferred size without any
given height or width
var c = new kony.ui.FlexContainer();
c.clipBounds = false;

var label1 = new Label("id", "text")
c.add(label1);

label1.text = // Data is fed from net work

label1.left = 20;
label1.top = 20;
```

5.9.12 Widget position in center and offset from left and right boundaries



```
//Sample code of widget position in center with width to grow
maximum and with a minimum
offset from left and right boundaries.
var c = new kony.ui.FlexContainer();

var labell = new Label("id", "text")
c.add(labell);

labell.text = // Data is fed from net work
labell.minWidth = 20;
labell.maxWidth = 80;
labell.height = kony.flex.USE_PREFERRED_SIZE

labell.center = {80,80};
```

5.10 Widget Level Animation Using Flex Forms

Widget Level Animation feature enables you to animate widgets placed in a container (for example, Form). This feature supports 2D and 3D animations and transformations. The 2D animations are supported on iOS, Android, Windows, and SPA platforms and the 3D animations are supported on iOS, SPA and Android (limited support) for the applications created in JavaScript language using the Kony Platform.

When the transformation matrix is identity, the 3D directions are as follows:

- x-axis is towards the right of the widget
- y-axis is towards bottom of the widget
- z-axis is coming out of the screen

Note: 3D animations should be applied on widgets when they are in normal state. Otherwise it may have undefined behavior.

The animation is classified into the following sub classes:

- [Key Frame Animations](#)
- [Animation Properties Events and Methods](#)

5.11 Key Frame Animations

Key frame animations allows you to define animations by changing the widget level properties.

5.11.1 Key Frame Animation Definition

Key frame definition consists of one or more properties of the widget along with changes in their values and the steps (key frame) involved in changing the value from the initial value to final value. Each step from initial value to the final value can be configured with an animation behavior.

Key frame animation can be defined dictionary of steps with each step configured again using a JavaScript Object (Dictionary). Step-config is optional parameter. Following is the syntax:

Note: If the animation definition does not follow the above syntax then animate API throws Invalid Animation Definition exception.

```
{  
  
  <step>: {  
  
    <widget_property> : <value> ,  
  
    <widget_property> : <value> ,  
  
    ...  
  
    stepConfig : { ... }  
  }  
  
  ,  
  
  <step>: {  
  
    <widget_property> : <value> ,  
  
    <widget_property> : <value> ,  
  
    ...  
  
    stepConfig : { ... }  
  
  }  
  
  ,  
  
  ...  
}
```

```
}
```

5.12 References

APIs does not completely adhere to the CSS3 specifications but is influenced by CSS3 animations and transformations.

The following URLs help as a reference:

- <http://dev.w3.org/csswg/css3-animations/>
- <http://dev.w3.org/csswg/css3-2d-transforms/>
- <http://cubic-bezier.com/>
- <http://dev.w3.org/csswg/css3-2d-transforms/>
- <http://dev.w3.org/csswg/css3-3d-transforms/>

5.13 Animation Properties Events and Methods

Following are the widget level properties, events, and methods that are allowed to be part of the animation definition.

animate

This method applies the animation to the widget immediately, if the widget is part of the currently visible view hierarchy. If widget is not part of the currently visible view hierarchy then this API invocation is ignored. This method is asynchronous and immediately returns and do not wait for the animation to start or complete.

All the animation callbacks will receive animate as a second parameter.

Signature

```
animate(  
  animationObj,  
  animationConfig,  
  animationCallbacks)
```

Input Parameters

animationObj

An object defined using `kony.ui.createAnimation()` API. Refer `kony.ui.createAnimation` for more details.

animationConfig

Optional. As defined in [Animation Configuration](#) section.

animationCallbacks[JSObject]

Optional. A dictionary represents JavaScript functions that work as animation call backs. Following are the animation callbacks:

- `animationStart` (source, animationHandle, elapsedTime): This event occurs at the start of the animation. If there is 'animation-delay' configured then this event will fire only after the delay period. This event gets called asynchronously.
- `animationEnd` (source, animationHandle, elapsedTime): The `animationEnd` event occurs when the animation finishes. This event gets called asynchronously.

source: A widget that is being animated.

animationHandle: A handle returned by `applyAnimation` method.

elapsedTime: The amount of time the animation has been running in seconds, when this event is fired.

Return Values

Returns a handle to the animation, that is platform-defined object. This handle is of no use for now, but is returned for the future requirements such as cancellation of animations etc.

Remarks

This method throws Invalid Animation Definition Exception if animation definition, does not follow the dictionary structure expected. This method is ignored if this is called on widget whose immediate parent is not FlexForm, FlexContainer or FlexScrollContainer.

Example

```
//Sample code of animation

function AnimateBoth()
{
    var getFuncName = frm1.listbox18.selectedKey;

    if(getFuncName == "BothLT") {
        frm1.textbox26.animate(myAnimDefinition(), animConfiguration(), {});
    }
    else if(getFuncName == "BothTBL"){
        frm1.textbox26.animate(myAnimDefinitionscl(), animConfiguration(), {});
    }
}
}
```

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

kony.ui.createAnimation

This method creates an animation object that can be used to animate the widgets using animate API.

Signature

```
kony.ui.createAnimation (
    animationDef)
```

Input Parameters

animationDef [Number]

A sample animation definition.

```
{
  <step>: {
    <widget_property>: <value>,
    <widget_property>: <value>,
    ---
    stepConfig: {...}
  }
  <step>: {
    <widget_property>: <value>,
    <widget_property>: <value>,
    ---
    stepConfig: {...}
  },
  ---
}
```

Return Values

None

Exceptions

WidgetError

Remarks

Values cannot be specified using pixels.

Example

```
//Sample code of animation

function myAnimDefinition()
{
    var animDefinition =
    {
        0 :
        {
            "width":50,
            "left":0
        },
        100 :
        {
            "width":50,
            "left":90
        }
    } ;
    animDef = kony.ui.createAnimation(animDefinition);
    return animDef;
}
```

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

anchorPoint

Specifies the anchor point of the widget bounds rectangle using the widgets coordinate space. The possible values are dictionary with x,y as possible keys with the values as numbers ranging from 0 to 1. All geometric manipulations to the widget occur about the specified point. For example, applying a rotation transform to a widget with the default anchor point causes the widget to rotate around its center.

Syntax

```
anchorPoint
```

Type

JSObject

Permissions

Read + Write

Remarks

The default value for this property is center ({ "x":0.5, "y":0.5}), that represents the center of the widgets bounds rectangle. The behavior is undefined if the values are outside the range zero (0) to one (1).

Availability

Not available in the IDE.

- iOS
- Android
- Windows
- SPA

transform

This property is set to identify transform by default. Any transformations applied to the widget, occur relative to the widget anchor point. Values that can be set to this property must be created using `kony.ui.makeAffineTransform`.

Syntax

```
transform
```

Type

JSObject

Permissions

Read + Write

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;
}
```

Availability

Not available in the IDE.

- iOS
- Android
- Windows
- SPA

`kony.ui.makeAffineTransform`

This method creates a 2D transform object. A 2D transform object can be used to scale, translate, and rotate the widgets in a two-dimensional space.

Signature

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

Input Parameters

None

Return Values

It returns the identity transform.

Exceptions

WidgetError

Remarks

Irrespective of the invocation order of operations, `kony.ui.makeAffineTransform` method applies the transform in the sequence of scale, translate, and rotate operations.

Example

```
//Sample code of animation
function animDeftranslate() {
    var transformProp1 = kony.ui.makeAffineTransform();
```

```
transformProp1.translate(10, 10);
var transformProp2 = kony.ui.makeAffineTransform();
transformProp2.translate(20, 20);
var transformProp3 = kony.ui.makeAffineTransform();
transformProp3.translate(30, 30);
var animDefinitionOne = {
    0: {
        "anchorPoint": {
            "x": 0.5,
            "y": 0.5
        },
        "transform": transformProp1
    },
    50: {
        "anchorPoint": {
            "x": 0.5,
            "y": 0.5
        },
        "transform": transformProp2
    },
    100: {
        "anchorPoint": {
            "x": 0.5,
            "y": 0.5
        },
        "transform": transformProp3
    }
}
animDef = kony.ui.createAnimation(animDefinitionOne);
return animDef;
}

Function getParent() {

    Var result = this.parent;
```



```
}
```

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

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.

Signature

```
rotate (angle)
```

Input Parameters

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.

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.

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;  
}
```

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)
```

Parameters

angle

Specify the angle, by which a widget to be rotated around rx, ry, and rz axes.

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.

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 180° to -180°. 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.

Example

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

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 JSONObject with keys `sx` and `sy` and allow numbers only.

Signature

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

Input Parameters

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.

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.

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;
}
```

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

scale3D ()

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

Syntax

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

Parameters

sx

Specify the value to be scaled in the x direction.

sy

Specify the value to be scaled in the y direction.

sz

Specify the value to be scaled in the z direction.

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 within 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.

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;
```

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.

Signature

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

Input Parameters

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.

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.

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;
}
}
```

Availability

- iOS
- Android/Android Tablet
- Windows
- SPA

translate3D ()

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

Syntax

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

Parameters

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.

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.

Example

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

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)
```

Parameters*distanceOfViewerToPlane*

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

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.

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;
```

Availability

Available in the IDE

iOS

SPA

5.13.1 Widget Skin Properties

Following widget skin properties can be animated:

backgroundColor

Specifies the background color of the widget in hex format.

Syntax

```
backgroundColor
```

Type

String

Permissions

Read + Write

Remarks

There is no default value. It accepts 6 dig or 8 digit with alpha position are allowed. For example, ffffff or ffffff00.

When the 4-byte color format (RGBA) string is used, an alpha (A) value of FF specifies that the color is transparent. If the value is 00, the color is opaque. For example, red complete opaque is FF000000. Red complete transparent is FF0000FF. This change is made for backward compatibility.

Note: This property has more priority compared to the values coming from the configured skin. The values 0x and # are not allowed in the hex format.

Note: Initial value of backgroundColor has to be specified explicitly. If not, platform will not deduce the values from the existing skin and will lead to undefined behavior.

Availability

Not available in the IDE.

- iOS
- Android
- Windows
- SPA

opacity

Specifies the opacity of the widget. The value of this property must be in the range 0.0 (transparent) to 1.0 (opaque). Any values outside this range are fixed to the nearest minimum or maximum value.

Syntax

```
opacity
```

Type

Number

Permissions

Read + Write

Remarks

This property has more priority compared to the values coming from the configured skin.

Availability

Not available in the IDE.

- iOS
- Android
- Windows
- SPA

5.14 Animation Configurations

Animation configuration specifies the render behavior of an object during the animation. For example, you can specify how long an animation occur or the direction the animation should occur. The animation configurations are part of overall animation and not the step level configuration.

Following are the components of animation configuration:

duration

This property defines the time in seconds that an animation takes to complete one cycle. This is overall animation level 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 animation is instantaneous. However, there will not be visible animation changes, technically animation occurs and all animation callbacks get triggered.

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

iterationCount

This property 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 time view is live in the current hierarchy.

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

direction

This property 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 are of odd counts are played in the normal direction, and the animation cycle iterations that are even counts are played in a reverse direction.

Note that this is overall animation level configuration and not the step level configuration.

Note: Values will be specified as a string containing one of the above values and any other values will be ignored and default is applied or may lead to undefined behavior.

delay

This property 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.

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. Default value is zero and any negative or invalid values will default this property to zero.

Note: This is overall animation level configuration and not the step level configuration.

fillMode

This property defines what values are applied to the widget state by the animation outside the time it is executing.

Following are the options:

- `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.

- `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.
- `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.
- `kony.anim.FILL_MODE_NONE` (default): 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.

Following is the table showing the behavior of the animatable properties when queried during or at the end of the animation.

Fill-mode	In delay state	Animation states	Final state	Model Update
Direction: None				
None	Y	RGB	Y	No update
Forwards	Y	RGB	B	A.E.A update with 100th step
Backwards	R	RGB	Y	A.B.A update with 0th step, A.E.A update with initial value
Both	R	RGB	B	A.B.A update with 0th step, A.E.A update with 100th step
Direction: Alternate, Iteration Count = even value				
None	Y	RGB => BGR	Y	No update
Forwards	Y	RGB => BGR	R	A.E.A update with 0th step

Fill-mode	In delay state	Animation states	Final state	Model Update
Backwards	R	RGB => BGR	Y	A.B.A update with 0th step, A.E.A update with initial value
Both	R	RGB => BGR	R	A.B.A update with 0th step, A.E.A update with 0th step
Direction: : Alternate, Iteration Count = odd value				
None	Y	RGB => BGR => RGB	Y	No update
Forwards	Y	RGB => BGR => RGB	B	A.E.A update with 100th step
Backwards	R	RGB => BGR => RGB	Y	A.B.A update with 0th step, A.E.A update with initial value
Both	R	RGB => BGR => RGB	B	A.B.A update with 0th step, A.E.A update with 100th step

Note: This is overall animation level configuration and not the step level configuration. Values will be specified as string containing one of the above values and any other values would be ignored or may lead to undefined behavior.

Example

```
function animConfig(){
  var config = {
    "duration":1,
```

```
"iterationCount":1,  
"delay":0,  
"fillMode":kony.anim.FILL_MODE_FORWARDS  
};  
return config;  
}
```

5.15 Applying Animations

Every widget provides animate API to animate the widgets.

Following are the types of animations you can apply:

1. [Sequential and Parallel Animations](#)
2. [Querying Widget Properties](#)
3. [Layout Callbacks during Animation](#)
4. [Flex Container and Child Widgets](#)
5. [Multiple Parallel Animations](#)
6. [Interactions on the Widget during Animation](#)

5.16 Sequential and Parallel Animations

To sequence the animations one after the other, animation events have to be used. You can start a new animation at the end of the existing animation using `animationEnd` event.

All animations initiated by animate API gets executed asynchronously. Essentially calling animate API on widgets sequentially one after the other leads to parallel execution of the animations.

5.17 Querying Widget Properties

Following is the table showing the behavior of the animatable properties when queried during or at the end of the animation. Only when the model is updated (last column) then the values are available when queried.

Fill-mode	In delay state	Animation states	Final state	Model Update
Direction: None				
None	Y	RGB	Y	No update
Forwards	Y	RGB	B	A.E.A update with 100th step
Backwards	R	RGB	Y	A.B.A update with 0th step, A.E.A update with initial value
Both	R	RGB	B	A.B.A update with 0th step, A.E.A update with 100th step
Direction: Alternate, Iteration Count = even value				
None	Y	RGB => BGR	Y	No update
Forwards	Y	RGB => BGR	R	A.E.A update with 0th step
Backwards	R	RGB => BGR	Y	A.B.A update with 0th step, A.E.A update with initial value
Both	R	RGB => BGR	R	A.B.A update with 0th step, A.E.A update with 0th step

Fill-mode	In delay state	Animation states	Final state	Model Update
Direction: : Alternate, Iteration Count = odd value				
None	Y	RGB => BGR => RGB	Y	No update
Forwards	Y	RGB => BGR => RGB	B	A.E.A update with 100th step
Backwards	R	RGB => BGR => RGB	Y	A.B.A update with 0th step, A.E.A update with initial value
Both	R	RGB => BGR => RGB	B	A.B.A update with 0th step, A.E.A update with 100th step

5.18 Layout Callbacks during Animation

doLayout callbacks are not guaranteed to be called in synchronization with steps configured in animation and may not get called in any determined fashion. It is suggested to unhook any layout events during the animations.

5.19 Flex Container and Child Widgets

When dimensional and positional properties of the flex container are animated then all child widgets sharing percentage (%) relationship with the parent also gets animated. Percentage (%) Relationship between parent and child can be established by specifying one or positional, dimensional properties of the widgets in percentage (%) units.

Actual animation that child widget goes through depends on the property that is animated on the container and the property of the child widget that shares the percentage (%) relationship with the parent.

If there is no percentage (%) relationship between the parent and child then child will not go through any animation as parent gets animated.

5.20 Multiple Parallel Animations

Parallel animations on widgets that do not have any dependencies on other widgets (for example, widgets can share dependency through parent and child widgets with percentage % relationship or the widgets inside `HORIZONTAL_FLOW`, `VERTICAL_FLOW` share dependency on siblings) will work across platforms consistently without any issues.

- Parallel animations on multiple widgets, that do not share any relationship, must not have any issues, and must work consistently.
- Parallel animations on widgets, that share dependency, may lead to inconsistent results and should be avoided.
- Parallel animations on the same widget, for example, calling `animate` method on the same widget, and then the first animation gets canceled due to the second animation.
- Parallel animations cancellation may lead to undefined behavior and `animation-fill-mode` property may not work as expected.

Note: Any implicit animations due to widget dependency are not treated as parallel animations.

5.21 Interactions on the Widget during Animation

Interacting with the widget during animation may also lead to undefined behaviors. Behavior depends on underlying platform.

As a guideline, developers must avoid writing code that changes the widget properties if they are being animated.

Note: For example, changing properties of the widget immediately after `widget.animate()` may lead to undefined behaviors. Ideally, any changes on the widget must happen after animation, in `animationEnd()` event.

6. Platform Specific Limitations

This section lists the limitations, properties or the widgets not supported by platforms.

6.1 Desktop Web Limitations

This section lists the properties that are not supported by the Desktop Web platform.

- ComboBox and ListBox, skin styles "Transparent" and "One Color" are supported in background color tab.
- ComboBox and ListBox, browser does not support if the properties defined in font tab and border are different for **skin** and **focusSkin**.
- On Firefox browser, TextBox and TextArea widgets does not support percentage (%) based padding, while other browsers does support.
- For all widgets in Internet Explorer 7 and 8, transparency (border/font) is not supported for skin.
- On safari browser, ListBox and ComboBox widgets does not support padding.
- Rounded Corners will not work for all widgets in Internet Explorer 8 because of border-radius property is not supported in Internet Explorer 8 and its lower versions.
- Vertical split and Horizontal split will not work for all widgets in Internet Explorer 9 and its lower versions.
- For non-modal popups (isModal = false), popup transparency (transparencyBehindThePopup) property is not applied as the background widgets are accessible to the user.
- A valid calendar year selection range is from 1900 to 2099. If you select an year beyond the range shows an alert message (you can customize this error message).
- In Internet Explorer 8 and below browsers do not support all geolocation APIs.

- focusSkin applied to the container widgets (like HBox, VBox, Segment) is not inherited by the inner widgets in IE browsers (IE8, IE9, IE10). To overcome this apply focusSkin at every widget inside the container widget.
- For ScrollBox and TabPane widgets, angle background Multi Step Gradient is not supported.
- Desktop Web platform does not support browser (Internet Explorer 8) running in compatibility mode.
- Vertical gradient and Horizontal gradient are supported for all widgets in Internet Explorer 8 and above versions.
- Preview of map widget is not supported.
- On Internet Explorer browsers, focusSkin applied to the widgets CheckBox and RadioButton will work on click of text, but not on icon.
- For Browser widget, Desktop Web platform supports BROWSER_REQUEST_METHOD_GET option only.
- Video widget in print API is not supported in Firefox browser.
- To apply focusSkin for dynamically created widgets, assign focusSkin dynamically after creating the widget.

```
formid.widgetid.focusSkin = "skinname";
```

- To apply hoverSkin for dynamically created widgets, assign hoverSkin dynamically after creating the widget.

```
formid.widgetid.hoverSkin = "skinname";
```

- In Desktop Web platform, only left, right, and center alignment options can be applied to the content. This limitation is applicable for all widgets.
- For Browser widget, resetting of URL does not work.
- In Desktop Web platform, nested containers in a non-percentage HBOX are not supported.

6.2 SPA Limitations

This section lists the properties that are not supported by the SPA platform.

- `focusSkin` is not supported in Windows NTH and Android devices.
- For Horizontal Image strip, the `stripview` and `slot view` are not supported. If the images are of different size, It is mandatory to mention the width and height property. Else, the alignment of the images may get disturbed on the screen.
- Only static maps are supported on Windows Phone 7.5 and BlackBerry NTH.
- The widgets `Slider`, `Chart2D3D`, and `PickerView` widgets are not supported by both the Windows and BlackBerry NTH devices.
- Opacity should not be applied to form background for Windows Phone 7.5 and may lead to erroneous results.
- The property `secureTextEntry` for `textarea` is not supported in IE (desktop and mobile).
- `HBox` position attribute is not supported in SPA (mobile and desktop). Instead use for header / footer to dock elements.
- `showLoadingScreen()` should be preferred over `blockedUI`, as `blockedUI` cannot cater to multiple service calls which may be either chained or nested depending upon the application logic.
- `setContext` for `popup`, `dockable header / footer /appmenu` is not supported on Windows Phone 7.5 since it doesn't support absolute positioned elements.
- A valid calendar year selection range is from 1900 to 2099. If you select an year beyond the range shows an alert message (you can customize this error message).
- On SPA (Windows 8 and Windows Tablet devices) platform, `focusSkin` applied to the widgets `HBox`, `VBox`, `Calendar`, `ComboBox`, `ListBox`, and `SegmentedUI` is not inherited by the inner widgets in IE browsers (IE8, IE9, IE10). To overcome this apply `focusSkin` at every widget inside the container widget.

- Multi Step Gradient is not supported for all widgets on Windows Mango (IE9) devices.
- ScrollBox does not support Blur radius option for skins on BB and BB NTH devices.
- Preview of map widget is not supported.
- On Windows device browsers, focusSkin applied to the widgets CheckBox and RadioButton will work on click of text, but not on icon.
- For Browser widget, SPA platform supports BROWSER_REQUEST_METHOD_GET option only.
- For calendar widget, font color for input cannot be changed for Windows Phone 7.5 (Mango) platform.
- On SPA platform, <script> tag is not supported.
- To apply focusSkin for dynamically created widgets, assign focusSkin dynamically after creating the widget.

```
formid.widgetid.focusSkin = "skinname";
```

- On SPA (iOS devices) platform, when accessibility is set for FlexContainer, the FlexContainer's child widgets will not be focused. Only the FlexContainer Widget will be focused.
- In SPA platform, only left, right, and center alignment options can be applied to the content. This limitation is applicable for all widgets.
- For Browser widget, resetting of URL does not work.
- In SPA platform, nested containers in a non-percentage HBOX are not supported.

6.3 Windows Kiosk

This section lists the properties that are not supported by Windows 7 Kiosk platform.

- The widgets ObjectSelector3D, Phone, pickerView, Switch, MenuItem and Video are not supported.
- As of today (10th, July 2013) Windows 7 Kiosk applications run only on Windows 8 PRO and not on Windows 8 RT.
- The application icon that is set from Application Properties > Common > Desktop icon size should be multiple of 8 pixel and less than 256 pixel. For example, the icon image should be of size 8x8 or 16x16 px, it should be not 16x17 px.
- Windows/Kiosk platform does not support Segment Pageview.
- In the Kiosk platform, the panning mode (touch) happens only when scrolling direction is in both horizontal and vertical. Because to this, the Scroll Indicators are enabled based on the scroll direction.

6.4 BlackBerry 10

This section lists the limitations and properties that are not supported by BlackBerry 10 platform.

- BlackBerry platform does not support the following widgets:
 - pickerView
 - TabPane
- Gradient skins are not supported on any widgets.
- The BlackBerry 10 supports application version only if the format is specified as x.x.x (For example, 2.3.6). The Build generation fails if you specify any other version format.
- Only three options (WIDGET_ALIGN_TOP_LEFT, WIDGET_ALIGN_CENTER, and WIDGET_ALIGN_BOTTOM_RIGHT) of the widgetAlignment is supported by respective widgets.
- The layout property *hExpand* is always *true* for respective widgets and there is no effect when you set as *false*.

- You application crashes- when an event is invoked dynamically by assigning a JSONObject. For example, the below code will not work.

```
form.button.onClick = callback method()  
//The callback method is a JSONObject
```

- In segmentedUI, you cannot change *sectionHeaderTemplate* and *rowTemplate* dynamically. For example, the below code will not work.

```
form1.segment1.rowTemplate = template1
```

- In calendar widget, if you use the method *setenabled*, the date gets cleared and *validStartDate* is displayed. If *validStartDate* is not set then current date is displayed.
- The property *focusSkin* is not supported for TextBox widget.
- The font style with underline is not supported for TextBox widget.
- Skin font style with underline is supported only for widgets RichText and Link.
- All BlackBerry 10 supported widget events are not writable.
- Following are the Map widget limitations:
 - From Kony Visualizer, you must set the permission for *access_location_service* as *true*. To set *access_location_service*, navigate to Application Properties > Native > BlackBerry10, select *access_location_services* and click **Add >** and then click **Finish**.
 - Your device location service setting must be on. To set device location service in your device, navigate to Device Settings > Location Services > turn on the location services.
 - BlackBerry Native maps are supported, but map key and provider not applicable.
 - The Map widget is available in the United States and Canada regions. It will not work in Asia Pacific region.

- The Map widget works with BlackBerry 10 OS version 10.0.9.2709 and above.
- If the network is slow, then rendering of the map is not smooth. The fonts and the user interface (UI) may be affected.
- For devices earlier than 10.1, a developer- specified or custom pin image is not displayed. Only BlackBerry 10 provided images can be displayed.
- Rendering of Map may takes 2 to 3 minutes or sometimes more than 5 minutes depending on your network.
- Templates for Map widget are not supported.
- The default pin is always shown by the BlackBerry 10 device.
- When a Map is loading you cannot display any alert messages as "Map is loading".
- Your application may crash when you perform any action while loading a map.
- Zoom level is decided by altitude. Hence user has to provide zoom level in terms of 1000. The default zoomlevel is 10000.
- Events associated with respective widgets are not writable.
- Following are the CheckBox widget limitations:
 - By default the *itemOrientation* of a CheckBox widget is set to vertical. Unlike other platforms, the BlackBerry 10 platform does not support the horizontal orientation.
 - When you define a skin for normal skin or focus skin, the options font style, font size, font color are applied to the text of the CheckBox. They are not applicable to CheckBox image.
- Following are the Button widget limitations:
 - Word Wrapping and Padding properties are not supported for Native button. Image button supports Word Wrapping.

- Rounded corner for borders and background is not supported. You can achieve this behavior using Image button.
- When you define a skin for normal skin or focus skin, the options font style, font size, font color are applied to the text of the button in Image Button. They are not applicable to Native Button widget.
- Following are the ComboBox widget limitations:
 - Word Wrapping and Padding properties are not supported.
 - Rounded corner for borders and background is not supported.
 - Overriding the down arrow is not supported.
- Following are the RadioButton widget limitations:
 - Word Wrapping and Padding properties are not supported.
 - Rounded corner for borders and background is not supported.
 - Overriding the default *ticked* and *unTicked* images is not supported.
- Following are the Calendar widget limitations:
 - Word Wrapping and Padding properties are not supported.
 - Rounded corner for borders and background is not supported.
 - Grid calendar view is not supported.
 - Skin is not supported for calendar widget.

7. App Submission Guidelines

This section explains a few guidelines to be followed while submitting an application.

7.1 iOS 11

The following are the guidelines for submitting an iOS application with XCode 9:

1. It is mandatory to provide 'Asset Catalog' resource for iOS applications
2. Go to Target -> General -> App Icons and Launch Images -> Click on User Asset Catalog -> Migrate
3. Ensure that all the resolutions of the App Icons are provided in the Asset catalog
4. In the **info.plist** file, add `CFBundleIconName` with values set as AppIcon that is created in the assets resource.

7.2 iOS 10

In Xcode 8, Apple introduced a new privacy feature to access hardware resources (like camera, gallery, and contacts). To avoid compliance issues, you must ensure that the required settings are included while building iOS applications using Xcode 8.

According to iOS 10 mandate, It is necessary to add descriptions for the features that are related to user privacy. You need to add descriptions for the features (that are provided in the below procedure) even if the app does not use the features. Kony Visualizer Enterprise integrates the required settings as a part of iOS applications if you follow the steps provided below.

Before building an iOS application, do the following:

1. Create a JSON file.
2. Add the below strings to the JSON file.

```
{
  "NSCalendarsUsageDescription" : "<description>",
  "NSCameraUsageDescription" : "<description>",
  "NSPhotoLibraryUsageDescription" : "<description>"
}
```

Important: If you use other framework, you need to update the JSON file with appropriate keys. For more information, refer to [Information Property List Key Reference](#).

3. Save the JSON file as `infoplist_configuration.json`.
4. Add the `infoplist_configuration.json` file to `/resources/mobile/native/iphone` in your project.

Now, Kony Visualizer Enterprise ensures that the required settings are packaged and added to the `info.plist` file in your Xcode project.

If the iOS application uses protected mode, ensure that you download and use the iOS 10 compatible Finalizer Utility for Xcode 8.

8. DBX App Design Guidelines

This style-guide specifies the visual design system applied to the Responsive applications of Retail Banking user interface. It is intended to provide implementation guidance and design standards that ensure a unified, optimistic, modern, and premium experience for Kony customers.

HOW TO USE THIS STYLE-GUIDE

This visual style guide defines Kony Retail Banking Applications and describes relevant, detailed interface elements and components. The visual design and the resulting style guide offered here have been formulated to ensure the most appropriate experience for users.

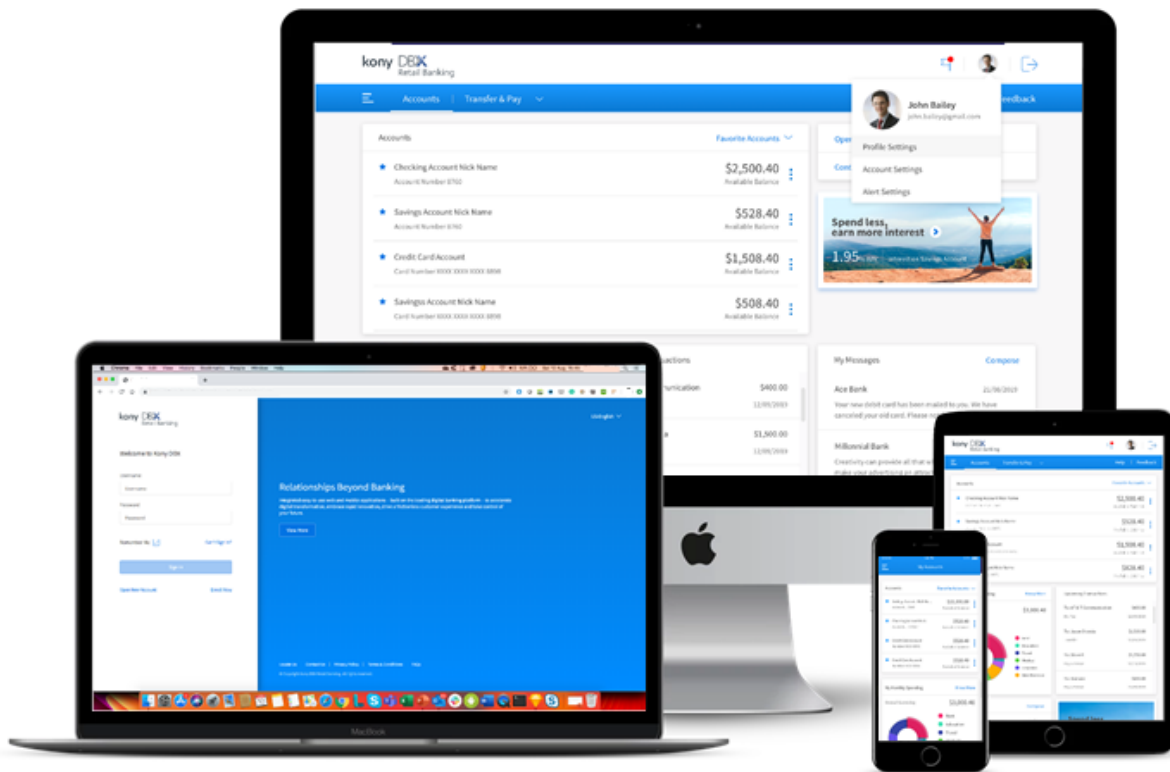
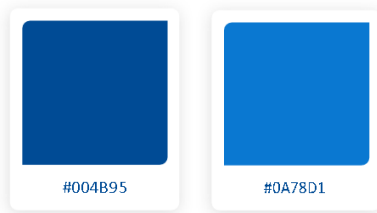
THE FINAL AND AUTHORITATIVE VISUAL SPECIFICATION

The images and written guidance found within this visual style guide represent the final and authoritative specification for the visual design of Kony Applications.

8.1 Color Theme

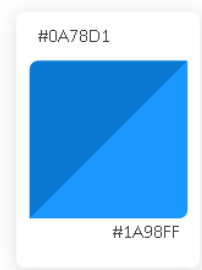
- Application reflects a Bold/ Progressive' theme.
- Modern-looking & aesthetically pleasing interface (choice of color, layout, flow) - closer to most modern apps that a retail banking end-user interacts with on a daily basis.
- The Gradient blue color with Drop Shadow is used as a 'Login , Menu Bar, Module landing screens and the Header Bar of each and every page are used as main Branding area. This can work as canvas to be customized later with the branding color of a customer.
- Primary actions, switches, swipe buttons etc are represented with a special Blue colour. selected Tabs, Icons, Header text are represented in Dark blue colour.

- Secondary actions are represented as links with simple text in blue color.

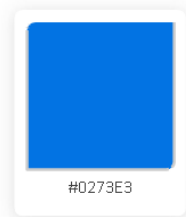


8.1.1 Primary Branding Color - Online Banking

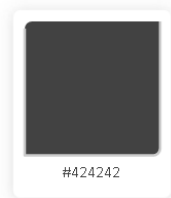
Gradient Colors for Header, Sign In, Can't Sign In and Enroll Screens



Color for Active Button, Icon, Link

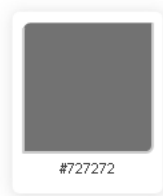


Default Color for Font Like Values



8.1.2 Secondary Branding Color - Online Banking

Secondary Font Colors like Label



Separator Color



Placeholder Color



Background Color



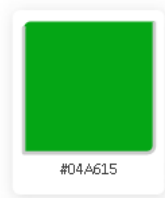
Shadow Color



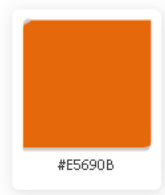
Container Color



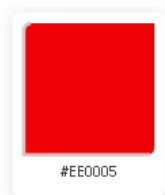
Positive Color



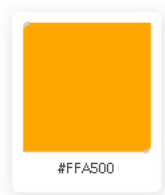
Negative Color



Error Color



Alert Color



8.1.3 Primary Branding Color - Retail Banking

Header Color



#0A78D1 & #1A98FF

Header Text



#FFFFFF

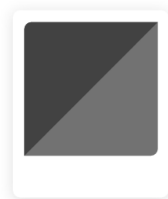
Link and Buttons



#0A78D1

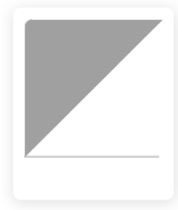
8.1.4 Secondary Branding Color - Retail Banking

Text Color



#424242 & #727272

Text Color



#A0A0A0 & #FFFFFF

Separator and Focus Color



#E3E3E3 & #F9F9F9

Swipe Buttons Color



#FF5D6E & #0A78D1

8.2 Fonts

8.2.1 Typography

In conversation, tone is everything. It gives words subtlety and intonation and reinforces the intended meaning. When dealing with the written word presented on the printed page or on the digital display, the choice of typeface is equally important. It gives the message a distinct personality and character.

SSP-(SSP) was designed by Paul D. Hunt under the guidance of Robert Slimbach. It was Adobe's first open source typeface family, conceived primarily as a typeface for user interfaces. SSP- draws inspiration from the clarity and legibility of twentieth-century American gothic typeface designs. Distilling the best archetypical qualities of these models, Paul followed a rational design approach by simplifying glyph shapes by paring them to their essential form. However, in order to more easily differentiate similar letter shapes (such as uppercase I and lowercase L), some additional details have been added. Besides providing such explicitly clarity in short text strings, another fundamental design consideration was to create a typeface that reads well in extended settings. This can be seen in the general proportions: SSP- has been designed with a more generous width than many other comparable gothics, and its shorter majuscule letters, combined with minuscule letters with longer extenders, create a more pleasant reading texture in longer text passages.

<p>Semi - Bold</p> <p>ABCDEFGHIJKLMNOPQRSTUVWXYZ</p> <p>Z abcdefghijklmnopqrstuvwxyz 0123456789 (@ \$ % & * - = ? / , .)</p>	<p>Bold</p> <p>ABCDEFGHIJKLMNOPQRSTUVWXYZ</p> <p>abcdefghijklmnopqrstuvwxyz 0123456789 (@ \$ % & * - = ? / , .)</p>
<p>Regular</p> <p>ABCDEFGHIJKLMNOPQRSTUVWXYZ</p> <p>abcdefghijklmnopqrstuvwxyz 0123456789 (@ \$ % & * - = ? / , .)</p>	<p>Light</p> <p>ABCDEFGHIJKLMNOPQRSTUVWXYZ</p> <p>abcdefghijklmnopqrstuvwxyz 0123456789 (@ \$ % & * - = ? / , .)</p>

Text Styles and Colors are one of the most important elements of any application. We have Primary Header, Label, Secondary Header, Value etc.

Font Style for Online Banking	Font Style for Retail Banking
<p>Header Level 1 - 20px</p> <p>Sub-Header Level 2 - 15px</p> <p>Action text Link - 15Px</p>	<p>Header Level 1 - 30px</p> <p>Sub-Header Level 2 - 28px</p> <p>Action text Link - 26px</p>

8.2.2 Where To Use

We have several types of Text Styles and Colors used:

- Primary Text Style on any page is provided as Values, Normal Texts, any important text.
- Secondary Text Styles used on any page as Labels, sub text, additional information.
- Buttons on the app header are mainly the Cancel, Back, Add.

8.2.3 Text Styles/ Colors

Primary Text (Camel Case)

Font	SSP Regular
------	-------------

Size	17px
Colour	#424242 / 66,66,66

Value (Camel Case)

Font	SSP Regular
Size	15px
Colour	#424242 / 66,66,66

Secondary Text (Camel Case)

Font	SSP Regular
Size	15px
Colour	#727272 / 66,66,66

Label (Camel Case)

Font	SSP Regular
Size	15px
Colour	#727272 / 114,114,114

8.2.4 Links**Normal Status**

Font	SSP Regular
Size	15px
Colour	#0273E3

Hover Status (Only for DesktopWeb)

Font	SSP Regular
Size	15px
Colour	#358FE9

8.3 Text Field/Button Styles

8.3.1 Statuses

Normal status

Border	E3E3E3
Thickness	1px

Selected status (Only for DesktopWeb)

Border	4A90E2
Thickness	1px
Radius	3px

Non Editable field / Disabled

Border	E3E3E3
Thickness	1px
Radius	3px
Filled	F7F7F7

Normal field / Error

Border	EE0005
Thickness	1px
Radius	3px

8.3.2 Icon/Option Button**Normal status**

Border	E3E3E3
Thickness	1px
Radius	2px
Filled	FFFFFF

Hover status (Only for DesktopWeb)

Border	E3E3E3
Thickness	1px
Radius	2px
Filled	FFFFFF
Shadows	x-0, y-2, Blur-10px

On Click / On Tap

Border	0273E3
Thickness	0.5px
Radius	2px
Filled	FFFFFF
Shadows	x-0, y-2, Blur-10px

8.3.3 Buttons

8.3.3.1 Primary Button

Normal status

Filled	0273E3
Text	FFFFFF, SSP Regular - 15

Hover status (Only for DesktopWeb)

Filled	358FE9
Text	FFFFFF, SSP Regular - 15

On Click / On Tap

Filled	0161C1
Radius	3px
Text	FFFFFF, SSP Regular - 15

Disabled

Filled	80B9F1
Text	FFFFFF, SSP Regular - 15

8.3.3.2 Secondary Button**Normal status**

Border	0273E3
Text	0273E3, SSP Regular - 15

Hover status (Only for DesktopWeb)

Filled	F7F7F7
Stroke	0273E3
Text	0273E3, SSP Regular - 15

On Click / On Tap

Filled	0161C1
Stroke	1px
Text	0161C1, SSP Regular - 15

8.4 Components and Iconography

On a Web page, an icon is a graphical image that represents the topic or information category of another Web page. Frequently, the icon is a hypertext link to that page. Typically, icons are gathered in one or two places on a page, either as separate graphic files or as a single image map .

8.4.1 Where To Use

Icons, other components used in a web based application to represent certain type of actions and indications. Components like number formats, Banners used in all breakpoints with a specific dimensions.

8.4.2 When To Use

Whenever there is a need to represent some actions or indicate some information. In a computer's graphical user interface (GUI), an icon is an image that represents an application, a capability, or some other concept or specific entity with meaning for the user. An icon is usually selectable but can also be a non-selectable image

Iconography for Online Banking

ICONOGRAPHY

TOGGLE ICONS

Unselected status (Inactive)



Hover status (Only for DW)

None

On Click / On Tap



ACTION ICONS

Normal status



Active Checkbox



Hover status (Only for DW)



In-Active Checkbox



IMPORTANT ICONS

Hamburger Menu Icons



Login Icons



Transfer Type Icons



Profile Management Icons

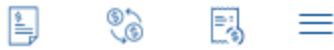


Locate Us Icons



Iconography for Retail Banking

Footer Nav Icons



Hamburger Menu Icons



Other App icons

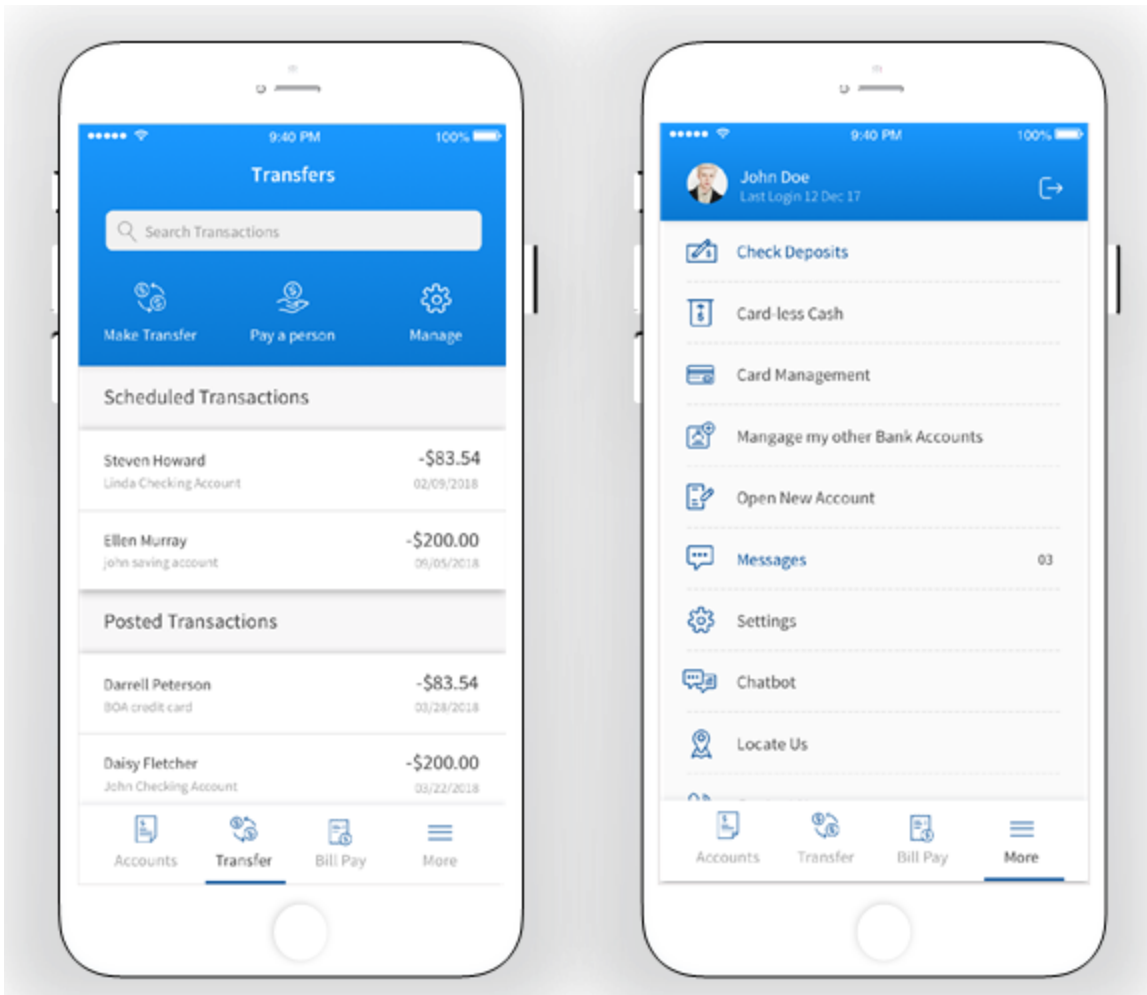


Section landing page

The icons in the module landing pages are big in size and white in color



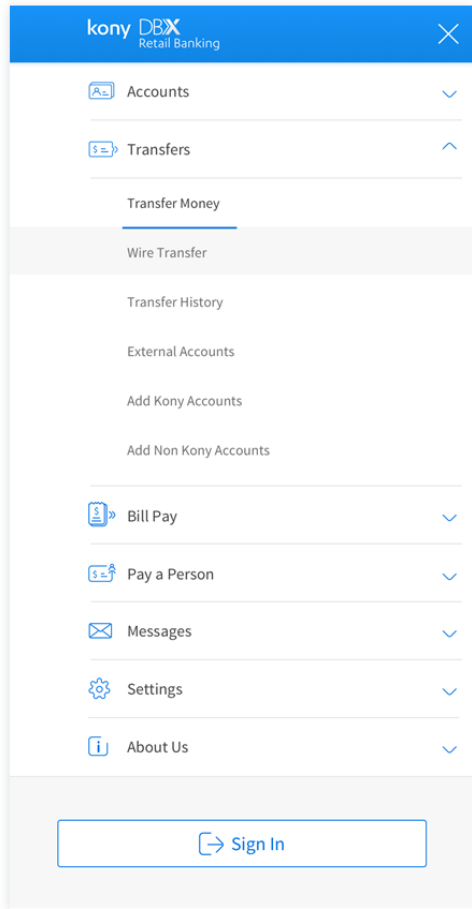
8.4.3 Screens



8.4.4 Components

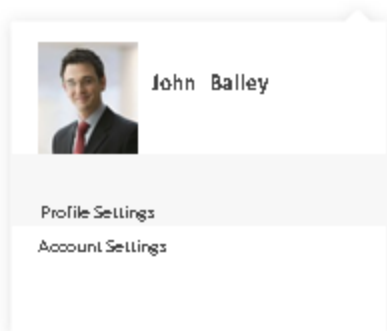
Hamburger Menu

Which contains the Primary actions and their sub items. Blue selections defines which module is used.



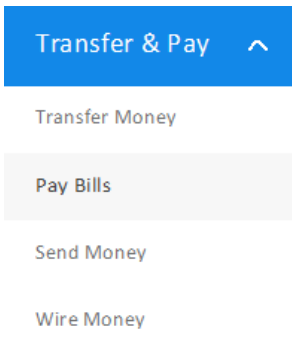
Settings Contextual Menu

Which contains Primary actions of the settings. There is no selection only hover background color.



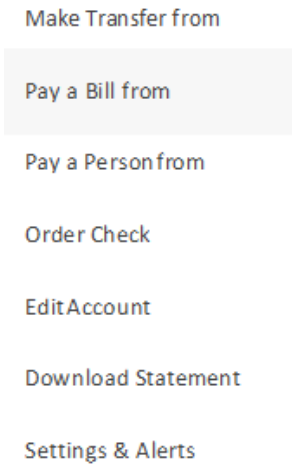
Transfers and Pay Menu

Which contains all the Transfer related actions. There is no selection only hover background color.



Account Specific Contextual Menu

Which contains all the account specific actions. There is no selection only hover background color.



8.4.5 Other Components

Calendar Format : MM/DD/YYYY

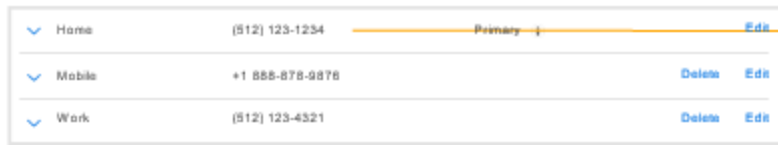
Send On

SSP, 15px,#424242

Phone Number: 10 Digits with Country Code

Or call our support team: 1 0008567888



Home	(512) 123-1234	Primary	Edit
Mobile	+1 888-878-9878		Delete Edit
Work	(512) 123-4321		Delete Edit

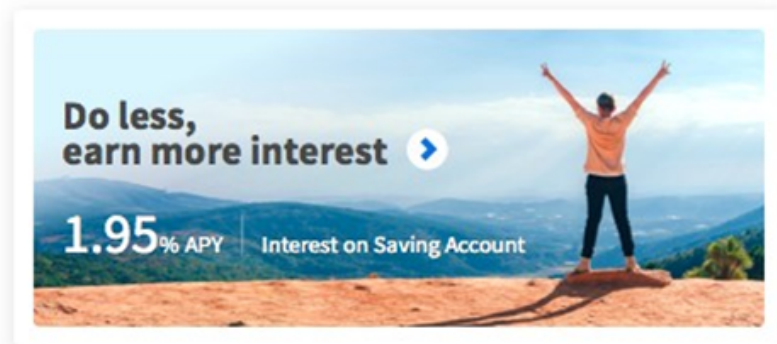
SSP, 15px, #424242

8.4.6 Banner Ads

Banner Ads sizes differs based on the responsive implementation.

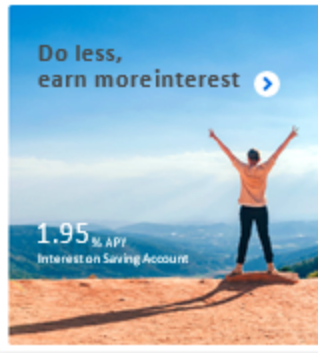
Desktop

Width 390px Height 172px



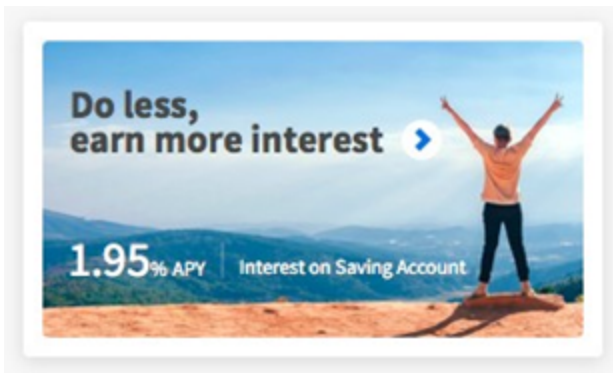
Tablet

Width 353px Height 380px



Mobile

Width 300px Height 174px



8.5 Page Size and Grid - Online Banking

Responsive web design (RWD) is an approach to web design that makes web pages render well on a variety of devices and window or screen sizes. Recent work also considers the viewer proximity as part of the viewing context as an extension for RWD. Content, design and performance are necessary across all devices to ensure usability and satisfaction.

A site designed with RWD adapts the layout to the viewing environment by using fluid, proportion-based grids, flexible images.

8.5.1 Where To Use

Responsive grid system in OLB follows two column, maintaining 1200px in the center which needs to be maintained throughout the application for responsive implementation.

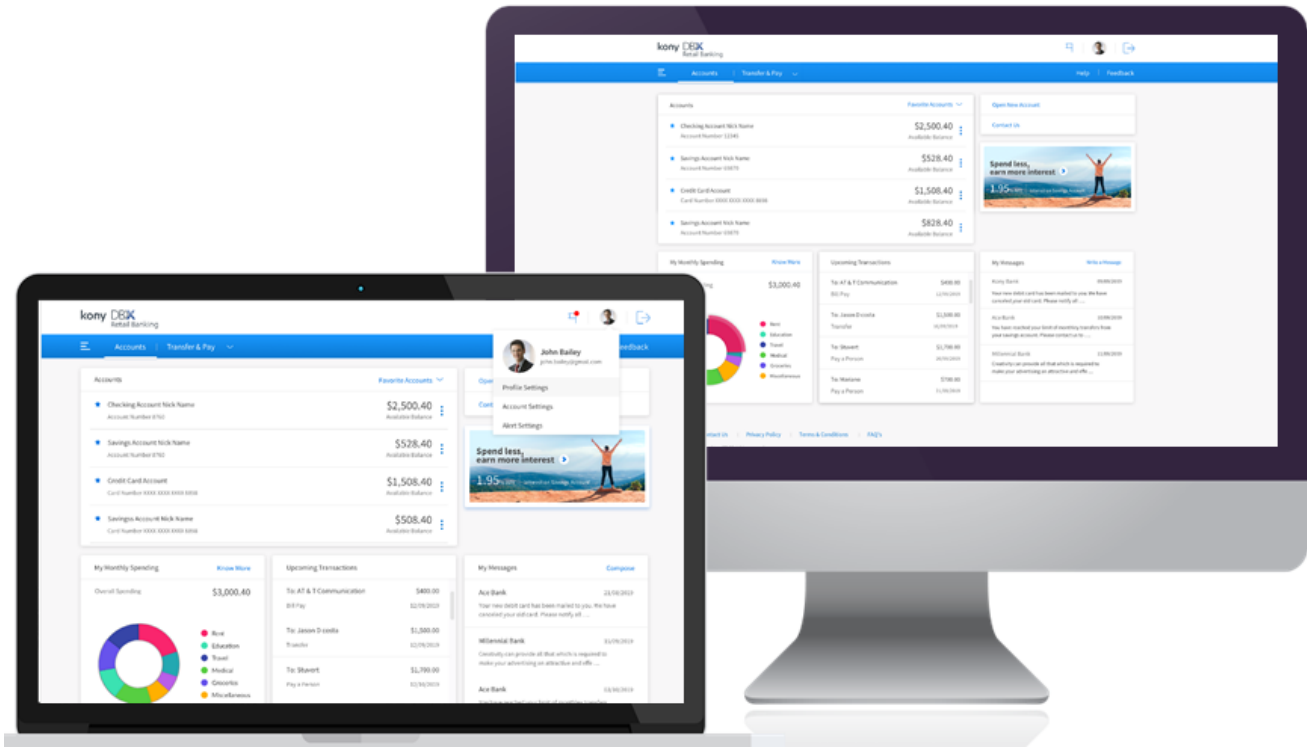
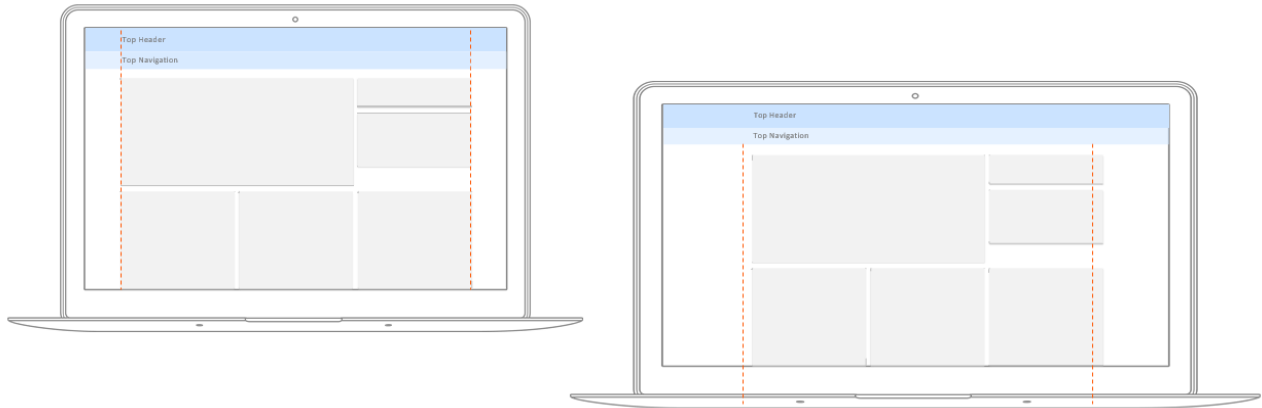
8.5.2 When To Use

Mobile is different from desktop, and designing with mobile top-of-mind is becoming increasingly important. Users are turning to their mobile devices more and more these days, and that means website designs need to adapt - in ways that go beyond a smaller screen size.

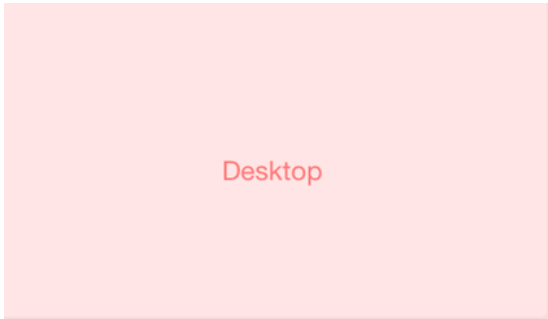
Responsive Web design is the approach that suggests that design and development should respond to the user's behaviour and environment based on screen size, platform and orientation.

8.5.3 Grid and Frame

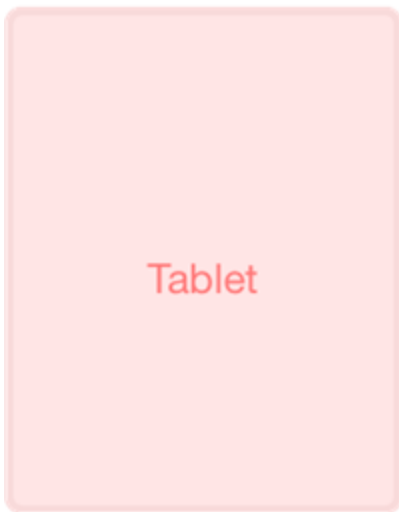




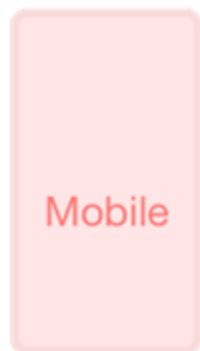
8.5.4 Resolutions



Width: 1366, Height: 768 / Scrollable



Width: 768, Height: 1024 / Scrollable



Width: 320, Height: 568 / Scrollable

8.6 Header and Footer - Online Banking

A header is the top margin of each page, and a footer is the bottom margin of each page. Headers and footers are useful for including material that you want to appear on every page of a document such as your name, the title of the document, or page numbers.

8.6.1 Where To Use

Headers and footers are typically used on top and on bottom of every web based pages to show the action that a user can perform immediately and on footer some other copyright texts and other contact us and actions etc.

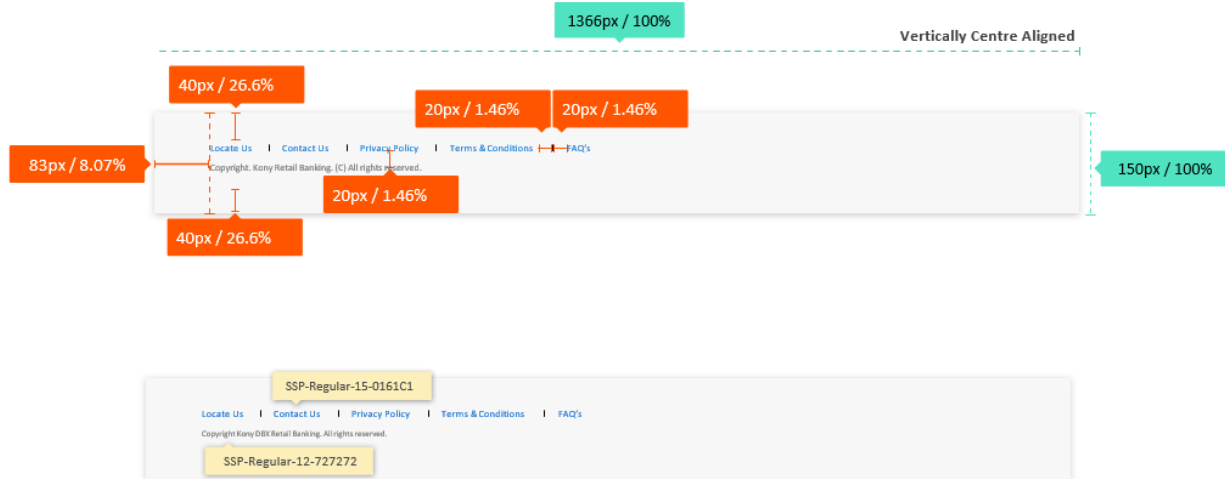
8.6.2 When To Use

Headers and footers are typically used in multiple-page application to display descriptive information. In addition to page numbers, a header or footer can contain information such as: The Logo name, the important action links, a graphic and other extra side links.

8.6.3 Header



8.6.4 Footer



8.7 Sign In - Online Banking

The page consists a list of objects presented in a form manner for the user to register or sign in to their own banking account. It consists all registered user in logged using same system.

8.7.1 Where To Use

The Sign In page should be used only starting of every web based online banking applications.

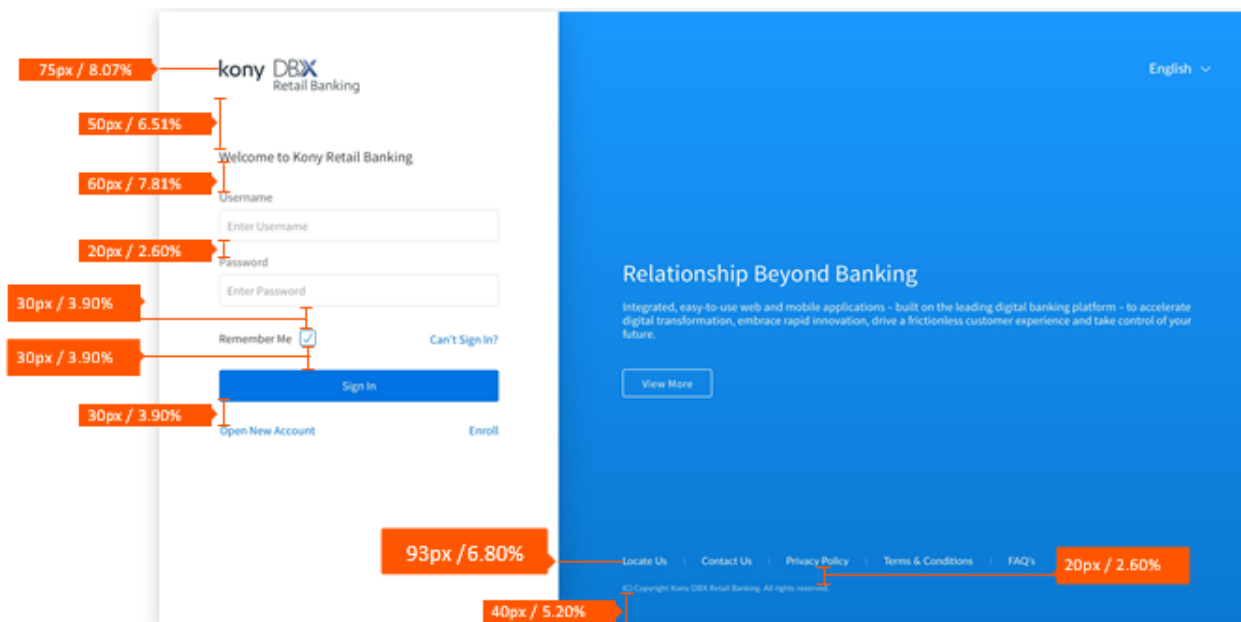
8.7.2 When To Use

The Sign In page should be used on starting of the online banking application to enter inside the application. Online Banking is very important and for many security reasons this is very important part of the application. User needs to be very careful and should go through Multifactor Authentications for better experience.

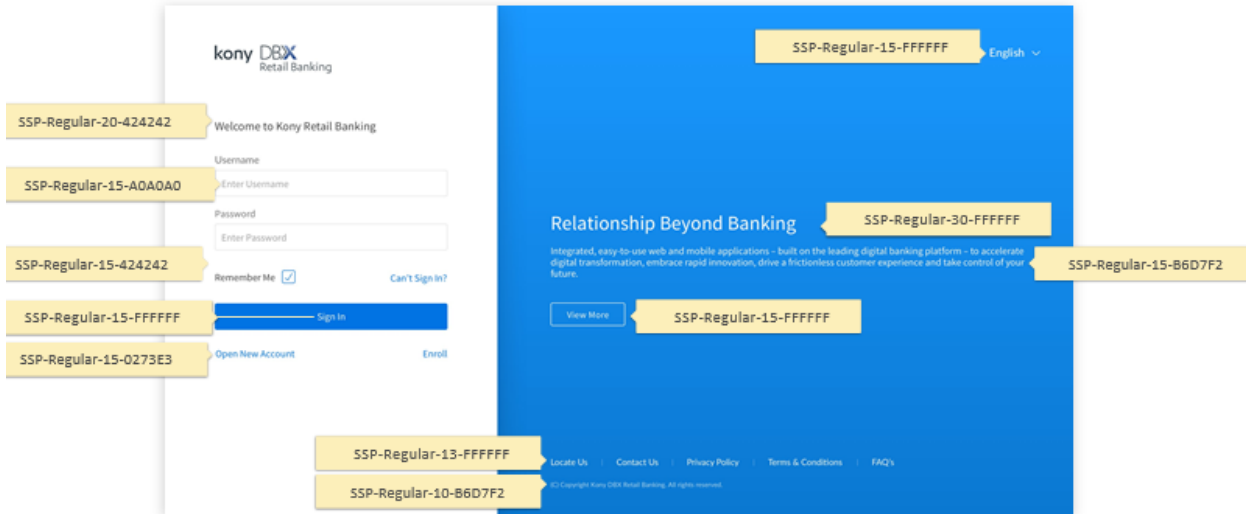
8.7.3 Screens



8.7.4 Specifications



8.7.5 Font Specifications

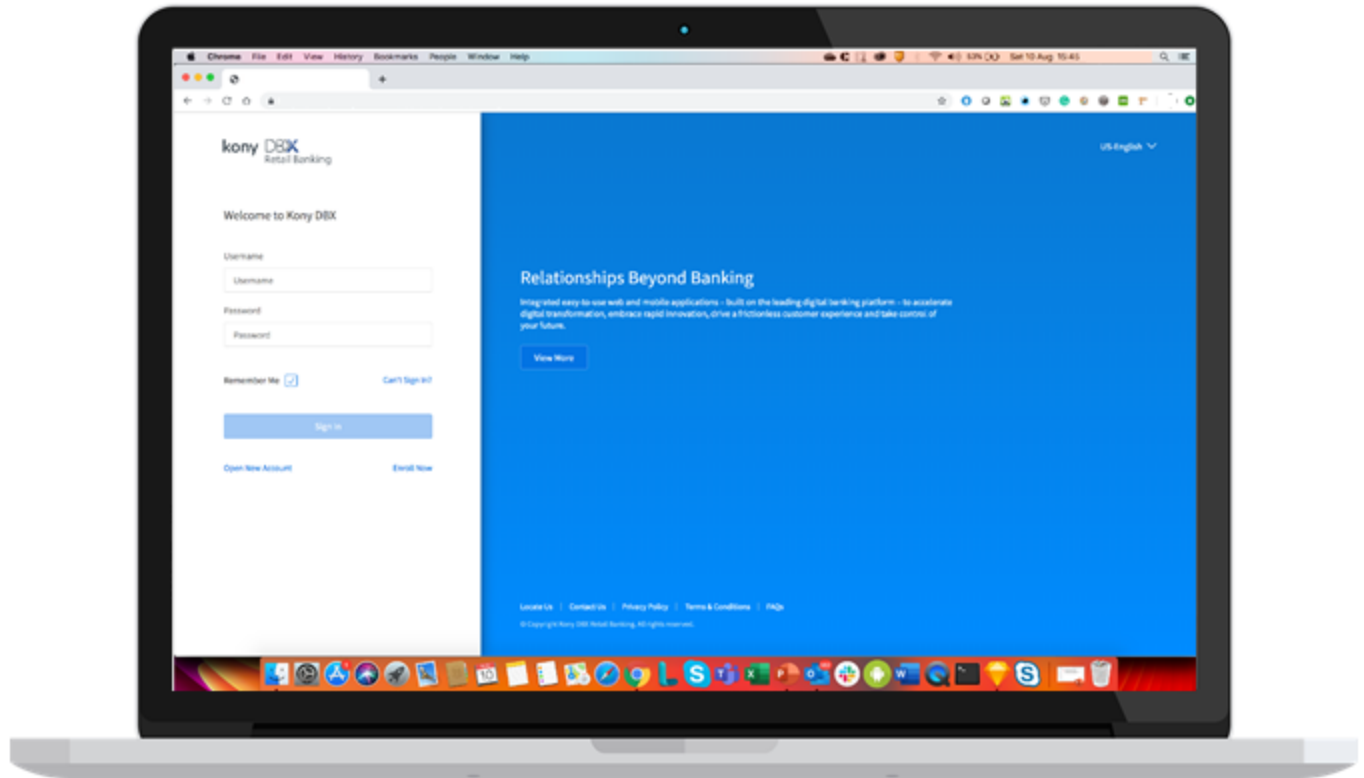


8.7.6 Screens

Login page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen.

The detail screen maintains the relationship, continuity and context of the object from the list.

It also provides additional information about the object. The related Information is presented in the form of drill down Page.



8.7.7 Design Principles

VISUAL RULES

- Do not provide any scroll within a segment.
- Proper TAPABLE areas are required for icons, links etc... Even if the image size is smaller, provide a bigger tapable area as per the standards (Ex: Minimum of 44px for non-retina display)
- If there is no data on any field within the segment leave the space blank and show all the other data in their respective positions. Consistency plays an important role in the list segments.

GESTURES

We have different interaction on the list segments.

- Tap on the segment area to go to the details view.
- Right to Left swipe on the segment will provide the list level actions to the user like 'Edit', 'Delete' etc..
- Swipe up is for scrolling the list.

INTERNATIONALIZATION (i18n)/ LOCALIZATION (l10n) GUIDELINE

- Even if there is difference in the number of characters, the position of the elements remains the same.
- Truncation rules that apply to English also applies to other languages.

8.8 Account List

The page consists a list of objects presented in a logical manner for the user to scan, browse or search in order to identify the object of interest. It consists all the important data to help the user to view the information, user can also perform some actions directly by not going to the detail view.

8.8.1 Where To Use

Lists are best suited to present a homogeneous data type or sets of data types, such as images and text. They are optimized for reading comprehension with the goal of differentiating between similar data types or similar qualities within a single data type.

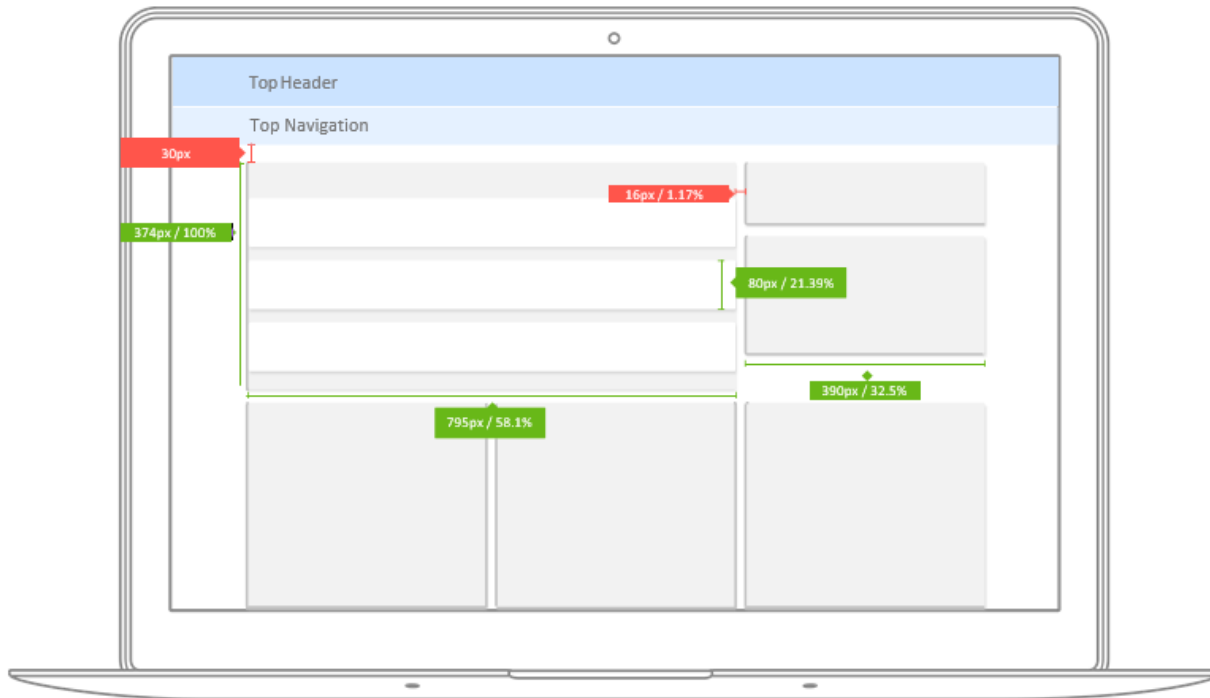
8.8.2 When To Use

This independent page is used when the user want to see the full list of available business objects and perform some actions on them. The 'Search' over the list help to quickly find the desired record.

When designing the List, it is important to present the information in a simple layout that is easy to understand. Visual hierarchy of the fields plays an important role in guiding the user through priority fields.

8.8.3 Grid and Frame

Online Banking



Retail Banking

There are several combinations in representing the data in a list segment. But in general, certain rules need to be followed to make them look like a single family of the application.

Position of some common elements like Header bar, Search bar, Left margin, Right margin etc remain the same in all types of list,



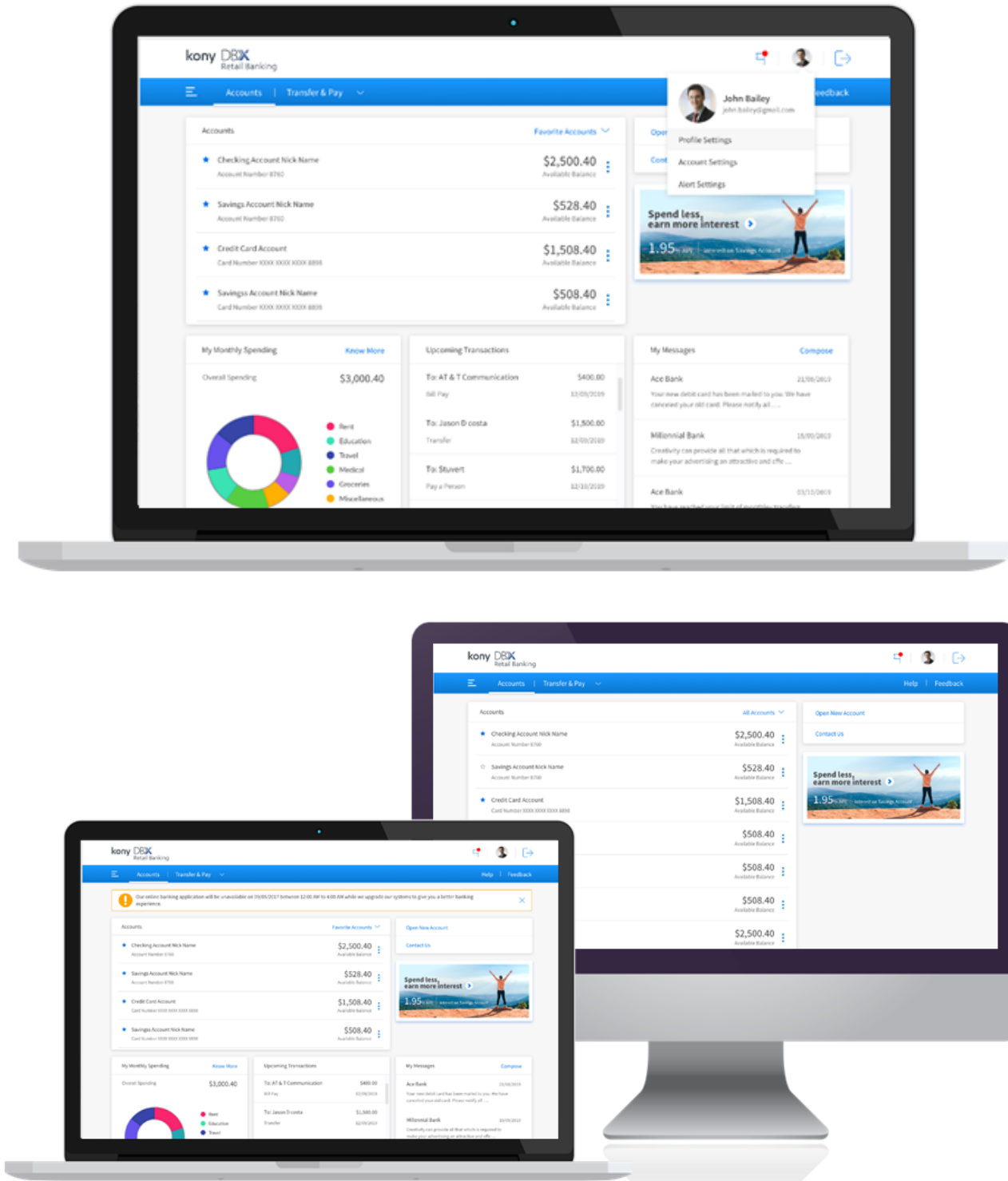
8.8.4 Screens

Online Banking

Login page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen.

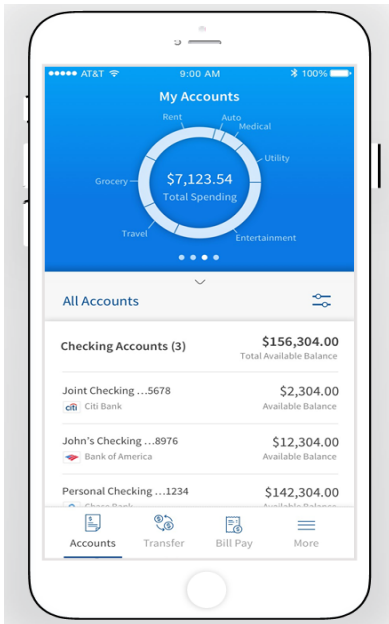
The detail screen maintains the relationship, continuity and context of the object from the list.

It also provides additional information about the object. The related Information is presented in the form of drill down Page.



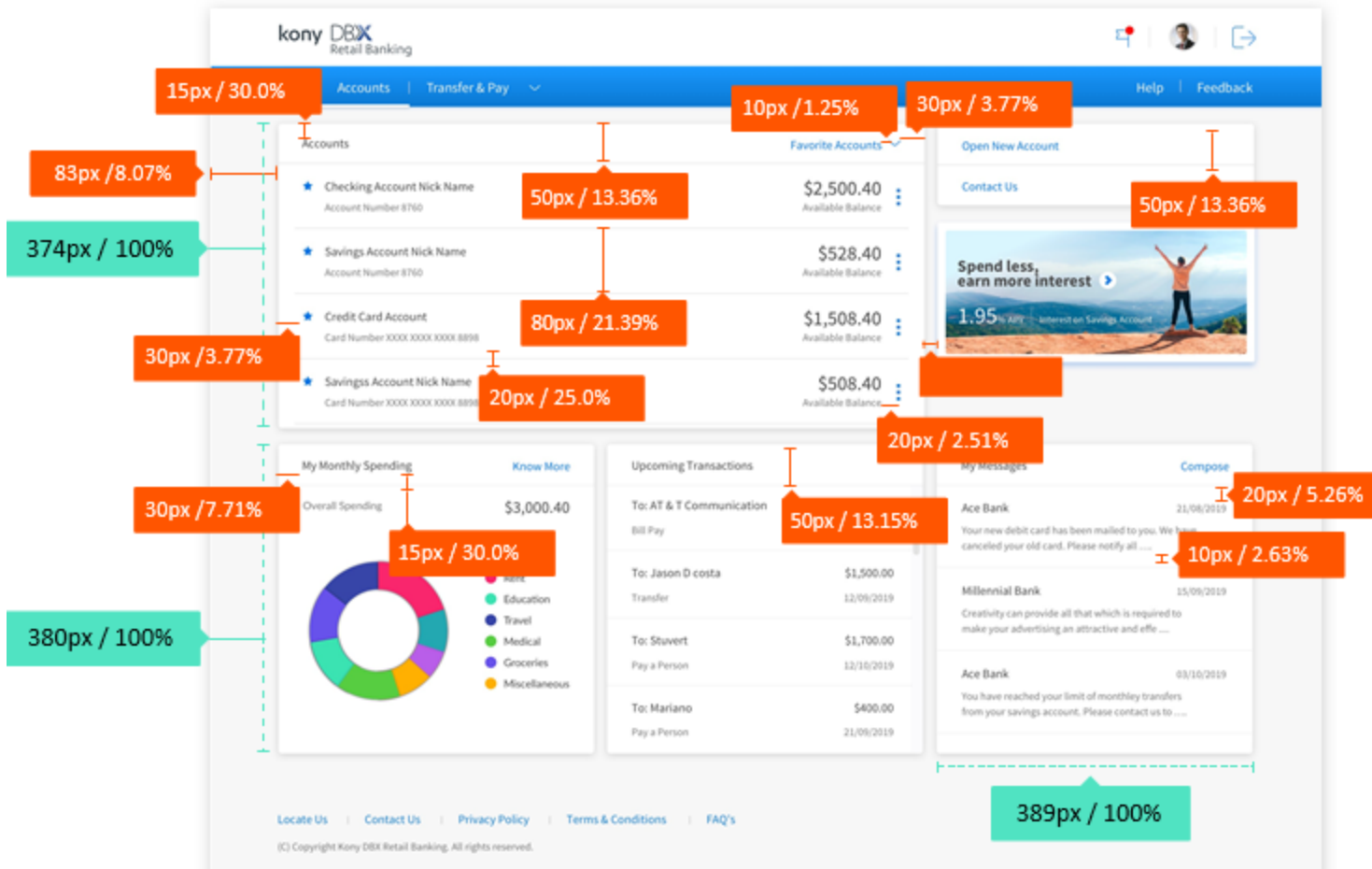
Retail Banking

Device: iPhone Screen Resolution: 375px/ 667px

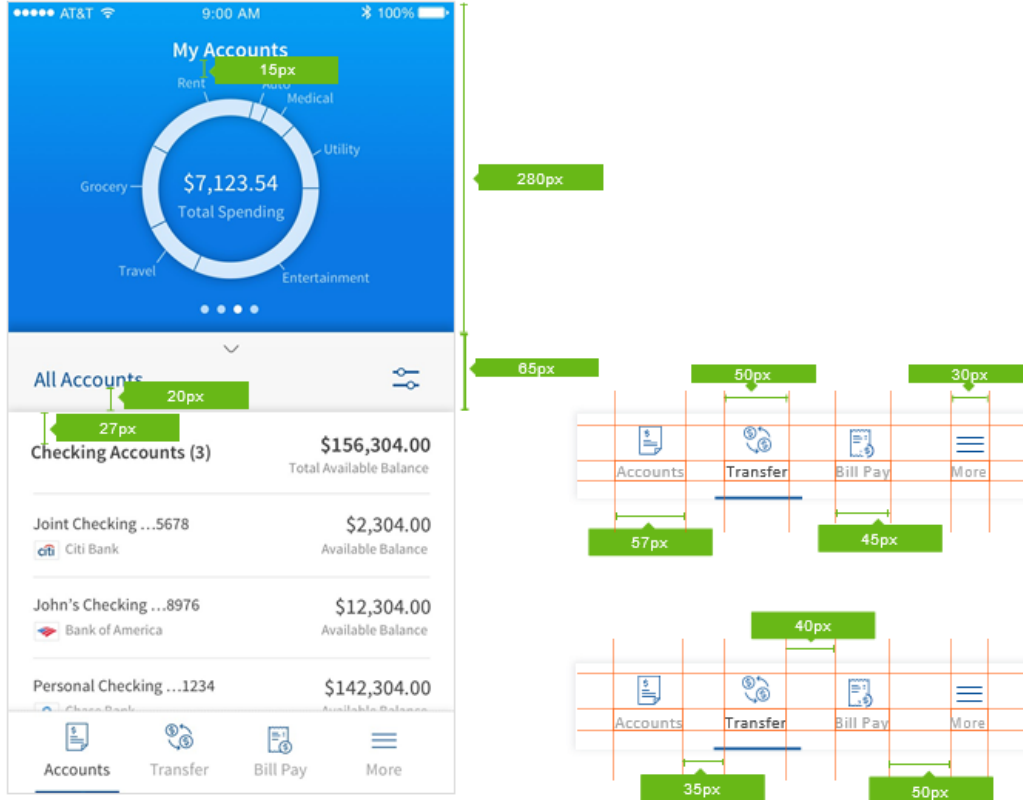


8.8.5 Specifications

Online Banking

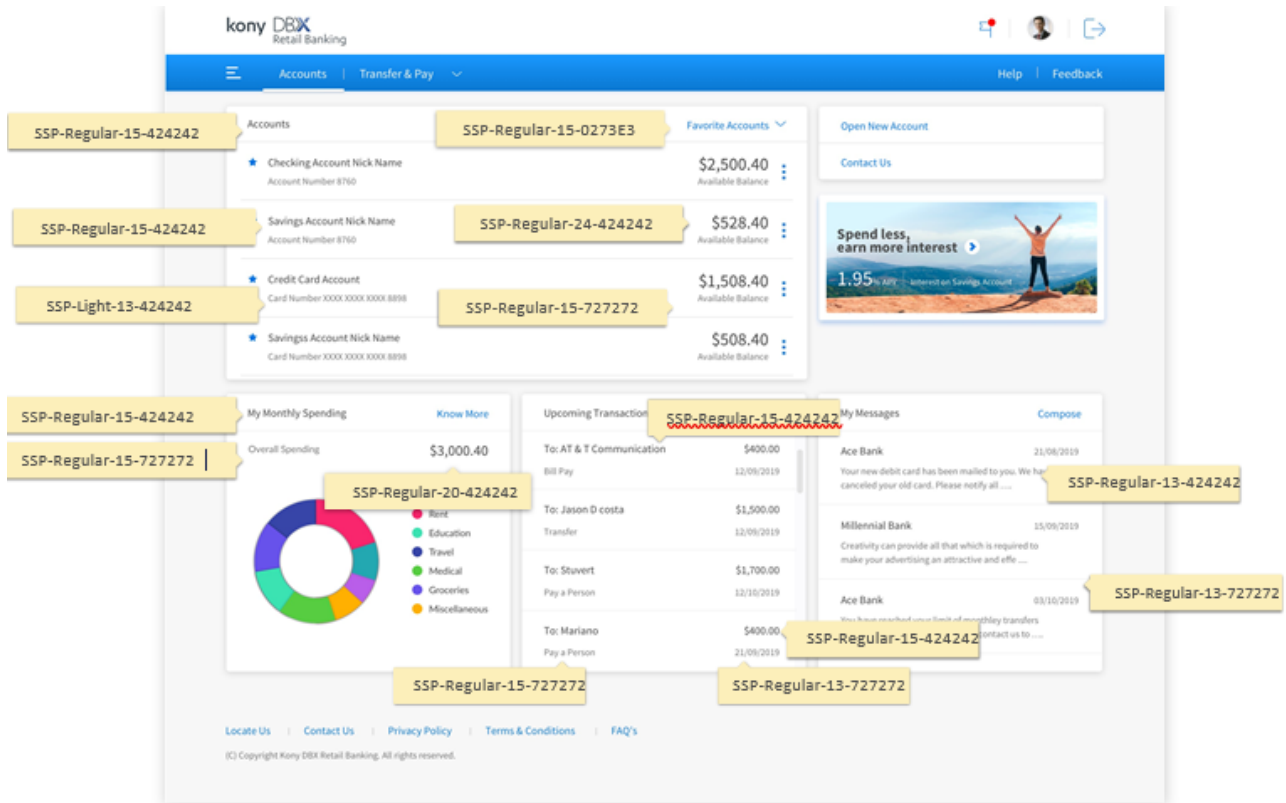


Retail Banking

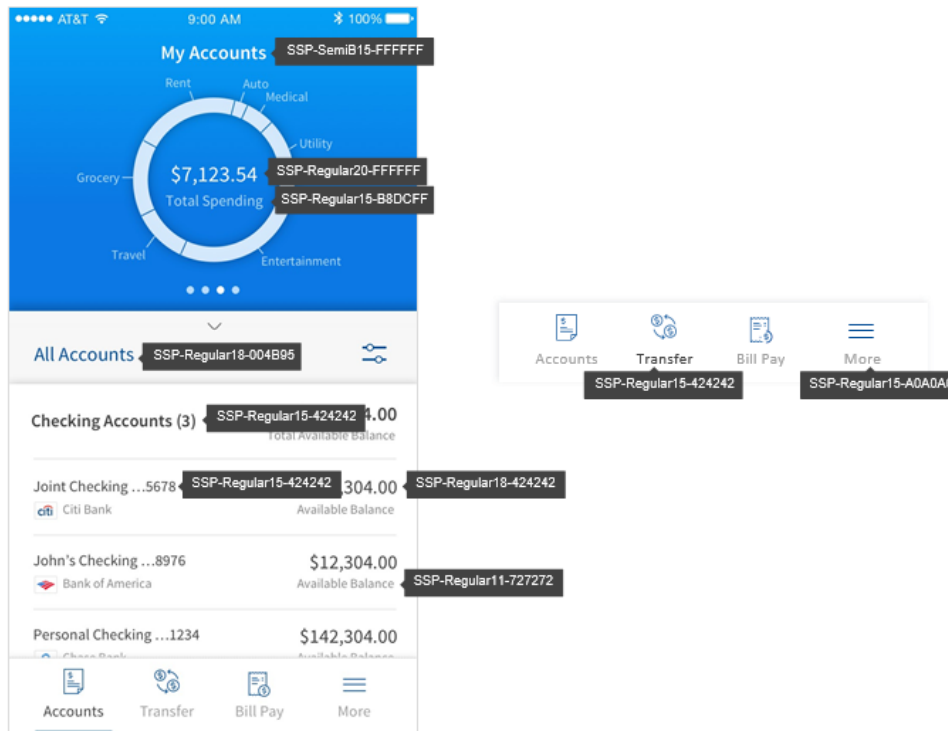


8.8.6 Font Specifications

Online Banking



Retail Banking

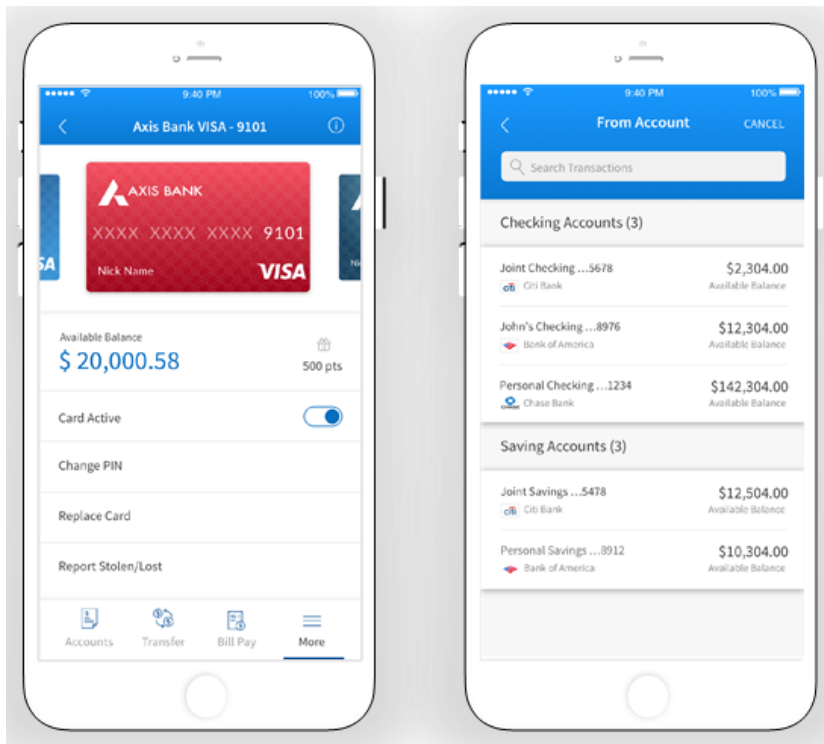


8.8.7 Types - Retail Banking

Types of List Views:

The list could get triggered from any one the following modes.

- Trigger from App Menu & displayed as a independent page or a separate entity
- Trigger from one of the Sub-menus in a detail page & displayed as an independent page
- Displayed as part of a Detail Page



8.8.8 Design Principles

VISUAL RULES

- Do not provide any scroll within a segment.
- Proper TAPABLE areas are required for icons, links etc... Even if the image size is smaller, provide a bigger tapable area as per the standards (Ex: Minimum of 44px for non-retina display)
- If there is no data on any field within the segment leave the space blank and show all the other data in their respective positions. Consistency plays an important role in the list segments.

GESTURES

We have different interaction on the list segments.

- Tap on the segment area to go to the details view.
- Right to Left swipe on the segment will provide the list level actions to the user like 'Edit', 'Delete' etc..
- Swipe up is for scrolling the list.

INTERNATIONALIZATION (i18n)/ LOCALIZATION (l10n) GUIDELINE

- Even if there is difference in the number of characters, the position of the elements remains the same.
- Truncation rules that apply to English also applies to other languages.

8.9 Account Details

Detail page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen. The detail screen maintains the relationship, continuity and context of the object from the list .

It also provides additional information about the object The related Information is presented in the form of drill down Page.

8.9.1 Where To Use

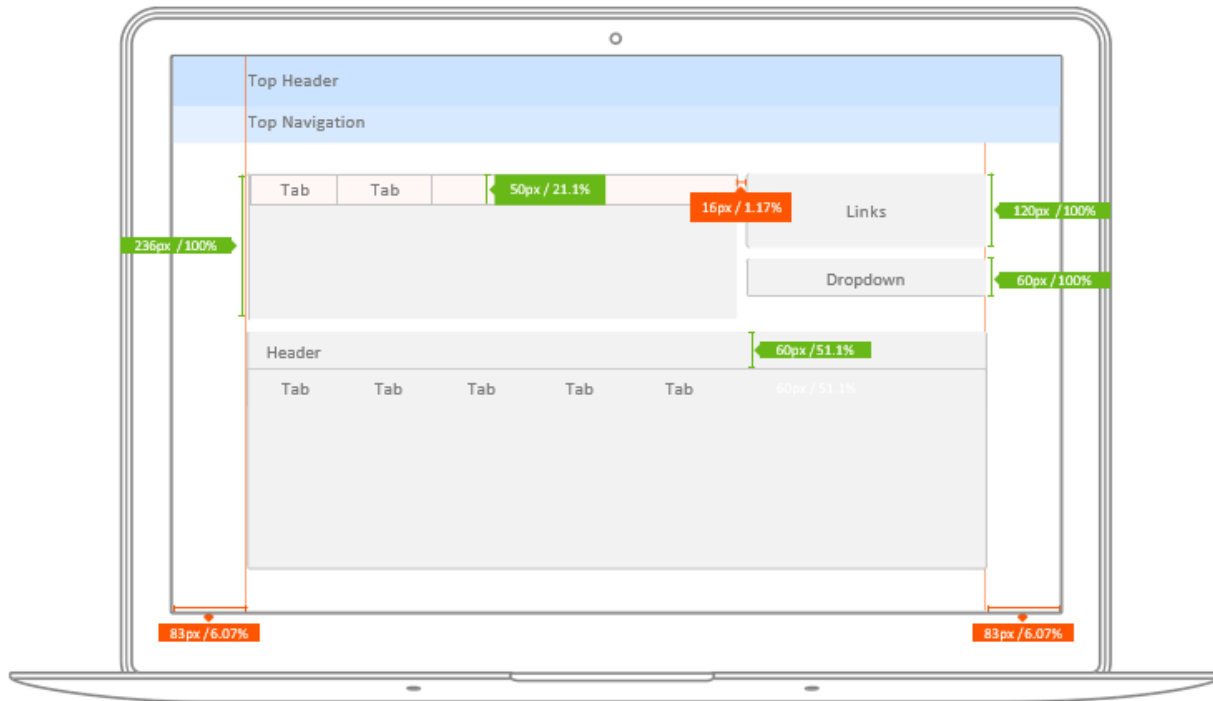
This page is designed to represent the complete detail of a list item. These pages should be represented as drilldown pages from a list view so that user can navigate back to list.

8.9.2 When To Use

Use these pages only when you need to represent more information on a particular list item. When there is very less information in a item, this page can be skipped by showing the complete detail in the list itself. But most of the time that is practically not possible. The detail Page enables real estate for page level actions such as edit, delete, remove etc

8.9.3 Grid and Frame

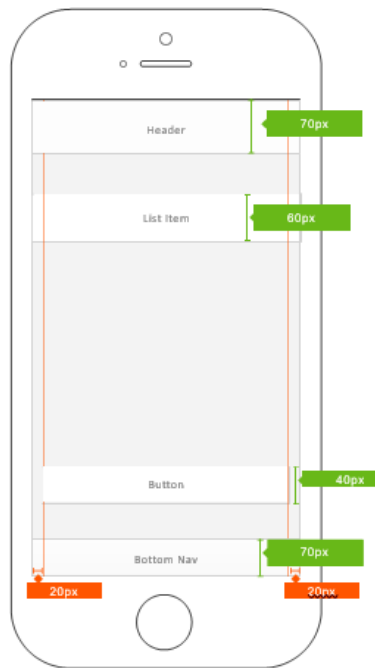
Online Banking



Retail Banking

There are several combinations in representing the data in a list segment. But in general, certain rules need to be followed to make them look like a single family of the application.

Position of some common elements like Header bar, Search bar, Left margin, Right margin etc remain the same in all types of list.



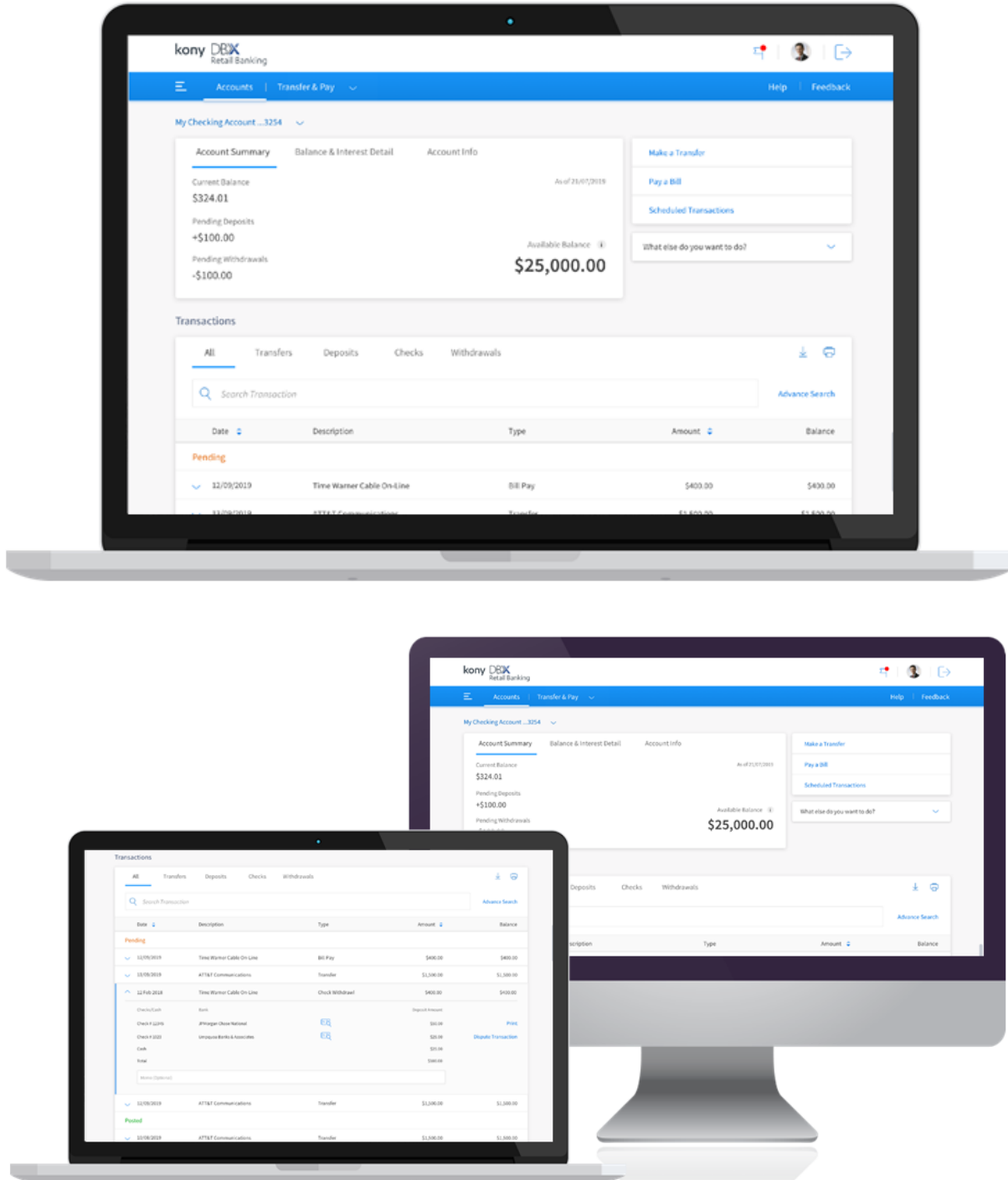
8.9.4 Screens

Detail page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen.

The detail screen maintains the relationship, continuity and context of the object from the list.

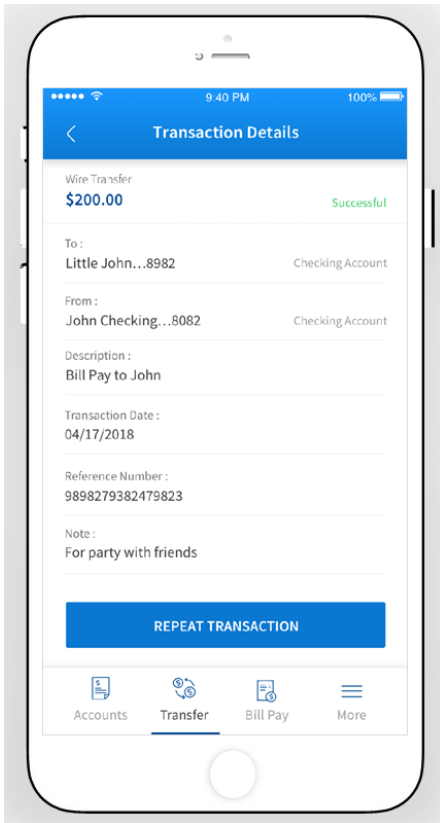
It also provides additional information about the object. The related information is presented in the form of drill down Page.

Online Banking



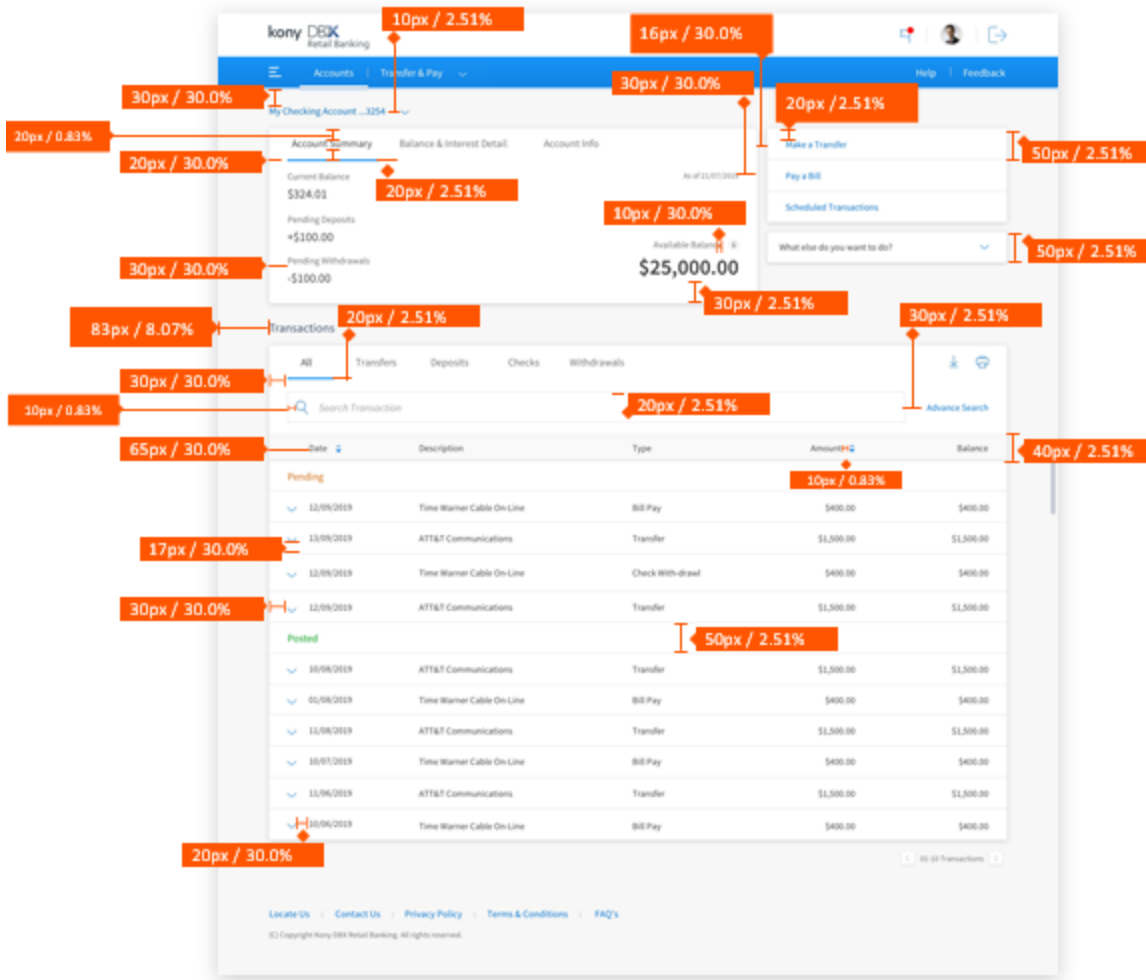
Retail Banking

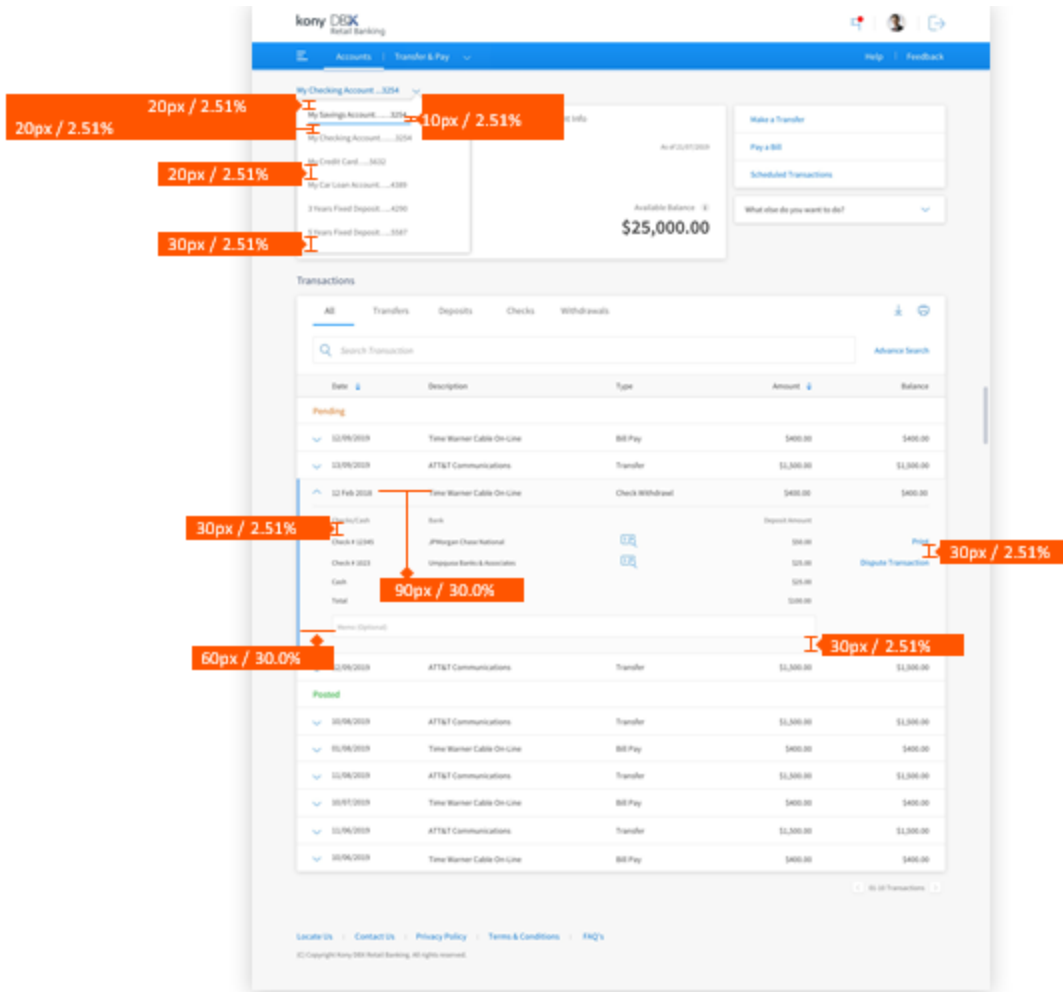
Device: iPhone Screen Resolution: 375px/ 667px



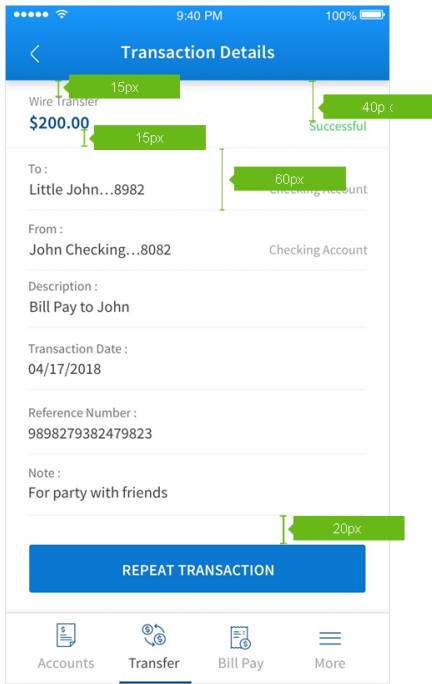
8.9.5 Specifications

Online Banking



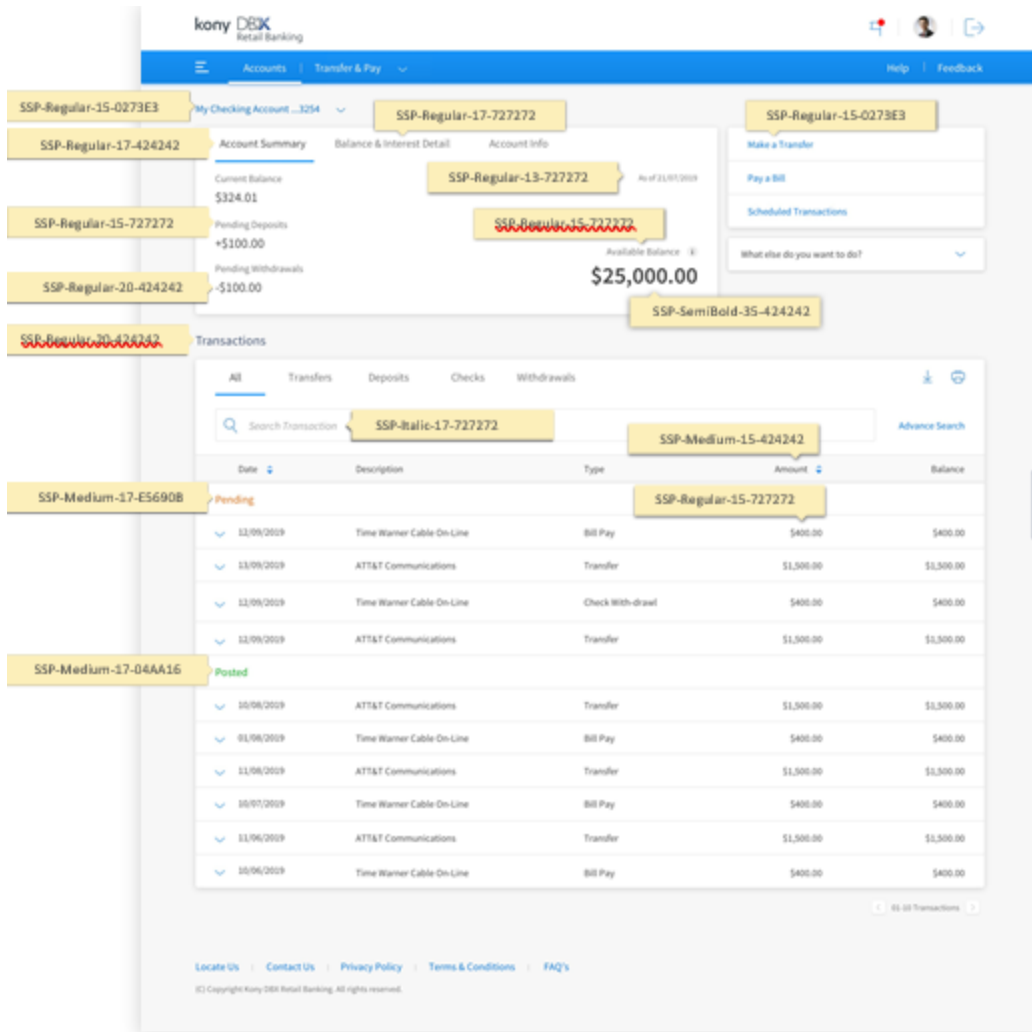


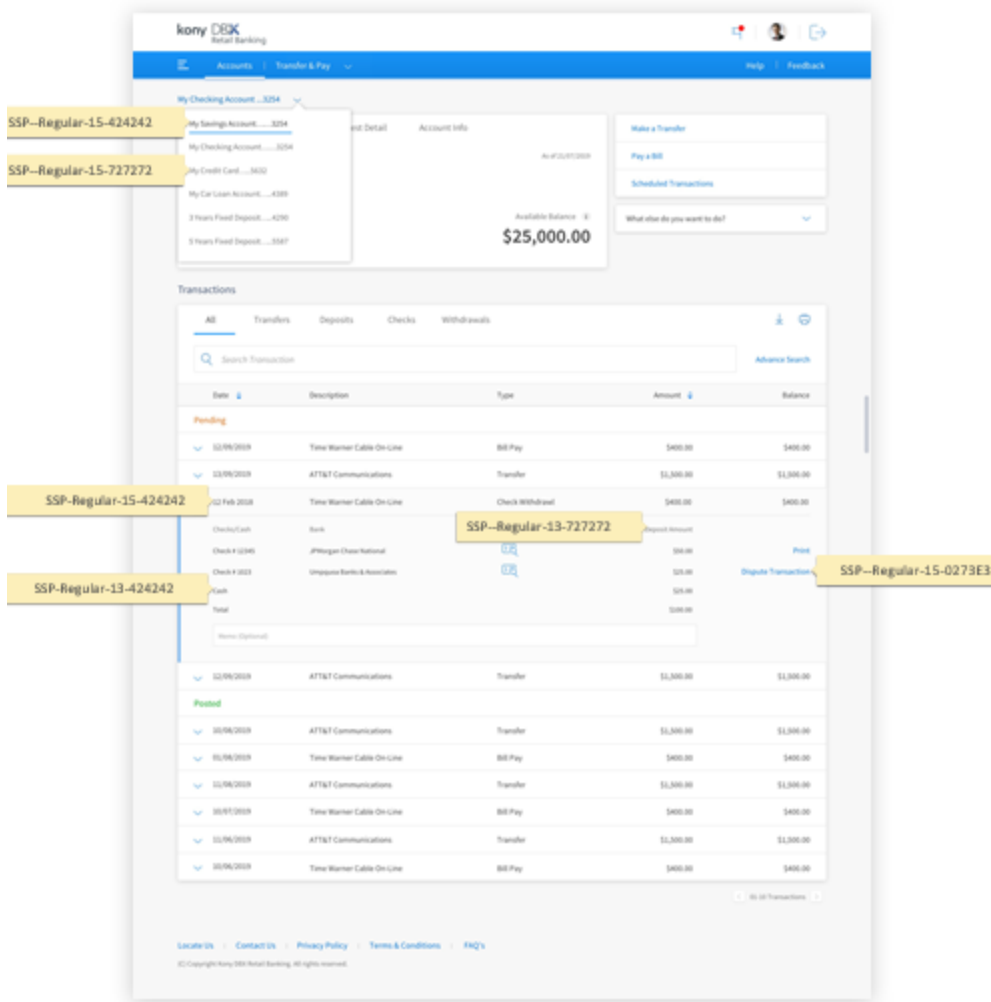
Retail Banking



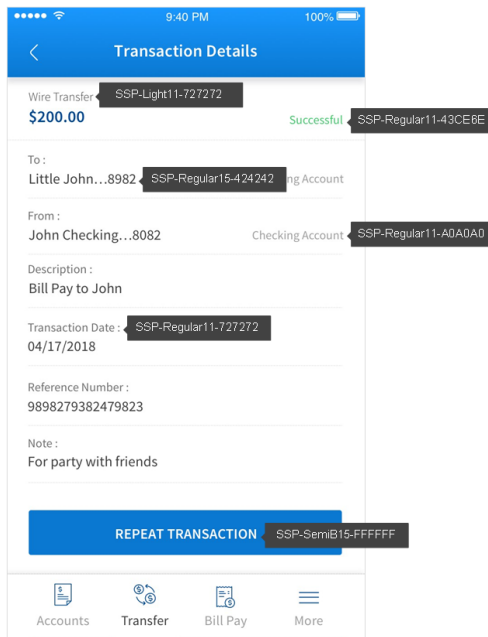
8.9.6 Font Specifications

Online Banking





Retail Banking

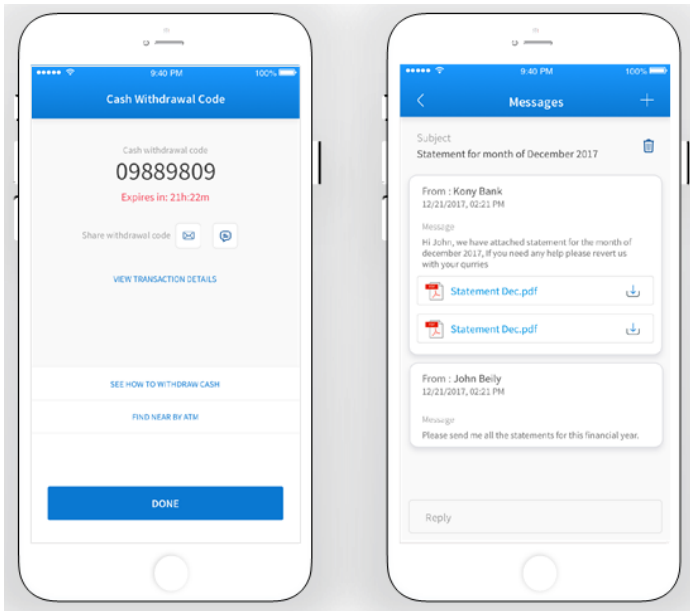


8.9.7 Types - Retail Banking

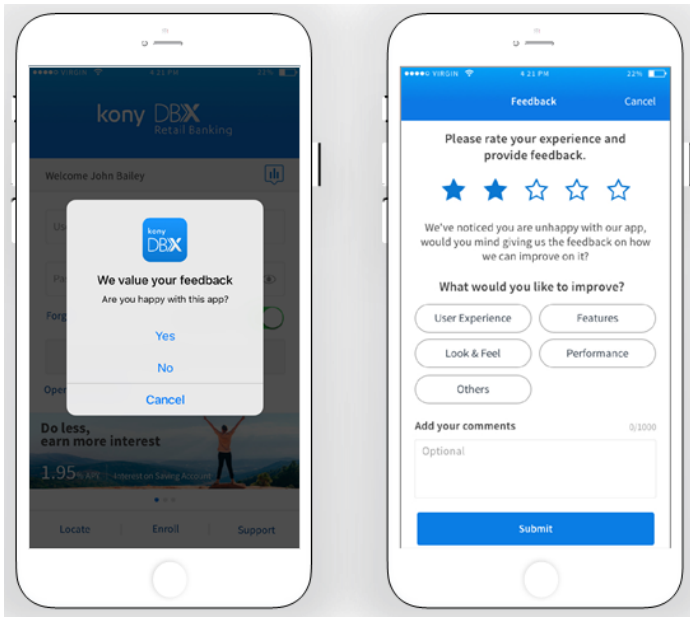
Types of Detail Views:

The Detail page could get triggered from any one the following modes.

- Trigger from Landing pages from segment & displayed as a independent page or a separate entry
- Trigger from separate completion of flow or process.
- Displayed as part of a Messages Page



- If user Logs out then feedback pop- up will appear with further flow
- Menu



8.9.8 Design Principles

VISUAL RULES

- Do not provide any scroll within a segment.
- Proper TAPABLE areas are required for icons, links etc... Even if the image size is smaller, provide a bigger tapable area as per the standards (Ex: Minimum of 44px for non-retina display)
- If there is no data on any field within the segment leave the space blank and show all the other data in their respective positions. Consistency plays an important role in the list segments.

GESTURES

We have different interaction on the list segments.

- Tap on the segment area to go to the details view.
- Right to Left swipe on the segment will provide the list level actions to the user like 'Edit', 'Delete' etc..
- Swipe up is for scrolling the list.

INTERNATIONALIZATION (i18n)/ LOCALIZATION (l10n) GUIDELINE

- Even if there is difference in the number of characters, the position of the elements remains the same.
- Truncation rules that apply to English also applies to other languages.

8.10 Add View

User can add a new business object by clicking on the ADD link on the top right corner of a List view. All the New Transactions are provided in the Module landing screen.

This page is the exact replica of a detail/ view page. User can add the informations with the help of mandatory field indications, place holder text.

8.10.1 Where To Use

User will use this to do all the new transactions.

8.10.2 When To Use

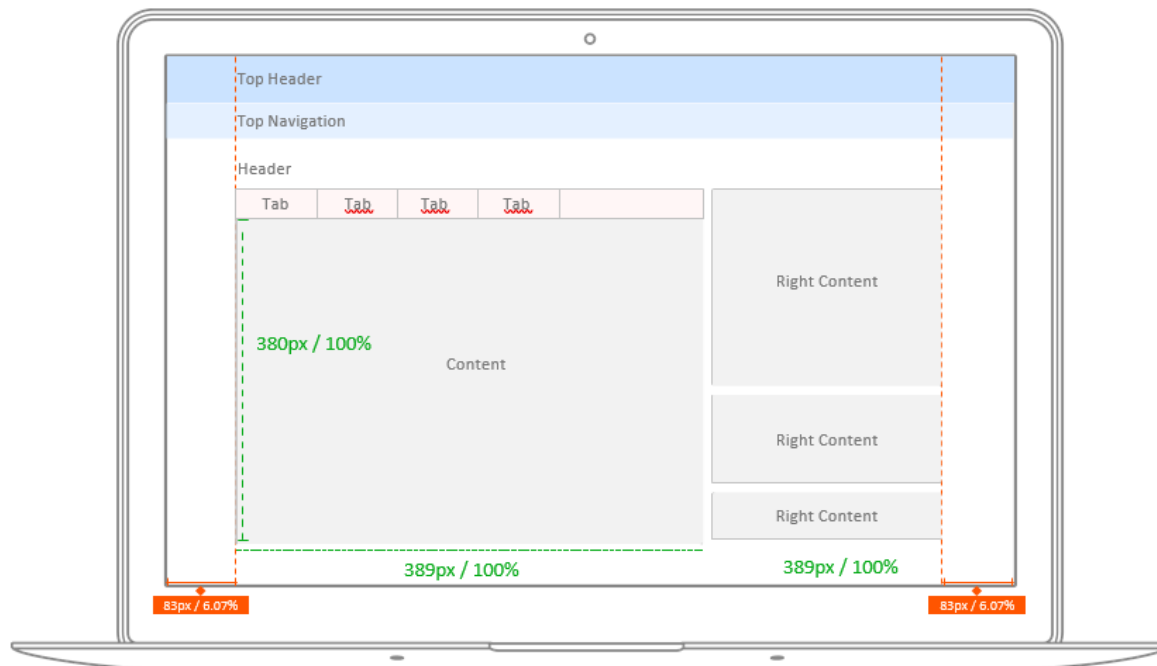
Use these pages only when you need to represent more information on a particular list item. When there is very less information in a item, this page can be skipped by showing the complete detail in the list itself. But most of the time that is practically not possible. The detail Page enables real estate for page level actions such as edit, delete, remove etc

8.10.3 Grid and Frame

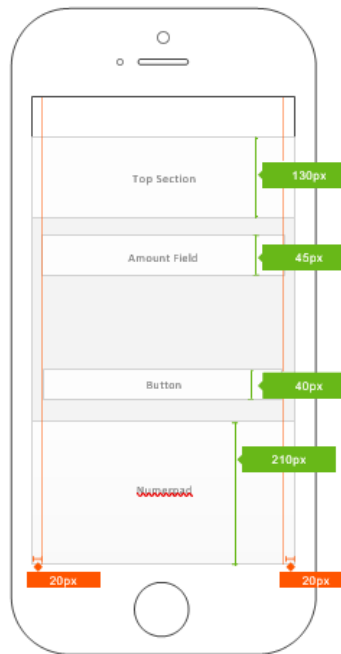
There are several combinations in representing the data in a list segment. But in general, certain rules need to be followed to make them look like a single family of the application.

Position of some common elements like Header bar, Search bar, Left margin, Right margin etc remain the same in all types of list.

Online Banking



Retail Banking

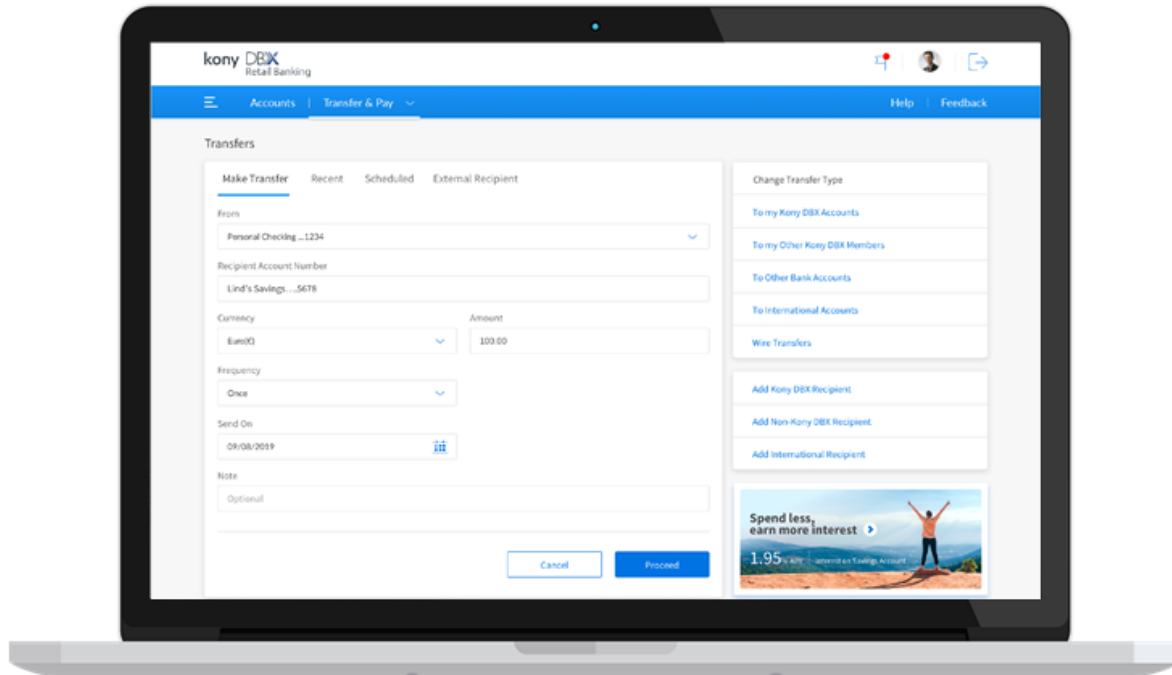


8.10.4 Screens

User can add a new business object by clicking on the ADD link on the top right corner of a List view. All the New Transactions are provided in the Module landing screen.

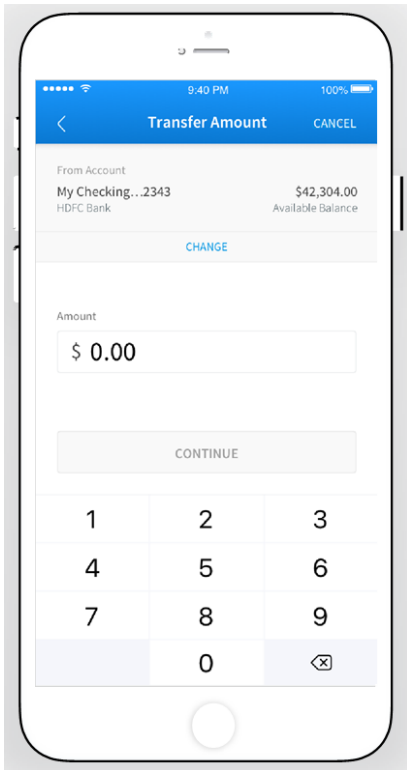
This page is the exact replica of a detail/ view page. User can add the informations with the help of mandatory field indications, place holder text.

Online Banking



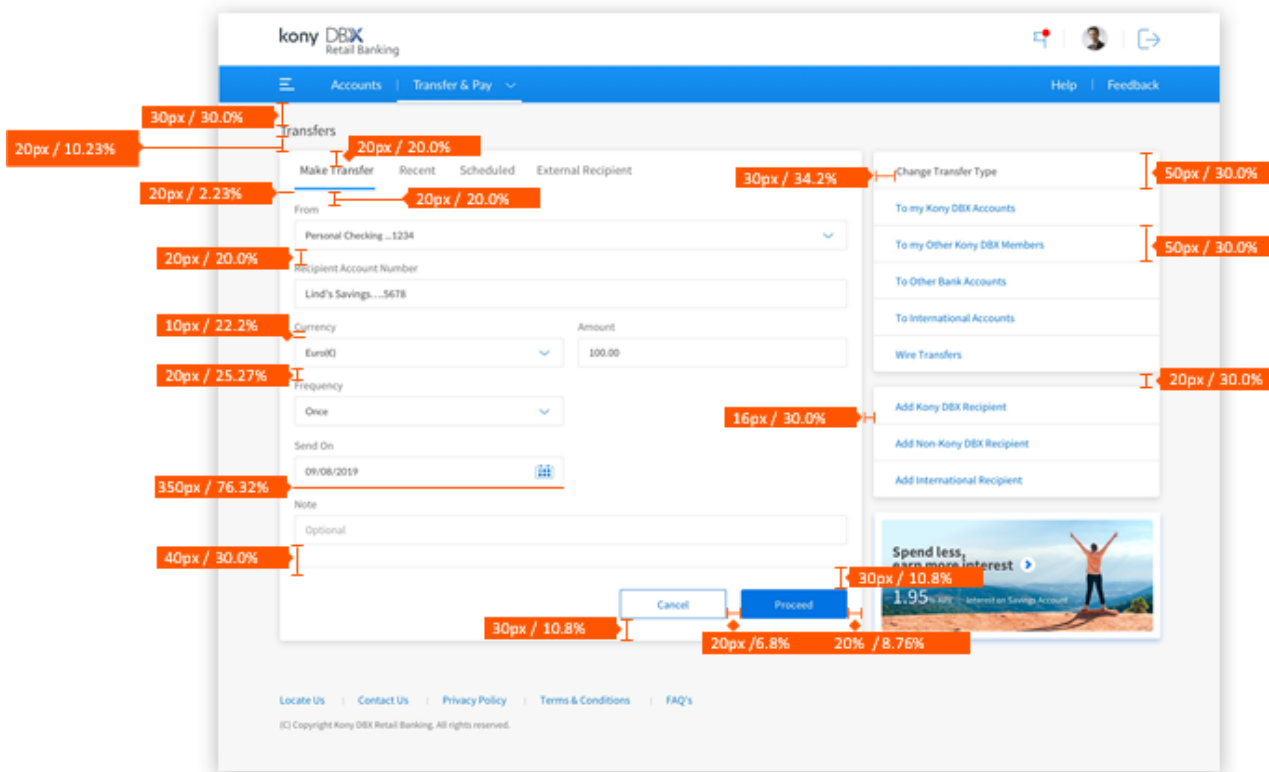
Retail Banking

Device: iPhone Screen Resolution: 375px/ 667px

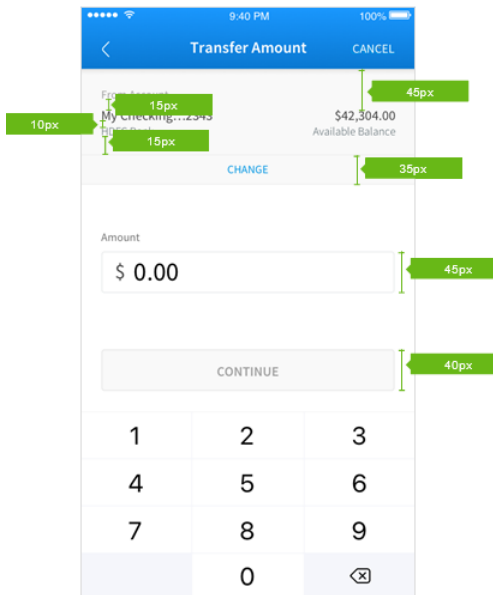


8.10.5 Specifications

Online Banking

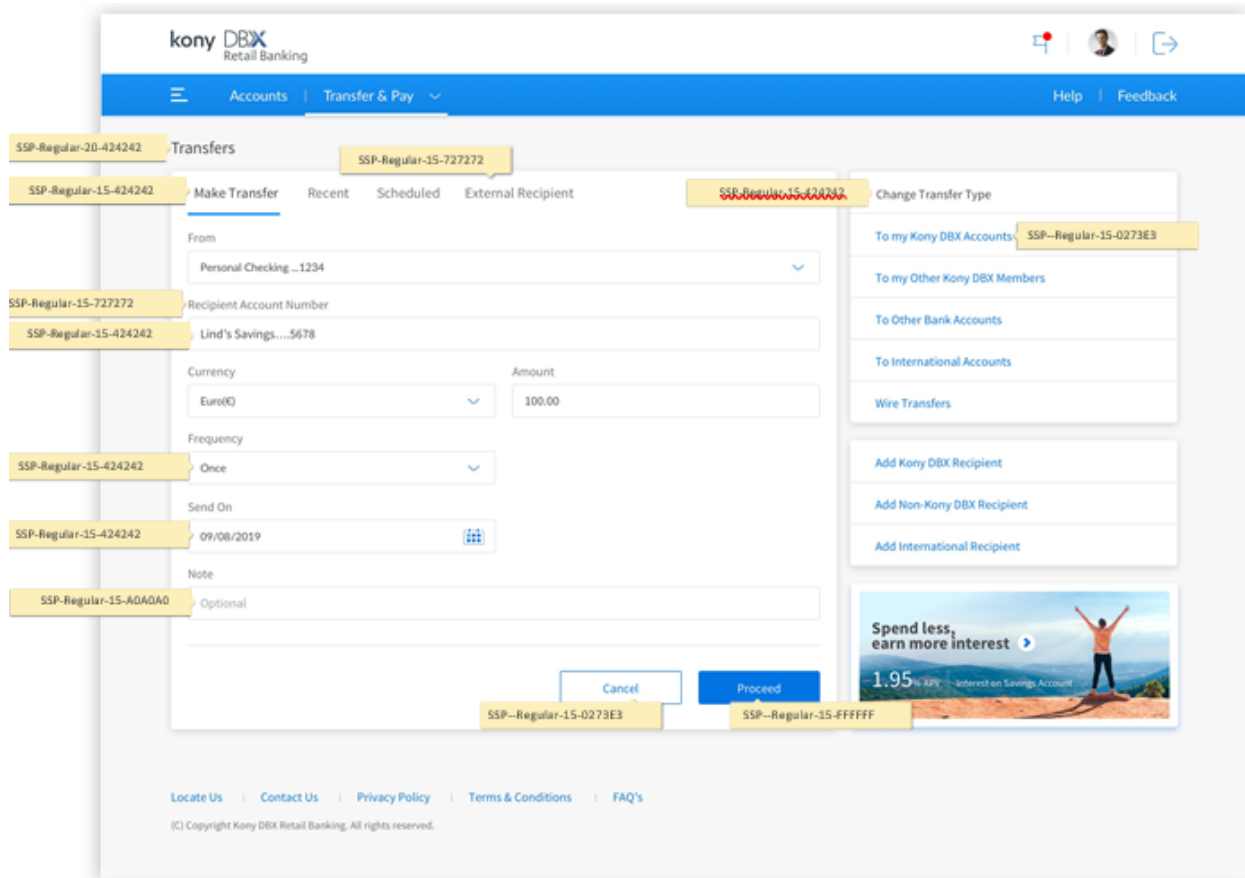


Retail Banking

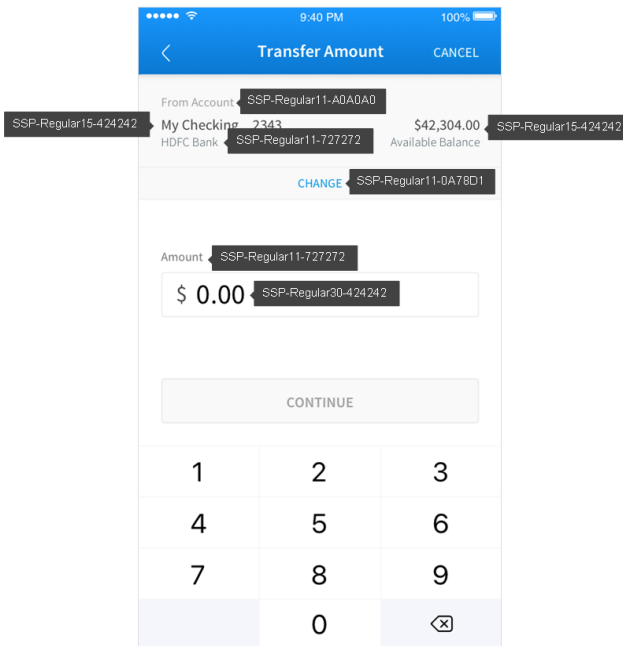


8.10.6 Font Specification

Online Banking



Retail Banking

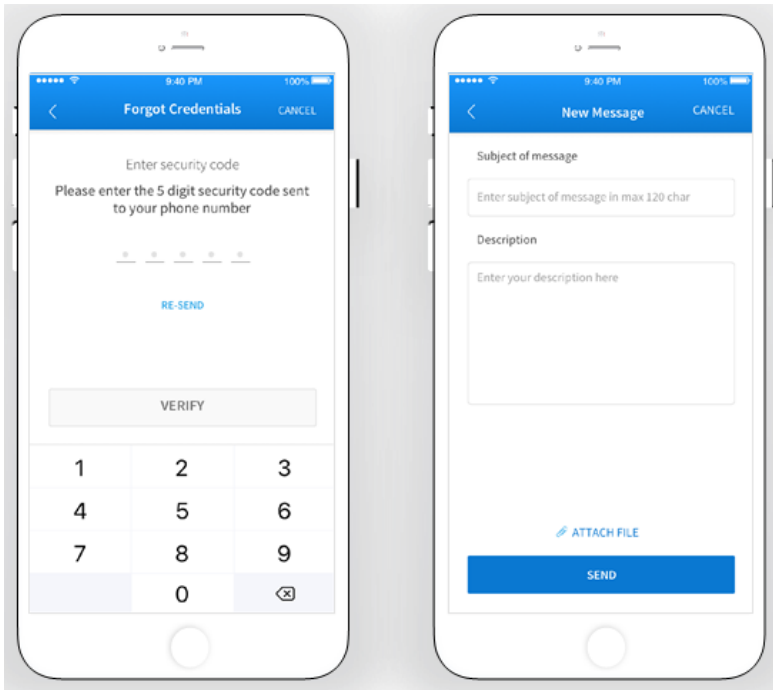


8.10.7 Types - Retail Banking

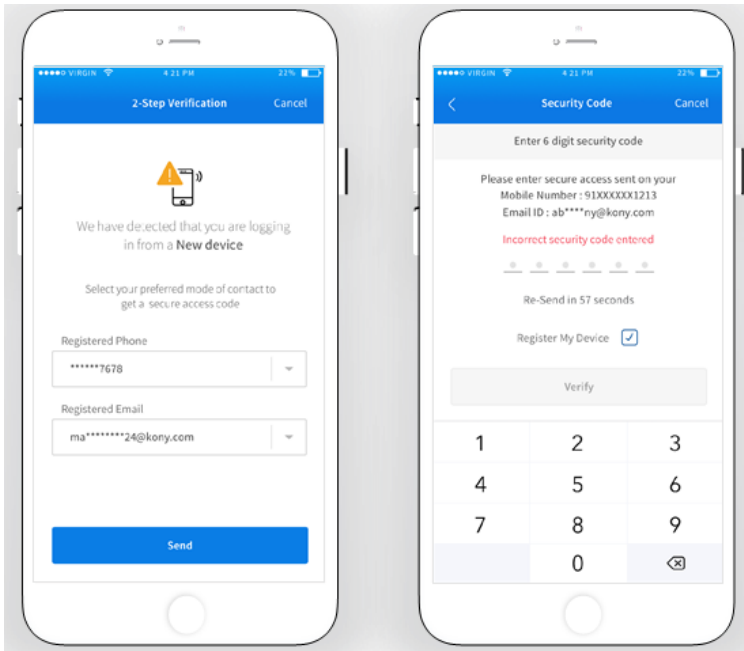
Types of Add View:

The Add page could get triggered from any one the following modes.

- Trigger from Login and will show up in between the flow.
- Trigger from transaction page.
- Displayed as part of a Message Page



- Trigger from Login and will show up in between the flow.



8.10.8 Design Principles

VISUAL RULES

- Do not provide any scroll within a segment.
- Proper TAPABLE areas are required for icons, links etc... Even if the image size is smaller, provide a bigger tapable area as per the standards (Ex: Minimum of 44px for non-retina display)
- If there is no data on any field within the segment leave the space blank and show all the other data in their respective positions. Consistency plays an important role in the list segments.

GESTURES

We have different interaction on the list segments.

- Tap on the segment area to go to the details view.
- Right to Left swipe on the segment will provide the list level actions to the user like 'Edit', 'Delete' etc...
- Swipe up is for scrolling the list.

INTERNATIONALIZATION (i18n)/ LOCALIZATION (l10n) GUIDELINE

- Even if there is difference in the number of characters, the position of the elements remains the same.
- Truncation rules that apply to English also applies to other languages.

8.11 PopUp Type - Online Banking

A pop-up is a graphical user interface (GUI) display area, usually a small window, that suddenly appears ("pops up") in the foreground of the visual interface. A variation on the pop-up window, opens a new browser window under the active window. Pop-up windows do not interrupt the user immediately, but appear when the user closes the covering window, making it more difficult to determine which website created them.

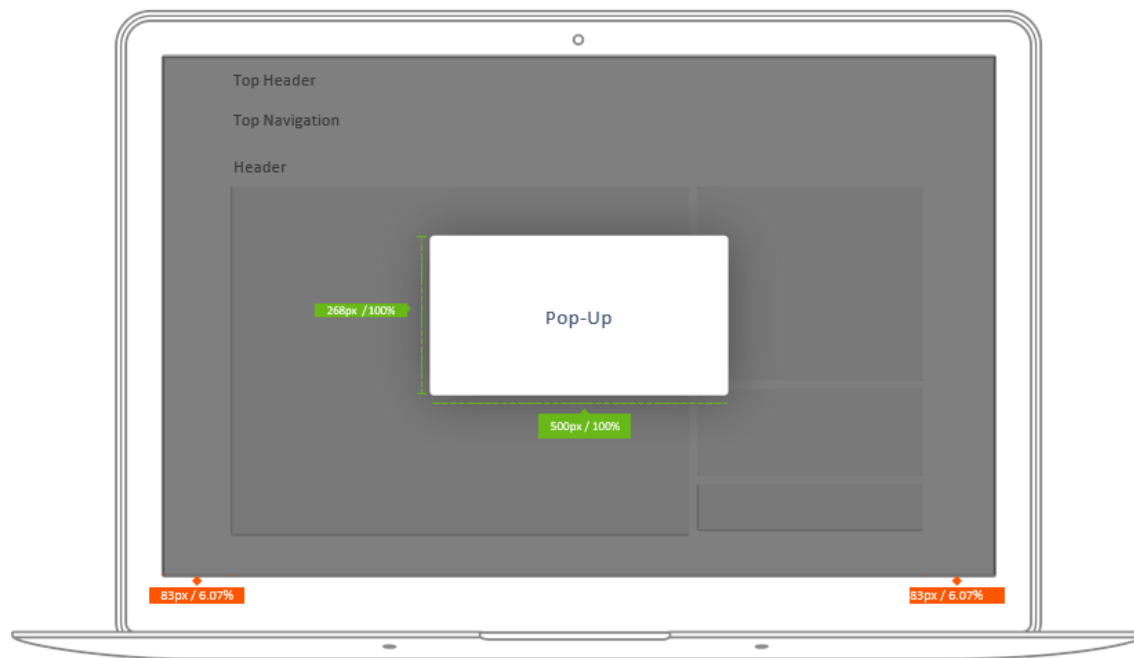
8.11.1 Where To Use

This page is designed to represent anything that requires the user attention and confirmation before cancelling any type of transaction or important actions.

8.11.2 When To Use

Use these pages only when you need to represent more information on a particular list item. When there is very less information in a item, this page can be skipped by showing the complete detail in the list itself. But most of the time that is practically not possible. The detail Page enables real estate for page level actions such as edit, delete, remove etc

8.11.3 Grid and Frame

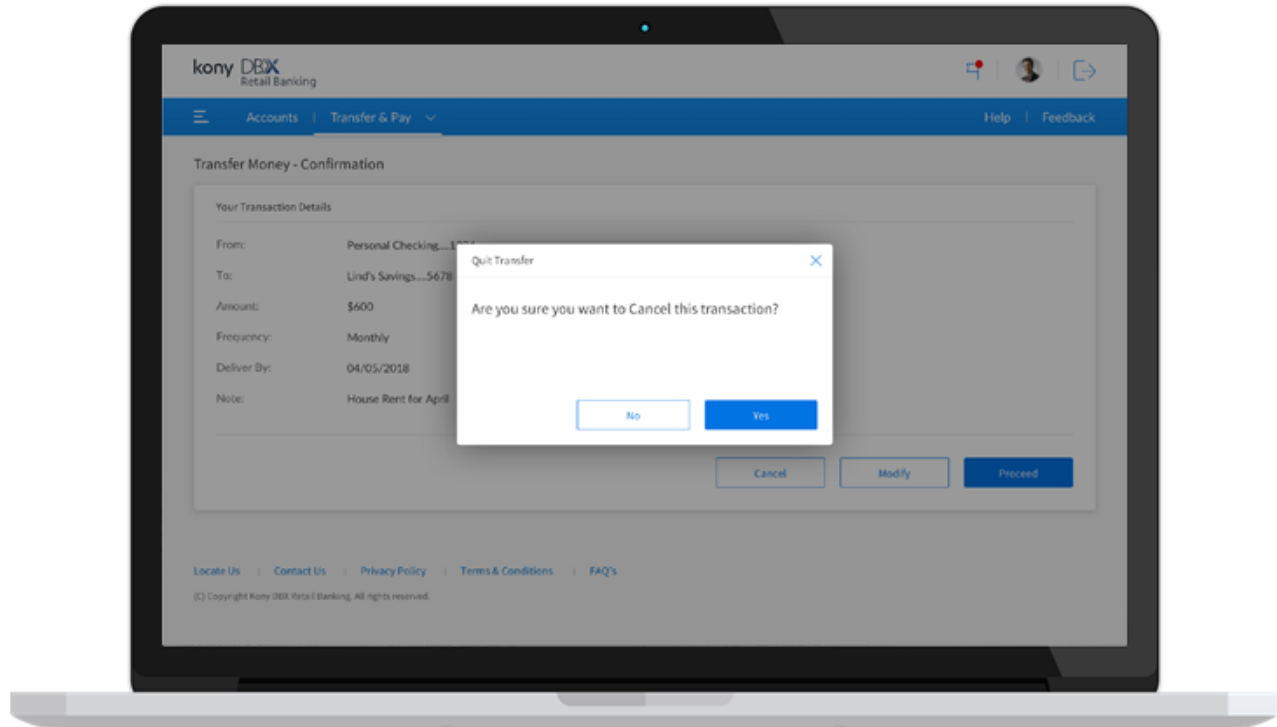


8.11.4 Screens

Detail page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen.

The detail screen maintains the relationship, continuity and context of the object from the list.

It also provides additional information about the object The related Information is presented in the form of drill down Page.



- Tap on the segment area to go to the details view.
- Right to Left swipe on the segment will provide the list level actions to the user like 'Edit', 'Delete' etc..
- Swipe up is for scrolling the list.

INTERNATIONALIZATION (i18n)/ LOCALIZATION (l10n) GUIDELINE

- Even if there is difference in the number of characters, the position of the elements remains the same.
- Truncation rules that apply to English also applies to other languages.

8.12 Confirmation - Online Banking

Detail page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen. The detail screen maintains the relationship, continuity and context of the object from the list .

It also provides additional information about the object The related Information is presented in the form of drill down Page.

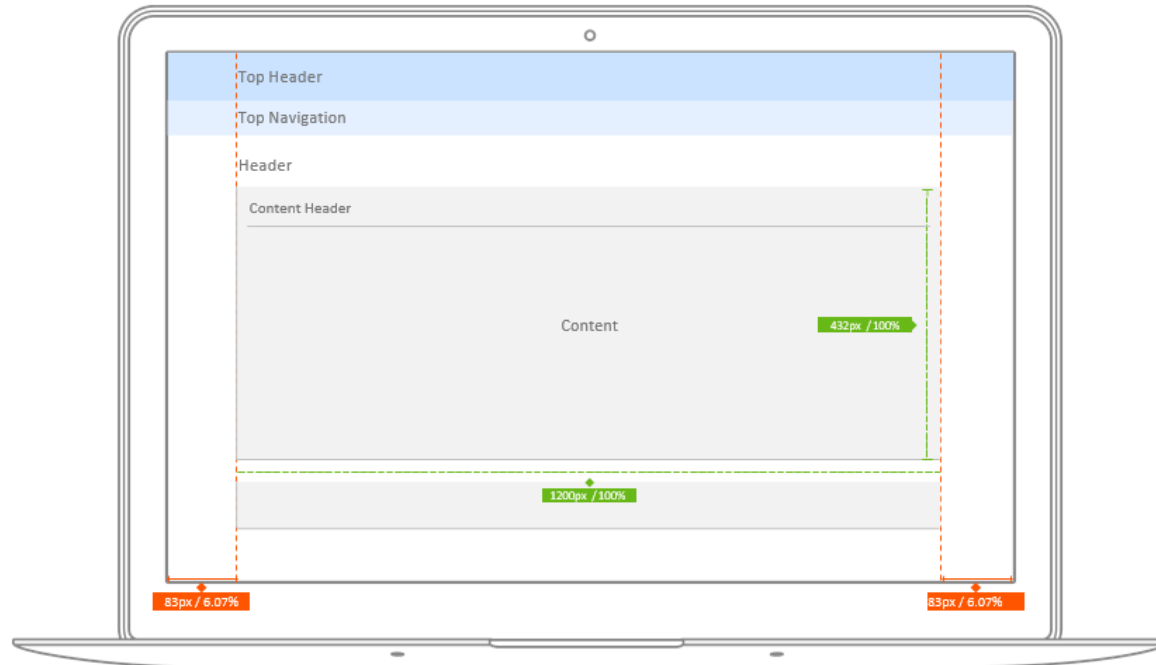
8.12.1 Where To Use

This page is designed to represent the complete detail of a list item. These pages should be represented as drilldown pages from a list view so that user can navigate back to list.

8.12.2 When To Use

Use these pages only when you need to represent more information on a particular list item. When there is very less information in a item, this page can be skipped by showing the complete detail in the list itself. But most of the time that is practically not possible. The detail Page enables real estate for page level actions such as edit, delete, remove etc

8.12.3 Grid and Frame

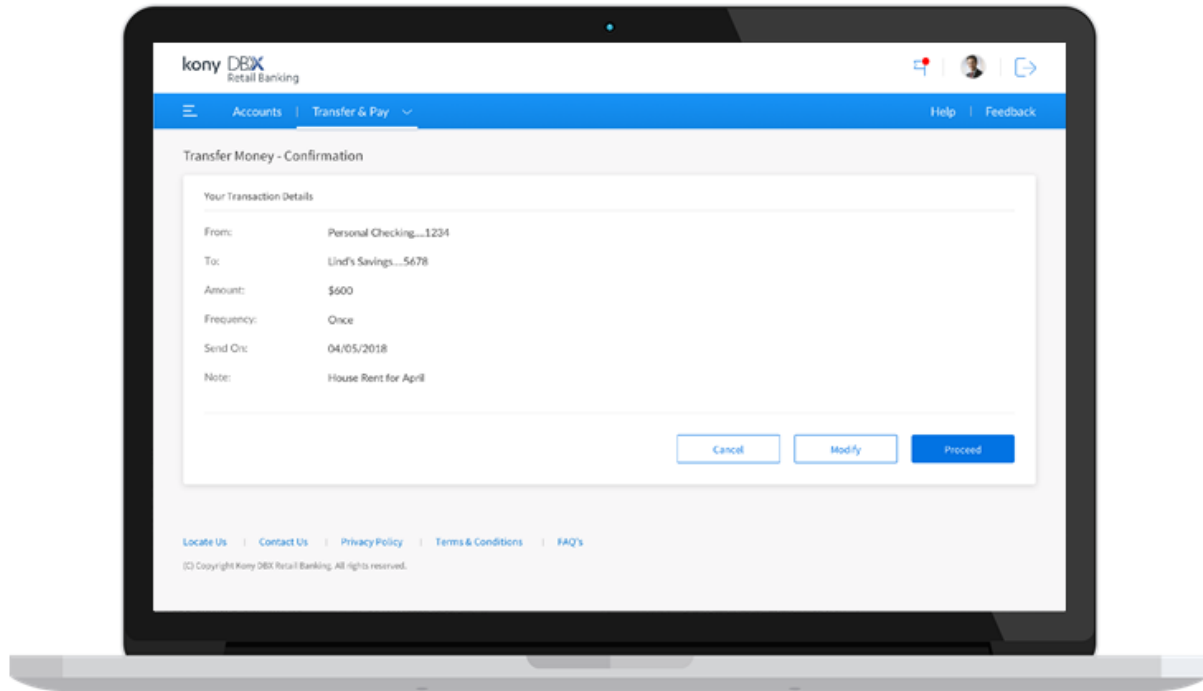


8.12.4 Screens

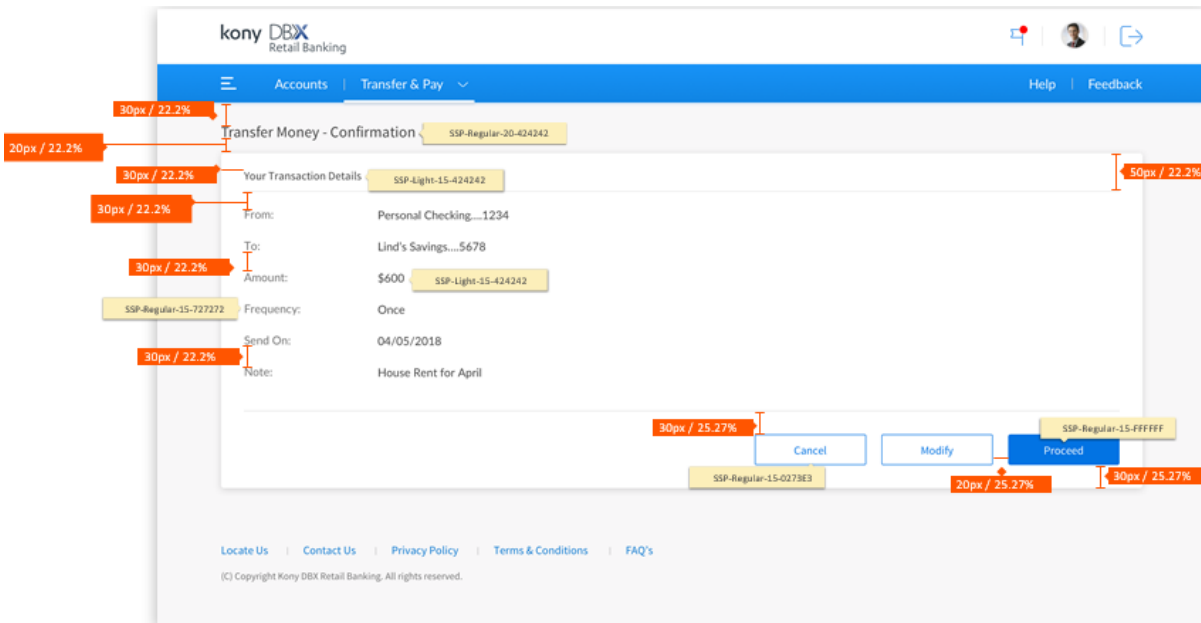
Detail page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen.

The detail screen maintains the relationship, continuity and context of the object from the list.

It also provides additional information about the object. The related Information is presented in the form of drill down Page



8.12.5 Specifications



8.12.6 Design Principles

VISUAL RULES

- Do not provide any scroll within a segment.
- Proper TAPABLE areas are required for icons, links etc... Even if the image size is smaller, provide a bigger tapable area as per the standards (Ex: Minimum of 44px for non-retina display)
- If there is no data on any field within the segment leave the space blank and show all the other data in their respective positions. Consistency plays an important role in the list segments.

GESTURES

We have different interaction on the list segments.

- Tap on the segment area to go to the details view.
- Right to Left swipe on the segment will provide the list level actions to the user like 'Edit', 'Delete' etc..
- Swipe up is for scrolling the list.

INTERNATIONALIZATION (i18n)/ LOCALIZATION (l10n) GUIDELINE

- Even if there is difference in the number of characters, the position of the elements remains the same.
- Truncation rules that apply to english also applies to other languages.

8.13 Acknowledgment - Online Banking

Detail page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen. The detail screen maintains the relationship, continuity and context of the object from the list .

It also provides additional information about the object The related Information is presented in the form of drill down Page.

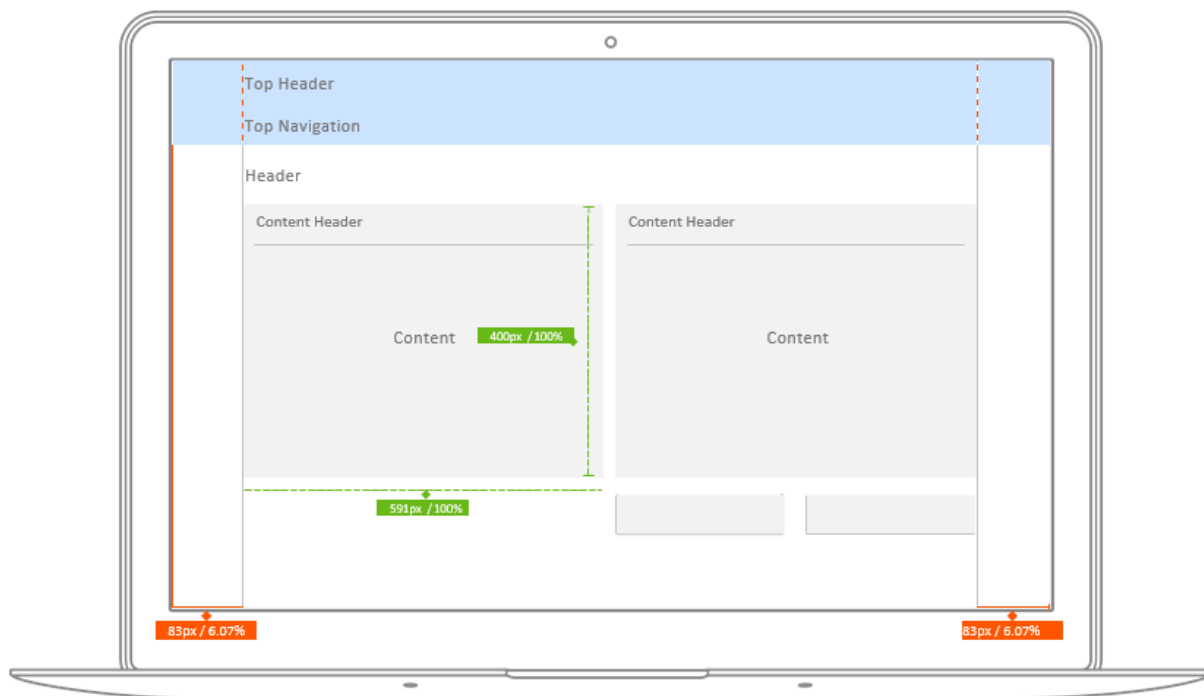
8.13.1 Where To Use

This page is designed to represent the complete detail of a list item. These pages should be represented as drilldown pages from a list view so that user can navigate back to list.

8.13.2 When To Use

Use these pages only when you need to represent more information on a particular list item. When there is very less information in a item, this page can be skipped by showing the complete detail in the list itself. But most of the time that is practically not possible. The detail Page enables real estate for page level actions such as edit, delete, remove etc

8.13.3 Grid and Frame

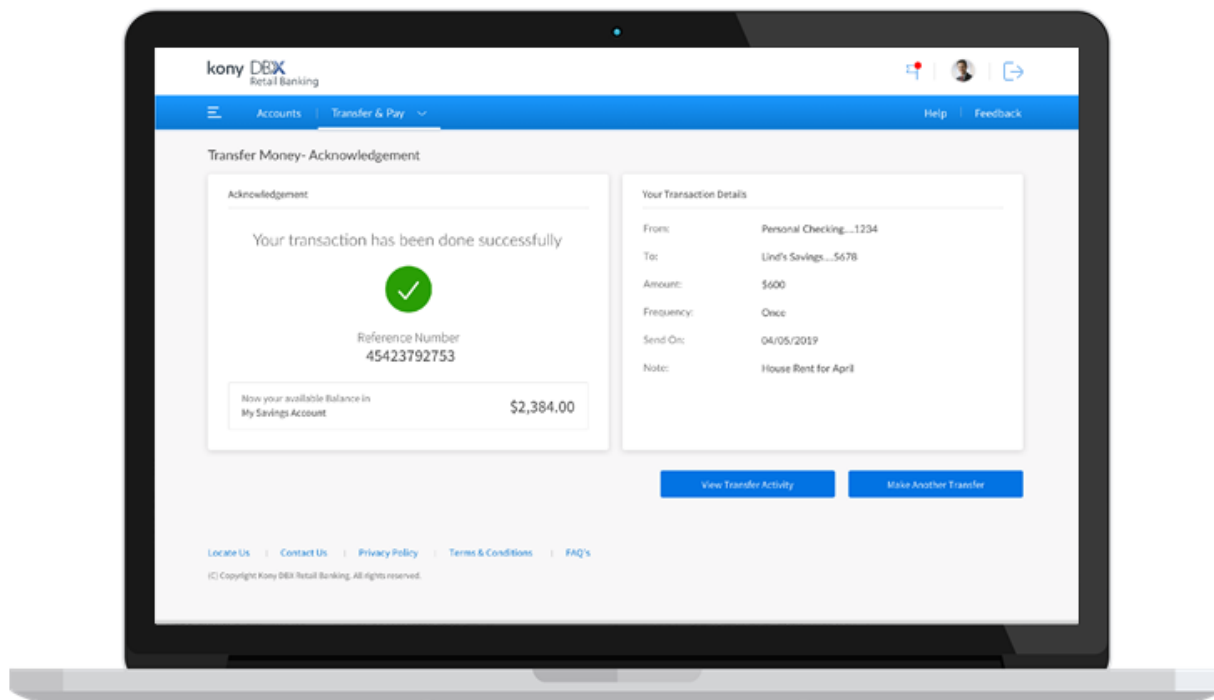


8.13.4 Screens

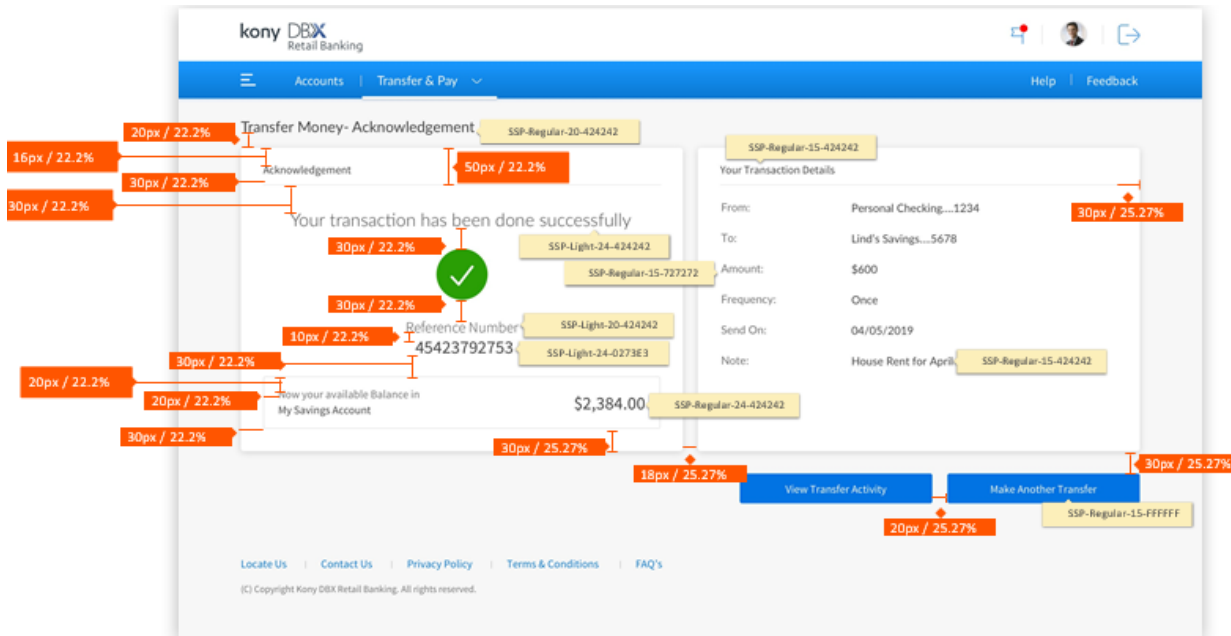
Detail page represents the complete detail of a particular business object. The user navigates to a detail page by tapping one of the records in a list screen.

The detail screen maintains the relationship, continuity and context of the object from the list.

It also provides additional information about the object. The related Information is presented in the form of drill down Page.



8.13.5 Specifications



8.13.6 Design Principles

VISUAL RULES

- Do not provide any scroll within a segment.
- Proper TAPABLE areas are required for icons, links etc... Even if the image size is smaller, provide a bigger tapable area as per the standards (Ex: Minimum of 44px for non-retina display)
- If there is no data on any field within the segment leave the space blank and show all the other data in their respective positions. Consistency plays an important role in the list segments.

GESTURES

We have different interaction on the list segments.

- Tap on the segment area to go to the details view.
- Right to Left swipe on the segment will provide the list level actions to the user like 'Edit', 'Delete' etc..
- Swipe up is for scrolling the list.

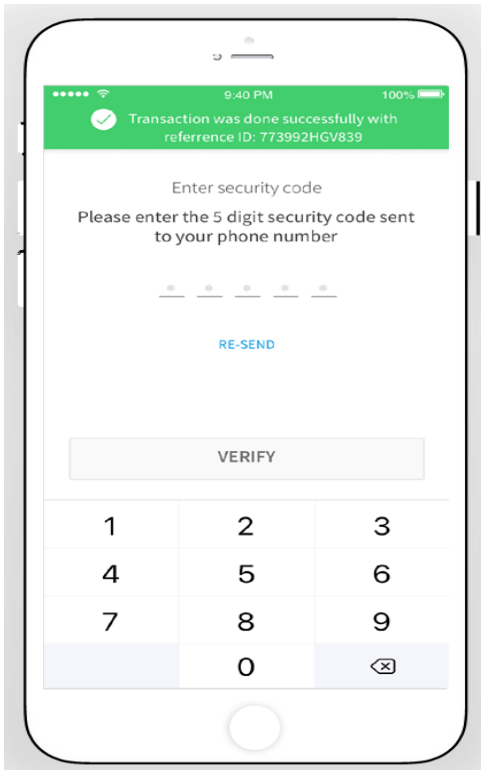
INTERNATIONALIZATION (i18n)/ LOCALIZATION (l10n) GUIDELINE

- Even if there is difference in the number of characters, the position of the elements remains the same.
- Truncation rules that apply to english also applies to other languages.

8.14 Toast Messages/ Notifications - Retail Banking

Toast Notifications should stay on the screen for 5-6 seconds, so that they do not block the information behind them for too long, but allows the user to read the message. They should not be interactive, you don't have to think on it or type on it and should be shown just to inform you with text/icons or both

Device: Iphone Screen Resolution: 375px/ 667px



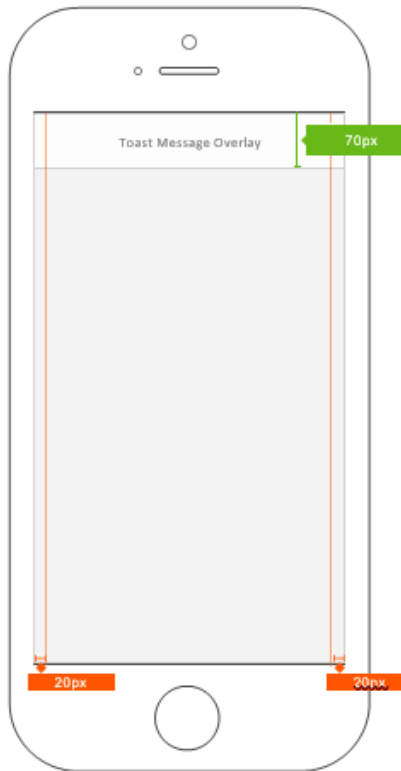
8.14.1 Where to Use

Normally these will be used when we perform any action on multiple items at a time on list view (Ex: delete), It may be during the action on single item, or submitting/ saving the record(s) etc...

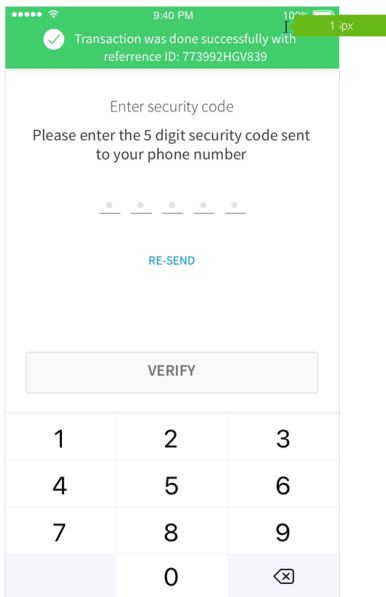
8.14.2 Frame

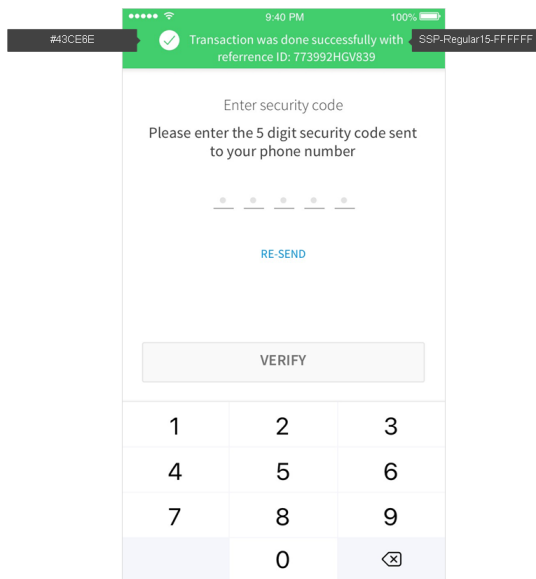
There are several combinations in representing the data in a list segment. But in general, certain rules need to be followed to make them look like a single family of the application.

Position of some common elements like Header bar, Search bar, Left margin, Right margin etc remain the same in all types of list.



8.14.3 Specification



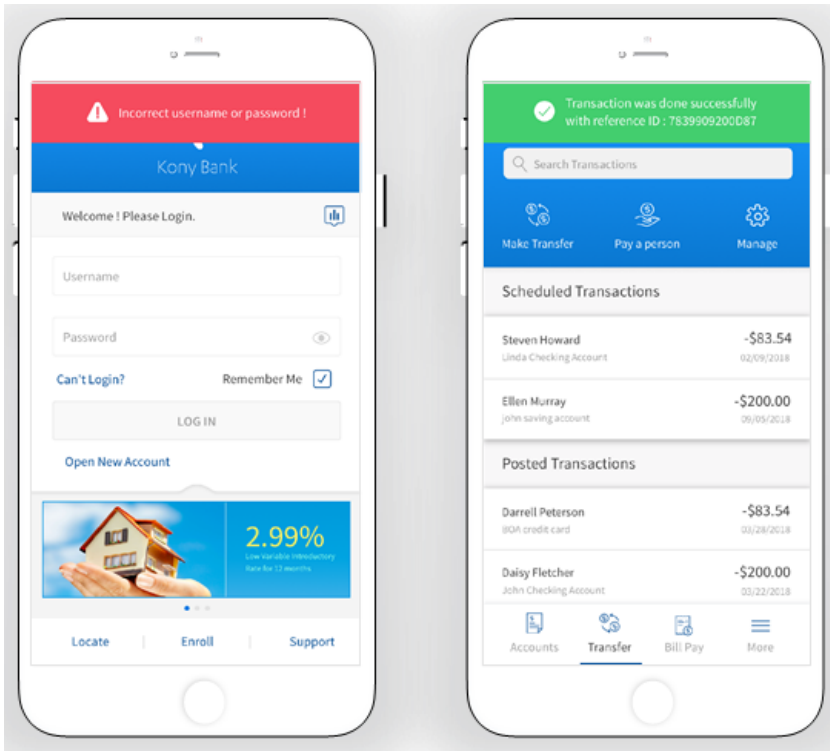


8.14.4 Types

Types of Toast Notification:

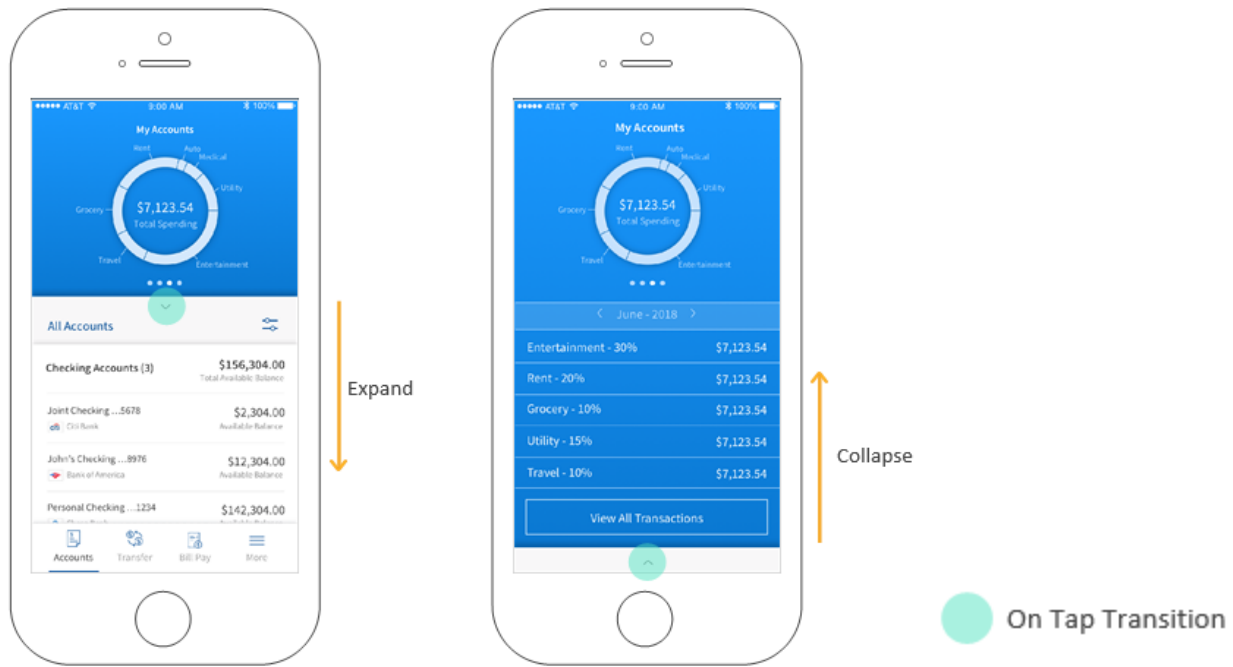
Use these messages when we perform any critical actions. Here are some of the examples...

- Confirmation/acknowledgement message
- Missing Fields
- Mandatory steps to perform before any action



8.15 Animation/Transition - Retail Banking

Page Transition/Component animations are part of app like page transition while navigation or transition of toast messages or expand and collapse mode or swipe mode, etc.

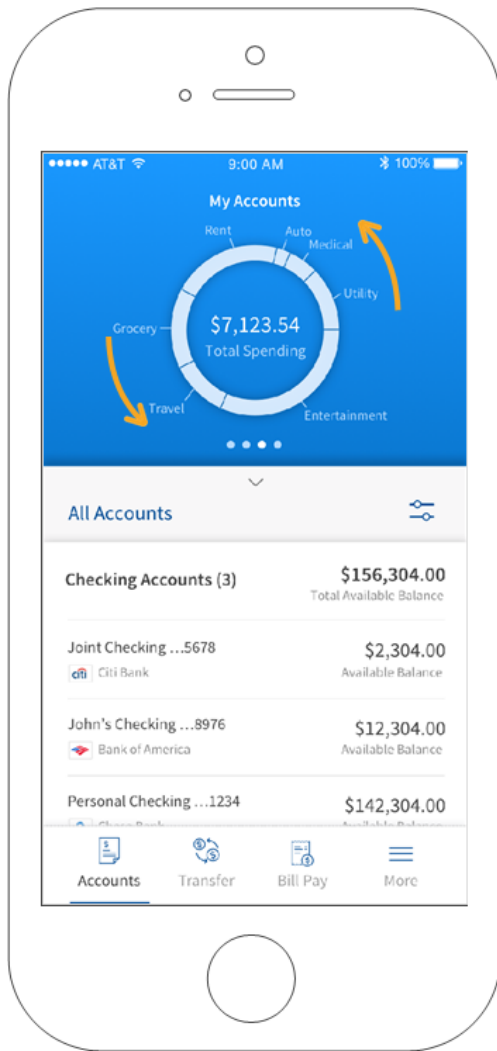


8.15.1 Where to Use

Normally these will be used everywhere when we perform any action (Ex: confirm message), It may be during the navigation from one page to other , or expand and collapse mode or submitting/saving the record(s) etc...

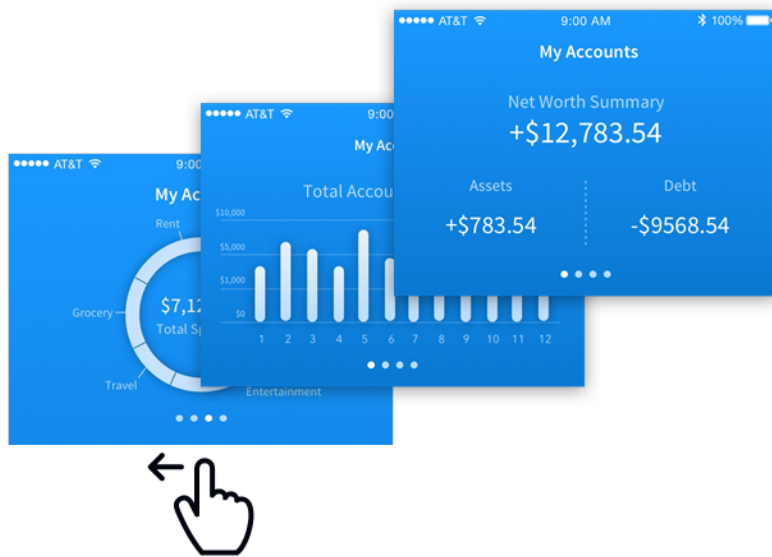
8.15.2 Types

Dashboard Chart has dynamic rotating feature. And also the has the swipe left carousel behavior.



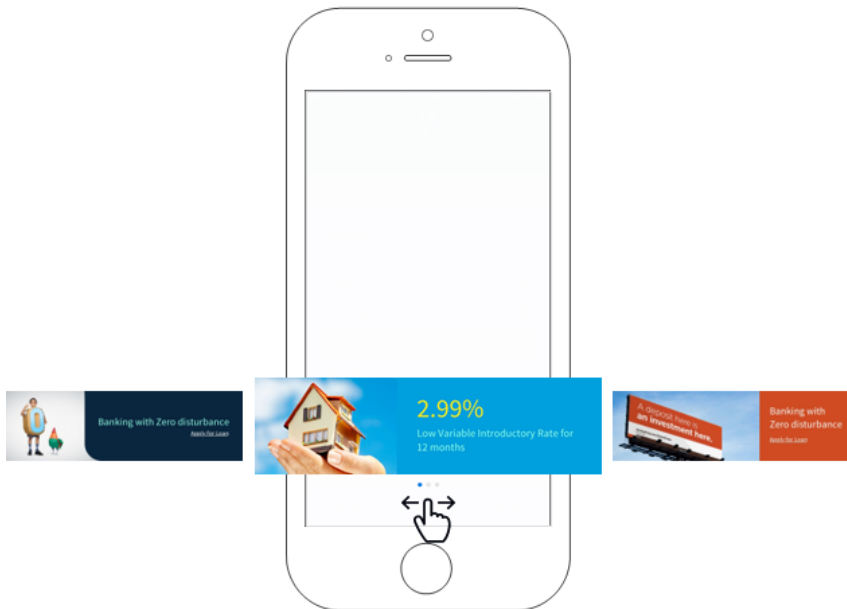
Rotating Chart

Swipe left carousel behavior



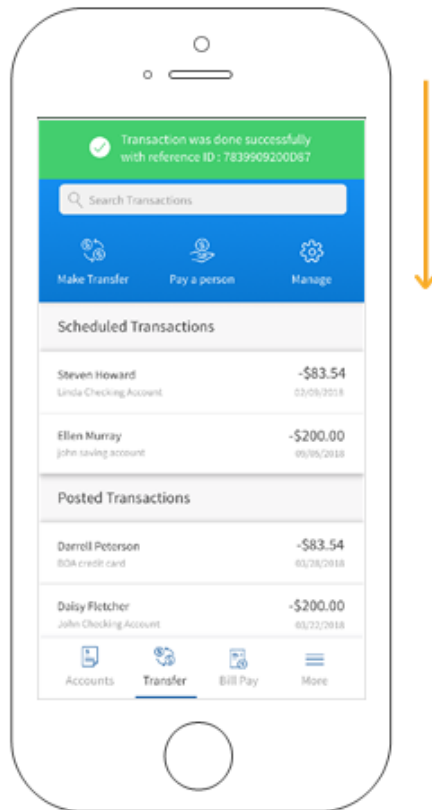
Carousel

Login Marketing ads has carousel behaviour to view different offers and information.



Transition

Toast notification has ease in and ease out transition. Toast Notifications stay on the screen for 5-6 seconds, so that they do not block the information behind them for too long, but allows the user to read the message.



8.16 Visual Indication and Button behavior - Retail Banking

Buttons are the most important action elements of any application. We have Primary buttons, text buttons, secondary buttons, icons etc.

Where to Use

We have several types of buttons.

- Primary Actions on any page is provided as Text button (Blue button- All caps)
- Secondary actions are provided as the text link buttons.
- Buttons on the app header are mainly are the Cancel, Back, Add.

8.16.1 Primary Button

Always use the button at the end of the page. (Active State)

A solid blue rectangular button with the word "VERIFY" in white, uppercase, sans-serif font centered on it.A solid blue rectangular button with the word "CONTINUE" in white, uppercase, sans-serif font centered on it.

(Inactive state)

A light gray rectangular button with a thin gray border and the word "VERIFY" in gray, uppercase, sans-serif font centered on it.A light gray rectangular button with a thin gray border and the word "CONTINUE" in gray, uppercase, sans-serif font centered on it.

8.16.2 Secondary Button

Use it as link buttons

A light gray rectangular button with a thin gray border and the text "Can't Login" in blue, sans-serif font centered on it.A light gray rectangular button with a thin gray border and the text "Open New Account" in blue, sans-serif font centered on it.

8.16.3 Do's and Dont's

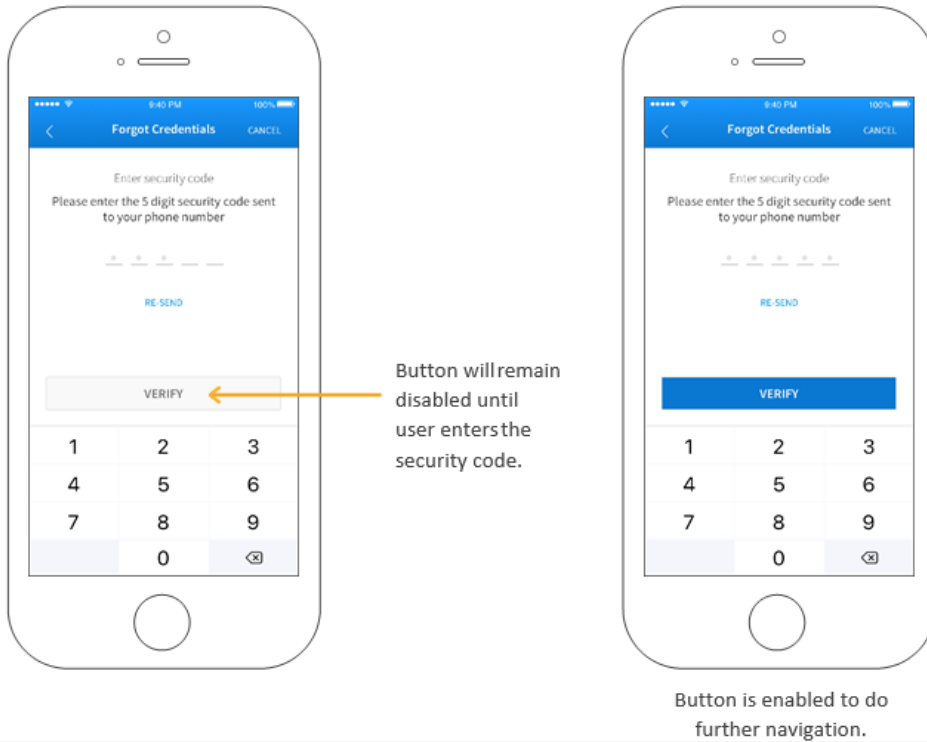
- Use very short labels. This helps in dealing with Internationalization.
- Always use same size boxes for all the controls.

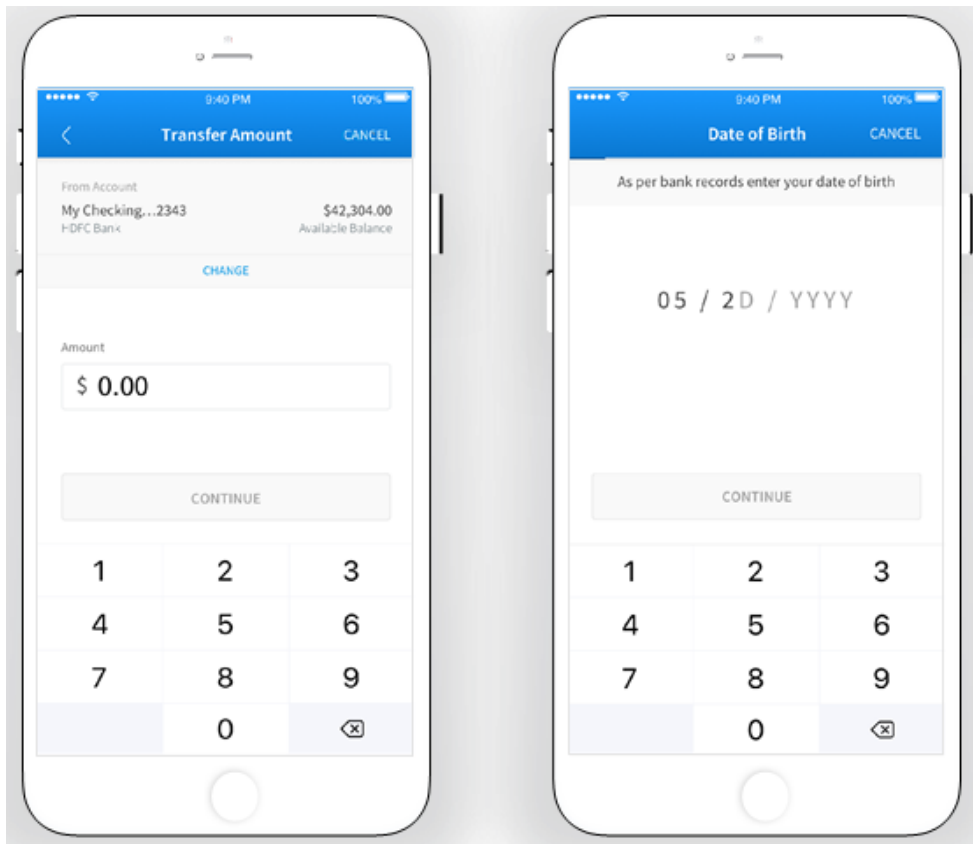
8.16.4 Types

Button 1

These types of buttons are used for security type pages/navigations. Ex: Entering CVV code, Security code, Date of birth etc. These buttons won't become enable unless user enter the require data asked to go further.

Button Type : 01

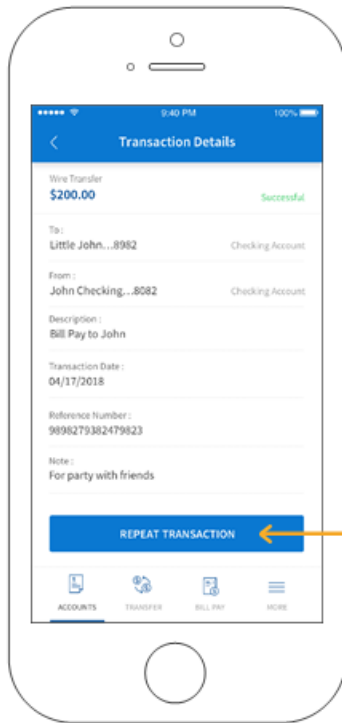




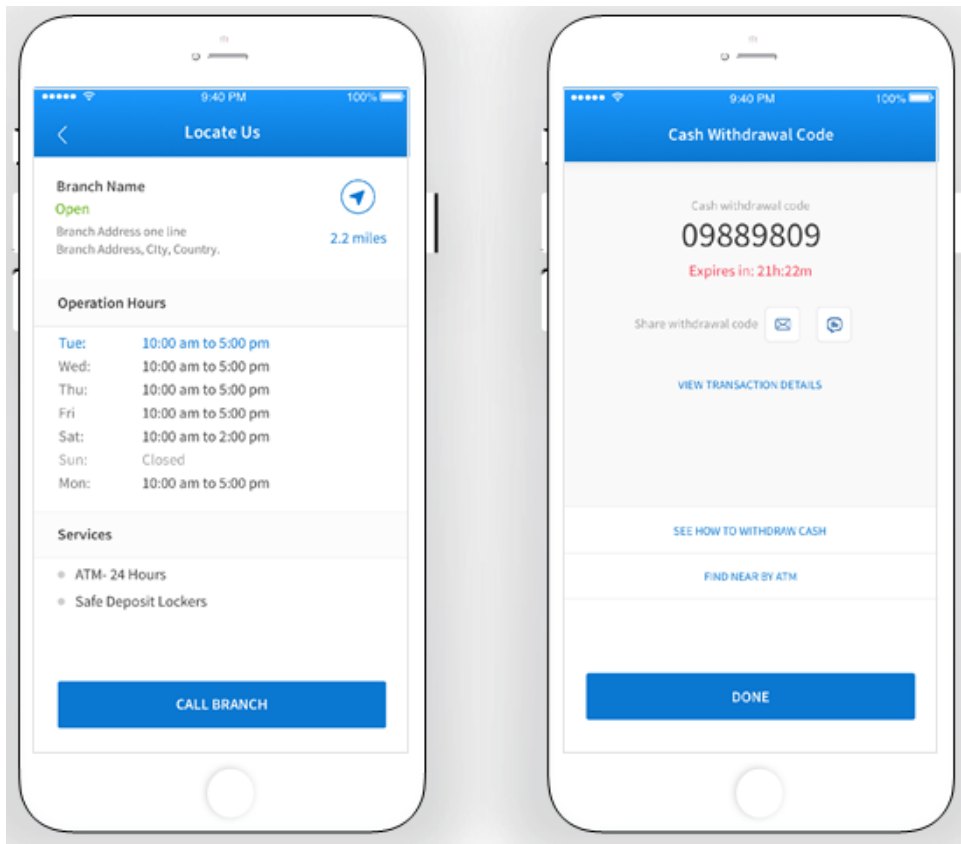
Button 2

These types of buttons are used in pages where user are not forced to enter data or required information.

Button Type : 02



Button will remain enabled in such forms.

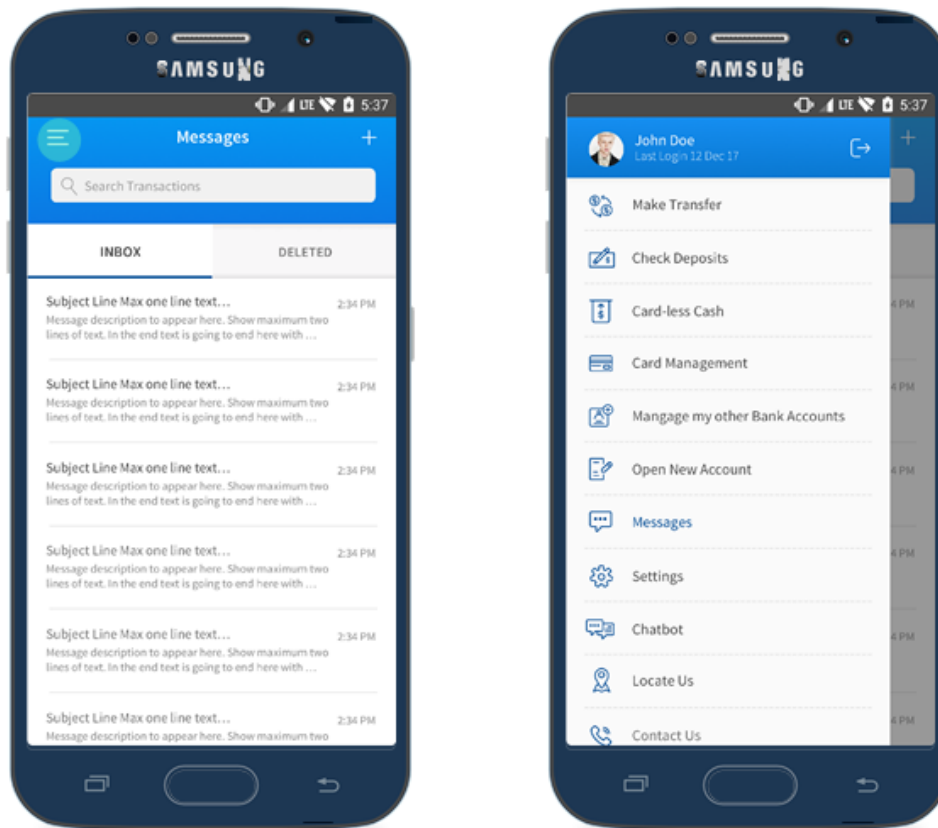


8.17 Menu - Retail Banking

Android Hamburger Menu

Hamburger menu instead of bottom menu - Move bottom bar menu options in to hamburger menu.

Android Device



iOS Footer Menu

Hamburger menu instead of bottom menu - Move bottom bar menu options in to hamburger menu.

