Migration Guide: 5.18 to 6.0

We're excited to bring you You.i Engine One version 6.0.0. As part of this release, there are some changes that may affect your application. For simplicity, the steps below are organized by the type of application you're developing.

Affecting All Apps

The following changes in release 6.0 affect both C++ and React Native applications.

Changes to Minimum Supported Versions of Third-Party Tools

Our minimum supported versions of some third-party tools have changed as follows. See also our full Hardware and Software Specifications.

Android NDK 21d

We've updated the version of Android's NDK that is supported by You.i Engine One to version 21d. Additionally, we now recommend that you use Android Studio 4.

Updating Android Studio will automatically update your `ANDROID_HOME` environment variable when it's installed. If you intend to use Android Studio for your app project, you must download the correct NDK and replace the `$ANDROID_HOME/ndk-bundle` as before. If you plan to develop and build from the command line, you can extract the NDK to a location like `$HOME/android-ndk-r21d` and build from there.

CMake 3.18

Due to the addition of support for Xcode 12, we've updated the version of CMake supported by You.i Engine One to version 3.18.

Visual Studio Target Platform Changes

Visual Studio 2019 is now used as the generator for the PS4, UWP, and Windows target platforms. When generating and building apps for Visual Studio, you must now specify the value for the `-p` option as `win64` instead of `vs2017`. Note that some third-party components may need to be reinstalled after upgrading to VS2019; contact us for support in upgrading your environment.

Prior to 6.0

```
youi-tv run -p vs2017
```

Now in 6.0

```
youi-tv run -p win64
```

Change to Accessibility Actions on Android Touch

We've fixed an issue where it wasn't possible to use the Android Touch hardware volume keys for the accessibility actions `increment` and `decrement`. Related to this fix, we've made some changes to behavior.

- When both the `increment` and `decrement` actions are added to a view with the `accessibilityRole` set to `adjustable`, the hardware volume keys can now be used to carry out the assigned increment and decrement actions as expected.
- If only the `increment` or the `decrement` action is added, but not both, the volume keys do not control the increment/decrement actions. Only the swipe gestures work in this case.
- Adding the `increment` and `decrement` actions to a view now overrides any default You.i Engine One handling for views such as lists and scrollbars. Your app needs to explicitly handle increment and decrement behavior for these elements in this case.

See also our Accessibility documentation.

Changes to Advertising ID on CYIDeviceInformationBridge and DeviceInfo

We've made changes to how a You.i Engine C++ or RN app retrieves advertising information from the devices your app is running on. We've moved the device `advertisingID` from `CYIDeviceInformationBridge` to the new `CYIAdvertisingInformationBridge` class. We've also moved the `advertisingId` method from the DeviceInfo module to a new AdvertisingInfo module.

These changes only affect your apps if you're currently making use of `advertisingId` on `CYIDeviceInformationBridge` for a C++ app, or `advertisingId` on DeviceInfo for an RN app. If you are, update your code as shown in the examples below.

C++ Apps

Prior to 6.0

```
CYIDeviceInformationBridge *pDeviceInfoBridge = CYIDeviceBridgeLocator::GetDeviceInformationBridge();
pDeviceInfoBridge->GetAdvertisingId([this](const CYIString &advertisingId) {
           // Do something with advertisingId
        });
```

Now in 6.0

```
CYIAdvertisingInformationBridge *pAdvertisingInformationBridge = CYIDeviceBridgeLocator::GetAdvertisingInforma
if (pAdvertisingInformationBridge)
{
    pAdvertisingInformationBridge->RequestAdvertisingId([](const CYIString &advertisingId) {
          // Do something with advertisingId
    });
}
```

RN Apps

Prior to 6.0

```
import { DeviceInfo } from '@youi/react-native-youi';

DeviceInfo.getAdvertisingId()
      .then((advertisingId) => {
        // Do something with advertisingId
      })
      .catch(() => {
        // advertisingId not available
      });
```

Now in 6.0

```
import { AdvertisingInfo } from "@youi/react-native-youi";

AdvertisingInfo.getAdvertisingId()
      .then((advertisingId) => {
        // Do something with advertisingId
      })
      .catch(() => {
```

```
        // advertisingId not available
        });
```

Additionally, for RN apps, you need to include the new module in your app's `UserInit()` as shown in the example below. For the `youi-tv init` app, for example, you can find the `UserInit` method in `/youi/src/App.cpp`.

```
#include <youireact/advertising/AdvertisingInfoModule.h>

bool App::UserInit()
{

    bool userInitSuccess = PlatformApp::UserInit();

    //To use the AdvertisingInfo module
    GetReactNativeViewController().AddModule<yi::react::AdvertisingInfoModule>();

    return userInitSuccess;
```

Upgrade to React Native 0.60

We've upgraded to React Native 0.60! When you install You.i Engine One version 6.0, React Native 0.60 is installed as well. Once that's done, you've got some work to do so your apps can enjoy the goodness of You.i Engine One with React Native 0.60. We'll guide you through:

- Configuring your project
- Clearing your JavaScript transform caches
- Updating your app code

Then come a few tasks required only if you use Appium or RAM bundling:

- Updating Appium
- Recreating module paths for RAM bundling

Finally, some things to be aware of:

- Enhancements
- Known issues and caveats

Configure Your Project

As a first step, there are a number of dependencies that need to be updated as a result of the upgrade to React Native 0.60.

Update package.json

Run these commands from your project folder to upgrade your app.

```
yarn upgrade react-native@0.60.6 react@16.8.6 metro-react-native-babel-preset@0.57.0 jest@^24.9.0 react-test-re
yarn add --dev @youi/babel-plugin-react-native-youi@~1.0.0
yarn add @babel/runtime@^7.6.2
yarn remove babel-preset-env babel-preset-react babel-register babel-preset-react-native
```

If you're using babel-jest, you need to upgrade it too.

```
yarn upgrade babel-jest@^24.9.0
```

Review Facebook RN's upgrade notes for 0.60.6 to determine whether updates are required for any other packages your app uses.

Create a Babel Configuration File

In your project's root folder, create a file named `babel.config.json`, with the following contents:

```
{
  "plugins": ["module:@youi/babel-plugin-react-native-youi"]
```

```
}
```

Update scheduler

Due to a Facebook React Native bug, our `react-native-youi` package now has an explicit dependency on version 0.14.0 of scheduler. If you've manually changed the version of scheduler in any of your You.i RN apps, you must now use version 0.14.0.

Remove the rn-cli.config.js File

Remove `rn-cli.config.js` from your project.

Add the react-native.config.js File

In your project's root folder, create a file named `react-native.config.js`, with the following content:

```
/**
 * React Native CLI configuration
 * https://github.com/react-native-community/cli/blob/master/docs/configuration.md

 */
const fs = require('fs');
const path = require('path');

module.exports = {
  reactNativePath: fs.realpathSync(
    path.resolve(require.resolve('@youi/react-native-youi/package.json'), '..')
  )
};
```

Add the metro.config.js File

In your project's root folder, create a file named `metro.config.js`, with the following content:

```
/**
 * Metro bundler configuration
 * https://facebook.github.io/metro/docs/configuration/

 */
const blacklist = require('metro-config/src/defaults/blacklist');
const path = require('path');
const fs = require('fs');

const reactNativeYouiPath = fs.realpathSync(
  path.resolve(require.resolve('@youi/react-native-youi/package.json'), '..')
);

module.exports = {
  resolver: {
    platforms: ['youi'],
    blacklistRE: blacklist([
      /\/youi\/build\/.*/,
      /node_modules\/react-native\/.*/
    ]),
    extraNodeModules: {
      // Redirect react-native to react-native-youi
      'react-native': reactNativeYouiPath,
      '@youi/react-native-youi': reactNativeYouiPath
    }
  },
  transformer: {
    getTransformOptions: async () => ({
      transform: {
        experimentalImportSupport: false,
        inlineRequires: false
```

```
      }
    })
  }
};
```

Clear Your JavaScript Transform Caches

Remove the `node_modules` folder from your app.

```
rm -r node_modules
```

Clear your watchman cache, if applicable.

```
watchman watch-del-all
```

Remove all directories in your temp directory that begin with the keywords `react`, `metro`, or `haste`.

```
rm -r $TMPDIR/react-*
rm -r $TMPDIR/metro-*
rm -r $TMPDIR/haste-*
```

Clear your Yarn cache.

```
yarn cache clean
```

Clear your Jest cache, if applicable.

```
yarn test --clearCache
```

For more information, see Clearing the Cache of Your React Native Project.

Make Changes to App Code

This set of tasks involves changes to your app code.

Change Your WebView Import

The WebView component is no longer available in Facebook's RN implementation. However, You.i Engine One includes an equivalent component. To access the component, ensure that your app imports `WebView` from `react-native-youi`.

Prior to 6.0

```
import { WebView } from 'react-native';
...
<WebView/>
```

Now in 6.0

```
import { WebView } from '@youi/react-native-youi';
...
<WebView/>
```

Changes to the ApplyProps Implementation for Native Components

We've renamed `ApplyProps` to `ApplyPropsPriv` to indicate that it's now a private function, and we no longer allow you to override it. If you have a native component that uses `ApplyProps`, you need to use `onPropsApplied` instead of `ApplyProps`.

As `onPropsApplied` does not take a folly dynamic object properties parameter, if you're using the properties parameter in your implementation, you now need to access the values using `GetProp`.

Note that this breaking change affects native components only.

Prior to 6.0

```
void MyNativeComponent::ApplyProps(folly::dynamic properties)
```

```
{
    ShadowViewRef::ApplyProps(properties);
    DoSomething();
    CYIString myProp;
    if(InitFromValue(myProp, properties["myProp"]))
    {
        DoSomethingElse(myProp);
    }
}
```

Now in 6.0

```
void MyNativeComponent::OnPropsApplied()
{
    DoSomething();
    folly::Optional<CYIString> myProp = GetProp<CYIString>("myProp");
    if(myProp)
    {
        DoSomethingElse(*myProp);
    }
}
```

Update the Transform Prop

Previously, the `rotate`, `rotateX`, `rotateY`, and `rotateZ` arguments of the Transform prop could take a `null`, `integer`, or `string` value. Now these arguments must take a string value with `rad` or `deg` appended after the value. See Facebook's React Native documentation on Transforms for more information.

Prior to 6.0

```
<View style={{transform: [{rotate: 0}]}} />
```

Now in 6.0

```
<View style={{transform: [{rotate: '0deg'}]}} />
```

Value for PlatformConstants isTesting Has Changed to False

Due to a change in how `Platform.isTesting` is used in Facebook React Native, the value for `isTesting` in You.i RN has changed from `true` to `false`. You may need to refactor any code that relies on the value for `isTesting`.

TextInput and TextInputRef prop returnKey changed to returnKeyType

A change was made to the `TextInput` and `TextInputRef` component prop `returnKey`. To be consistent with Facebook React Native, we changed the name to `returnKeyType`. The usage of the prop is the exact same as before, so there is no need to rewrite or refactor any code. You simply need to replace all instances of `returnKey` with `returnKeyType`.

Change Imports of the NetInfo Module

If you're using the `NetInfo` module in your project, some changes are required due to its removal from React Native 0.60.

Install a You.i Engine React Native–compatible version of the community-maintained `NetInfo` module (`@youi/netinfo`) using Yarn.

```
yarn add @youi/netinfo@^5.92
```

Change any existing `NetInfo` import statements in your code to import from `@youi/netinfo`.

Prior to 6.0

```
import { netinfo } from 'react-native';
```

Now in 6.0

```
import { netinfo } from '@youi/netinfo';
```

Changes to the AccessibilityInfo Module

The Facebook RN AccessibilityInfo module has changed significantly in RN 0.60. As a result, our support for this module has been updated. If you use `AccessibilityInfo`, you may need to update your code.

The following events have been renamed:

Prior to 6.0	Now in 6.0
`voiceOverDidChange`	`screenReaderChanged`
`announcementDidFinish`	`announcementFinished`

The `fetch` method of the `AccessibilityInfo` module is deprecated. We recommend using the `isScreenReaderEnabled` method instead.

The following methods have been added to the `AccessibilityInfo` module in React Native 0.60, but are currently unsupported by You.i React Native. Invoking these methods automatically rejects the supplied promise.

- `getCurrentBoldTextState`
- `getCurrentGrayscaleState`
- `getCurrentInvertColorsState`
- `getCurrentReduceMotionState`
- `getCurrentReduceTransparencyState`

See also AccessibilityInfo.

Update Appium

If you use Appium, make the changes in this section. All of these changes must be made from your project's `__tests__/appium` folder.

Update package.json

Update `package.json` by running the following commands from your project's `__tests__/appium` folder:

```
npm uninstall babel-preset-env babel-preset-react babel-register
npm install @babel/core@^7.6.2 @babel/preset-env@^7.6.2 @babel/register@^7.6.2
```

Edit wdio.conf.js

Edit `wdio.conf.js`, changing `mochaOpts` to use `@babel/register` as follows:

Prior to 6.0

```
mochaOpts: {
        ui: 'bdd',
        timeout: 60000,
        compilers: ['js:babel-register'],
    },
```

Now in 6.0

```
mochaOpts: {
        ui: 'bdd',
        timeout: 60000,
        require: ['@babel/register'],
    },
```

Configure Babel

Delete `.babelrc` from your project's `__tests__/appium` folder.

In the same folder, create a file named `babel.config.js` with the following contents:

```
module.exports = {
    presets: [
        [ '@babel/preset-env', {
            targets: {
                node: 11,
            },
        } ],
    ],
};
```

Optional: Use yarn Instead of npm for Existing Projects

Going forward, all newly generated You.i Engine React Native projects will use yarn instead of npm for Appium tests. If you want to use yarn for Appium testing on existing projects, here's how.

From your project's `__tests__/appium` folder, run:

```
yarn import
```

Note: Calling `yarn import` is required because Appium uses a number of packages that don't declare their dependencies properly. For more information on `yarn import` and its usage in solving these problems, see yarn import in the Yarn documentation.

Delete `package-lock.json`.

Edit the generated Readme file in your project's `__tests__/appium` folder, replacing `npm` with `yarn` as follows:

- Change `npm install` to `yarn`
- Change `npm run-script <platformName>` to `yarn <platformName>`

Recreate Module Paths for RAM Bundling

If you use RAM bundling to optimize app startup performance, you need to recreate your list of module paths. This requirement is due to two changes in Facebook React Native 0.60:

- Relative path resolution is now used.
- Files in the `react-native` library have been added, removed, and moved.

Here's an example showing how JS module paths have changed in You.i RN:

Prior to 6.0

```
'node_modules/react-native/Libraries/Core/InitializeCore.js',
```

Now in 6.0

```
'node_modules/@youi/react-native-youi/Libraries/Core/InitializeCore.js',
```

Because of the changes to the `react-native` library, rather than just editing the paths in your module paths file, you need to regenerate the module paths by completing the following steps.

Clear the contents of the module.exports array

In the module paths file where you specify the initially loaded modules for the RAM bundle, clear the entire contents of the array. For example:

```
module.exports = [
]
```

Generate your project with RAM bundling

As explained in Generate a Project with RAM Bundling, generate your project by running `youi-tv build` with the `--ram_bundle` option. For example:

```
youi-tv build -p osx --ram_bundle --file index.youi.js --bundler_configuration <App Path>/config.js --sourcemap
```

Gather module information from the output sourcemap JSON file

As explained in Gather Module Information, use the JSON file specified with the `--sourcemap_output` option for `youi-tv build` to gather JS module information.

Add the new JS module paths to the module.exports array

As explained in Add JS Module Dependencies, fill the `module.exports` array with the new JS modules and their paths. For example:

```
module.exports = [
'node_modules/@youi/react-native-youi/Libraries/Core/InitializeCore.js',
'node_modules/@youi/react-native-youi/Libraries/Core/setUpGlobals.js',
'node_modules/@youi/react-native-youi/Libraries/Core/polyfillES6Collections.js',
'node_modules/@youi/react-native-youi/Libraries/vendor/core/_shouldPolyfillES6Collection.js',
'node_modules/@youi/react-native-youi/Libraries/Core/setUpSystrace.js',
'node_modules/@youi/react-native-youi/Libraries/Core/setUpErrorHandling.js',
'node_modules/@youi/react-native-youi/Libraries/Core/ExceptionsManager.js',
]
```

React Native Known Issues

This section provides workarounds for issues you may notice after migration to React Native 0.60.

Error with File Watcher System Limit on Linux

When building a React Native app on a Linux system, you may run into the following error:

```
Error: ENOSPC: System limit for number of file watchers reached
```

This is due to the size of your project exceeding the system limit for the number of file watchers. All of the files within your project are watched, including `node_modules`.

According to Facebook, this issue should be resolved by increasing your file watcher limit to the maximum. Before you do this, however, restart your computer as this sometimes remedies the problem.

If restarting your computer doesn't work, increase your file watcher limit. To do this, edit `/etc/sysctl.conf`, and add this line to the end of the file:

```
fs.inotify.max_user_watches=524288
```

Load the new value:

```
sudo sysctl -p
```

Jest Tests May Not Mock Properly

In React Native, if you include the following line in your project's `package.json` file for Jest testing:

```
"transform": {
      "^.+\\.js$": "<rootDir>/node_modules/react-native/jest/preprocessor.js"
    }
```

And if you import your component to test, and mock part of its render as follows:

```
import App from '../App';
jest.mock('../OtherComponent', () => () => 'OtherComponent');
```

```
it('renders correctly', () => {
  const tree = renderer.create(<App />).toJSON();
  expect(tree).toMatchSnapshot();
});
```

You'll notice that the resulting snap file hasn't mocked out what you requested to be mocked.

This happens because the default `babel-jest` transform for Jest testing, which includes `jest-hoist`, has been overwritten.

To address this issue, do one of the following:

- Change the test to require the component as follows:

```
const App = require('../App').default;
jest.mock('../OtherComponent', () => () => 'OtherComponent');

it('renders correctly', () => {
  const tree = renderer.create(<App />).toJSON();
  expect(tree).toMatchSnapshot();
});
```

- Or, add jest-hoist to your project's `.babelrc`.

Jest Testing Type Error

The following error may occur during Jest testing. It's caused by using the preprocessor as the transform in your Jest configuration.

```
TypeError: Cannot read property 'default' of undefined

    > 6 | export default class MyComponent extends React.Component {
```

Here's the Jest configuration that causes the error:

```
"transform": {
      "^.+\\.js$": "<rootDir>/node_modules/react-native/jest/preprocessor.js"
    }
```

We recommend adding the following code to your components to prevent this error:

```
constructor(props){
 super(props);
}
```

We've observed that this issue occurs on components that:

- have member variables declared outside of the constructor with no constructor defined
- use arrow functions

This error also occurs with Facebook React Native.

Additional Changes Affecting React Native Apps

React Native Breaking Changes

The following changes in release 6.0 may affect your React Native applications.

Thin Lines Rendering Changes

Prior to 6.0, the rendering of a View component that imitated a line (through a small height value) depended on both its location on screen and the device type. In this release, we've changed the way a View with a fixed height and background is rendered. This new behavior requires the following code changes.

Ensure Rendering Precision with Anti-aliasing

As You.i Engine React Native doesn't use anti-aliasing by default on solid color backgrounds, the rendering precision for a view may vary depending on its location on the screen. For example, given a grouping of lines with less than a pixel in height, you'll notice some lines disappearing:

```
renderLineItems = () =>
    [...Array(50).keys()].map((_, i) => (
        <>
        <Text>Line - {i}</Text>
        <View
            style={{
                height: 0.7f / Dimensions.get('screen').scale, // Line will be 0.7 pixels high.
                backgroundColor: 'blue'
            }}
        />
    </>
));
```

By placing a small and unnoticeable border on these lines as shown in the code below, you can add anti-aliasing on a view, so that the lines render as expected:

```
renderLineItems = () =>
    [...Array(50).keys()].map((_, i) => (
        <>
        <Text>Line - {i}</Text>
        <View
            style={{
                height: 0.7f / Dimensions.get('screen').scale, // Line will be 0.7 pixels high.
                borderRadius: 0.1f, // Adds anti-aliasing due to imperceptible radius.
                backgroundColor: 'blue'
            }}
        />
    </>
));
```

Platform-specific DIPs Rounding No Longer Required

Prior to this change, you had to implement platform-specific rounding on the DIPs value. This is no longer required. The value for the `height` variable is used as-is.

```
renderLineItems = () =>
    [...Array(50).keys()].map((_, i) => (
        <>
        <Text>Line - {i}</Text>
        <View
            style={{
                height: 0.7f / Dimensions.get('screen').scale, // Line will be 0.7 pixels high.
                borderRadius: 0.1f, // Adds anti-aliasing due to imperceptible radius.
                backgroundColor: 'blue'
            }}
        />
    </>
));
```

React Native Behavior Changes

Requesting Initial Focus on a List Item

We've made it easier to request initial focus on a list item. Previously, this involved the following steps:

1. Store the reference of the item that you want to focus.
2. Wait for the list item to become visible, or use a delay such as a timeout or a debounce.
3. Pass the ref to `FocusManager.focus`.

Now in release 6.0, you can simply request focus once the item ref is available.

Prior to 6.0

```
// Example 1: Use setTimeout or some other mechanism like debounce to wait an extra frame.
componentDidMount() {
    setTimeout(FocusManager.focus(componentRef));
}

// Example 2: Call this using the component's onLayout and make sure it's only called the first time onLayout
onLayout() {
    if (this.props.defaultFocus && !this.isButtonMounted) {
        FocusManager.focus(this.buttonRef);
        this.isButtonMounted = true;
```

```
    }
}

// Example 3: Call this on the list's handleViewableItemsChanged and focus when the component is visible.
handleViewableItemsChanged = ({ viewableItems }) => {
    if (this.shouldRequestFocus) {
        const buttonMounted = viewableItems.find(({ index, isViewable }) => {
        return isViewable && index === this.defaultButtonToFocus.index;
    });

        if (buttonMounted) {
            FocusManager.focus(this.defaultButtonToFocus.ref);
            this.shouldRequestFocus = false;
        }
    }
};
```

Now in 6.0

```
// Example 1: Store the ref and reference it in componentDidMount.
componentDidMount() {
    FocusManager.focus(componentRef);
}

// Example 2: Focus when the ref callback is called.
class ListItem extends Component {
    _onRef = ref => {
        this.props.defaultFocus && FocusManager.focus(ref);
    }
    render() {
        return <Button ref={this._onRef} />;
    }
}
```

More Consistent Layout for Views

To match Facebook React Native, and to prevent graphical inconsistencies, we now render scenes in You.i RN apps with an orthographic camera instead of a perspective projection camera. This feature requires no code changes, but if you're using the View `transform` parameters `rotateX` and `rotateY`, you'll notice that views now render as expected.

TouchableOpacity Now Changes Opacity when Focused

Facebook React Native has changed the focus and blur logic of TouchableOpacity. Opacity now changes to `150` when selected and `250` when blurred. Verify your app's behavior and make any necessary changes to account for this change.

Roku Cloud Applications

Roku Cloud Breaking Changes

The following changes in release 6.0 may affect your Roku applications.

Update CMakeLists.txt: YI_BUILD_CLOUD Removed from Build System

We've cleaned up our build system by removing an unnecessary build flag, `-d YI_BUILD_CLOUD`. If you're building for Roku Cloud, the generate flag to use is still `-d YI_BUILD_CLOUD_SERVER`, but you need to make some changes. Add

```
include(Modules/YiCloud)
```

to the `CMakeLists.txt` file under the first occurrence of

```
if (YI_BUILD_CLOUD_SERVER)
```

Optionally, you can remove the following lines from `CMakeLists.txt`. If you leave these lines in the file, they won't cause errors.

```
include(Modules/${YI_PLATFORM_LOWER}/YiInitializePlatform OPTIONAL)
if(COMMAND yi_initialize_platform)
    yi_initialize_platform()
endif()
```

Update mainDefault.cpp if Modified: Functions Moved From CYICloudInterface to CYICloud

A number of internally used functions that were exposed as public functions on `CYICloudInterface` were moved to `CYICloud`. These functions should only be used by the `mainDefault.cpp` provided by the You.i Engine SDK when compiled for Roku Cloud. Please see the table below for the new mapping.

Previous Mapping	New Mapping
`CYICloud::GetInterface().RunMainLoop()`	`CYICloud::RunMainLoop()`
`CYICloud::GetInterface().Begin()`	`CYICloud::Begin()`
`CYICloud::GetInterface().Shutdown()`	`CYICloud::Shutdown()`
`CYICloud::GetInterface().SetUseNullRenderer()`	`CYICloud::EnableRokuInterface()`

While these functions typically only affect the `mainDefault.cpp` file for Roku Cloud, and shouldn't alter your application, if your app happens to have a modified `mainDefault.cpp` file, this change may affect it.

Update Customized mainDefault.cpp: Simplified Cloud Service Startup Sequence

To enhance usability and prevent potential errors, we've consolidated the APIs used in the Cloud Service startup sequence into a single API. This change affects you only if:

- your app has a customized `mainDefault.cpp` file (`/youiengine/<version>/templates/mains/src/mainDefault.cpp`), and
- you're using separate `CYICloud` `Init()` and `Begin()` APIs in your app's initialization sequence.

In this case, you must remove the `Init()` call and move the `CYIApp` parameter into the `Begin()` call. `Begin()` stays where it was in the sequence.

Prior to 6.0

```
g_pApp = AppFactory::Create();
// Configure g_pApp
...
CYICloud::GetInterface().Init(g_pApp.get());
...
...
CYICloud::GetInterface().Begin(startImmediately)
```

Now in 6.0

```
g_pApp = AppFactory::Create();
// Configure g_pApp
...
...
...
CYICloud::GetInterface().Begin(g_pApp.get(), startImmediately)
```

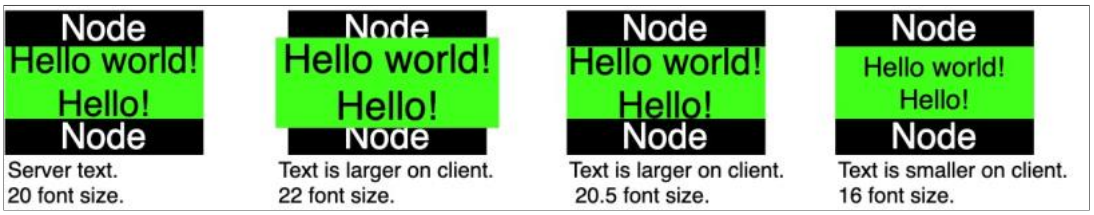
Text Rendering Differences on Roku Cloud

Prior to 6.0, You.i Engine One added padding to text bounds to account for the rendering size differences between the server and client. In 6.0, padding is not used anymore, resulting in accurate text alignment. We've also fixed the issue related to incorrect exporting of React Native vertical alignment information. If you added text alignment workarounds to your application previously, you need to remove them in 6.0.

The same text displayed in the server renders with a different size on the Roku client. This happens for the following reasons:

- The Roku device renders the same size text slightly larger than the server-side text renderer. There can be rendering differences between various supported Roku devices as not every Roku device uses the same renderer. For example, the Roku 3 device renders text differently than the Roku Ultra device.
- The server application renders scaled fonts, but the Roku client doesn't. You have to multiply the scale into the font size on the Roku client. The resulting font size is then rounded to the nearest integer because fractional font sizes are not supported by Roku devices.

In most cases, rendering size differences will be less than 0.5 font size. If scaled text is avoided, the rendering differences are smaller.



Roku Video Playback Changes

We've made the following changes to the Roku video playback feature:

Exit on Video Complete Key Changes

The `ExitOnVideoComplete` key in the video metadata has been replaced with the `exitonvideocomplete` key in the surface object. See Exit on Video Complete for more details.

Prior to 6.0

```
(metadata.insert({"ExitOnVideoComplete", "false"});)
```

Now in 6.0

```
surface: {
  exitonvideocomplete: false
}
```

Bookmark Interval Changes

The `BookmarkInterval` key in the video metadata has been replaced by `notificationinterval` key in the video node properties. See Specifying the Notification Interval for more details.

Prior to 6.0

```
metadata["BookmarkInterval"] = "3"; // seconds;
```

Now in 6.0

```
{
  notificationinterval: 3  //seconds
}
```

Metadata Prop Structure Changes

Previously, all fields in the `metadata` prop were flat structure of strings. Now, the `metadata` prop accepts any types of value, such as object and strings.

Prior to 6.0

```
metadata: {
```

```
  title: video.title,
  id: video.video_id,
  BookmarkInterval:3,
  PlayStart:this.getPlayStartTime(video.video_id)
 }
```

Now in 6.0

```
 metadata: {
  content: {
   title: video.title,
   id: video.video_id,
   playstart:this.getPlayStartTime(video.video_id)
  },
  notificationinterval:3
 }
```

The video node properties will also have the `analytics` and `ads` keys in the surface object. This was initially part of the video metadata. See You.i Engine React Native Apps for more details.

Beacon Signal API Changes

In 6.0, we have made a change to the `AppLaunchComplete` beacon to adhere with the new Roku certification requirements for `AppDialogue` beacons. The `AppDialogue` beacons are required to send a signal for any screens displayed before the user arrives at the Home screen of the app. The `AppLaunchComplete` beacon is not sent automatically anymore by the client after the first scene has loaded. Instead, a new API is added to the server that enables developers to choose what beacons to send and when to do so. These beacons are now sent via a command that runs on the server, which means they introduce a minor latency when these beacons are processed on the client.

As the Roku OS already handles erroneous use of beacons, if incorrect string is passed as the beacon name, or if a beacon is sent at the incorrect time, errors can be seen on the telnet logs when the Roku device is connected to the 8085 port. Some of the common errors are:

- Sending the `AppLaunchComplete` beacon more than once during the lifetime of the app.
- Sending the `AppDialogueInitiate` beacon without the `AppDialogueComplete` beacon before sending the `AppLaunchComplete` beacon.
- Sending the `AppDialogueInitiate` beacon after the `AppLaunchComplete` beacon.

For C++

Use `CYICloudInterface::SendAppBeacon(const CYIString &beaconName)` to send the command to the Roku client.

```
void LanderScreenViewController::OnLoadScreen(const CYIPersistentStore &)
{
 //Once the screen has finished loading, you need to send the AppLaunchComplete beacon.
 //This beacon should only be sent after the screen had loaded for the first time.
 CYICloud::GetInterface().SendAppBeacon("AppLaunchComplete");
}
```

For React Native

Use the Cloud Module's `sendAppBeacon(string)` API to send the command to the Roku client.

```
const Cloud = NativeModules.Cloud;

 componentDidMount() {
 Cloud.sendAppBeacon('AppLaunchComplete');
}
```

Roku Cloud Deprecations and Removals

Check whether you're using any of the functionality we've deprecated or removed in this version.

Cloud Module setInitialScrollIndex Method Deprecated

React Native apps on Roku Cloud no longer require the Cloud module's `setInitialScrollIndex` method for lists; this method is deprecated as of You.i Engine One version 6.0. In anticipation of the removal of `setInitialScrollIndex` in a future release, we recommend using `initialScrollIndex` as you would have on other RN apps, as it now works the same on all RN-supported platforms.

Prior to 6.0

```
const Cloud = NativeModules.Cloud;
componentDidMount() {
    Cloud.setInitialScrollIndex(findNodeHandle(this.listRef), 14);
}
<FlatList
    ref={(ref) => (this.listRef = ref)}
    initialNumToRender={this.state.listData.length}
    style={{ backgroundColor: '#fff', flexDirection: 'row' }}
    data={this.state.listData}
    keyExtractor={(item, index) => {
        return item.key;
    }}
    horizontal={true}
    renderItem={({ item }) => (
        <TheListItem index={item.key} hasFocus={false} />
    )}
    getItemLayout={(data, index) => {
        return { length: 540, offset: 540 * index, index };
    }}
/>
```

Now in 6.0

```
<FlatList
    ref={(ref) => (this.listRef = ref)}
    initialScrollIndex={14}
    style={{ backgroundColor: '#fff', flexDirection: 'row' }}
    data={this.state.listData}
    keyExtractor={(item, index) => {
        return item.key;
    }}
    horizontal={true}
    renderItem={({ item }) => (
        <TheListItem index={item.key} hasFocus={false} />
    )}
    getItemLayout={(data, index) => {
        return { length: 540, offset: 540 * index, index };
    }}
/>
```

CYICloud Init API Removed

The `CYICloud Init()` API has been removed, and its functionality has been moved into the `Begin()` API. See Simplified Cloud Service Startup Sequence to learn more.

C++ Applications

C++ Breaking Changes

The following changes in release 6.0 may affect your C++ applications.

Cloud Module setinitialScrollIndex Method Deprecated

The `CYICloud Init()` API has been removed, and its functionality has been moved into the `Begin()` API. See Simplified Cloud Service Startup Sequence to learn more.

VideoMetadata and SetVideoMetadata Removal

`CYIAbstractVideoPlayer::VideoMetadata` and `CYCloudInterface::SetVideoMetadata` are no longer available. Instead use `SetVideoNodeProperties`. Any type of value is acceptable, while still limiting the value to `CYIAny` for object and array.

Prior to 6.0

```
CYIAbstractVideoPlayer::VideoMetadata metadata;

metadata["Id"] = m_assetID;
metadata["ContentType"] = "movie";
metadata["ClosedCaptions"] = "false";
metadata["Length"] = 960;
metadata["HDBranded"] = "true";
metadata["IsHD"] = "true";
metadata["Live"] = CYIString::FromValue(MediaDataModel::GetInstance()->IsLive(m_assetID));
metadata["StreamFormat"] = "hls";
metadata["Title"] = MediaDataModel::GetInstance()->GetTitle(m_assetID);
metadata["Url"] = m_currentURL.ToString();
metadata["PlayStart"] = 300;

// ADB mobile
metadata["name"] = MediaDataModel::GetInstance()->GetTitle(m_assetID);
metadata["isAuth"] = "false";
metadata["reqAuth"] = "false";
metadata["streamtype"] = "hls";
CYICloud::GetInterface().SetVideoMetadata(m_pPlayer.get(), metadata);
```

Now in 6.0

```
std::map<CYIString, CYIAny> content;

content.emplace("id", m_assetID);
content.emplace("contenttype", CYIString("movie"));
content.emplace("closedcaptions", false);
content.emplace("length", 960);
content.emplace("hdbranded", true);
content.emplace("ishd", true);
content.emplace("live", MediaDataModel::GetInstance()->IsLive(m_assetID));
content.emplace("title", MediaDataModel::GetInstance()->GetTitle(m_assetID));
content.emplace("playstart", lastSessionInfo.bookmark.currentTimeInMilliSeconds / 1000);

CYIBundle properties;
properties.Put("content", CYIAny(content));

// Analytics: ADB mobile
std::map<CYIString, CYIAny> adbMobile;
 adbMobile.emplace("name", MediaDataModel::GetInstance()->GetTitle(m_assetID));
 adbMobile.emplace("isauth", false);
 adbMobile.emplace("reqauth", false);
 adbMobile.emplace("streamtype", GetStreamFormatString(m_currentFormat));
```

```
std::map<CYIString, CYIAny> analytics;
 analytics.emplace("adbmobile", CYIAny(adbMobile));
 properties.Put("analytics", CYIAny(analytics));

 if (!CYICloud::GetInterface().SetVideoNodeProperties(properties))
 {
  YI_LOGE(LOG_TAG, "Video node properties setting wasn't successful.");
 }
```

Changes to Camera Functions

We've changed some C++ camera functions to support rendering scenes in You.i RN apps with an orthographic camera. We've also added some new functions that you might find useful. Note that C++ apps still use a perspective projection camera by default for scene rendering, but if you're using the changed functions mentioned here, you may need to modify your code slightly.

- `DEFAULT_FRUSTUM_DISTANCE` was moved from `CYIPerspectiveSceneNode` to `CYIAbstractCameraSceneNode`, for usability in `CYIOrthographicCameraSceneNode`.

  Prior to 6.0

  ```
  pCamera->SetFarPlane(CYIPerspectiveCameraSceneNode::DEFAULT_FRUSTUM_DISTANCE);
  ```

  Now in 6.0

  ```
  pCamera->SetFarPlane(CYIAbstractCameraSceneNode::DEFAULT_FRUSTUM_DISTANCE);
  ```

- `SetNearPlane` and `GetNearPlane` were moved from `CYIPerspectiveSceneNode` to `CYIAbstractCameraSceneNode` `CYIOrthographicCameraSceneNode`.

  ```
  auto pCamera = std::make_unique<CYIOrthographicCameraSceneNode>();
  pCamera->SetNearPlane(0.0f);
  float nearPlane = pCamera->GetNearPlane();
  ```

- `SetFarPlane` and `GetFarPlane` were moved from `CYIPerspectiveSceneNode` to `CYIAbstractCameraSceneNode` `CYIOrthographicCameraSceneNode`.

  ```
  auto pCamera = std::make_unique<CYIOrthographicCameraSceneNode>();
  pCamera->SetFarPlane(0.0f);
  float farPlane = pCamera->GetFarPlane();
  ```

- `CYIOrthographicCameraSceneNode` now has a static function `BuildDefaultCamera` to match the feature set of `CYIPerspectiveCameraSceneNode`. This function builds a new camera for the user.

  ```
  auto pCamera = CYIOrthographicCameraSceneNode::BuildDefaultCamera("My New Camera");
  ```

- `CYIOrthographicCameraSceneNode` now has a static function `ConfigureOrthographic` to match the feature set of `CYIPerspectiveCameraSceneNode`. This function updates the camera projection based on the given parameters.

  ```
  auto pCamera = CYIOrthographicCameraSceneNode::BuildDefaultCamera("My New Camera");
  CYIOrthographicCameraSceneNode::ConfigureOrthographic(pCamera.get(), cameraDistance, width, height);
  ```

- `CYISceneManager LoadScene` now takes an optional function `CYICameraFactory`. This function configures the camera used when loading a scene.

  ```
  std::unique_ptr<CYISceneNode> pMainComposition = pSceneManager->LoadScene(
      "MyComp.layout",
      screenRegion,
      CYISceneManager::ScaleType::Fit,
      CYISceneManager::VerticalAlignmentType::Center,
      CYISceneManager::HorizontalAlignmentType::Center,
      CYISceneManager::MissingClassHandlingMode::Abort,

      // New parameter to configure your own camera.
      [&pCamera](CYISceneManager *pSceneManager, std::shared_ptr<CYIViewTemplate>, CYISceneView *) {
      auto pOwnedDefaultCamera = std::make_unique<CYIOrthographicCameraSceneNode>();
      pOwnedDefaultCamera->SetSceneManager(pSceneManager);
  ```

```
        pCamera = pOwnedDefaultCamera.get();
        pCamera->Init();

        return pOwnedDefaultCamera;
        }
    );
```

- `CYIAbstractCameraSceneNode` methods `SetViewport` and `GetViewport` now use a `YI_FLOAT_RECT_REL` instead of a `CYIViewport`. This change ensures that thin lines are always rendered the same, irrespective of the location on a device's screen and across various supported devices.

  Prior to 6.0

  ```
  pCamera->SetViewport(CYIViewport(0, 0, width, height));
                                    const CYIViewport &GetViewport() const;
  ```

  Now in 6.0

  ```
  pCamera->SetViewport(YI_FLOAT_RECT_REL(0.0f, 0.0f, width, height));
                                    YI_FLOAT_RECT_REL &GetViewport() const;
  ```

Behavior Changes to Overflow Modes in Row and Column Layout Types in After Effects

We've fixed an issue affecting two overflow modes, `Offscreen` and `WrapAndMoveOffscreen`, within After Effects Row and Column layout types. The issue caused these modes to push any items not fitting within the container far offscreen. Now, layout items that don't fit in the container when these two modes are used are instead hidden, via a call to `CYISceneNode::Hide()`. If you've manually hidden or shown layout items when using these two overflow modes, you'll have to update your AE layouts to change that. It isn't necessary to re-export your After Effects compositions, however.

Additionally, we've renamed the following enum values:

Old Enum Name	New Enum Name
`CYILinearLayout::Overflow::Offscreen`	`CYILinearLayout::Overflow::Hide`
`CYILinearLayout::Overflow::WrapAndMoveOffscreen`	`CYILinearLayout::Overflow::WrapAndHide`

C++ Deprecations and Removals

Check whether you're using any of the functionality we've deprecated or removed in this version.

CYIScreenReaderStatusBridge Removals

We've removed the `StopObservingStatusChanges()` and `StartObservingStatusChanges()` functions of `CYIScreenReaderStatusBridge`. You need to remove any uses of these functions. The work previously handled by these functions is now handled internally, so the removal does not introduce any behavioral changes.

Prior to 6.0

```
// Set initial values from the bridge
    CYIScreenReaderStatusBridge *pBridge = CYIAccessibilityInformationBridgeLocator::GetScreenReaderStatusBri
    if (pBridge)
    {
        if (pBridge->IsScreenReaderEnabled())
        {
            m_pScreenReaderStatusText->SetText("Screen Reader is ON.");
        }
        else
        {
            m_pScreenReaderStatusText->SetText("Screen Reader is OFF.");
        }
```

```
        pBridge->ScreenReaderStatusChanged.Connect(*this, &ScreenreaderbridgetesterApp::OnScreenReaderStatusCh
        pBridge->StartObservingStatusChanges();
```

Now in 6.0

```
// Set initial values from the bridge
    CYIScreenReaderStatusBridge *pBridge = CYIAccessibilityInformationBridgeLocator::GetScreenReaderStatusBrid
    if (pBridge)
    {
        if (pBridge->IsScreenReaderEnabled())
        {
            m_pScreenReaderStatusText->SetText("Screen Reader is ON.");
        }
        else
        {
            m_pScreenReaderStatusText->SetText("Screen Reader is OFF.");
        }
        pBridge->ScreenReaderStatusChanged.Connect(*this, &ScreenreaderbridgetesterApp::OnScreenReaderStatusCh
    }
```

CYIEvent::m_pTarget Field Removed

We've removed the previously deprecated `m_pTarget` field from `CYIEvent`. If you were sending a `CYIEvent` with its `m_pTarget` field set, modify your code as follows.

Prior to 6.0

```
pEvent->m_pTarget = pTarget;
CYISceneNode *pNode = YiDynamicCast<CYISceneNode>(pEvent->m_pTarget);
```

Now in 6.0

```
pEvent->SetTarget(pTarget);
CYISceneNode *pNode = YiDynamicCast<CYISceneNode>(pEvent->GetTarget());
```

Addendum A: Resolving a React Native Memory Leak

We've discovered a memory issue when using React Native versions lower than 0.62. As you exercise your application, FiberNodes can increase alarmingly and are not properly released. Facebook has resolved the issue in the React Native 0.62.0 Renderer.

Currently, You.i Engine React Native works with Facebook React Native 0.60 and therefore doesn't benefit from the fix by Facebook. The good news is that you can easily incorporate the Facebook fix by using a node monkey patch and a version of the Facebook fix that we've backported to work with our Engine.

Note that you may not have noticed this problem in your app. We find it happens when there are frequent transitions between complex screens without a return to a screen where React Native runs its cleanup. For example, without this fix, React Native may only clear the FiberNodes when returning to a main screen such as the lander. To learn more about the problem and Facebook's solution, see this GitHub issue for React.

Monkey Patch Your Application

Request access to the zip file here. Once you have access, unzip the file. Inside the zip file you'll find:

- `TextInput.youi.js` - an updated TextInput that is compatible with the Renderer
- `ReactNativeRenderer-prod.js` - the new Renderer you'll apply to your module

Install a monkey patch tool. We recommend using `patch-package` (available here). The rest of these instructions assume you're using `patch-package`, but you can extrapolate the instructions necessary for whatever means you choose.

Copy the file and apply the patch:

```
cp ReactNativeRenderer-prod.js <yourproject>/node_modules/@youi/react-native-youi/Libraries/Renderer/oss
cp TextInput.youi.js <yourproject>/node_modules/@youi/react-native-youi/Libraries/Components/TextInput
npx patch-package @youi/react-native-youi
git add patches/@youi+react-native-youi+6.0.0.patch
git commit -m “Fix Facebook React Native issue with FiberNodes”
```

That’s it! Rebuild your application and it’s now patched.

```
youi-tv clean --all
youi-tv build -p <platform>
```