

BlackBerry Developer Summit

Launcher Integration

Table of Contents

- Launcher Integration 3**
 - Integrate the launcher into a BlackBerry Dynamics app3**
 - Your Application’s Visibility in the Launcher5**

Launcher Integration

Integrate the launcher into a BlackBerry Dynamics app

The Launcher can be integrated into Gradle, Ant or IDE based projects. An example of such integration is shown in project AppKinetics_Launcher app available at <http://www.blackberrydevsummit.com>. You can start by running this ready-made sample or follow the steps below which cover how to integrate the Launcher into an existing BlackBerry Dynamics application. If you don't have an existing BlackBerry Dynamics application, you can use these steps to integrate the launcher into the complete AppKinetics Sample project included with the BlackBerry Dynamics SDK.

The steps here cover integration into your BlackBerry Dynamics application Gradle project:

1. Unzip the Launcher distribution zip file. The lib is downloadable from <https://developers.blackberry.com/us/en/resources/downloads.html> - look for 'BlackBerry Launcher Library' under the BlackBerry Dynamics SDK section.
2. Copy the *launcherlib.aar* file to the *libs* folder of your application. If the *libs* folder doesn't exist create it in your /app or root folder.
3. Declare the *libs* folder to be a repository of your project. Add the following snippet to the repository configuration in the build.gradle of your module within the repositories section of allprojects. The bolded content below is what was added.

```
allprojects {
    repositories {
        mavenCentral()
        jcenter()

        flatDir{
            dirs 'libs'
        }
    }
}
```

4. Declare the *launcherlib* to be a dependency of your module. Add the following snippet to the dependencies configuration in the app's build.gradle of your module (adjust the version of support libraries as needed in your project):

```
implementation 'com.android.support:support-v13:24.2.1'
implementation 'com.android.support:appcompat-v7:24.2.1'
implementation 'com.android.support:design:24.2.1'
implementation 'com.android.support:cardview-v7:24.2.1'
implementation fileTree(dir: 'libs', include: ['*.jar'])
implementation(name:'launcherlib', ext:'aar')
```

5. The Launcher must be initialised, to do so invoke the initialization method below at an early stage of the app creation

- a. Typically, you would invoke the below method in the `onCreate()` of your `Application` subclass. This is the `AppKineticsApplication` class of the `AppKinetics` sample. If you do not have a class in your project that is a subclass of `Application`, you can create a new one by right clicking your app and selecting `New` → `Java Class`. Make sure to name this class the same name as your application, and then enter 'Application' as the Superclass. Also, ensure that your app's name appears in your `AndroidManifest.xml` file, like so:

```
<application
    android:name=".AppName"
    ...
```

- b. This method doesn't need to wait for the app to be authorized:

```
LauncherButton.initForApplication(Application context,
Collection<Class> activities, ActivitiesTargetingMethod
method);
```

- *context* is the Android Application Context, you can pass this as the value for context if you are calling the initialization from your `Application` subclass.
- *activities* is a Collection of the Android Activities in which the Launcher should or should not be displayed. For example, `Arrays.<Class>asList(MainActivity.class)`
- *method* indicates whether the *activities* Collection is exclusive or inclusive. To include the list of activities that will display the launcher, you would use: `LauncherButton.ActivitiesTargetingMethod.Inclusive`

The following line can be used in the `AppKinetics` Sample:

```
LauncherButton.initForApplication(this,
Arrays.<Class>asList(AppKinetics.class),
LauncherButton.ActivitiesTargetingMethod.Inclusive);
```

6. BlackBerry Dynamics applications observe certain callbacks from `GDStateListener` in order to be notified of authorization state. These callbacks will be implemented in your class that implements the `GDStateListener` interface. In the `AppKinetics` sample, this is done in the `AppKineticsModel` class.

In these callbacks the Launcher `HostingApp` class must have these messages passed on:

```
@Override
public void onAuthorized() {
    //Launcher needs to know whether the app is authorized
    //or not. If authorized flag is not set, launcher
    //button is not shown.
    HostingApp.getInstance().setAuthorized(true);
}
```

```
@Override
public void onLocked() {
    HostingApp.getInstance().setAuthorized(false);
}
```


7. The Launcher uses the GDServiceClient to launch applications however there can only be one GDServiceClientListener registered. If you don't have a GDServiceClientListener implemented, you can skip this step. This step is required for the AppKinetics sample.

- a) if your application already has registered a listener it must pass it onto the Launcher HostingApp class. For the AppKinetics sample, this is done in the AppKineticsApplication in the onCreate method in the if block checking if serviceListener is null.

```
HostingApp.getInstance().setClientServiceListener(GDService
ClientListener serviceClientListener);
```

The following line can be used in the AppKinetics sample.

```
HostingApp.getInstance().setClientServiceListener(serviceLi
stener);
```

8. Handle the Settings  button by implementing the interface *HostingApp.LauncherCommandCallback* and register the implementing class with the launcher lib using the method `HostingApp.getInstance().setOnCommandCallback(LauncherCommandCallback callback);` This button is intended to invoke your application settings screen so you need to add the necessary code to display the settings screen. Typically, this method call will be made from the same class that has initialized the Launcher button.
9. Configure proguard to silence warnings about the launcher lib by adding the following line to your proguard file: `-dontwarn com.good.launcher.**`

Your Application's Visibility in the Launcher

How can your application be made to appear in the Launcher of other applications? Your application will appear in the Launcher of other applications, if it provides at least one AppKinetics service, for example:

- Transfer File (com.good.gdservice.transfer-file)
- Presence Service (com.good.gdservice.enterprise.presence)

What if your application doesn't provide any AppKinetics services?

There is a dummy service that you can register as providing:

- Launch (com.good.gdservice.launch)

There is no need to write any code to handle service requests for the Launch service.

Details of all shared services are accessible through the *community.blackberry.com* site, or via this shortcut: <https://apps.good.com/#/services>. To add an app kinetics service to your app, view your app in UEM and add the service with the '+' button under 'Version', as seen in the screenshot below:

APP Dev Summit App 312

Settings Assigned to 358 users Assigned to 2 groups

BlackBerry Dynamics app

BlackBerry Dynamics app ID
com.bbdevsummit.devapp312

BlackBerry Dynamics IOS macOS Android Windows

Override BlackBerry Dynamics profile
- Select -

Override compliance profile
- Select -

App configuration Upload a template

Name	Ranking	
None assigned		

Server configuration payload
Add

Version

Version	Release status	Service bindings	Note	
1.0.0.0	PROD	Launch Service 1.0.0.0, Launchable Service 1.0.0.0		X

User certificates
 Allow BlackBerry Dynamics apps to use user certificates and user credential profiles

© 2015 BlackBerry. All rights reserved. BlackBerry® and related trademarks, names and logos are the property of BlackBerry Limited and are registered and/or used in the U.S. and countries around the world. All other trademarks are the property of their respective owners. This documentation is provided "as is" and without condition, endorsement, guarantee, representation or warranty, or liability of any kind by BlackBerry Limited and its affiliated companies, all of which are expressly disclaimed to the maximum extent permitted by applicable law in your jurisdiction.