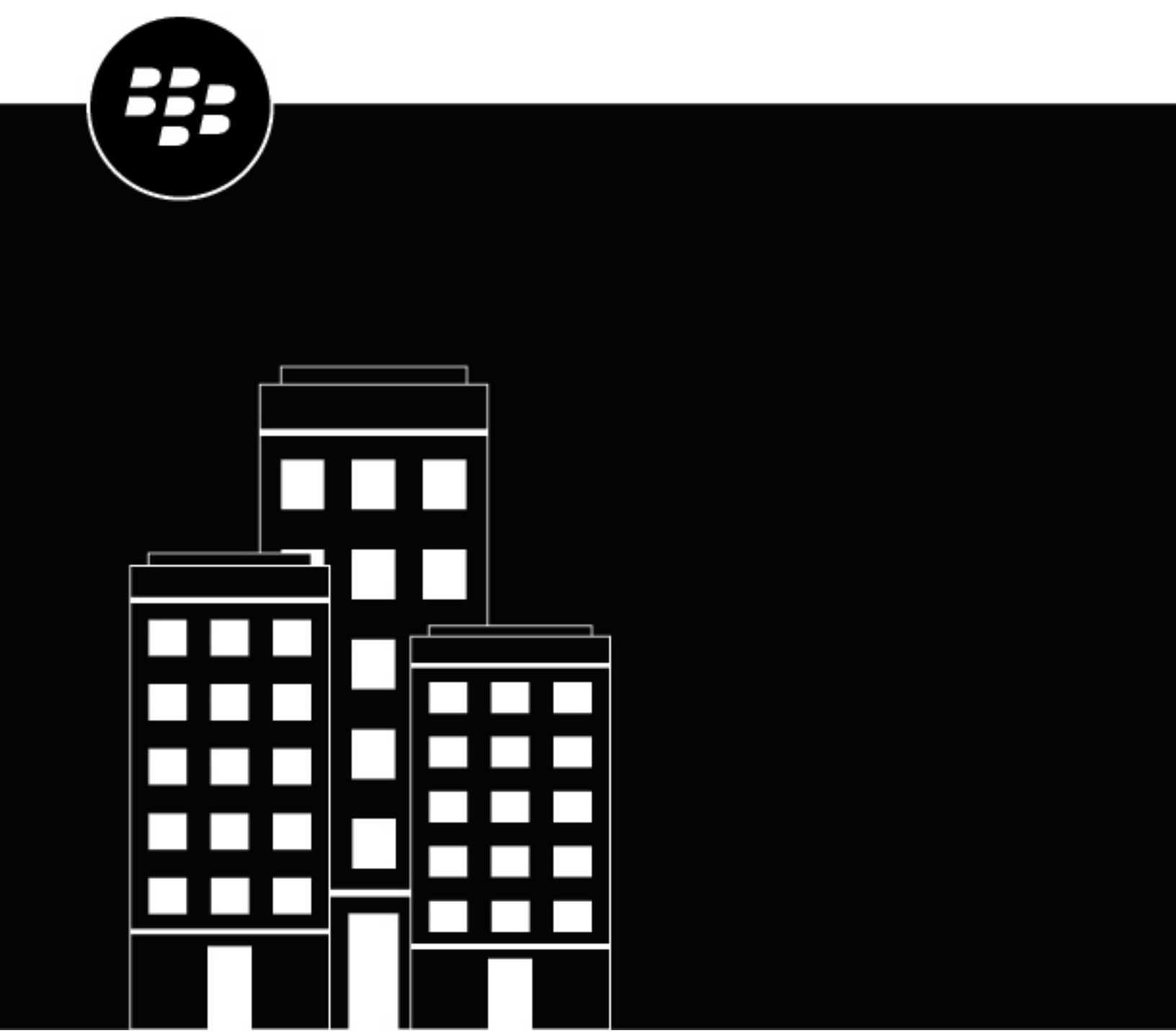

BlackBerry Dynamics Launcher Framework

Development Guide

12.1

Contents

About the BlackBerry Dynamics Launcher Library

The BlackBerry Dynamics Launcher is the blue BlackBerry icon located in your BlackBerry Dynamics apps. It allows you to perform the following actions:

- Quickly switch between the BlackBerry Dynamics app that is currently open and any other BlackBerry Dynamics apps on your device
- Move between BlackBerry Work Mail, Calendar, Contacts, and Docs
- Access your work apps catalog
- Access Quick Create tools for email, contacts, and calendar events
- Launch browser-based web clips and non-BlackBerry Dynamics apps installed on your device. This feature requires admin permission and the UEM Client.
- Search for BlackBerry Dynamics Launcher items (iOS only)
- Access settings

You can use the BlackBerry Dynamics Launcher Library to enable your BlackBerry Dynamics apps so that they are displayed in the Launcher. The library consists of header files and a compiled library for Objective-C for iOS and Java for Android.

This guide describes how to integrate the Launcher Library in your apps. It does not describe how to enable them so that they appear in the Launcher. Features are enabled in BlackBerry UEM or Good Control. For more information on how to add an app to the BlackBerry Dynamics Launcher in BlackBerry UEM, see Add an app shortcut for iOS, macOS, and Android devices in the BlackBerry UEM Administration content.

For iOS, if you plan to switch to the BlackBerry Dynamics Launcher dylib framework starting in version 3.2, integrate the “BlackBerryLauncher” dylib.

Server requirements

To take advantage of advanced features that do not require programming, the BlackBerry Dynamics Launcher and the Launcher Library require the following servers:

Item	Requirement
BlackBerry UEM	To allow the app to be displayed in the BlackBerry Dynamics Launcher, your environment must have a version of BlackBerry UEM that is officially supported by BlackBerry. For more information, see the UEM Software Lifecycle.
BlackBerry Enterprise Mobility Server	To create custom icons, see user profile information and presence status, and display BlackBerry Dynamics Launcher apps' policy-enabled sections, your environment must have BlackBerry Enterprise Mobility Server 2.4 or later. **Note:** The Launcher Library does not strictly require the BlackBerry Enterprise Mobility Server. Without it, your app still has the default BlackBerry icon and the IT administrator can add other app icons to the Launcher, but you will not have the features of policy-based sections in the Launcher (like docs or RSA SecurID), BEMS-Presence, or users' profile photos.

Compatibility updates for version 12.x

- Support for BlackBerry Dynamics SDK version 12.x
- Android SDK API level that is supported by the BlackBerry Dynamics SDK. See the BlackBerry Dynamics SDK Development Guide for the required SDK API level.
- BlackBerry Dynamics Launcher now supports building and debugging BlackBerry Dynamics apps on an Apple M1 Mac Xcode iOS simulator. This can only be done for BlackBerry Dynamics apps using the BlackBerry Dynamics dynamic library

BlackBerry Dynamics SDK and Launcher Library version requirements

The following information only applies if you use the BlackBerry Dynamics Launcher as a front-end to your apps.

- The Launcher Library works on top of the BlackBerry Dynamics SDK.
- The BlackBerry Dynamics SDK and the Launcher Library are mutually dependent. Make sure that you have installed the latest BlackBerry Dynamics SDK for your platform that corresponds to this release of the Launcher Library and make sure that you have installed the version of the Launcher Library that corresponds to your installed version of the BlackBerry Dynamics SDK.

Important: If you do not ensure that you are using the corresponding versions of the BlackBerry Dynamics SDK and the Launcher Library, your apps will crash.

Integrating the Launcher Library for display on Android devices

The BlackBerry Dynamics Launcher Library can be integrated like any Android library project. After the library is integrated into the build system (Gradle/Ant) or into your favorite IDE, there are few basic steps you need to follow to integrate Launcher functionality into your apps.

Note: The BlackBerry Dynamics Launcher Library supports AndroidX.

Set up the sample app in Gradle and Android Studio for sample app

The sample app for the BlackBerry Dynamics Launcher Library is based on the RSS Reader sample app delivered with the BlackBerry Dynamics SDK. Apps that rely on the BlackBerry Dynamics Launcher Library must use Android support library com.android.support:support-v13. This is already configured in the Android sample app's build.gradle file, but you must add these dependencies in any new project.

1. Download and unzip the BlackBerry Dynamics SDK release .zip distribution package.
2. Download and unzip the BlackBerry Dynamics Launcher Library release .zip distribution package.
3. Go to GoodLauncherLibrary/SampleLauncherApp/app/build.gradle file and modify maven url to point to your locally downloaded BlackBerry Dynamics SDK folder, for example: `maven { url '<dynamics-sdk-root-folder>/m2repository' }`

Configure an existing Android project

These steps assume you have a project already created and a `libs` directory in that project. The `libs` directory must not be at the top level of your project. It must be at the same level as your `build.gradle` file.

1. Download and unzip the BlackBerry Dynamics SDK release .zip distribution package.
2. Download and unzip the BlackBerry Dynamics Launcher Library release .zip distribution package.
3. Copy GoodLauncherLibrary/launcherlib.aar to the project libs directory.
4. Make the following declarations in your `build.gradle` file, as shown in these snippets:
   a) Declare a dependency on the `launcherlib.aar` library and gd library:

```
dependencies {
  implementation
'com.blackberry.blackberrydynamics:android_handheld_platform:<version>'
  implementation(name: 'launcherlib', ext: 'aar')
}
```

   b) Add the following repositories for Blackberry Dynamics Launcher and BlackBerry Dynamics SDK components:

```
repositories{
  //Copy launcherlib.aar inside app/libs folder
   flatDir {
   dirs 'libs'
   }
  //Maven url points to the local Dynamics SDK download folder path ex:
 gdsdk-release-<version>/m2repository/maven { url '<dynamics-sdk-root-
folder>/m2repository'
```

```
    }
  }
```

c) Add the Android support library dependencies:

```
    implementation 'androidx.legacy:legacy-support-v13:1.0.0'
    implementation 'androidx.appcompat:appcompat:1.0.0'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
  }
```

5. To build your project with Gradle, use either `gradle--info --stacktrace clean assembleDebug` or `gradle --info --stacktrace clean assembleRelease`.

Using a global instance of GDStateListener instead of a per-activity listener

The BlackBerry Dynamics Launcher Library and BlackBerry Dynamics SDK for Android do not have a mechanism to determine whether the global state listener is set in order to decide whether it should use its own `GDStateListener` in its activity. You should set a global instance of `GDStateListener`.

Initialization, registration, and program setup

Apps that rely on the BlackBerry Dynamics Launcher Library must use Android support library `com.android.support:support-v13`. Before any interaction with the Launcher, it must be initialized.

1. Initialize the Launcher Library:

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

The LauncherButton.initForApplication should be called inside the onCreate of 'android.app.Application' derived class using application context.

Item	Description
`context`	This is a context of the app where the Launcher Library is integrated.
`activities`	This is a list of the app activities where the the Launcher button should or should not be shown. The third parameter, `ActivitiesTargetMethod`, defines whether this is an excluded activities list or included activities list.
`method`	This can be one of two possible values: • `LauncherButton.ActivitiesTargetingMethod.Exclusive` • `LauncherButton.ActivitiesTargetingMethod.Inclusive`

2. Notify the Launcher of the BlackBerry Dynamics authorization state change. The app that hosts the Launcher Library should let the Launcher Library know the authorization state with the following method:

- • Set to `true` when the BlackBerry Dynamics state EventListener notifies `onAuthorized` state.
  - Set to `false` when EventListener notifies `onLocked` state.

3. If necessary, register `GDServiceClientListener`.

   The Launcher Library uses `GDServiceClient` to start apps listed on the Launcher pad.

   The `GDServiceClient` requires an instance of `GDServiceClientListener` but only one listener can be set. In case your app already defines `GDServiceClientListener`, it should be passed to the Launcher Library. Otherwise, the default `GDServiceClientListener` is set to `GDServiceClient` and the app's `GDServiceClientListener` is reset.

   ```
   HostingApp.getInstance().setClientServiceListener(GDServiceClientListener
    serviceClientListener);
   ```

4. Handle the settings icon in the Launcher.

   This button is intended to invoke your app settings screen. Your app must implement and then register the `LauncherCommandCallback` interface.

   ```
     HostingApp.getInstance().setOnCommandCallback(LauncherCommandCallback
   commandCallback)
     public interface HostingApp.LauncherCommandCallback {
     public void onSettingsCommand();
   }
   ```

ProGuard configuration

To make sure that the build does not fail with too many warnings, apps that integrate the BlackBerry Dynamics Launcher Library must have the following directive in the ProGuard file:

```
-dontwarn com.good.launcher.**
```

Integrating the BlackBerry Dynamics Launcher Library on iOS

Add either the BlackBerry Dynamics Launcher Library or the BlackBerry Dynamics Launcher dylib to an Xcode project. Choose dylib if you are already using the BlackBerry Dynamics dylib framework in your app.

Add the Launcher Library to an Xcode project

Adding BlackBerry Dynamics Launcher Library to an Xcode project is no different than adding other libraries, with a few minor additions.

1. Move the unzipped Launcher Library files to any desired directory and add each item to the Xcode project file.
2. Under the **Build Phases** of the app target, add `Launcher.framework` to the **Link Binary With Libraries** phase.
3. Verify that the `LauncherBundle.bundle` is added as part of the **Copy Bundle Resources** phase.
4. Ensure that the **Main Interface** setting for both iPhone and iPad under the target's **General settings** is cleared. The Launcher provides its own root interface.
5. To ensure icons are downloaded, in the app's info.plist file add the following key/valuepair:

   ```
   (string: boolean): GDFetchResources : YES
   ```

6. Under the **Other Linker Flags** build setting, add the following: `-ObjC`.

Add the Launcher dylib to an Xcode project

Adding BlackBerry Dynamics Launcher dylib to an Xcode project is no different than adding other libraries, with a few minor additions.

1. Move the unzipped Launcher Library files to any desired directory and add each item to the Xcode project file.
2. Under the **Build Phases** of the app target, add `BlackBerryLauncher.xcframework` to the **Embed Framework**.
3. Ensure that the **Main Interface** setting for both iPhone and iPad under the target's **General settings** is cleared. The Launcher provides its own root interface.
4. To ensure icons are downloaded, in the app's info.plist file add the following key/valuepair:

   ```
   (string: boolean): GDFetchResources : YES
   ```

5. Under the **Other Linker Flags** build setting, add the following: `-ObjC`.

Initialize the Launcher

Add the following link to the end of your initialization code, as shown in Example of onAuthorized with startServicesWithOptions:

```
[launchPadVC
 startServicesWithOptions:GTLInternalGDAuthTokenAndPushConnectionManagement];
```

Starting the Launcher

After the GDiOS delegate (usually also the UIApplicationDelegate) receives the onAuthorized: callback, the BlackBerry Dynamics Launcher Library needs to start up various services that call into the BlackBerry Dynamics API. Starting the Launcher requires that startup options be provided. Internally, the library uses the GDUtility and GDPushConnection classes. These classes are singleton classes with a single delegate. If the host app is using these classes, the Launcher Library needs to either retrieve the required information through a delegate callback or through method calls into the Launcher.

Startup options	Description
GTLInternalGDAuthTokenAnd PushConnectionManagement	This is the default option. Use this option if the host app does not use BlackBerry Dynamics auth tokens (GDUtilityclass) or BlackBerry Dynamics push connection. The specified classes are used internally in the Launcher.
GTLHostGDAuthTokenManagement	Use this option if the host app uses the GDUtility class internally. The Launcher makes a delegate callback into the app whenever it requires an auth token (see below).
GTLHostGDAuthTokenManagement	Use this option if the host app uses GDPushConnection and specifies a delegate. Whenever the host app receives the onStatus: call, the value provided should be passed to the Launcher.

Example of onAuthorized with startServicesWithOptions

The following fragment is extracted from the sample apps AppNameGDiOSDelegate.m.

Note: Make sure you add the line highlighted below at the end of your initialization code. Otherwise, your icons might not be displayed properly.

AppNameGDiOSDelegate.m with launchPadVC

```
-(void) onAuthorized:(GDAppEvent*)anEvent
{
    /* Handle the Good Libraries authorized event. */

    switch (anEvent.code) {
        case GDErrorNone: {
            if (!self.hasAuthorized) {
                // launch application UI here
                .
                .
                .
                //Detect the device
                if ([[UIDevice currentDevice] userInterfaceIdiom] ==
 UIUserInterfaceIdiomPhone)
                {
                    /*
                     *      iPhone start - based on single UINavigationController
```

```
                     */
                    // Setup the rootviewcontroller and a navigation controller
 (for the splitviewcontroller)
                    self.navController = [[Utilities storyboard]
 instantiateViewControllerWithIdentifier:@"iPhoneNC"];

                    GTLauncherViewController *launchPadVC =
 [[GTLauncherViewController alloc] initWithBaseViewController:self.navController];
                    appWindow.rootViewController = launchPadVC;

                    [launchPadVC
 startServicesWithOptions:GTLInternalGDAuthTokenAndPushConnectionManagement];

                }
                else
                {

                    /*
                     *      iPad start - based on UISplitViewController
                     */
                    self.splitViewController = [[Utilities storyboard]
 instantiateViewControllerWithIdentifier:@"SplitVC"];

                    self.navController = [self.splitViewController.viewControllers
 firstObject];
                    self.detailNavigationController =
 [self.splitViewController.viewControllers lastObject];
                    GTLauncherViewController *launchPadVC =
 [[GTLauncherViewController alloc]
 initWithBaseViewController:self.splitViewController];
                    self.rssReaderAppDelegate.window.rootViewController =
 launchPadVC;

                    [launchPadVC
 startServicesWithOptions:GTLInternalGDAuthTokenAndPushConnectionManagement];
  .
  .
  .
```

`GTLHostGDAuthTokenManagement` startup option

```
//This method must be implemented if GTLHostGDAuthTokenManagement is a specified
 startup option
- (void)launcherViewController:(GTLauncherViewController *)controller
 didRequestGDAuthTokenForServerName: (NSString *)name completion:
 (void (^)(NSString *token))completion { //Implementation specific to the host
 app
   [[HostAuthTokenManager tokenManager] getAuthTokenForServer:name
 completion:^(NSString *token) {
  if (completion) {
  completion(token);
  };
  }];
 }
```

`GTLHostGDPushConnectionManagement` startup option

```
//GDPushConnectionDelegate callback
 - (void)onStatus:(int)status
 {
   .
   .
   .
   //The following is required if GTLHostGDPushConnectionManagement is specified
 as a startup option
   GTLauncherViewController *launcherVC = ...;
   [launcherVC setGDPushConnectionStatus:status];
 }
```

Troubleshooting

If BlackBerry Launcher Framework for iOS does not display an icon for a custom ISV app, check the BlackBerry PCE portal to ensure that the icon for the BlackBerry Dynamics app entitlement appears in the PCE tool. If it does not, upload or refresh the app icon in the PCE tool. For more information, see Add an internal app to the app list.

Hiding or showing the BlackBerry Dynamics Launcher

The following code samples show how to hide or show the BlackBerry Dynamics Launcher button:

For iOS, use the `GTLauncherViewController.launcherButtonHidden` boolean property.

To hide the Launcher button:

```
GTLauncherViewController *launcherVC = (GTLauncherViewController *)viewController;
 launcherVC.launcherButtonHidden = YES;
```

To show the Launcher button:

```
GTLauncherViewController *launcherVC = (GTLauncherViewController *)viewController;
 launcherVC.launcherButtonHidden = NO;
```

For Android, use the `HostingApp.getInstance().setVisible(boolean isVisible, boolean updateView)` property.

To hide the Launcher button:

```
 HostingApp.getInstance().setVisible(false, false)
```

To show the Launcher button:

```
 HostingApp.getInstance().setVisible(true, false)Paragraph
```

Sample app for Android and iOS

The distribution comes with a sample RSS Reader app that you can examine to see how the BlackBerry Dynamics Launcher Library is implemented.

Moving to production

Service binding

When your application is ready, it needs to be bound to a service in Good Control or BlackBerry UEM. You can bind it to the BlackBerry Dynamics Launcher Library service itself.

For details about how to bind an app to a service, see Manage BlackBerry Dynamics app services.

Customizing the Launcher icon in BEMS

You can set a custom icon for your app in BlackBerry UEM. See Setting a customized icon for the BlackBerry Dynamics Launcher.

Validate the BEMS SSL certificate

You can have the BlackBerry Dynamics Launcher Library validate the SSL certificate of any BlackBerry Enterprise Mobility Server that is associated with your deployment to secure the communications between the servers.

As of the 3.6.0 release of the BlackBerry Dynamics Launcher Library, this entitlement is deprecated and will not have any impact on the Launcher apps section or the UEM web apps shown in the Launcher.

Before you begin:

- Verify that you have a copy of the BEMS SSL certificate or the BEMS SSL certificate chain. BEMS supports replacing the BEMS self-signed certificate with one issued by an internal or external certificate authority. If the BEMS certificate has been replaced, then you should deploy the Root CA and any intermediates or subordinate signing certificates in the chain, not the actual BEMS certificate. For more information, see Replacing the autogenerated SSL certificate in the BEMS Configuration content.
- Verify that you have administrative access to your Good Control and BlackBerry UEM server.
- Verify that you have the entitlement ID: `com.blackberry.feature.validatebemscertificate`

1. On the BEMS server, export the SSL certificate.
2. If you are using a Good Control server, go to **Certificates > Trusted Authorities** tab, **Upload New Certificate** to import the BEMS certificate.
3. If you are using a BlackBerry UEM server, go to **Policies and Profiles** > **Certificates** > **CA certificate** to add the certificate and assign it to users. For more information, see Assign the BEMS SSL certificate to users.
4. Assign the entitlement ID to users. Do one of the following:
    - Assign it to the Everyone user group.
    - Create a user group of only those users you want to assign it to.
    - Assign it to individual users.

Legal notice

WERE FORESEEN OR UNFORESEEN, AND EVEN IF BLACKBERRY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, BLACKBERRY SHALL HAVE NO OTHER OBLIGATION, DUTY, OR LIABILITY WHATSOEVER IN CONTRACT, TORT, OR OTHERWISE TO YOU INCLUDING ANY LIABILITY FOR NEGLIGENCE OR STRICT LIABILITY.

THE LIMITATIONS, EXCLUSIONS, AND DISCLAIMERS HEREIN SHALL APPLY: (A) IRRESPECTIVE OF THE NATURE OF THE CAUSE OF ACTION, DEMAND, OR ACTION BY YOU INCLUDING BUT NOT LIMITED TO BREACH OF CONTRACT, NEGLIGENCE, TORT, STRICT LIABILITY OR ANY OTHER LEGAL THEORY AND SHALL SURVIVE A FUNDAMENTAL BREACH OR BREACHES OR THE FAILURE OF THE ESSENTIAL PURPOSE OF THIS AGREEMENT OR OF ANY REMEDY CONTAINED HEREIN; AND (B) TO BLACKBERRY AND ITS AFFILIATED COMPANIES, THEIR SUCCESSORS, ASSIGNS, AGENTS, SUPPLIERS (INCLUDING AIRTIME SERVICE PROVIDERS), AUTHORIZED BLACKBERRY DISTRIBUTORS (ALSO INCLUDING AIRTIME SERVICE PROVIDERS) AND THEIR RESPECTIVE DIRECTORS, EMPLOYEES, AND INDEPENDENT CONTRACTORS.

IN ADDITION TO THE LIMITATIONS AND EXCLUSIONS SET OUT ABOVE, IN NO EVENT SHALL ANY DIRECTOR, EMPLOYEE, AGENT, DISTRIBUTOR, SUPPLIER, INDEPENDENT CONTRACTOR OF BLACKBERRY OR ANY AFFILIATES OF BLACKBERRY HAVE ANY LIABILITY ARISING FROM OR RELATED TO THE DOCUMENTATION.

Prior to subscribing for, installing, or using any Third Party Products and Services, it is your responsibility to ensure that your airtime service provider has agreed to support all of their features. Some airtime service providers might not offer Internet browsing functionality with a subscription to the BlackBerry® Internet Service. Check with your service provider for availability, roaming arrangements, service plans and features. Installation or use of Third Party Products and Services with BlackBerry's products and services may require one or more patent, trademark, copyright, or other licenses in order to avoid infringement or violation of third party rights. You are solely responsible for determining whether to use Third Party Products and Services and if any third party licenses are required to do so. If required you are responsible for acquiring them. You should not install or use Third Party Products and Services until all necessary licenses have been acquired. Any Third Party Products and Services that are provided with BlackBerry's products and services are provided as a convenience to you and are provided "AS IS" with no express or implied conditions, endorsements, guarantees, representations, or warranties of any kind by BlackBerry and BlackBerry assumes no liability whatsoever, in relation thereto. Your use of Third Party Products and Services shall be governed by and subject to you agreeing to the terms of separate licenses and other agreements applicable thereto with third parties, except to the extent expressly covered by a license or other agreement with BlackBerry.

The terms of use of any BlackBerry product or service are set out in a separate license or other agreement with BlackBerry applicable thereto. NOTHING IN THIS DOCUMENTATION IS INTENDED TO SUPERSEDE ANY EXPRESS WRITTEN AGREEMENTS OR WARRANTIES PROVIDED BY BLACKBERRY FOR PORTIONS OF ANY BLACKBERRY PRODUCT OR SERVICE OTHER THAN THIS DOCUMENTATION.

BlackBerry Enterprise Software incorporates certain third-party software. The license and copyright information associated with this software is available at http://worldwide.blackberry.com/legal/thirdpartysoftware.jsp.

BlackBerry Limited
2200 University Avenue East
Waterloo, Ontario
Canada N2K 0A7

BlackBerry UK Limited
Ground Floor, The Pearce Building, West Street,
Maidenhead, Berkshire SL6 1RL
United Kingdom

Published in Canada