About this guide

This guide is an introduction to the BlackBerry Dynamics bindings for Xamarin. The guide focuses on how to install the SDK, how to use the provided plugins, and introduces the sample apps that are packaged with the SDK.

This guide is intended for software developers who already have an understanding of developing software with Xamarin. It is not a tutorial on programming with Xamarin. It assumes that you have working knowledge of the language and concepts. The guide also assumes that you have installed the supported version of Xamarin. For information about how to install, see the instructions at https://www.xamarin.com/.

Relation to BlackBerry Dynamics SDK

Apps developed with the Xamarin bindings rely on the underlying BlackBerry Dynamics SDK. The full set of the SDK’s APIs has been bound to a C#/.NET counterpart. For example:

- **BlackBerry Dynamics SDK for Android Java**: `GDAndroid.getInstance().activityInit(this);`
- **Xamarin C#/.NET**: `GDAndroid.Instance.ActivityInit(this);`
BlackBerry Dynamics background

The following sections provide some background information that can help you understand the features of the BlackBerry Dynamics SDK.

The way that these features are implemented in your environment will depend on how your administrator has configured your organization's servers, your network, and other infrastructure.

BlackBerry Dynamics API reference

The BlackBerry Dynamics SDK API reference describes the available interfaces, classes, methods, and much more.

You can access the Android API reference:

• In the installed directories for the BlackBerry Dynamics SDK for Android.

FIPS 140-2 compliance

BlackBerry Dynamics apps must comply with U.S. Federal Information Processing Standards (FIPS) 140-2. The BlackBerry Dynamics SDK distribution contains FIPS canisters and tools and, by default, enforces FIPS compliance.

On Android, FIPS compliance is automatic. No special build steps are required.

There are two components involved in enabling FIPS:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlackBerry Dynamics app</td>
<td>The app must start in FIPS-compliant mode. The BlackBerry Dynamics SDK determines whether a service is running in FIPS mode when the app communicates with the server to receive policies. All apps must be written for FIPS compliance.</td>
</tr>
<tr>
<td>Policy server (either standalone Good Control or BlackBerry UEM)</td>
<td>For more details on FIPS policies, see Reading your app for deployment: server setup.</td>
</tr>
</tbody>
</table>

FIPS compliance enforces the following constraints:

• MD4 and MD5 are prohibited. As a result, access to NTLM-protected or NTLM2-protected web pages and files is blocked.
• Wrapped apps are blocked.
• In secure socket key exchanges with ephemeral keys, with servers that are not configured to use Diffie-Hellman keys of sufficient length, BlackBerry Dynamics retries with static RSA cipher suites.

Note: When you enable FIPS compliance, user certificates must use encryption that meets FIPS standards. If a user tries to import a certificate with encryption that is not compliant, the user receives an error message indicating that the certificate is not allowed and cannot be imported.
Easy Activation

The Easy Activation feature simplifies the provisioning process by allowing a BlackBerry Dynamics app to hand off activation to an app that is already installed on the device and can act as the activation delegate. The user has to retrieve and manually enter an access key only the first time they install a BlackBerry Dynamics app.

Securing cut-copy-paste on devices (Data Leakage Prevention, or DLP)

You can use the BlackBerry Dynamics SDK to protect certain data copied and pasted between apps on your users’ devices.

Server administrators must enable the Data Leakage Prevention policies in the management console.

To enable sharing among a group of apps, the apps must be provisioned from the same BlackBerry Control service for each user.

If the Data Leakage Prevention settings are enabled in your environment, you can work around them when you need to debug your app. For more information, see the BlackBerry Dynamics SDK API Reference.

Shared Services Framework

BlackBerry Dynamics-enabled apps can communicate with each other using the Shared Services Framework.

There are two kinds of shared services:

- Server-side services
- Client-side services

The BlackBerry Dynamics SDK contains sample apps that show how these services work.

For a conceptual background, see BlackBerry Dynamics Services Framework.

Support for fingerprint authentication

Support for fingerprint recognition is a supplement to standard BlackBerry Dynamics secure user authentication, not a replacement for it. BlackBerry Dynamics includes the following policies related to fingerprint authentication. These settings are configured using policies in the management console:

- Allow or disallow fingerprint authentication for BlackBerry Dynamics-based apps in general.
- If fingerprint authentication is allowed, you can also allow or disallow it for BlackBerry Dynamics apps immediately after app coldstart. If you do not allow it after app coldstart, the user must enter the password for the app.
- Require the end user to enter a password after a specified interval.

Note: For app developers, no additional programming work is necessary for fingerprint authentication. If the app supports Samsung Pass, some linking directives are required. See Info: Add Samsung Pass to your BlackBerry Dynamics app.

For more information, see BlackBerry Dynamics and Fingerprint Authentication.
Support for client certificates

BlackBerry Dynamics supports many popular uses of client-side Public Key Infrastructure (PKI) certificates to secure apps and communications:

- General requirements for working with PKI certs
- Description of client certificate sharing among BlackBerry Dynamics apps on a device
- Kerberos PKINIT: client certificates in the Kerberos authentication model. (This is not Kerberos Constrained Delegation, or KCD).

Support for the "Do not require password" setting

The BlackBerry Dynamics Runtime supports the "Do not require password" setting in a BlackBerry Dynamics profile in UEM or in a security policy in standalone Good Control. When this setting is enabled by an administrator, users cannot set a password for a BlackBerry Dynamics app or BlackBerry Dynamics container. Note that this setting does not apply to the device password.

This setting is available in BlackBerry UEM 12.7 or later and standalone Good Control 3.0.50.70 or later.

Security considerations

- Consider the security impact to your organization's environment before an administrator enables this setting. If enabling this feature does not meet security standards, consider other options, including authentication delegation or assigning the profile to specific users or groups that are already assigned device management profiles or other controls.
- Do not enable the "Do not require password" setting and authentication delegation in the same policy set.
- When the "Do not require password" setting is enabled, authentication can be accomplished only through user interaction or autonomously. For more information, see “canAuthorizeAutonomously” in the SDK programming reference for iOS or Android.

User experience when the rule is enabled or disabled

If a BlackBerry Dynamics app requires a password and the administrator enables the "Do not require password" setting, the next time the user opens the app, the app displays a message that a password is no longer required. As long as the feature is enabled, the user is not prompted for a password.

If the administrator disables the "Do not require password" setting, the next time the user opens the app, the app displays a message that a password is required. The user is prompted to specify a password.

Android: Optional APIs for the "Do not require password" policy rule

You can call the GDAndroid.getInstance().canAuthorizeAutonomously() method to determine whether this feature is enabled for a BlackBerry Dynamics app.

Call GDAndroid.serviceInit() to start the authorization process when the app has received a GCM message or background broadcast/explicit intent.

The GDInteraction sample app illustrates the use of these methods.

For syntax and details, see the SDK programming reference.
Bypassing the App Lock screen

BlackBerry Dynamics supports the ability of an app to bypass the BlackBerry Dynamics user authentication/lock screen. Some organizations want this feature, particularly in VoIP apps where the user needs to respond quickly to an incoming call.

Note: Enabling this policy weakens the security inherent to BlackBerry Dynamics.

For information about requesting this feature, the necessary programming for bypassing the lock screen, the setup of a required app policy, and other details, see Bypass Unlock: BlackBerry Dynamics app Developer Guide.

Supported languages

The BlackBerry Dynamics SDK supports the following languages. No SDK calls are required to use a particular language; the interface selects the appropriate language based on the language setting the user has configured on their device.

- English (US)
- Chinese (Simplified)
- Danish
- Dutch
- French
- German
- Italian
- Japanese
- Korean
- Portuguese (Brazil)
- Portuguese (Portugal)
- Spanish
- Swedish
Requirements

BlackBerry Dynamics software versions

- BlackBerry Dynamics SDK for Android 4.2.x
- BlackBerry Dynamics Launcher Library for Android 2.8.x
- Bindings for Xamarin.Android 4.2.x

Compatibility with earlier releases

The latest release of the BlackBerry Dynamics SDK is compatible with the previous two releases.

Note: You should always build with and test against the most recent release. The most recent release has bug fixes and new features that you should test and deploy regularly.

Licensing for Xamarin

To create BlackBerry apps with Xamarin, you must purchase a valid software license from Xamarin. BlackBerry does not provide this license. You can use any of these license types:

- Community
- Professional
- Enterprise

Software requirements

Development environment:

- Latest version of the BlackBerry Dynamics SDK for Android
- Latest version of the Android SDK
- Xamarin.Android 9.0.0.20 or later
- Microsoft Visual Studio for macOS 7.6 or later
- Minimum Android API version: 21
- Target Android API version: 28
- Target Framework API version: 28

BlackBerry Dynamics entitlement ID and version

BlackBerry Dynamics apps are uniquely identified by a BlackBerry Dynamics entitlement ID (GDApplciationID) and entitlement version (GDApplciationVersion). The entitlement ID and entitlement version are used to manage end-user entitlement for your apps, as well as for publishing and service provider registration. The BlackBerry Dynamics entitlement ID was formerly known as the app ID or GD App ID.

The entitlement ID is used in the app, in the BlackBerry UEM or standalone Good Control management console for app management, and in some administrative user interfaces on the application developer portal.
**Note:** The entitlement ID and entitlement version are different from the native application ID and native application version. The native application ID is a unique identifier for the app that is used by the OS and associated platforms (for example, the package name for Android or bundle identifier for iOS). The native application version is the app version number that you must change if you want to distribute a new version of an app (see Android Studio: Version Your App). You only need to change the entitlement version if the app starts to provide a new shared service or shared service version, or if the app stops providing a shared service or shared service version. For more information about when to change the entitlement ID and entitlement version, see the BlackBerry Dynamics API reference.

### Requirements for the entitlement ID and entitlement version

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required for apps</td>
<td>You must define both the entitlement ID and the entitlement version for all your BlackBerry Dynamics apps, regardless of whether you use the BlackBerry Dynamics Shared Services Framework. Developers and administrators should ensure that the value specified for the GDApplicationVersion key in the app configuration files is the same as the value the administrator specifies in BlackBerry UEM or in standalone Good Control. The entitlement version is independent of any native version identifier. For more information, see Distinction from and use with native language identifiers.</td>
</tr>
<tr>
<td>Represent the same app across all platforms</td>
<td>The same entitlement ID must be used to represent the app across all platforms. By default, access to apps varies by the type of app:</td>
</tr>
<tr>
<td></td>
<td>• By default, all versions of partner or ISV apps are available to all authorized users in any organization that the app has been published to.</td>
</tr>
<tr>
<td></td>
<td>• By default, each version of a BlackBerry Dynamics app requires that the administrator grant access in BlackBerry UEM or in standalone Good Control before users can run the app on users’ devices.</td>
</tr>
<tr>
<td>Naming scheme</td>
<td>Develop a naming scheme to meet your needs. For example:</td>
</tr>
<tr>
<td></td>
<td>• Entitlement ID: com.manufacturingco.gd</td>
</tr>
<tr>
<td></td>
<td>• Entitlement version: 1.0.0.0</td>
</tr>
<tr>
<td></td>
<td>• Native application version: 2.0</td>
</tr>
<tr>
<td>Entitlement ID format</td>
<td>• The general form of an entitlement ID is &lt;company_name&gt;.&lt;app_name&gt;.</td>
</tr>
<tr>
<td></td>
<td>• The ID must use reverse domain name form, for example, com.company.example. Use a domain name owned by your organization.</td>
</tr>
<tr>
<td></td>
<td>• The ID must not begin with com.blackberry, com.good, com.rim, or net.rim.</td>
</tr>
<tr>
<td></td>
<td>• The ID can contain only lower-case letters, numeric digits, hyphens, and periods.</td>
</tr>
<tr>
<td></td>
<td>• The string must follow the &lt;subdomain&gt; format defined in section 2.3.1 of RFC 1035, as amended by Section 2.1 of RFC RFC 1123.</td>
</tr>
<tr>
<td>Entitlement version value</td>
<td>• The value must use one to four segments of digits, separate by periods (x.x.x.x).</td>
</tr>
<tr>
<td></td>
<td>• Each segment can be up to three digits and must not use a leading zero (for example, 01.02 is not valid). A segment can use a single 0.</td>
</tr>
<tr>
<td></td>
<td>• The first release of an app should use the entitlement version 1.0.0.0.</td>
</tr>
</tbody>
</table>
Distinction from and use with native language identifiers

The Entitlement ID and Entitlement Version are BlackBerry Dynamics specific metadata and are independent of the identifiers needed by the app platforms themselves. The key point is that the values and the native language identifiers’ values can be the same but they do not necessarily have to be. Listed below by platform are the equivalent native identifiers, which are where the values of Entitlement ID and version are stored.

- `AndroidManifest.xml`
  - `packageName`
  - `packageVersion`

Unique native identifiers for enterprise apps

If you are developing a private app for use in your enterprise, make sure that the value you choose for the app's native identifiers (Bundle ID and others constructs used on other platforms) is unique, especially with respect to apps that are available through the public app stores.

Duplicate native identifiers can prevent the proper installation or upgrade of your own app.

For all your native identifiers, devise a naming scheme that you can be relatively certain is unique.

Mapping BlackBerry Dynamics entitlement ID to native identifiers

To take advantage of many features, such as Easy Activation, multi-authentication delegation, and the shared services framework, developers need to set up a map in the server between your defined Entitlement ID and the native identifiers on the platforms for which your app is distributed. The native platforms have no knowledge of the Entitlement ID; thus the mapping is needed for the operating systems to take over the actual function of the app.

Native version identifiers: * wildcard allowed for blocking app

The SDK supports use of native version identifiers in keeping with the conventions described by the major vendors. These same conventions apply to the use of the * wildcard in the server to deny apps by native version.

- Platform: `AndroidManifest.packageVersion`
- A string of the format major.minor.point with no explicit requirement to use integers, although this is implied and followed by convention.
- More information from Google

The * character can be used in native version identifiers, but must always be preceded by a period (.) and must be the last character in the native version string. Examples:

- Allowed: 2.3.0
- Not allowed: 2.*.3
- 2.0 includes 2.0.*

Set the entitlement ID and entitlement version in settings.json

You must create the file `assets/settings.json` and add to it the correct configuration information for your app. You can copy the file from one of the sample apps and edit it. The `GDApplicationID` and `GDApplicationVersion` settings described here enable Inter Container Communication (ICC), among other features. For more background, see BlackBerry Dynamics entitlement ID and version. The content needs to look like this, substituting your own `GDApplicationID` and `GDApplicationVersion` values:

```json
{
...
```
"GDLibraryMode": "GDEnterprise",
"GDApplicationID": "com.yourcompany.appname",
"GDApplicationVersion": "1.2.3.0",
"GDConsoleLogger": [
    "GDFilterErrors_",
    "GDFilterWarnings_",
    "GDFilterInfo_",
    "GDFilterDetailed_"
]
}

Restricted key prefix

Note that the key prefix "blackberry" is reserved by BlackBerry and should not be used for key values, key attributes, or key elements. For more information and examples, see the "Application Policies Definition" page in the appendix of the BlackBerry Dynamics SDK API Reference for Android or iOS.
Steps to get started with the BlackBerry Dynamics Bindings for Xamarin

Follow the steps below to start working with the BlackBerry Dynamics Bindings for Xamarin.

1. Install the BlackBerry Dynamics Bindings for Xamarin.
2. Install the BlackBerry Dynamics SDK for Android needed along with Xamarin. For details, see the BlackBerry Dynamics SDK for Android Development Guide.
3. Be sure to do the Set up for Xamarin.Android.
4. Familiarize yourself with the features of the software:
   - For general information, see BlackBerry Dynamics background.
   - To view code samples, see Sample apps.
5. Understand the Requirements and possible constraints on your programming.
7. Set up your projects:
   - For new projects see Load bindings for a new project.
   - For existing projects, see Implementing a BlackBerry Dynamics event listener.
8. Optionally implementing the BlackBerry Dynamics Launcher.
9. Test your app.
   - Use the testing tools supplied with Microsoft Visual Studio.
   - You might need the Debug settings to disable ARM v8 with Xamarin.Android.
10. Deploy your app. For options, see Ready your app for deployment.

Install the BlackBerry Dynamics Bindings for Xamarin

The bindings are delivered in the form of Dynamic Link Libraries (DLL) in nested .zip files for each platform. They are Xamarin projects:

- A project with bindings for the BlackBerry Dynamics SDK for Android
- A project for each of the Sample apps
- A project for the BlackBerry Dynamics Launcher Library

1. Download the package for your target platform (Android or iOS).
2. Uncompress the package.
3. Uncompress the Sample apps.
4. If you want to implement the BlackBerry Dynamics Launcher Library, uncompress the included project.

Location and names of DLLs

The DLLs are in the top directory of the project and for each sample app. The DLLs are named as follows:

- The SDK .zip package `BlackBerry_Dynamics_SDK_for_Xamarin_Android_<version>.zip` unzips `GoodDynamics.Android.dll`
- The .zip package `Examples.zip` unzips the Sample apps to individually named zip files and DLLs.
Install the BlackBerry Dynamics SDK

The BlackBerry Dynamics SDK for Android or BlackBerry Dynamics SDK for iOS or both must be installed along with Xamarin. For details, see the development guide for the appropriate platform.

Set up for Xamarin.Android

After installing the bindings and the BlackBerry Dynamics SDK for Android, follow these setup steps to prepare the installation for your project builds.

Rely on packaged GoodDynamics.Android binding project

You don’t need to create your own binding project. For standard use, you can rely on the GoodDynamics.Android.dll packaged with every sample app.

Set Attribute.AlwaysRetainTaskState = true

In your projects for Xamarin, be sure to set AlwaysRetainTaskState = true for proper behavior of the app. Any Activity that is exported (and thus has some form of Intent Filter that allows an external entity to start it) must have AlwaysRetainTaskState = true. Other activities do not need this setting. For more details, see https://developer.xamarin.com/api/field/Android.Content.PM.ActivityInfoFlags.AlwaysRetainTaskState/.
Load bindings for a new project

Before you begin: BlackBerry supplies Xamarin Sample apps. You can use these samples as a starting point.

If you are starting a new project that has not already been configured for the BlackBerry Dynamics Bindings for Xamarin, you need to configure the project to load them. To add the BlackBerry Dynamics bindings to your project, in Microsoft Visual Studio follow these steps:

1. Install the BlackBerry Dynamics Bindings for Xamarin.
2. Open your project.
3. In the left nav, right click the References folder.
4. Select Edit References.
5. On the .Net Assembly tab, click Browse to navigate your computer to find where you uncompressed the package with the appropriate DLL for your platform.
6. Follow the remaining prompts to add the DLL to your project.
Implementing a BlackBerry Dynamics event listener

The Application object manages the global app state of an app. Many of the sample apps installed with the BlackBerry Dynamics Bindings for Xamarin implement the event handling lifecycle. There are several key parts to implementing the life cycle of events. You have your choice of approach, one of which is presented here.

1. Specify GDappID and Version.
2. Implement the IGDAppEventListener and IGDStateListener interfaces.
   - Special-case the OnNotAuthorized events.
   - On authorization, relying on GDErrorNone, start the app UI.
3. Assign IGDEventListener and IGDStateListener implementations to the Android class.
4. Initialize your Activity for use with the BlackBerry Dynamics runtime.

Specify GDappID and Version

Add a settings.json file under the Assets folder of your app, where you will define the GDappID and Version. An example file is below. For background, see BlackBerry Dynamics entitlement ID and version.

```json
{
  "GDLibraryMode": "GDEnterprise",
  "GDappID": "com.good.gd.example.apachehttp",
  "GDappVersion": "1.0.0.0",
  "GDConsoleLogger": [ "GDFilterNone_",
    "GDFilterErrors_",
    "GDFilterWarnings_",
    "GDFilterInfo_",
    "GDFilterDetailed_" ]
}
```

Implement the IGDAppEventListener and IGDStateListener interfaces

```csharp
using System.Collections.Generic;
using Com.Good.GD;
using Android.Util;
namespace Example.Droid
{
    public class GDEventListener : Java.Lang.Object, IGDAppEventListener
    {
        private static GDEventListener _instance;
        private bool _started;
        string tag = "GDEventListener";
        public static GDEventListener Instance
        {
            get
            {
                if (_instance == null)
                {
                    _instance = new GDEventListener();
                }
                return _instance;
            }
```
public void OnGDEvent(GDAppEvent anEvent)
{
    if (anEvent.EventType == GDAppEventType.GDAppEventAuthorized)
    {
        OnAuthorized(anEvent);
    }
    else if (anEvent.EventType == GDAppEventType.GDAppEventNotAuthorized)
    {
        OnNotAuthorized(anEvent);
    }
    else if (anEvent.EventType == GDAppEventType.GDAppEventRemoteSettingsUpdate)
    {
        //handle app config changes
        ...
    }
    else if (anEvent.EventType == GDAppEventType.GDAppEventServicesUpdate)
    {
        //handle event service update
        ...
    }
    else if (anEvent.EventType == GDAppEventType.GDAppEventEntitlementsUpdate)
    {
        //handle event entitlement update
        ...
    }
    else
    {
        Log.Info(tag, "Unhandled GDEvent " + anEvent.EventType);
    }
}

private void OnNotAuthorized(GDAppEvent anEvent)
{
    GDAppResultCode code = anEvent.ResultCode;
    if (code == GDAppResultCode.GDErrorActivationFailed ||
    code == GDAppResultCode.GDErrorProvisioningFailed ||
    code == GDAppResultCode.GDErrorPushConnectionTimeout ||
    code == GDAppResultCode.GDErrorIdleLockout ||
    code == GDAppResultCode.GDErrorRemoteLockout ||
    code == GDAppResultCode.GDErrorPasswordChangeRequired ||
    code == GDAppResultCode.GDErrorSecurityError ||
    code == GDAppResultCode.GDErrorBlocked ||
    code == GDAppResultCode.GDErrorAppDenied ||
    code == GDAppResultCode.GDErrorWiped)
    {
        Log.Info(tag, "OnNotAuthorized " + anEvent.Message);
    }
    else if (code == GDAppResultCode.GDErrorIdleLockout)
    {
        Log.Info(tag, "OnNotAuthorized");
    }
    else
    {
        Log.Info(tag, "Unhandled not authorized event");
    }
}
private void OnAuthorized(GDAppEvent anEvent) {
    GDAppResultCode code = anEvent.ResultCode;
    if (code == GDAppResultCode.GDErrorNone) {
        if (!_started) {
            _started = true;
            //Launch app UI Here
            ...
        }
    } else {
        Log.Info(tag, "Authorized startup with an error");
    }
}

public class GDStateListener : Java.Lang.Object, IGDStateListener {
    private static GDStateListener _instance;
    public static GDStateListener Instance {
        get {
            if (_instance == null) _instance = new GDStateListener();
            return _instance;
        }
    }
    public void OnAuthorized() {
        //Handle Authorized
        ...
    }
    public void OnLocked() {
        //Handle Locked
        ...
    }
    public void OnUpdateConfig(IDictionary<String, Java.Lang.Object> settings) {
        //Handle UpdateConfig
        ...
    }
    public void OnUpdatePolicy(IDictionary<String, Java.Lang.Object> policyValues) {
        //Handle UpdatePolicy
        ...
    }
    public void OnUpdateServices() {
        //Handle UpdateServices
        ...
    }
    public void OnWiped() {
        //Handle Wiped
        ...
    }
    public void OnUpdateEntitlements() {

    
Implementing a BlackBerry Dynamics event listener | 19
Assign IGDEventListener and IGDStateListener implementations to the Android class

In your app file, usually defined as `path_to_your_project/app.cs`, assign the static IGDEventListener and IGDStateListener implementations to the GDAndroid class.

Authorization occurs after OnCreate method completes. Any interaction with the BlackBerry Dynamics Runtime outside of authorization within this method results in an exception.

```csharp
public class GoodApplication : Application
{
    public GoodApplication (IntPtr handle, JniHandleOwnership transfer)
    : base (handle, transfer)
    {
        // do any initialisation you want here (for example initialising properties)
    }

    public override void OnCreate ()
    {
        base.OnCreate ();

        GDAndroid.Instance.SetGDAppEventListener (GDEventListener.Instance);
        GDAndroid.Instance.SetGDStateListener(GDStateListener.Instance);
        ...
    }
}
```

Initialize your Activity for use with the BlackBerry Dynamics runtime

Make sure to initialize each Activity of your app for use with the BlackBerry Dynamics Runtime. In the OnCreate method, pass the current instance of the activity to the GDAndroid.Instance.ActivityInit method.

```csharp
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);
    GDAndroid.Instance.ActivityInit (this);
    // Set our view from the "main" layout resource
    SetContentView(Resource.Layout.Main);
    //Rest of Activity code below.
    ...
}
```
Implementing the BlackBerry Dynamics Launcher

You have the option to implement the BlackBerry Dynamics Launcher, the popular UI front-end to many BlackBerry Dynamics apps. For more information, see the documentation for the BlackBerry Dynamics Launcher Library.

To implement the BlackBerry Dynamics Launcher, you add the BlackBerry Dynamics Launcher button to an activity and include its authentication logic in the appropriate place in your app.

Before you begin:

• Install the BlackBerry Dynamics Bindings for Xamarin.
• Install the BlackBerry Dynamics SDK.

1. Download the BlackBerry Dynamics Launcher software from BlackBerry Developer Downloads.
2. Unzip the packages.
3. In your IDE, load the bindings into the desired projects. For each project, do the following:
   a) Right-click References and click Edit References to reference the Launcher project. The important file here is AndroidLauncherBinding.dll.
   b) In the Components folder, add the "Android Support Library v13" component for backwards compatibility.
4. Verify that the AndroidLauncherBinding project references the following Android Support packages. If they are not there, add them to the project's packages and reference them within the AndroidLauncherBinding project.
   • Xamarin.Android.Support.v13.25.3.1 or later
   • Xamarin.Android.Support.v4.25.3.1 or later
5. In your app, include the package Using Com.Good.Launcher;
6. Add the BlackBerry Dynamics Launcher to the activity you want it to appear on. The BlackBerry Dynamics Launcher can be initialized using either an inclusive or an exclusive list of activities. If an inclusive list is used, the BlackBerry Dynamics Launcher is shown only on the included activities and vice versa.

   In this example, it is only added to the main activity:

   ```csharp
   List includedActivities = new List();
   includedActivities.Add(Java.Lang.Class.FromType(typeof(MainActivity)));
   LauncherButton.InitForapp(this, includedActivities,
                            LauncherButton.ActivitiesTargetingMethod.Inclusive);
   ```
7. Add the following authorization code to the appropriate place in your app that authenticates users. Use true or false as appropriate.

   ```csharp
   HostingApp.Instance.SetAuthorized (true);
   ```
# Sample apps

The easiest way to get started with the BlackBerry Dynamics Bindings for Xamarin is to open one of the projects for sample apps. Uncompress the desired sample and double-click either the `.csproj` or `.sln` file to open the project.

<table>
<thead>
<tr>
<th>Sample app</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppKinetics shared services</td>
<td>Provides examples of how to use the BlackBerry Dynamics <a href="#">Shared Services Framework</a>.</td>
</tr>
<tr>
<td>Apache HTTP client</td>
<td>Demonstrates how to use BlackBerry Dynamics Secure Communication APIs to access resources behind the enterprise firewall. These secure communications APIs can be used to exchange data between the mobile app on the device and an app server utilizing the secure BlackBerry Dynamics proxy infrastructure.</td>
</tr>
<tr>
<td>App-specific policy</td>
<td>Demonstrates how to use BlackBerry Dynamics app-specific policy APIs. App policies control specific features of a single app, in contrast to built-in policies that apply to all apps.</td>
</tr>
<tr>
<td>BlackBerry Dynamics interaction/events</td>
<td>Demonstrates how to interact with the BlackBerry Dynamics Library and use the Remote Settings API.</td>
</tr>
</tbody>
</table>
| Greetings server            | Demonstrates how to write a server that illustrates the BlackBerry Dynamics Inter-container Communications (ICC) API. The ICC system is a means of exchanging data securely between two BlackBerry Dynamics apps running on the same device. One app is the server (or "provider") and the other is the client (or "consumer").  
  For background on the ICC, see [Shared Services Framework](#). |
| Greetings client            | Demonstrates how to write a client that illustrates the BlackBerry Dynamics Inter-container Communications (ICC) API. The ICC system is a means of exchanging data securely between two BlackBerry Dynamics apps running on the same device.  
  For background on the ICC, see [Shared Services Framework](#). |
<p>| Push channel                | Provides an example of the BlackBerry Dynamics Push infrastructure, showing how to control the connection, create channels, and send messages in a loopback manner to the client. The Push Channel framework is a BlackBerry Dynamics feature to receive real-time notifications from an app server. |</p>
<table>
<thead>
<tr>
<th>Sample app</th>
<th>Description</th>
</tr>
</thead>
</table>
| Secure copy-cut-paste   | Compares the use of the Android default UI text controls and the secure counterparts offered by the BlackBerry Dynamics UI text controls. Text data is encrypted/decrypted before copy/paste operations are performed. The BlackBerry Dynamics UI text controls in the sample are as follows:  
  • GDTextView  
  • GDEditText  
  • GDAutoCompleteTextView  
  • GDSearchBar  
  • GDWebView  
  For background, see Securing cut-copy-paste on devices (Data Leakage Prevention, or DLP).                                                                                           |
| Secure store            | Demonstrates how to use the BlackBerry Dynamics SDK to create, manage, and access files stored in an app’s secure container. Both the file names and file contents are encrypted, stored on the device, and can only be accessed when the BlackBerry Dynamics app is unlocked. |
Testing and troubleshooting

Rely on Visual Studio testing tools

You can use the testing and debugging tools in Microsoft Visual Studio to test and debug your app developed with the BlackBerry Dynamics bindings for Xamarin.

Debug settings to disable ARM v8 with Xamarin.Android

Follow these steps to avoid issues if ARMv8 architecture is enabled in Android builds:

1. Go to Project Options > Android Build > General tab.
2. Uncheck Use shared Mono runtime.
3. Go to Project Options > Android Build > Advanced tab > Supported ABIs.
4. Uncheck arm64-v8a.
5. Save your changes.
About the BlackBerry Dynamics SDK for Xamarin.Forms

This release of the BlackBerry Dynamics SDK adds support for Xamarin.Forms for Android and iOS. Xamarin.Forms is a framework that developers can use to create user interfaces that can be used across platforms. For more information, visit developer.xamarin.com to see Getting Started with Xamarin.Forms and An Introduction to Xamarin.Forms.

Visit BlackBerry Developer Downloads to download the BlackBerry Dynamics SDK for Xamarin.Forms package. See the BlackBerry Dynamics SDK for Xamarin Release Notes to review the lists of fixed and known issues in each release.

Principal interfaces

The SDK provides a unified API that supports the following principal interfaces:

- Runtime Object
- Secure Storage
  - Secure SQL Database
  - Secure File System
- Secure Communication
  - Socket
  - HTTP Communication
- Inter-Application Data Exchange
- Single Sign-On

For complete details about each interface, see the BlackBerry Dynamics SDK for Xamarin.Forms API reference.

Using the BlackBerry Dynamics SDK for Xamarin.Forms

- Follow the requirements, prerequisites, and setup instructions for the BlackBerry Dynamics SDK for Android or the BlackBerry Dynamics SDK for iOS (or both, if you develop apps for both platforms).
- Follow the instructions in Steps to get started with the BlackBerry Dynamics Bindings for Xamarin.
- Use the following Xamarin.Forms libraries that are distributed in the SDK package:
  - Xamarin.Forms common libraries:
    - BBDXamarinForms.Common.Interfaces.dll
    - BBDXamarinForms.Common.Library.dll
    - BBDXamarinForms.Common.Utils.dll
  - Xamarin.Forms platform libraries:
    - BBDXamarinForms.Droid.Interfaces.dll
    - BBDXamarinForms.Droid.Library.dll
    - BBDXamarinForms.iOS.Interfaces.dll
    - BBDXamarinForms.iOS.Library.dll
  - Xamarin binding libraries:
    - GoodDynamics.Android.dll
    - GoodDynamics.iOS.dll
• Review the sample projects in the BlackBerry Dynamics SDK for Xamarin.Forms package to familiarize yourself with the project structure and configurations.
  
  • BlankApp: The main project that contains application views that render on both platforms and allow the execution of the unified API. Demonstrates how to implement the BlackBerry Dynamics SDK event handling lifecycle. See the App.xaml.cs file and Implementing a BlackBerry Dynamics event listener.
  
  • AppKinecticsService: Demonstrates how to write a server application that uses the BlackBerry Dynamics Inter-Container Communications API (AppKinetics). AppKinetics allows BlackBerry Dynamics apps running on the same device to exchange data and commands securely.
  
  • SampleAppSuite: Demonstrates how to use the rest of the available APIs.

Note that the sample apps require the following NuGet packages that can be obtained using Microsoft Visual Studio: AutoMapper, NLog, Unity, SQLite-net, and System.ValueTuple.
Readying your app for deployment: server setup

You want to test your app on a BlackBerry server before you deploy it into production. You need to become familiar with how to setup and configure such a server. You have options available: either BlackBerry UEM or the older Good Control.

Check with your IT or other department to see what test servers might already be available in your organization.

**BlackBerry UEM preferred**

BlackBerry UEM is the primary server configuration to test and deploy your app.

If you upgrade from Good Control to BlackBerry UEM, you not only get to use the great feature set that Good Control provides but you also get to take advantage of an enhanced feature set such as:

- Support for more policies for operating systems
- Better app management
- More container types
- Improved administration and provisioning
- Advanced connectivity and networking
- Expanded compliance and integrity checking
- Additional email, content, location, and certificate features
- Access to BlackBerry Web Services APIs

For information on how to use BlackBerry UEM to manage BlackBerry Dynamics apps, see the BlackBerry UEM Administration Guide.

**Standalone Good Control**

The standalone Good Control server is also available, but BlackBerry encourages you to use BlackBerry UEM for your tests and deployment. For information about getting started with Good Control, see Developer Bootstrap: Good Control Essentials.
Details of support for client certificates

BlackBerry Dynamics SDK support for personal certificates (PKCS12 or PKI certs)

The BlackBerry Dynamics SDK has been enhanced to support personal certificates for authentication of applications at runtime.

No programming is required by the BlackBerry developer on any of the client BlackBerry Dynamics SDK platforms to take advantage of this feature. All operations are carried about by the BlackBerry Dynamics Runtime. The app must use the BlackBerry Dynamics Secure Communication Networking APIs provided in prior releases, the employee’s account must be correctly configured, and the GC must be the 2.0.xx.yy release later.

An enterprise can deploy corporate services requiring two-way SSL/TLS mutual authentication in order to authenticate their employees. Through the enterprise, the employee may be issued or otherwise obtain a password protected Personal Information Exchange file (PKCS12/p12/pfx) containing a SSL/TLS client certificate and private key required by such services for authentication purposes. This file may be installed on various machines and devices, including BlackBerry Dynamics apps, so that access can be granted to these services.

Setup in Good Control

Requirements of the certificates themselves are described in Certificate requirements and troubleshooting.

To deploy Personal Information Exchange files with BlackBerry Dynamics apps, the following steps must be taken to configure the GC and employee’s account. For more information, see the Good Control and Good Proxy Admin Help.

• After the GC is installed, an administrator may choose to extend the default 24-hour period that an employee’s protected Personal Information Exchange file shall be cached by the GC server.
• An administrator must add all BlackBerry Dynamics apps that access services requiring client authentication to the Certificates -> App Usage tab,
• An administrator must enable Use PKCS12 Certificate Management in the employee’s security policy,
• An administrator or employee must upload their Personal Information Exchange files to the Certificates tab.

Behavior of personal certificates in the app

After the employee activates a BlackBerry Dynamics app enabled for access to server resources requiring client authentication, it receives their Personal Information Exchange files, provided they are still cached on the GC. For each file, the employee is asked to enter their password protecting the file contents, so the identification material can be installed. Once installed, provided the identification is correct, the BlackBerry Dynamics app is granted access to server resources requiring two-way SSL/TLS mutual authentication when connecting.

If there is more than one Personal Information Exchange file required per employee, the BlackBerry Dynamics Runtime ensures that the certificate chosen to send to the server meets all of the following criteria:

1. Only client certificates suitable for SSL/TLS client authentication are eligible for sending to the server. That is, certificates that have no Key Usage and Extended Key Usage, or Key Usage contains "Digital Signature" or "Key Agreement", or Extended Key Usage contain "TLS Web Client Authentication", and those whose Key Usages and Extended Key Usages do not contradict allowances for SSL/TLS client auth.
2. If the server advertises the client certificate authority in the SSL/TLS handshake, only client certificates issued by these authorities will be considered
3. Only current client certificates will be considered (that is, certificates that have not expired or are not yet valid)
Usually this is sufficient to identify the correct client certificate, but if there is still more than one certificate meeting all of the above criteria then the first one is used. If the certificate chosen is not the desired one, the administrator or employee can manage this by removing the undesired client certificate from Good Control. The administrator can also increase the chance of success by ensuring the server is configured to advertise the client certificate authority in the SSL/TLS handshake.

Certificate requirements and troubleshooting

Make sure your certificates conform to these requirements:

- Certificates must be in PKCS 12 format: Certificate Authority (CA), public key, and private key, all in the same file.
- The PKCS12 file must end with the extension .p12 or .pfx.
- The PKCS 12 file must be password-protected.

There are many sources of certificates:

- Your own internal certification authority (CA)
- A well-known public CA
- Tools from the Internet, such as OpenSSL's keytool command. For example, the following is sufficient to generate a PKCS 12 certificate that is usable with Good Control; substitute your own values for alias the keystore name and the keystore password. If in doubt consult information on the Internet about all the possible options on the keytool command:

  keytool -genkeypair -alias good123 -keystore good123.pfx -storepass good123 -validity 365 -keyalg RSA -keysize 2048 -storetype pkcs12

Beware of weak ciphers from export

Personal Information Exchange files are encrypted, and therefore must be encrypted with FIPS-strength ciphers if to be used when FIPS is enabled on the employee’s security policy.

For their own maximum interoperability with other systems, it is common for third-party applications, for example the macOS keychain, to export identity material (credentials) using weak ciphers.

The administrator or employee can use a tool such as the OpenSSL command line to re-encrypt the file with a FIPS-strength cipher like so, which re-encrypts with the AES-128-CBC cipher:

  openssl pkcs12 -in weak.p12 -nodes -out decrypted.pem
  <enter password>
  openssl pkcs12 -export -in decrypted.pem -keypbe AES-128-CBC -certpbe AES-128-CBC -out strong.p12
  <enter password>
  rm decrypted.pem

Client certificate sharing among BlackBerry Dynamics-based applications

The BlackBerry Dynamics SDK supports the “sharing” of a single client certificate among all BlackBerry Dynamics-based applications for an end-user.

If the security policy for authentication via client certificates is enabled in Good Control or UEM and one or more client certificates have been uploaded to the server, those certificates are used for user authentication by all BlackBerry Dynamics-based applications on the user’s device.
• No programming is required.
• Client certificates must be enabled in Good Control or UEM and at least one PKCS 12 certificate for a user must be uploaded to server.

Kerberos PKINIT: User authentication with PKI certificates

The BlackBerry Dynamics SDK supports Kerberos PKINIT for user authentication using PKI certificates.

No programming is required to use Kerberos PKINIT.

Important: Kerberos PKINIT is distinct from Kerberos Constrained Delegation (KCD). PKINIT relies on the Key Distribution Center (KDC), which should not be confused with "KCD".

<table>
<thead>
<tr>
<th>Kerberos PKINIT</th>
<th>Kerberos Constrained Delegation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kerberos PKINIT authentication is between the BlackBerry Dynamics app and the Windows Key Distribution Center (KDC), which communicate directly, and user authentication is based on certificates issued by Microsoft Active Directory Certificate Services.</td>
<td><strong>Note:</strong> For PKINIT, Kerberos Constrained Delegation must not be enabled. If Kerberos Constrained Delegation has been configured, a BlackBerry Dynamics app does not use Kerberos PKINIT to access the defined KCD realms. Instead, when Kerberos Constrained Delegation is used, a trust relation has been previously established between BlackBerry Control and the Key Distribution Center, and BlackBerry Control communicates with the service on behalf of the app. Kerberos Constrained Delegation takes precedence over Kerberos PKINIT, even if the user has a valid certificate.</td>
</tr>
</tbody>
</table>

Key requirements for PKINIT

Organizations that want to use Kerberos for BlackBerry Dynamics apps must make sure the following requirements are met.

Servers

• Kerberos Constrained Delegation must not be enabled.
• Windows Key Distribution Center (KDC) services for KDC server certificates issued by a Certificate Authority (CA) via the Active Directory Certificate Services must come only from the following Windows Server versions. No other server versions are supported.
  • Internet Information Server with Windows Server 2008 R2
  • Internet Information Server with Windows Server 2012 R2
• In BlackBerry Control:
  • The KDC hosts must be in the Allowed Domains of the Connectivity Profile applied to the affected users' policy sets.
  • Valid KDC service certificates must be located either in the BlackBerry Dynamics Certificate Store or the Device Certificate Store.
**Client certificates**

- The minimum keylength for the certificates must be 2,048 bytes.
- Client certificates must include the User Principal Name (UPN) (for example, user@domain.com) in the Subject Alternative Name (SAN) of object ID (OID) szOID_NT_PRINCIPAL_NAME 1.3.6.1.4.1.311.20.2.3, as specified by Microsoft. See Microsoft Support: Object IDs associated with Microsoft cryptography.
- The domain of the UPN must match the name of the realm of the Windows Key Distribution Center (KDC) service.
- The Extended Key Usage (EKU) property of the certificate must be Microsoft Smart Card logon (1.3.6.1.4.1.311.20.2.2).
- Certificates must be valid. Validate them against the servers listed above.

**Client applications**

- In BlackBerry Work, to allow the use of client certificates, you must enable the `useEASAuthCert` setting.
- Apps must not send any password in the HTTP/HTTPS request.
- Apps must either set the HTTP/HTTPS header `WWW-Authenticate: Negotiate`, or not set any authorization method in the HTTP or HTTPS request, to which the server has responded with 401 `WWW-Authenticate: Negotiate`. For more information, see SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows.

**Key points**

The following are key points to note when integrating BlackBerry Dynamics and Kerberos infrastructure:

- The KDC host must be in the Allowed Domains of the Connectivity Profile applied to the affected users’ policy sets in BlackBerry Control.
- The KDC host must be listening on TCP port 88 (Kerberos default port).
- BlackBerry Dynamics does not support KDC over UDP.
- BlackBerry Dynamics does not use Domain Name System (DNS) records such as SRV, CNAME, or TXT to locate the correct KDC. That is, the KDC must have an A record (IPv4) or AAAA record (IPv6) in your DNS.
- BlackBerry Dynamics does not use Kerberos configuration files (such as `krb5.conf`) to locate the correct KDC.
- The KDC can refer the client to another KDC host. BlackBerry Dynamics will follow the referral, as long as the KDC host that is referred to can be reached by BlackBerry Dynamics. This setting is defined in the Allowed Domains of the Connectivity Profile that is applied to the affected users’ policy sets in BlackBerry Control.
- The KDC can obtain the TGT transparently to BlackBerry Dynamics from another KDC host.

**Background on PKINIT, with FAQ**

Consider the interactions in this KDC diagram.

Kerberos PKINIT authentication requires the client (in the drawing, the human John, running a BlackBerry Dynamics-enabled application) to be able to contact:

- When initializing the user session, the user’s Key Distribution Center (KDC) Authentication Service (AS) to obtain a Ticket-Granting Ticket (TGT)
- When establishing a connection to a resource (in the drawing, Service "A"), the resource’s KDC Ticket-Granting Service (TGS)

In a large organization users and resources might belong to various realms and there may be many KDCs, so how does BlackBerry Dynamics find the right one?
1. How does the client locate the user’s KDC Authentication Service when initializing the user’s session?
   • Password-based authentication
     The realm in the user name must contain the host name of the KDC AS. For example:
     User: user@MY.REALM.COM
     Password: myPassword
   • Certificate-based authentication: This is PKINIT.
     The realm in the UPN of the user’s certificate must contain the host name of the KDC AS. For example:
     UPN (OID 1.3.6.1.4.1.311.20.2.3): user@MY.REALM.COM

2. How does the client locate the resource’s KDC Ticket-Granting Service (TGS) when retrieving the resource?
   BlackBerry Dynamics attempts to obtain a TGS from the host in the domain of the resources URL. For example:
   URL: http://resource.myrealm.com/index.html
   The client will connect to KDC TGS running on host myrealm.com on TCP port 88.
EXPECTED SAVINGS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF BUSINESS OPPORTUNITY, OR CORRUPTION OR LOSS OF DATA, FAILURES TO TRANSMIT OR RECEIVE ANY DATA, PROBLEMS ASSOCIATED WITH ANY APPLICATIONS USED IN CONJUNCTION WITH BLACKBERRY PRODUCTS OR SERVICES, DOWNTIME COSTS, LOSS OF THE USE OF BLACKBERRY PRODUCTS OR SERVICES OR ANY PORTION THEREOF OR OF ANY AIRTIME SERVICES, COST OF SUBSTITUTE GOODS, COSTS OF COVER, FACILITIES OR SERVICES, COST OF CAPITAL, OR OTHER SIMILAR PECUNIARY LOSSES, WHETHER OR NOT SUCH DAMAGES 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