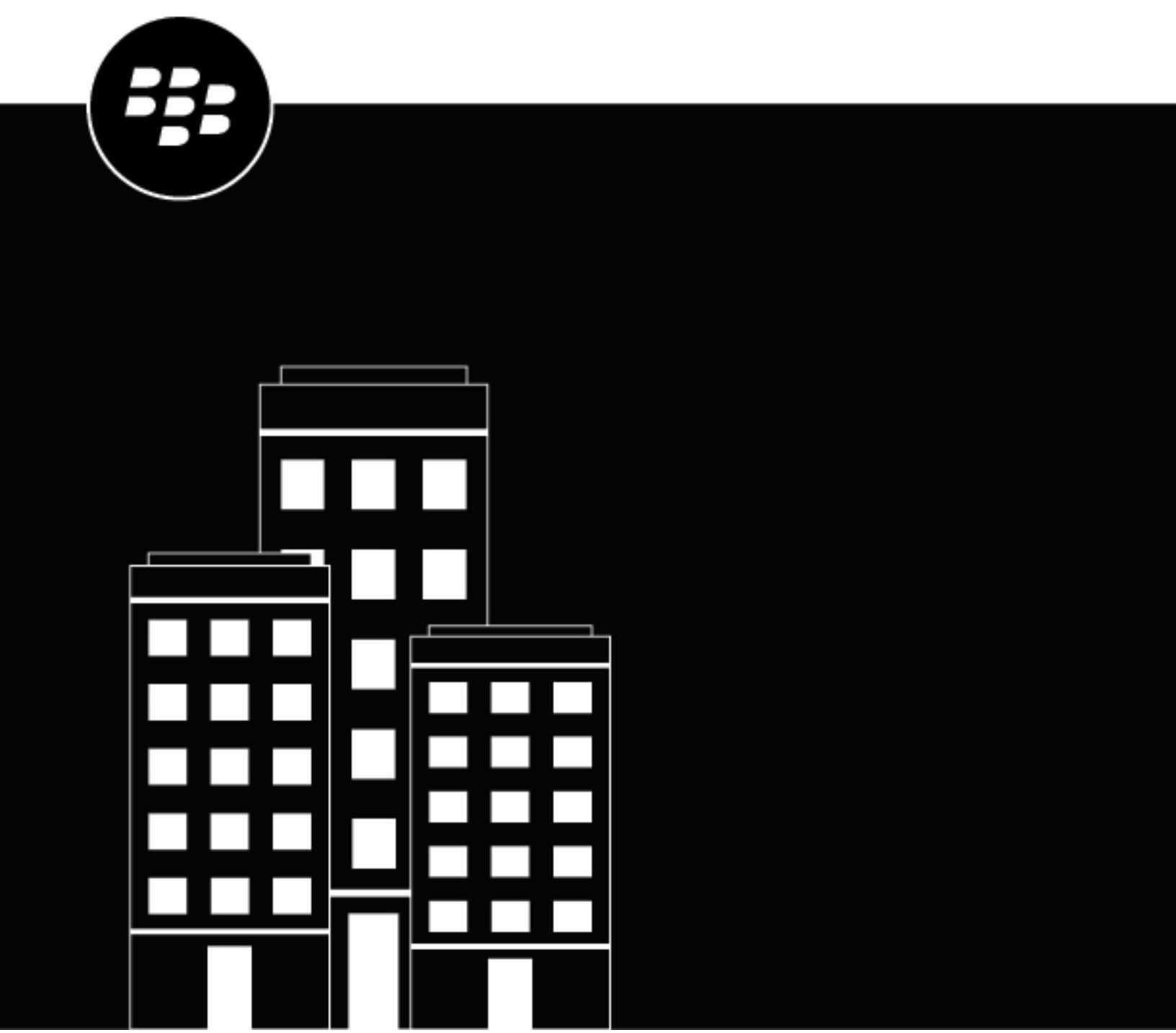

BlackBerry Dynamics Bindings for Microsoft.Android

Development Guide

12.1

Contents

What is the BlackBerry Dynamics SDK? 5

Reliance on the underlying BlackBerry Dynamics SDK 5
BlackBerry Dynamics API reference 5
Key features of the BlackBerry Dynamics SDK 5
Activation 6
Secure storage 7
Secure communication 7
Shared Services Framework 7
Data Leakage Prevention 8
User authentication 9
Administrative controls 11
Advanced security features with CylancePROTECT Mobile 11
Data collection and metrics with BlackBerry Analytics 11

Requirements and support for platform-specific features 13

Software requirements 13
Using an entitlement ID and version to uniquely identify a BlackBerry Dynamics app 14
Relationship between the entitlement ID and version and native identifiers 15
Specify the entitlement ID and entitlement version for your app 15
Using the entitlement version for the Shared Services Framework 15
FIPS compliance 16
FIPS-linking on Android 16
Supported CPU architectures 16
Requirements and prerequisites for Android platform features 17
Support for Android for Work and Samsung Knox APIs 17
Support for spannable text 17
Supported TLS protocols and cipher suites 18

Steps to get started with the BlackBerry Dynamics SDK 19

Install the BlackBerry Dynamics bindings for .NET 19
Using the BlackBerry Dynamics SDK framework 20
Prepare an existing BlackBerry Dynamics app to use the NuGet packages 20
Load bindings for a new project 20
Implement a BlackBerry Dynamics event listener 21
Implement the BlackBerry Dynamics Launcher 24

Implementing Play Integrity attestation for BlackBerry Dynamics apps 26

Prerequisites for Play Integrity attestation 26
Adding the GDSafetyNet library to the app project 26
Updating the BlackBerry Dynamics application policy file 27
Testing the app 28

About the BlackBerry Dynamics SDK for Microsoft.Maui.....29
Principal interfaces.....29
Using the BlackBerry Dynamics SDK for Microsoft.Maui.....29

Sample apps.....31

Testing and troubleshooting.....32
Troubleshooting common issues.....32
Disable ARM v8 to avoid potential issues.....32

Deploying your BlackBerry Dynamics app.....33
Configuring library version compliance.....33
Using an obfuscation tool in your build and release process.....34

Deploying certificates to BlackBerry Dynamics apps.....35
Using Personal Information Exchange files.....35
Configuring support for client certificates.....36
Certificate requirements.....36
Using Kerberos.....37

Legal notice.....38

What is the BlackBerry Dynamics SDK?

The BlackBerry Dynamics SDK provides a powerful set of tools that you can use to create secure productivity apps for a BlackBerry UEM domain. The SDK leverages the full capabilities of the secure BlackBerry Dynamics platform, so you can focus on building your apps rather than learning how to secure, deploy, and manage those apps.

The BlackBerry Dynamics SDK is available for all major development platforms. It allows you to leverage many valuable services, including secure communication, securing data in file systems and databases, inter-app data exchange, presence, push, directory lookup, single sign-on authentication, identity and access management, and more.

This guide will provide:

- Information about supported features
- Development requirements and prerequisites
- Instructions for installing, configuring, and using the SDK
- Considerations for key platform features
- Information about the sample apps provided with the SDK
- Testing and troubleshooting guidance
- Guidance for deploying your app

This guide is intended for intermediate and experienced developers with an understanding of how to create apps for the intended platform. It is not a basic tutorial.

Reliance on the underlying BlackBerry Dynamics SDK

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

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

BlackBerry Dynamics API reference

The BlackBerry Dynamics SDK API reference describes the available interfaces, classes, methods, and much more. The API reference for each SDK platform is available at https://developers.blackberry.com/us/en/resources/api-reference.html.

Key features of the BlackBerry Dynamics SDK

This section provides more information about key features of the BlackBerry Dynamics SDK. It does not detail the complete feature set. For more information about the full list of supported features and APIs, see the BlackBerry Dynamics SDK API reference for your platform.

For more information about the requirements and prerequisites to support platform-specific features, see Requirements and support for platform-specific features.

The implementation of some of the features discussed in this section will depend on how the UEM administrator has configured your organization's servers, network, and other infrastructure components. Contact the

administrator to clarify whether there are components of a feature that are configured or managed using the management server.

Activation

Infrastructure and enterprise activation

After a BlackBerry Dynamics app is installed on a user's device, the user must activate the app in order to use it. The activation process registers the app with the management server and gives the app access to the full capabilities of the BlackBerry Dynamics platform. The activation process ensures that all end users are fully authorized and permitted to use the app.

Users can activate a BlackBerry Dynamics app manually using an activation password, QR code, or access key provided by the administrator or obtained from UEM Self-Service, by using the UEM Client, through a third-party IDP, such as Active Directory or Okta, or by using the Easy Activation feature described below.

For more information about activating BlackBerry Dynamics apps, see the Activation section in the GDAndroid class reference or GDiOS class reference and Managing BlackBerry Dynamics apps in the UEM Administration content.

Easy Activation

Easy Activation simplifies the process of activating multiple BlackBerry Dynamics apps on a user's device. With Easy Activation, a device user only needs to activate the first BlackBerry Dynamics app on their device; when the user installs additional BlackBerry Dynamics apps, the user can choose to delegate the activation process to the previously activated app. Any BlackBerry Dynamics app can be an activation delegate, but priority is given to the app that is configured as the authentication delegate.

Easy Activation is automatically enabled for all BlackBerry Dynamics apps that are produced by BlackBerry. To enable Easy Activation for your custom BlackBerry Dynamics app, the UEM administrator must specify the app package ID (Android) or bundle ID (iOS) in the BlackBerry Dynamics app settings in the management console. Contact your organization's administrator to provide this information. For instructions for specifying the package ID or bundle ID for an app, see Manage settings for a BlackBerry Dynamics app in the UEM Administration content.

On the application side, Easy Activation is enabled by default by the BlackBerry Dynamics Runtime.

For more information, see the Easy Activation section in the BlackBerry Dynamics Security White Paper.

Programmatic activation

The programmatic activation feature enables a BlackBerry Dynamics app to activate without any user interaction and without displaying activation prompts or progress screens. This can be useful when targeting your apps to a consumer audience or for developing apps for devices that have limited or no means of user input.

For more information about programmatic activation, see `programmaticActivityInit` in the BlackBerry Dynamics SDK for Android API Reference or `programmaticAuthorize` in the BlackBerry Dynamics SDK for iOS API Reference.

Note the following implementation details:

- Decide whether you want users to specify a password to unlock a BlackBerry Dynamics app after the initial activation. You can use programmatic activation while still requiring users to type a password to unlock the app. This setting is configured in a BlackBerry Dynamics profile in UEM.
- To activate the app, your application server must use the BlackBerry Web Services REST APIs to retrieve the user credentials and to generate an access key. You may need to create a new UEM user account or to lookup an existing user account. See the User resource in the BlackBerry Web Services REST API reference for the available REST APIs that can be used to create a user, lookup a user, and to generate an access key.
- Pass the user credentials to the app and call `programmaticActivityInit` or `programmaticAuthorize` with the retrieved credentials, setting `ShowUserInterface` to false.

- For Android, receive the broadcast event `GD_STATE_ACTIVATION_ACTION` to track activation progress from `NotActivated` to `InProgress` to `Activated`. You can choose to display a progress indicator during this short period.
- For iOS, observe `GDState.BBActivationState` to track activation progress from `NotActivated` to `InProgress` to `Activated`. You can choose to display a progress indicator during this short period.
- Once activation completes, the user is prompted to set a password (unless you've configured the BlackBerry Dynamics profile to not require a password). The app should wait for the `GD_STATE_AUTHORIZED_ACTION` notification.
- You can use `configureUI` (Android/iOS) to customize the UI of the password screen (for example, with a custom logo and colors).

Secure storage

Secure file system

BlackBerry Dynamics apps store data in a secure, encrypted file system. For more information, see BlackBerry Dynamics File I/O Package in the BlackBerry Dynamics SDK for Android API reference, or GDFileManager and GDFileHandle in the BlackBerry Dynamics SDK for iOS API reference.

Secure SQL database

BlackBerry Dynamics apps can leverage a secure SQL database that stores and encrypts data on the user's device. The secure SQL database is based on the SQLite library.

For more information, see the BlackBerry Dynamics SQL Database page in the BlackBerry Dynamics SDK API reference (Android/iOS).

Secure communication

The BlackBerry Dynamics platform enables secure data exchange between a BlackBerry Dynamics app on an end user's device and a back-end application server on the Internet or behind the enterprise firewall. Any communication through the enterprise firewall uses the secure BlackBerry Dynamics proxy infrastructure. One app can communicate with multiple application servers.

To learn more about the programming interfaces for secure communication, see GDSocket and GDHttpClient in the BlackBerry Dynamics SDK for Android API reference, or GDSocket, GDURLLoadingSystem, and NSURLSession Support in the BlackBerry Dynamics SDK for iOS API reference.

AppKinetics

AppKinetics, or Inter-Container Communication (ICC), is a method for securely exchanging data and commands between two BlackBerry Dynamics apps on the same device. The exchange uses a consumer-provider model: one app initiates a service request that the other app receives and responds to as a service provider.

For more information about AppKinetics, see the Inter-Container Communication Package in the BlackBerry Dynamics SDK for Android API reference, or GDService and GDServiceClient in the BlackBerry Dynamics SDK for iOS API reference.

Shared Services Framework

BlackBerry Dynamics apps can communicate with each other and application servers using the Shared Services Framework, a collaboration system that is defined by two components: one that provides a service and another that consumes the service.

The provider can be a client-side service, which is a BlackBerry Dynamics app that uses the GDService APIs (Android/iOS), or a server-side service that is provided by an application server or other remote system. The service is consumed by a BlackBerry Dynamics app that communicates with the provider using AppKinetics (a proprietary BlackBerry ICC protocol) for client-side services or a protocol such as HTTPS for server-side services.

The typical steps that are required to consume a service:

1. Service discovery: The BlackBerry Dynamics app (the consumer) queries for service providers using the GDAndroid.getServiceProvidersFor API or the GDiOS.getServiceProvidersFor API. Service discovery is optional but recommended for both types of services because it respects user entitlements and permissions.
2. Provider selection: The consuming app selects the provider. This is handled by the app code.
3. Service request: The consuming app sends a service request to the provider using the GDServiceClient API (Android/iOS) for client-side services or TCP sockets or HTTP over BlackBerry Dynamics secure communication (Android/iOS) for server-side services.
4. Service response: The consuming app receives the provider response using the same interface that was used for the request (the GDServiceClient API (Android/iOS) for client-side services or BlackBerry Dynamics secure communication (Android/iOS) for server-side services).

Client-side services can be used offline and are ideal if the service requires specific user interaction.

Server-side services can be provided by a clustered application server and are ideal if the server software already exists outside of the BlackBerry Dynamics platform.

Client-side and server-side services both require user entitlement in the management console.

If you want your custom BlackBerry Dynamics app to use the Shared Services Framework, the UEM administrator must specify the app package ID (Android) or bundle ID (iOS) in the BlackBerry Dynamics app settings in the management console. Contact your organization's administrator to provide this information. For instructions for specifying the package ID or bundle ID for an app, see Manage settings for a BlackBerry Dynamics app in the *UEM Administration Guide*.

Sample apps that are included with the SDK demonstrate how to use the Shared Services Framework. For more information about how to use the Shared Services Framework, see the following resources:

- GDAndroid.getServiceProvidersFor
- GDiOS.getServiceProvidersFor
- GDService (Android/iOS)
- GDServiceClient (Android/iOS)
- BlackBerry Dynamics Service Definition (Android/iOS)
- Definitions and descriptions of published services

Server-side services can use the Push Channel API (Android/iOS) to send notifications to BlackBerry Dynamics apps. The channel is end-to-end secure at the same level as BlackBerry Dynamics secure communication. As a result, the BlackBerry Dynamics app does not need to poll the application server, which decreases the load on both the app and the application server. Any application server that is a service provider can use the Push Channel.

Data Leakage Prevention

The BlackBerry UEM administrator can use Data Leakage Prevention (DLP) settings in BlackBerry Dynamics profiles (UEM) to configure data protection standards, including enabling or disabling copy and paste between BlackBerry Dynamics apps and non-BlackBerry Dynamics apps, screen captures, dictation, FIPS, and more.

Contact your organization's administrator to configure DLP standards as necessary for your custom BlackBerry Dynamics apps.

Note the following for the different platforms of the SDK:

SDK platform	Notes
BlackBerry Dynamics SDK for Android	The SDK provides the following classes to manage the secure copy, cut, and paste of data. These classes are replacements for the equivalent standard Android classes. The secure copy-cut-paste sample app demonstrates uses of the secure clipboard and the secure widgets. • com.good.gd.content.ClipboardManager • com.good.gd.widget.GDTextView • com.good.gd.widget.GDEditText • com.good.gd.widget.GDAutoCompleteTextView • com.good.gd.widget.GDMultiAutoCompleteTextView • com.good.gd.widget.GDSearchView • com.good.gd.widget.GDAppCompatTextView • com.good.gd.widget.GDAppCompatCheckedTextView • com.good.gd.widget.GDAppCompatAutoCompleteTextView • com.good.gd.widget.GDAppCompatMultiAutoCompleteTextView • com.good.gd.widget.GDAppCompatEditText • com.good.gd.widget.GDAppCompatSearchView • com.good.gd.widget.GDWebView • com.blackberry.bbwebview.BBWebView The widget classes starting with "GDAppCompat" are replacements for the corresponding Android AppCompat widgets. Due to current DLP controls, the setOnReceiveContentListener method is not supported in the components and widgets listed above.

Automatic widget substitution in AppCompat apps

Apps that use AppCompatActivity with a theme derived from AppCompat can use the following to install the provided GDAppCompatViewInflater in their theme:

```
<item name="viewInflaterClass">com.good.gd.app.GDAppCompatViewInflater</item>
```

When this is installed, widgets that are defined in layout files are automatically replaced by their GDAppCompat equivalents. For more information, see the com.good.gd.widget API documentation.

User authentication

BlackBerry UEM offers the following options to adjust the user experience for accessing BlackBerry Dynamics apps.

Fingerprint and biometric authentication

Various forms of biometric authentication are supported by the BlackBerry Dynamics SDK, including fingerprint authentication and for Android and Touch ID and Face ID for iOS. The BlackBerry UEM administrator can use a BlackBerry Dynamics profile (UEM) to enable biometric authentication. Contact your organization's administrator to enable and configure these features.

For more information, see BlackBerry Dynamics and Fingerprint Authentication.

Authentication delegation

The BlackBerry UEM administrator can configure up to three BlackBerry Dynamics apps on users' devices to act as an authentication delegate (a primary, secondary, and tertiary delegate). When a user opens any BlackBerry Dynamics app, the device will display the login screen of the authentication delegate app. After the user logs in successfully, all of the BlackBerry Dynamics apps on the device are unlocked. The user does not need to enter a password again until the idle timeout is reached.

If you want your custom BlackBerry Dynamics app to be an authentication delegate, the UEM administrator must specify the app package ID (Android) or bundle ID (iOS) in the BlackBerry Dynamics app settings in the management console. Contact your organization's administrator to provide this information. For instructions for specifying the package ID or bundle ID for an app, see Manage settings for a BlackBerry Dynamics app in the *UEM Administration Guide*.

The administrator configures one or more authentication delegate using a BlackBerry Dynamics profile. It is a best practice to configure the most commonly used app as the authentication delegate. Contact your organization's administrator to configure one or more authentication delegates.

Note: If the administrator configures a secondary authentication delegate, the administrator must notify users that if they delete the primary authentication delegate app, the user must unlock the secondary delegate app and set the app password again so that it can be used to authenticate any additional BlackBerry Dynamics apps. The same requirement applies if a tertiary delegate is configured and the primary and secondary delegate apps are deleted.

Do not require a password

Enabled using a BlackBerry Dynamics profile, this setting removes the password login for BlackBerry Dynamics apps. Users cannot choose whether to use a password.

Do not enable authentication delegation and this setting in the same profile or policy set. This feature is supported in UEM 12.7 or later. If the setting is enabled and then disabled at a later date, users are prompted to create a password the next time they log in to a BlackBerry Dynamics app.

You can use the GDAndroid.getInstance().canAuthorizeAutonomously() or [GDiOS sharedInstance].canAuthorizeAutonomously method to check if this feature is enabled. See the GDInteraction sample app (Android) or the SecureStore sample app (iOS) for examples of this method.

Bypass the app unlock screen

Enabled in the UEM Client settings for a specific BlackBerry Dynamics app (UEM), this setting allows an app to completely bypass the password login screen.

For more information and programming guidance, see the Bypass Unlock Developer Guide.

Background Authorize for iOS

Background Authorize is a restricted API that allows a recently locked BlackBerry Dynamics app to use the principal BlackBerry Dynamics APIs (such as secure storage and secure communication) when the app is running in the background.

This feature can be useful in scenarios where the app has stopped unexpectedly and is started in the background in response to an APNS message (for example, a new email). If Background Authorize is enabled, the app can download new data and store it in the secure container. When the user brings the app to the foreground they can authorize and immediately access the data (for example, messages).

To access this restricted API, submit a request to the BlackBerry Dynamics Registrar program at BlackBerryDynamicsRegistrar@blackberry.com.

For more information about this feature, see the Background Authorize Developer Guide.

Background Authorize for Android

GDAndroid.canAuthorizeAutonomously allows BlackBerry Dynamics apps to background unlock, receive state callback, and use credential-protected storage. The app can use canAuthorizeAutonomously() to check if it is possible to use background unlock, and if possible, authorize with serviceInit().

Administrative controls

The BlackBerry UEM administrator can use various server settings, policies, and profiles to manage BlackBerry Dynamics apps and ensure that app usage meets the organization's security standards. Consult with your organization's administrator to ensure that your custom apps adhere to the configured settings in the management console.

For more information, see Controlling BlackBerry Dynamics on users devices, Enforcing compliance rules for devices, and Managing BlackBerry Dynamics apps in the *UEM Administration Guide*.

You also have the option to add new management policies and settings that are specific to your custom BlackBerry Dynamics app to the UEM management console so that they can be configured and applied to users by UEM administrators. For more information, see Adding custom policies for your app to the UEM management console.

Advanced security features with CylancePROTECT Mobile

The BlackBerry Dynamics SDK integrates the CylancePROTECT library to support CylancePROTECT Mobile for UEM. CylancePROTECT is a licensed service that offers a suite of features that enhances UEM's ability to detect, prevent, and resolve security threats without disrupting the productivity of your workforce. CylancePROTECT is configured and managed by the UEM administrator. No additional development or integration effort is required if your organization wants to leverage the CylancePROTECT features for custom BlackBerry Dynamics apps.

CylancePROTECT uses a combination of advanced technologies, including:

- The cloud-based CylanceINFINITY service that uses sophisticated AI and machine learning to identify malware and unsafe URL
- The UEM server that provides a complete device management and compliance infrastructure for your organization
- BlackBerry apps that monitor and enforce security standards at the device and user level

The seamless integration of these technologies establishes a secure ecosystem where data is protected and malicious activities are identified at all endpoints and eliminated proactively.

CylancePROTECT includes the following features:

- Malware detection for Android apps (including BlackBerry Dynamics apps) that are uploaded to UEM for internal deployment
- Malware detection on Android devices
- Sideloaded app detection on iOS and Android devices
- Safe browsing with BlackBerry Dynamics apps
- Insecure network detection on iOS and Android devices.
- Insecure Wi-Fi access point detection on iOS and Android devices.
- Integrity checking for BlackBerry Dynamics apps on iOS devices using the Apple DeviceCheck framework
- Hardware certificate attestation for BlackBerry Dynamics apps on Android devices

For more information about CylancePROTECT, see the CylancePROTECT Mobile for UEM documentation.

Data collection and metrics with BlackBerry Analytics

BlackBerry Analytics is a cloud-based portal that you can use to view information about the BlackBerry Dynamics apps and devices that are used in your organization's environment. Previously, the BlackBerry Analytics

functionality was offered in a separate SDK that you could integrate with your BlackBerry Dynamics apps. In SDK version 8.0 and later, BlackBerry Analytics functionality is now included in the BlackBerry Dynamics SDK.

For more information about BlackBerry Analytics, see the BlackBerry Analytics documentation. For more information about integrating BlackBerry Analytics with your BlackBerry Dynamics apps, see Integrating BlackBerry Analytics.

Requirements and support for platform-specific features

This section provides the software requirements for using the SDK, as well as prerequisites that are required to support platform-specific features.

Software requirements

Item	Requirement
Compatibility with previous releases of the BlackBerry Dynamics Bindings for Microsoft.Android	This release of the BlackBerry Dynamics Bindings for Microsoft.Android is compatible with the 11.0.x, 11.1.x, and 12.0.x releases of the SDK. However, the project must be migrated from Xamarin.Android to Microsoft.Android. For more information, see Upgrade from Xamarin to .NET in the Microsoft documentation.
BlackBerry Dynamics SDK for Android	The latest compatible version of the BlackBerry Dynamics SDK for Android is bundled with the SDK.
Microsoft.Android	34.0.95 or later
Microsoft Visual Studio for macOS	17.6.12 or later
Minimum Android API version	30
Target Android API version	34
Supported CPU architectures	• ARMv7 • ARMv8 • x86 • x86_64
AndroidX libraries	Apps must use the AndroidX support NuGets to compile successfully. The SDK supports the following minimum versions. It is strongly recommended to use the latest stable version of each library, with the required dependencies: • Xamarin.AndroidX.AppCompat:1.2.0.4 • Xamarin.AndroidX.CardView:1.0.0.5 • Xamarin.AndroidX.ConstraintLayout:1.1.3.2 • Xamarin.AndroidX.Core:1.3.0.3 • Xamarin.AndroidX.Legacy.Support.V4:1.0.0.5 • Xamarin.Androidx.Preference:1.1.1.5 • Xamarin.AndroidX.RecyclerView:1.1.0.5

Item	Requirement
Google Play Services	The SDK uses Google Play Services version 17.0.0 to support some of its functionality. If your app uses the following Google Play Services libraries, verify that you are using the following minimum version or later: • Xamarin.GooglePlayServices.SafetyNet:117.0.0 • Xamarin.GooglePlayServices.Location:117.0.0
Java compatibility	You must build applications using Java 17 or later. For more information, see the Microsoft resources for Downloading OpenJDK 17 and Installing the JDK.
BlackBerry Dynamics Launcher Library	The BlackBerry Dynamics Launcher is a user-friendly interface that allows users to easily access and switch between BlackBerry Dynamics apps, configure app settings, and take advantage of other useful features. For more information, see the BlackBerry Dynamics Launcher Framework documentation. The BlackBerry Dynamics SDK and the BlackBerry Dynamics Launcher Library are mutually dependent. See the BlackBerry Dynamics SDK for Android Release Notes for the required version of the BlackBerry Dynamics Launcher Library.

Note: 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 in the appendix of the API Reference.

Using an entitlement ID and version to uniquely identify a BlackBerry Dynamics app

BlackBerry Dynamics apps are uniquely identified by a BlackBerry Dynamics entitlement ID (GDApplicationID) and entitlement version (GDApplicationVersion). The entitlement ID and entitlement version are used to manage end-user entitlement in the management console. The values are also used for app publishing and service provider registration.

These values are specified in the assets/settings.json file for Android or in the Info.plist file for iOS.

You must define both the entitlement ID and the entitlement version for all of your BlackBerry Dynamics apps, regardless of whether you use the Shared Services Framework. The same entitlement ID must be used to represent the app across all platforms.

For more information about setting and checking the entitlement ID and version, the proper format to use for these values, and other requirements and considerations:

- See the GDAndroid Class reference in the BlackBerry Dynamics SDK for Android API Reference, especially these sections:
  - Identification
  - Indirect Authorization Initiation
  - void authorize (GDAppEventListener eventListener) throws GDInitializationError
- See the GDiOS Class reference in the BlackBerry Dynamics SDK for iOS API Reference, especially these sections:
  - Identification

- Authorization
- - (void) authorize: (id< GDiOSDelegate > _Nonnull) delegate

Relationship between the entitlement ID and version and native identifiers

The entitlement ID (GDApplicationID) and entitlement version (GDApplicationVersion) of a BlackBerry Dynamics app are different from the native identifiers that are required by the app OS platform. The native identifiers for Android are the packageName and packageVersion values in the AndroidManifest.xml file. The native identifiers for iOS are the CFBundleIdentifier and CFBundleVersion in the Info.plist file. The values of the entitlement ID and entitlement version and the platform native identifiers can be the same, but do not have to be.

To take advantage of BlackBerry Dynamics features such as Easy Activation, authentication delegation, certificate sharing, the Shared Services Framework, and more, the UEM administrator must specify the entitlement ID and version and the native identifier (package name or bundle ID) for a custom BlackBerry Dynamics app in the management console. For more information, see Add an internal BlackBerry Dynamics app entitlement and Manage settings for a BlackBerry Dynamics app in the *UEM Administration Guide*.

The native identifiers for your custom BlackBerry Dynamics app should be unique, especially with respect to apps that are available through public app stores. Duplicate native identifiers can prevent the proper installation or upgrade of an app.

You must change the native app version if you want to distribute a new version of the 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.

Specify the entitlement ID and entitlement version for your app

Specifying the entitlement ID and version for your app enables Inter Container Communication (ICC) and other BlackBerry Dynamics features.

Copy the **assets/settings.json** file from one of the sample apps that is included in the SDK and populate it with the configuration information for your BlackBerry Dynamics app, including the entitlement ID (GDApplicationID) and entitlement version (GDApplicationVersion).

You can also use this file to configure logging for the Android Debug Bridge console.

Sample contents of a settings.json file

```
{
"GDLibraryMode": "GDEnterprise",
"GDApplicationID": "com.yourcompany.appname",
"GDApplicationVersion": "1.0.0.0",
"GDConsoleLogger": [
       "GDFilterErrors_",
       "GDFilterWarnings_",
       "GDFilterInfo",
       "GDFilterDetailed"
    ]
}
```

Using the entitlement version for the Shared Services Framework

For BlackBerry Dynamics apps that provide a service that is consumed by other BlackBerry Dynamics apps through the Shared Services Framework, you should include the entitlement version in the app's AndroidManifest.xml file. This allows the SDK routines that work with the services to identify the required version of the service provider.

See the snippet below, or the AppKinectics sample apps, for examples of how to add the entitlement version to the AndroidManifest.xml file for the service-provider app. The GDApplicationVersion value must match the same value that you specified in assets/settings.json.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.good.gd.example.appkinetics
      [...]
     <app
         [...]
         <meta-data android:name="GDApplicationVersion" android:value
 ="your_value_here"/>
     </app>
</manifest>
```

FIPS compliance

It is a best practice to make your BlackBerry Dynamics apps compliant with U.S. Federal Information Processing Standards (FIPS) 140-2.The BlackBerry Dynamics SDK distribution contains FIPS canisters and tools.

The BlackBerry UEM administrator enables FIPS compliance using a BlackBerry Dynamics profile (UEM). If enabled, BlackBerry Dynamics apps must start in FIPS-compliant mode. The SDK determines whether a service is running in FIPS mode when the app communicates with the server to receive policies.

FIPS compliance enforces the following constraints:

- The use of MD4 and MD5 are prohibited. As a result, access to NTLM-protected or NTLM2-protected web pages and files is 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.
- For iOS, when you build for testing with the x86 64-bit simulator, FIPS mode is not enforced. As a result, you might see a difference in behavior with the simulator compared to actual operation. BlackBerry recommends that you always test your app on actual iOS hardware and not rely exclusively on the simulation.

FIPS-linking on Android

When you build an app, the BlackBerry Dynamics SDK links for FIPS compliance automatically. No special directives are required. FIPS compliance is not supported on x86 emulators. If you want to test on x86 emulators, you must disable FIPS compliance.

To verify that FIPS has been included, verify that the following line is in the log output. This same line is printed at the start of the app launch:

```
IDeviceBase::initInstance: FIPS MODE REQUESTED
```

Supported CPU architectures

The BlackBerry Dynamics SDK includes the native libraries built for the ARMv7, ARMv8, x86, and x86_64 CPU architectures. If you include the library project in your app, all of the ARMv7, ARMv8, x86, and x86_64 libraries are

included when your app is built. Unless you use `RuntimeIdentifier(s)` as described below, all library types will be included.

Consider the following:

- Including multiple CPU architectures can increase the size of the app.
- x86 and x86_64 are supported for Android emulators as a faster alternative to ARM architectures. You may only want to include the x86 or x86_64 library in test versions of your app that you want to run on an emulator.
- Ensure that your app build includes the native library for each CPU architecture that it uses. It is a best practice to exclude native libraries for architectures that the app does not use. If the included libraries do not match the architecture used by the app, you may experience issues when running the app. For example, if your app uses a native library helloWorld_armv8.so, the app build must include the BlackBerry Dynamics ARMv8 library, and it does not need to include the ARMv7, x86, or x86_64 libraries.

  You can specify the required CPU architectures using `untimeIdentifier(s)` in the .csproj file. For example, to include only ARMv7 libraries in your project:

```
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <RuntimeIdentifier>android-arm</RuntimeIdentifier>
</PropertyGroup>
```

Requirements and prerequisites for Android platform features

This section provides specific requirements or considerations to support features of the Android OS platform in your BlackBerry Dynamics apps.

Support for Android for Work and Samsung Knox APIs

The SDK supports the Android APIs that were formerly part of Android for Work and Samsung Knox. No extra programming work is required for Android for Work or Samsung Knox to interoperate with the BlackBerry Dynamics SDK for Android.

Support for spannable text

The BlackBerry Dynamics SDK for Android supports copying and pasting Spannable text. The following Span objects are supported:

- AbsoluteSizeSpan
- AlignmentSpan
- Annotation
- BackgroundColorSpan
- BulletSpan
- EasyEditSpan
- ForegroundColorSpan
- LeadingMarginSpan
- LocaleSpan
- QuoteSpan
- RelativeSizeSpan
- ScaleXSpan
- StrikethroughSpan
- StyleSpan
- SubscriptSpan

- SuggestionSpan
- SuperscriptSpan
- TextAppearanceSpan
- TypefaceSpan
- UnderlineSpan
- URLSpan

Supported TLS protocols and cipher suites

The SDK uses CURL 8.0.0, supporting the TLS protocols and cipher suites of TLS version 1.3.

Steps to get started with the BlackBerry Dynamics SDK

Step	Action
1	Install the BlackBerry Dynamics bindings for .NET.
2	If applicable, Load bindings for a new project
3	Optionally, implement an event listener. Alternatively, you can use the GDStateAction interface to implement an event handling mechanism. The GreetingsServer sample app demonstrates the use of this API.
4	Optionally, implement the BlackBerry Dynamics launcher.
5	Consult the appropriate BlackBerry Dynamics SDK API reference for instructions for implementing the desired features of the BlackBerry Dynamics platform. See the sample apps included in the SDK package for examples of how to implement key features. If you want to use Microsoft.Maui, see About the BlackBerry Dynamics SDK for Microsoft.Maui and the corresponding API reference.
6	Test and debug your app.
7	Deploy your app.
8	Optionally, deploy certificates to the BlackBerry Dynamics apps on users' devices.

Install the BlackBerry Dynamics bindings for .NET

The bindings are delivered as a local NuGet package distribution with the required resources. The package includes .NET/Microsoft.Android projects for the BlackBerry Dynamics bindings, for each of the sample apps, and for the BlackBerry Dynamics Launcher Library.

Before you begin: Visit BlackBerry Developer Downloads to download the BlackBerry Dynamics Bindings package for the desired platform. When you click the link, you are prompted to log in to the Developer site with your BlackBerry Online Account. If you don't already have an account, you can register and create one.

Extract the contents of the BlackBerry Dynamics Bindings SDK package (the projects for the bindings package, sample apps, and if desired, BlackBerry Dynamics Launcher Library).

After you finish: It is a best practice to use the delivered NuGet packages and corresponding resources directory that are included with every sample app instead of creating your own binding project.

Using the BlackBerry Dynamics SDK framework

The BlackBerry Dynamics SDK is currently offered as a dynamic framework, consisting of the BlackBerryDynamics.xcframework and two BlackBerryCerticom xcframeworks packed as NuGet packages. Follow the tasks below to configure your existing apps and new apps to use the BlackBerry Dynamics SDK.

Prepare an existing BlackBerry Dynamics app to use the NuGet packages

Complete the steps below if you have an existing BlackBerry Dynamics app that uses the DLL binding libraries (GoodDynamics.Android.dll / AndroidLauncherBinding.dll).

1. Open your project in Microsoft Visual Studio.
2. In the **References** section, remove the GoodDynamics.Android / AndroidLauncherBinding references.
3. Remove the AndroidAarLibrary/AndroidJavaLibrary references:
   - android_handheld_resources-<sdk_version>.aar
   - android_handheld_blackberry_protect_support-<sdk_version>.aar
   - android_handheld_gd_safetynet-<sdk_version>.aar
   - zxing-android-embedded-4.1.0-<sdk_version>.aar
   - core-3.4.0-<sdk_version>.jar
4. Remove the GoodDynamics.Android.dll / AndroidLauncherBinding.dll files from the disc.
5. Remove the .aar/.jar files from the disc (sample projects):
   - android_handheld_resources-<sdk_version>.aar
   - android_handheld_blackberry_protect_support-<sdk_version>.aar
   - android_handheld_gd_safetynet-<sdk_version>.aar
   - zxing-android-embedded-4.1.0-<sdk_version>.aar
   - core-3.4.0-<sdk_version>.jar
6. Visit https://learn.microsoft.com/en-us/dotnet/maui/migration/ and follow the instructions to upgrade from Xamarin to .NET.

Load bindings for a new project

If you are starting a new project that has not already been configured for the BlackBerry Dynamics bindings for .NET, you must configure the project to load them.

Before you begin:

If required, Prepare an existing BlackBerry Dynamics app to use the NuGet packages.

1. In Microsoft Visual Studio, open your project.
2. Provide the NuGet.Config file and point it to the local source that contains the binding packages.
3. Right click the **Dependencies** folder and click **Manage NuGet Packages...**.
4. Choose the package source with the local binding packages that you provided in step 2, then choose the require items.
5. Follow the prompts to add the NuGet to your project.

Implement a BlackBerry Dynamics event listener

The **Application** object manages the app's global app state. Many of the sample apps that are included with the BlackBerry Dynamics Bindings demonstrate the event handling lifecycle. This topic demonstrates one approach to implementing the lifecycle of events.

Alternatively, you can use the GDStateAction interface to implement an event handling mechanism. The GreetingsServer sample app demonstrates the use of this API.

Before you begin: Define the BlackBerry Dynamics entitlement ID and entitlement version for your app.

1. Implement the `IGDAppEventListener` and `IGDStateListener` interfaces:

```
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()
        {
            //Handle Wiped
                ...
        }
    }
}
```

2. In your app file (<*project_path*>**/Application.cs**), assign the static `IGDEventListener` and `IGDStateListener` implementations to the `GDAndroid` class. Authorization occurs after the `OnCreate` method completes. Any interaction with the BlackBerry Dynamics Runtime outside of authorization within this method results in an exception.

```
[Application]
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);
        …
    }
}
```

3. 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.

```
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.
  …
}
```

Implement the BlackBerry Dynamics Launcher

You have the option to implement the BlackBerry Dynamics Launcher, an intuitive front-end UI that makes it easy for device users to access and modify settings for BlackBerry Dynamics apps. For more information, see the documentation for the BlackBerry Dynamics Launcher Framework.

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.

1. Download the BlackBerry Dynamics Launcher Bindings for .NET software from BlackBerry Developer Downloads.
2. In your IDE, load the NuGet binding packages into the desired projects. For each project, right-click **Dependencies** and click **Manage NuGet Packages...** to reference the Launcher binding. For more information, see Load bindings for a new project.
3. In your app, include the following packages:

```
using Com.Blackberry.Launcher;
using Com.Good.Launcher;
```

4. Add the BlackBerry Dynamics Launcher to the activity that you want it to appear on. The BlackBerry Dynamics Launcher can be initialized using either an inclusive or exclusive list of activities. If an inclusive list is used, the BlackBerry Dynamics Launcher is shown only on the included activities.

   In this example, the BlackBerry Dynamics Launcher is only added to the main activity:

```
var includedActivities = new List<Java.Lang.Class>
{
    Java.Lang.Class.FromType(typeof(MainActivity))
};
GDLauncher.InitForApplication(this, includedActivities,
 LauncherButton.ActivitiesTargetingMethod.Inclusive, this);
```

You can also use an initialization API for all activities:

```
GDLauncher.InitForApplication(this, this);
```

5. Implement ILauncherDelegate and pass it to InitForApplication. Optionally, override the default implementation.

```
// Override default interface implementation
void ILauncherDelegate.OnSettingsCommand()
{
    Toast.MakeText(this, $"Launcher version: {GDLauncher.Version}",
 ToastLength.Short).Show();
}
```

6. To get broadcast receivers to pass authorization events, ApplicationInit is mandatory before calling GDLauncher.InitForApplication in the application OnCreate method.

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

    GDAndroid.Instance.ApplicationInit(this);

    // Initialize launcher
    GDLauncher.InitForApplication(this, this);
}
```

7. Add the following authorization code to the appropriate place in your app that authenticates users. Use `true` or `false` as appropriate.

```
HostingApp.Instance.SetAuthorized (true);
```

Implementing Play Integrity attestation for BlackBerry Dynamics apps

BlackBerry UEM version 12.18 and later supports Play Integrity attestation for BlackBerry Dynamics apps. BlackBerry UEM version 12.17 and earlier supports SafetyNet attestation for BlackBerry Dynamics apps. For information on SafetyNet attestation, refer to previous versions of the SDK documentation.

You can use Play Integrity to extend BlackBerry root and exploit detection and to enhance app security and integrity. For more information about Play Integrity attestation, implementation considerations, and instructions for enabling the feature, see the BlackBerry UEM documentation. This chapter details considerations for developers who want to enable Play Integrity support for their BlackBerry Dynamics apps.

To support Play Integrity, you must complete the Play Integrity prerequisites, add a new library component to the app project, and update the BlackBerry Dynamics application policy file.

Note: You cannot run an app on an Android emulator while Play Integrity is enabled as Play Integrity attestation will fail during app provision. You may need to contact your IT admin to turn off Play Integrity if you are using an emulator.

Prerequisites for Play Integrity attestation

Play Integrity attestation is dependent on configurations made within the Google Play console. Use the following steps to configure Play Integrity attestation for your apps:

1. In the Google Play console, select the app you want to configure for Play Integrity attestation, then click **Setup > App signing**.
2. On the Integrity API tab, in the Device Integrity section, ensure that the checkboxes for **MEETS_BASIC_INTEGRITY** and **MEETS_STRONG_INTEGRITY** are enabled.
3. On the App Signing tab, make note of the SHA-256 certificate fingerprint for the **App signing key certificate** and the **Upload key certificate**. These certificate fingerprints are used when you configure your BlackBerry Dynamics application policy file.

After you finish:

- For more information on the Google Play Console, see the Google Play console documentation.
- For more information on device integrity, see the Android developer documentation.

Adding the GDSafetyNet library to the app project

The BlackBerry Dynamics SDK for Android version 5.0 and later includes a new GDSafetyNet library. To support Play Integrity, add the GoodDynamics.Android.SafetyNet NuGet library and refer to the Load bindings for a new project steps and the Apache HTTP sample configuration.

The GDSafetyNet library includes all of the client-side source code that is required to support Play Integrity. No additional app code is required. The GDSafetyNet library requires Xamarin.GooglePlayServices.SafetyNet NuGet 71.0 (which is included in the NuGet library above) or later to use device Play Integrity APIs. Verify that your BlackBerry Dynamics app is dependent on only a single version of Google Play Services.

Updating the BlackBerry Dynamics application policy file

During a Play Integrity attestation process, BlackBerry UEM uses the app response to verify that it is communicating with the official version of the app. You must provide this information in the application policy file.

In order to configure Play Integrity, you will need to provide a Play App signing key. You have two options for a Play app signing key: you can use the Google Play generated app signing key or upload your own private app signing key. For information on finding your app signing keys in your Google Play Console, see Prerequisites for Play Integrity attestation. The digest hash in your application policy file must correspond to your Play app signing key in your Google Play Console.

Consider the following example from the Greetings Client sample app in the BlackBerry Dynamics SDK:

```
<?xml version="1.0" encoding="utf-8"?>
<apd:AppPolicyDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:apd="urn:AppPolicySchema1.good.com"
    xsi:schemaLocation="urn:AppPolicySchema1.good.com AppPolicySchema.xsd" >
    <pview>
        <pview>
            <sendto client="None" />
            <desc>Play Integrity Attestation Supported</desc>
            <pe ref="apkCertificateDigestSha256"/>
            <pe ref="apkPackageName" />
            <pe ref="Description" />
        </pview>
    </pview>
    <setting name="apkCertificateDigestSha256">
        <hidden>
            <key>blackberry.appMetadata.android.apkCertificateDigestSha256</key>

 <value>DD:83:CA:47:09:FA:C5:33:75:FE:F4:A1:B5:FB:F4:A8:E8:C2:7A:DF:AF:24:
0D:7B:E3:BA:BD:FB:A9:2B:F9:D6</value>
    </hidden>
    </setting>
    <setting name="apkPackageName">
        <hidden>
            <key>blackberry.appMetadata.android.apkPackageName</key>
            <value>com.good.gd.example.services.greetings.client</value>
        </hidden>
    </setting>
    <setting name="Description" >
        <text>
            <key>snet</key>
            <label>Play Integrity</label>
            <value>Play Integrity</value>
        </text>
    </setting>
</apd:AppPolicyDefinition>
```

The app is uniquely identified by the combination of the official package name (in the example above, blackberry.appMetadata.android.apkPackageName) and the digest hash of the official signing key (in the example above, blackberry.appMetadata.android.apkCertificateDigestSha256). If the app is not publicly listed in the Google Play Store, you may extract the certificate using keytool. To determine the digest hash, you can use the following keytool command, specifying the keystore and key name that was used to sign the app:

```
keytool -list -v -keystore <KEYSTORE_NAME> -alias <KEY_NAME>
```

This command will provide a response like the following:

```
Creation date: 4-Sep-2018
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Sample
Issuer: CN=Sample
Serial number: 27c738c9
Valid from: Tue Sep 04 08:28:10 BST 2018 until: Wed Aug 22 08:28:10 BST 2068
Certificate fingerprints:
     MD5:  4C:30:85:93:5E:96:12:90:CF:A0:77:48:A5:CA:63:8F
     SHA1: 3C:52:A0:2A:76:63:15:C9:20:C1:06:D9:4D:75:7C:14:D6:7C:30:BC
     SHA256:
 DD:83:CA:47:09:FA:C5:33:75:FE:F4:A1:B5:FB:F4:A8:E8:C2:7A:DF:AF:24:0D:7B:E3:
BA:BD:FB:A9:2B:F9:D6
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
```

After you update the application policy file, coordinate with the BlackBerry UEM administrator to upload the app to UEM (see Deploying your BlackBerry Dynamics app) and to upload the application policy file in the management console (see Manage settings for a BlackBerry Dynamics app in the UEM Administration Guide). Before the administrator uploads the application policy file, verify that the Android app package ID has been specified or that the app source file has been uploaded; both settings are configured in the app entitlement settings (Android tab) in the management console.

UEM validates the format of the input package name and digest hash. If you update the application policy file and upload the app again, it can take up to 24 hours for the change to synchronize to all UEM instances. When the app is uploaded again, it is removed from the current list of apps that are enabled for attestation and must be added again.

Testing the app

After completing the integration tasks, your app is Play Integrity ready and you can proceed with further testing. Verify that the app displays in the list of Play Integrity capable apps in the UEM management console. If the app does not display in the list, it is likely that UEM was not able to parse the application policy file.

When the app appears in the list of Play Integrity capable apps, the administrator can then enable the app for Play Integrity. For instructions, see the BlackBerry UEM documentation. An app that was signed correctly will activate successfully. An app that was not signed correctly will not activate and a Play Integrity validation failure message will display in the app.

About the BlackBerry Dynamics SDK for Microsoft.Maui

Note: Due to the deprecation and upcoming end of support for Xamarin products, Microsoft has switched from Xamarin.Forms to Microsoft.Maui.

Microsoft.Maui is a framework that developers can use to create user interfaces that can be used across platforms. For more information, visit https://learn.microsoft.com/en-us/dotnet/maui/ to see Getting Started with Microsoft.Maui, What is .NET Maui?, and instructions to upgrade from Xamarin to .NET.

Visit BlackBerry Developer Downloads to download the BlackBerry Dynamics SDK for Microsoft.Maui package. See the BlackBerry Dynamics SDK for .NET 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
- Push Channel
- Inter-Application Data Exchange
- Single Sign-On

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

Using the BlackBerry Dynamics SDK for Microsoft.Maui

- 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 SDK.
- Use the following Microsoft.Maui NuGet libraries that are distributed in the SDK package:
  - Microsoft.Maui NuGet common libraries: BBDXamarinForms.Common.Library
  - Microsoft.Maui platform NuGet libraries:
    - BBDXamarinForms.Droid.Library
    - BBDXamarinForms.iOS.Library
  - Microsoft.Android and Microsoft.iOS binding NuGet libraries:
    - GoodDynamics.Android
    - GoodDynamics.iOS

  The NuGet libraries listed above are distributed as a local .nupkg file source (located in the Platform/nuget subdirectory) with a NuGet.Config settings file in each of the sample apps for iOS and Android. You should also use the .csproj file as a reference for NuGet project dependency configuration, as Microsoft Visual Studio Dependency Manager does not always compile with the actual state.

- Review the sample projects in the BlackBerry Dynamics SDK for Microsoft.Maui 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 Implement 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: The sample apps require the following NuGet packages that can be obtained using Microsoft Visual Studio: NLog, Unity.

Sample apps

The easiest way to get started with the BlackBerry Dynamics Bindings for Microsoft.Android 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.

By default, all of the sample apps use R8 as the obfuscation tool in the build and release process. For more information, see Using an obfuscation tool in your build and release process.

Sample app	Description
Apache HTTP client	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.

Testing and troubleshooting

This section provides guidance for testing and troubleshooting issues with your BlackBerry Dynamics apps.

Troubleshooting common issues

Problem	Possible solution
After upgrading the BlackBerry Dynamics SDK for Microsoft.Maui, you get link errors that prevent you from building your project.	Verify that you upgrade to the latest supported version of the core BlackBerry Dynamics SDK for iOS or Android. See the software requirements.
GenerateLibraryResources or MANIFESTMERGER errors.	Clean the local XamarinBuildDownload cache at /Users/<*user*>/Library/Caches/XamarinBuildDownload.

Disable ARM v8 to avoid potential issues

To avoid potential issues, it is a best practice to disable ARM v8 in Android builds.

1. In **Project Options > Android Build > General**, clear **Use shared Mono runtime**.
2. In **Project Options > Android Build > Advanced > Supported ABIs**, clear **arm64-v8a**.
3. Save the changes.

Deploying your BlackBerry Dynamics app

Before you deploy your BlackBerry Dynamics app to your organization's production environment, you should test the app and the deployment process in a BlackBerry UEM environment that is reserved for development testing and evaluation. Coordinate with your organization's administrator to get access to a dedicated test environment.

BlackBerry Dynamics apps are fully supported for BlackBerry UEM. BlackBerry UEM is the recommended enterprise management solution to implement and use going forward, because it provides advanced app and user management features, advanced connectivity and networking options, expanded compliance and integrity checking, and the most recent BlackBerry Web Services REST APIs that your apps can leverage.

See the following resources for more information about distributing and managing your app in a BlackBerry UEM environment:

Task	Resource
Add your app to BlackBerry UEM and distribute it to users	See the following topics in the *BlackBerry UEM Administration Guide*: • Apps • Add an internal BlackBerry Dynamics app entitlement • Managing BlackBerry Dynamics apps
Configure BlackBerry Dynamics profiles that impact app functionality	See the following topics in the *BlackBerry UEM Administration Guide*: • Controlling BlackBerry Dynamics on users devices • BlackBerry Dynamics profile settings • BlackBerry Dynamics connectivity profile settings • Using profiles to manage device features
Collect activity and compliance violation information for BlackBerry Dynamics apps	See the following topic in the *BlackBerry UEM Administration Guide*: Export a BlackBerry Dynamics app report to a CSV file

Configuring library version compliance

In a compliance profile or compliance policy, an administrator can enable the BlackBerry Dynamics library version verification compliance rule to specify an enforcement action if a BlackBerry Dynamics app is using a version of the BlackBerry Dynamics library that is not permitted. The available enforcement actions are "Do not allow BlackBerry Dynamics apps to run" and "Delete BlackBerry Dynamics app data."

In UEM, by default the BlackBerry Dynamics library version verification compliance rule is not selected and all versions are permitted. An administrator can enable this option and select specific versions to disallow.

Using an obfuscation tool in your build and release process

After your app is fully tested and ready to deploy, it is recommended that you use an obfuscation tool as part of your formal build and release process. It is a best practice to use ProGuard because it is the default obfuscation tool for Android.

For more information about the code obfuscation configuration and sample code that you can use, see the Build-Time Configuration appendix in the BlackBerry Dynamics SDK for Android API Reference.

If you are using the BlackBerry Dynamics Bindings for Microsoft.Android, note that ProGuard is not supported. It is recommended to use R8 as the obfuscation tool. See the Build-Time Configuration appendix in the BlackBerry Dynamics SDK for Microsoft.Maui API Reference instead.

The BlackBerry Dynamics SDK uses Platform APIs, some of which rely on an API level later than the current minimum supported API level. A target SDK below the latest API level might throw warnings. The BlackBerry Dynamics SDK ensures that the appropriate runtime checks are made before it attempts to use an API. You can ignore warnings about APIs that aren't found in BlackBerry Dynamics SDK classes with `-dontwarn com.good.gd`.

Deploying certificates to BlackBerry Dynamics apps

You can use any of the following options to deploy certificates to BlackBerry Dynamics apps. Each method requires configuration in the management console. Coordinate with your organization's administrator to select and configure the desired option.

Option	For more information
Personal Information Exchange files	See Using Personal Information Exchange files in this section.
CA certificate profile	See Sending CA certificates to devices and apps in the *UEM Administration Guide*.
User credential profile	See Sending client certificates to devices and apps using user credential profiles in the *UEM Administration Guide*.
SCEP profile	See Sending client certificates to devices and apps using SCEP in the *UEM Administration Guide*.
Shared certificate profile	See Sending the same client certificate to multiple devices in the *UEM Administration Guide*.

After certificates are distributed to a user's device, those certificates are shared and used by all of the BlackBerry Dynamics apps on the device. No additional programming is required by the app developer to support client certificates.

The management server and BlackBerry Dynamics apps also support the use of Kerberos for service authentication. For more information, see Using Kerberos in this section.

The SDK also provides a Crypto C language programming interface that allows an app to retrieve public key certificates that are stored in the BlackBerry Dynamics credentials store and use those certificates for signing and verification of messages and documents such as PDFs. Note that BlackBerry Infrastructure certificates cannot be retrieved from the store and that the private key will remain inaccessible. For more information, see the Crypto C Programming Interface appendix (Android/iOS) in the API reference.

Using Personal Information Exchange files

An organization can deploy corporate services that require two-way SSL/TLS authentication for users. A user is issued a password-protected Personal Information Exchange file (PKCS12 format, .p12 or .pfx) containing an SSL/TLS client certificate and a private key. This file can be provided to BlackBerry Dynamics apps to grant access to secure corporate services.

The BlackBerry Dynamics SDK supports the use of Personal Information Exchange files to authenticate BlackBerry Dynamics apps and to access secure services. All of the required operations to support client certificates are carried out by the BlackBerry Dynamics Runtime, with no additional programming required to handle the authentication challenge. For more information on how this is handled, refer to *HttpViewController.swift* in the Dynamics-iOS-Swift sample app. The app can use client certificates if:

- The app uses the BlackBerry Dynamics Secure Communication Networking APIs.
- The device user's UEM account is configured to support certificates.
- The certificates satisfy the certificate requirements.

After a user activates a BlackBerry Dynamics app, the app receives the Personal Information Exchange files. For each file, the user is prompted to provide the issued password so that the files and identification material can be installed. When this process is complete, the app can access the server resources that require two-way SSL/TLS authentication.

If more than one Personal Information Exchange file is required per user, the BlackBerry Dynamics Runtime selects the appropriate certificate using the following criteria:

1. Only client certificates that are suitable for SSL/TLS client authentication are eligible to send to the server. Certificates must have no Key Usage or Extended Key Usage, or Key Usage that contains “Digital Signature" or "Key Agreement”, or Extended Key Usage that contains "TLS Web Client Authentication”. Key Usages and Extended Key Usages must not contradict allowances for SSL/TLS client authentication.
2. If the server advertises the client certificate authority in the SSL/TLS handshake, only client certificates that have been issued by that authority are considered.
3. Expired certificates and certificates that are not yet valid cannot be selected.
4. If more than one certificate satisfies the above criteria, the BlackBerry Dynamics Runtime selects the most recently issued certificate.

Configuring support for client certificates

Certificate support is configured in the management console by the administrator. Contact your organization’s administrator to configure certificate support for BlackBerry Dynamics apps.

For more information about configuring certificate support in BlackBerry UEM, see the following:

- Manage settings for a BlackBerry Dynamics app in the *UEM Administration Guide*
- Sending certificates to devices using profiles in the *UEM Administration Guide*
- Connect BlackBerry UEM to a BlackBerry Dynamics PKI Connector in the *UEM Administration Guide*

Certificate requirements

- Client certificates must be in PKCS12 format, with the Certificate Authority (CA), public key, and private key in the same file.
- The PKCS12 file must have a .p12 or .pfx extension
- The PKCS12 file must be password-protected
- The source of the certificate can be your own internal CA, a well-known public CA, or an online tool such as OpenSSL or the Java keytool. You can use the following keytool example to generate a certificate, substituting your own values as required:

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

- If the organization’s security policy uses FIPS standards, Personal Information Exchange files must be encrypted with FIPS-strength ciphers. If Personal Information Exchange files use a weak cipher, which is common for third-party applications when exporting identity material, you can use a tool like OpenSSL to re-encrypt the files with a FIPS-strength cipher. See the following example:

```
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
```

Using Kerberos

BlackBerry Dynamics apps support both Kerberos PKINIT with PKI certificates and Kerberos Constrained Delegation. Kerberos PKINIT and Kerberos Constrained Delegation are distinct implementations of Kerberos. You can support one or the other for BlackBerry Dynamics apps, but not both.

With Kerberos PKINIT, authentication occurs directly between the BlackBerry Dynamics app and the Windows Key Distribution Center (KDC). User authentication is based on certificates that are issued by Microsoft Active Directory Certificate Services. No additional programming is required by the app developer to use Kerberos PKINIT.

With Kerberos Constrained Delegation, authentication is based on a trust relationship between the management server (BlackBerry UEM and a KDC. The management server communicates with the service on behalf of the app.

For more information about how to configure the desired Kerberos implementation in UEM, including requirements and prerequisites, see Configuring Kerberos for BlackBerry Dynamics apps in the *UEM Administration Guide*.

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 .https://www.blackberry.com/us/en/legal/third-party-software

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