BlackBerry Dynamics SDK for Android

Development Guide

7.1

Contents

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 or Good Control 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.

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 or standalone Good Control 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 access key provided by the administrator, by using the UEM Client, 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 Guide*.

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 using an access key or the UEM Client; 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 or standalone Good Control 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*.

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 UEM or standalone Good Control management console.

If you want your custom BlackBerry Dynamics app to use the Shared Services Framework, the UEM or standalone Good Control 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 or standalone Good Control administrator can use Data Leakage Prevention (DLP) settings in BlackBerry Dynamics profiles (UEM) or security policies (Good Control) 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.

The BlackBerry Dynamics SDK for Android provides the following classes to manage the secure copy, cut, and paste of data: ClipboardManager, GDTextView, GDEditText, GDAutoCompleteTextView, and GDSearchView. The secure copy-cut-paste sample app demonstrates uses of the secure clipboard.

User authentication

BlackBerry UEM and standalone Good Control offer 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 or standalone Good Control administrator can use a BlackBerry Dynamics profile (UEM) or a security policy (Good Control) 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 or standalone Good Control 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 or standalone Good Control 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 (UEM) or a security profile (Good Control). 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 (UEM) or security policy (Good Control), 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 and Good Control 3.0.50.70 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) or by using an app policy (Good Control), this setting allows an app to completely bypass the password login screen.

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

Administrative controls

The BlackBerry UEM or standalone Good Control 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*. If your organization uses standalone Good Control, see the Good Control Online Help.

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

The BlackBerry Dynamics SDK integrates the CylancePROTECT library to support CylancePROTECT for BlackBerry UEM. CylancePROTECT is a licensed service that offers a suite of features that enhances BlackBerry UEM's ability to detect, prevent, and resolve security threats without disrupting the productivity of your workforce. CylancePROTECT is configured and managed by the BlackBerry 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 apps (including BlackBerry Dynamics apps) that are uploaded to UEM for internal deployment
- Malware detection on Android devices
- Sideloaded app detection on iOS devices
- Safe browsing with BlackBerry Dynamics apps
- 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 documentation.

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

Android development

Item	Requirement
Compatibility with previous versions of the SDK	The latest release of the BlackBerry Dynamics SDK for Android is compatible with these previous releases of the SDK: • 7.0.x • 6.1.x It is recommended that you always build and test with the most recent release of the SDK to take advantage of new fixes and features.
Supported Android OS	Android 7.0 or later
BlackBerry Dynamics Handheld Library	Minimum API level: 24
BlackBerry Dynamics Wearable Library	Minimum API level: 24
Supported CPU architectures	• ARMv7 • ARMv8 • x86 • x86_64
Android Wear dependencies	• Google Play Services 11.0.1 • Wear OS Emulator API level 24

Suggested versions of platform and tools	• Android Studio 3 or later
	• The following values specified in sdk/libs/handheld/gd/build.gradle. Other versions of tools will work, but the BlackBerry Dynamics library gradle files might need to be updated accordingly.
	  • com.android.tools.build:gradle: 3.1.4
	  • compileSdkVersion 29
	  • buildToolsVersion "29.0.0"

Character encoding for build files	Build files (for example, settings.json) must use UTF-8 character encoding. Verify that the editor that you plan to use does not add non-UTF-8 characters or headers. In general, Java does not work with UTF-8-BOM (byte order mark).

Item	Requirement
Supported launch modes	Apps built with the BlackBerry Dynamics SDK for Android support the following launch modes in AndroidManifest.xml: • android:launchMode="standard" • android:launchMode="SingleTop" • android:launchMode="singleTask"
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.
Restricted key prefix	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 BlackBerry UEM or standalone Good Control 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 or standalone Good Control 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*, or the Good Control Help.

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 or standalone Good Control administrator enables FIPS compliance using a BlackBerry Dynamics profile (UEM) or security policy (Good Control). 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.

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
```

For information about logs, see Logging and diagnostics.

Note: FIPS compliance is not supported on x86 emulators. If you want to test on x86 emulators, you must disable FIPS compliance.

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 `abiFilters` 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 `abiFilters` in the build.gradle file. For example, to include only ARMv7 libraries in your project:

```
defaultConfig {
   ndk {
          abiFilters "armeabi-v7a"
    }
}
```

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 supports the following TLS protocols and cipher suites. Note that SSL version 3 or older is no longer supported. A BlackBerry Dynamics app that is upgraded to SDK version 7.0 or later might stop connecting to SSL application servers if the app uses weak ciphers.

Supported TLS version 1.2 cipher suites:

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
- TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 (0x00a3)
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
- TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
- TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 (0x00a2)
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
- TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x006b)
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x0038)
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x0067)
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 (0x0040)
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
- TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)
- TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
- TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)

- TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
- TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)

Supported TLS version 1.1 cipher suites:

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x0038)
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
- TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
- TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
- TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)

Supported TLS version 1.0 cipher suites:

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
- TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
- TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x0038)
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
- TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)

Steps to get started with the BlackBerry Dynamics SDK

Step	Action
1	Install the Android SDK with Android Studio.
2	Download and install the BlackBerry Dynamics SDK. You can use the Android SDK manager or install the SDK manually. Review the contents of the SDK.
3	Optionally, integrate the SDK in .aar format that can be published to an internal repository.
4	Consult the 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. For additional guidance and code examples, see the Getting started workflow for BlackBerry developers.
5	See the following sections if you want to use any of these optional features: • Adding a custom logo and colors with the branding API • Integrating BlackBerry Enterprise Mobility Server services • Creating wearable apps for Wear OS devices • Implementing SafetyNet attestation for BlackBerry Dynamics apps
6	Test and debug your app. The SDK allows you to test in enterprise simulation mode and supports automated testing.
7	Review configuration and build recommendations and then deploy your app.
8	Optionally, deploy certificates to the BlackBerry Dynamics apps on users' devices.

Install the Android SDK with Android Studio

You must install the Android SDK before you install the BlackBerry Dynamics SDK. Visit https://developer.android.com/studio/index.html to download and setup the Android SDK.

Verify that your installation is properly configured by writing and running a small "Hello, World!" program that is not based on the BlackBerry Dynamics SDK.

Install the BlackBerry Dynamics SDK with the Android SDK Manager

Before you begin: Visit BlackBerry Developer Downloads to download the SDK package. 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.

Start the Android SDK Manager and follow the steps detailed in Getting Started with the BlackBerry Dynamics SDK for Android on the BlackBerry Developer website.

Note: In the BlackBerry Dynamics SDK version 4.0 and later, there is a change to the extracted folder structure in the extras/blackberry/dynamics_sdk folder. At this level there is now an m2repository/ folder for .aar distribution and an sdk/ folder that contains the original .jar distribution and resources. If you are updating the SDK to version 4.0 or later, you must edit your settings.gradle files as necessary to account for the extra sdk/ folder at this level.

Manually install the BlackBerry Dynamics SDK for Android

Before you begin: Visit BlackBerry Developer Downloads to download the SDK package. 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.

1. Open your Android home directory in a file manager application or change to the directory in a terminal window.
2. From the sdk/ directory, navigate to the extras/ sub-directory. If there isn't a directory named blackberry/ at this location, create it using the file manager or by running the `mkdir` command.
3. Change to the blackberry/ sub-directory.
4. Copy the .zip file for the SDK to the blackberry/ directory and extract the files.

   This creates a directory structure with a directory named sdk/ as its root.
5. Rename the new sdk/ directory to dynamics_sdk/ using the file manager or by running the `mv` or `ren` command.

Contents of the BlackBerry Dynamics SDK for Android

The BlackBerry Dynamics SDK for Android includes APIs to support:

- The development of BlackBerry Dynamics apps for handheld devices
- The development of wearable apps for Wear OS devices
- Android backup capabilities
- SafetyNet attestation
- Automated testing

All libraries are distributed as both a maven repository that can be added to an existing repository and as library projects that can be added to an app project.

Integrating the BlackBerry Dynamics SDK in .aar format

The BlackBerry Dynamics SDK libraries are available in the SDK package as .aar files that can be published to an internal repository. The files are found in m2repository/com/blackberry/blackberrydynamics.

Use one of the following methods to incorporate the .aar files into your BlackBerry Dynamics app projects. The appropriate method depends on how you installed the SDK.

Using Maven if you installed the SDK using the Android SDK Manager

If you installed the SDK using the Android SDK Manager, the .aar files are added to the following path: ANDROID_HOME/extras/BlackBerry/dynamics_sdk/m2repository.

1. Define the path to the .aar files in the build.gradle repositories block.

   Example: Hard-coded path

   ```
   allprojects {
       repositories {
           maven { url android.sdkDirectory.path+'/extras/blackberry/dynamics_sdk/
   m2repository' }
           //other maven URLs ...
       }
   }
   ```

   Example: Path from local properties

   ```
   allprojects {
       repositories {
           def localProperties = new File(rootDir, "local.properties")
           Properties properties = new Properties()
           localProperties.withInputStream { instr ->
               properties.load(instr)
           }
           def sdkDir = properties.getProperty('sdk.dir')
           maven { url sdkDir+'/extras/blackberry/dynamics_sdk/m2repository' }
           //other maven URLs ...
       }
   }
   ```

2. Add Gradle compile dependencies in the format *<group_ID>:<artifact_ID>:<version>*. For example:

   ```
   dependencies {
       api 'com.blackberry.blackberrydynamics:android_handheld_platform:5.0.0.47'
   }
   ```

   The following artifacts are included in the SDK:

   - `android_handheld_platform`
   - `android_handheld_wearable_support`
   - `android_wearable_platform`
   - `android_handheld_backup_support`
   - `android_handheld_gd_safetynet`
   - `atsl`

Add .aar files if you installed the SDK manually

If you installed the BlackBerry Dynamics SDK manually, note the location of the m2repository directory and complete the following steps:

1. Define the path to the .aar files in the build.gradle repositories block. For example:

   ```
   allprojects {
       repositories {
   ```

```
        maven { url '/path/to/m2repository' }
        //other maven URLs ...
    }
}
```

Note: You can define the URL in the local.properties file to avoid VCS conflicts.

2. Add Gradle compile dependencies in the format *<group_ID>:<artifact_ID>:<version>*. For example:

```
dependencies {
    api 'com.blackberry.blackberrydynamics:android_handheld_platform:5.0.0.47'
}
```

The following artifacts are included in the SDK:

- `android_handheld_platform`
- `android_handheld_wearable_support`
- `android_wearable_platform`
- `android_handheld_backup_support`
- `android_handheld_gd_safetynet`
- `atsl`

Add .aar files as plain resources

1. Add the .aar files to an appropriate folder (for example, libs).
2. Define the path to the folder in the build.gradle file. For example:

```
allprojects {
    repositories {
        flatDir {
            dirs '<path_to_libs_folder>'
        }
        //other maven URLs ...
    }
}
```

3. Add the Gradle dependencies. For example:

```
compile ( name : 'android_handheld_platform-5.0.0.47' , ext : 'aar' )
```

Integrating optional features

This section provides more information about integrating optional features into your custom BlackBerry Dynamics apps.

Enforcing local compliance actions

The BlackBerry Dynamics SDK includes the following APIs that you can use to block or unblock a user's access to the UI of a BlackBerry Dynamics app locally:

- GDAndroid.executeBlock
- GDAndroid.executeUnblock

You can use these APIs to temporarily prevent access to an app under certain conditions. For example, if the user accesses a public Wi-Fi network that is not trusted, you can use GDAndroid.executeBlock to prevent access to the app until the user is on a trusted Wi-Fi network. While the app UI is blocked, the app's network activity and container storage access is not affected.

You can use GDAndroid.executeBlock to display a message to the user that explains why access to the app has been blocked and how the user can restore compliance and unblock the UI.

The GDInteraction sample app has been updated to demonstrate the use of these APIs.

Note: It is possible to circumvent a UI block if the user is able to restore a backup that was created before the block occurred. Take this condition into account when you develop and test your app.

Adding custom policies for your app to the UEM management console

You can add new management policies and settings that are specific to your custom BlackBerry Dynamics app to the BlackBerry UEM management console so that they can be configured and applied to users by UEM administrators. The new management policies that you define are in addition to the full suite of BlackBerry Dynamics and device management policies already offered by UEM.

You can add a custom policy for a BlackBerry Dynamics app by creating and uploading an application policy definition file to the management console. For implementation details and more information about the file format and elements, see the Application Policies Definition appendix in the API Reference (Android/iOS) and the BlackBerry Dynamics App Policies Technical Brief.

For instructions for uploading the file to the management console, see Manage settings for a BlackBerry Dynamics app in the *UEM Administration Guide*.

Adding a custom logo and colors with the branding API

You can use the branding API to add a custom logo and colors to the app UI. For more information, see void configureUI in the API Reference.

Support for Night Mode

The BlackBerry Dynamics SDK for Android version 6.1 and later supports Night Mode. The Greetings server and Application policy sample apps in the SDK package demonstrate how to support this feature. The other sample apps use light mode.

You can turn off support for Night Mode in the BlackBerry Dynamics UI by adding the following code to the application class:

```
import android.support.v7.app.AppCompatDelegate;
...
    AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

This can be useful for apps that do not support Night Mode, to ensure a consistent UI experience (all screens will use light mode). Most of the sample apps in the SDK package include this code for reference.

Note: The internal style and theme definitions included with the BlackBerry Dynamics SDK have changed. If your app was previously making use of any of these definitions, you may get errors when building your app, and may have to rework your UI to avoid some dependencies. It is a best practice that you do not rely on BlackBerry Dynamics styles when coding your UI. Please contact BlackBerry support if your app relied on a particular style and you need to know how it was defined.

Using zero sign-on for SaaS services through BlackBerry Enterprise Identity

Your custom BlackBerry Dynamics apps can leverage zero sign-on (ZSO) for SaaS through BlackBerry Enterprise Identity. For more information, see the BlackBerry Enterprise Identity docs.

Integrating BlackBerry Enterprise Mobility Server services

This section covers the general approach for programming with the BlackBerry Dynamics SDK and the BlackBerry Enterprise Mobility Server services. The approach consists of two parts:

- Programming an app to interact with the desired BEMS services
- Entitling users to the necessary applications

BEMS services conform to the Shared Services Framework. A service consists of two applications: A program that provides the service, and an app that consumes the service. BEMS is the service provider that must be configured for use in BlackBerry UEM or in standalone Good Control. You create the app that consumes this service.

BEMS services APIs

The BEMS services are described in the BEMS API Reference Guides.

Programming your service consumer app

You must define a unique BlackBerry Dynamics app ID for your application (for complete details, see Using an entitlement ID and version to uniquely identify a BlackBerry Dynamics app). The BlackBerry Dynamics SDK has functions to discover services, and each BEMS service has specific programming interfaces.

To discover the BEMS services, use `GDServiceType`. This API and other APIs for shared services are described in other sections of this guide and in the BlackBerry Dynamics API reference.

After your consumer app discovers the service, the way the app communicates with the service depends on the service definition.

Note: Most BEMS services run over SSL (HTTPS) on port 8443. Be sure your consumer application connects to the correct server and port.

Discovering the BlackBerry Enterprise Mobility Server services

Described here is a general approach to using the BlackBerry Dynamics SDK and Server-based Services Framework to programmatically discover the Docs services offered by your BEMS installation.

Item	Description
Service identifier	First you need to know the service identifier and version. For more information about the available services, see Mobile Services.
Service discovery	Next, code a service discovery query in your application program. See the `getServiceProvidersFor` API in the `GDAndroid`, `GDiOS`, and `GDMac` classes.
Server cluster	The result of the service discovery query is an array of GDServiceProvider objects. Each object corresponds to a BlackBerry Dynamics entitlement ID that is registered as a provider of the service. Your best result is that the array has one element. If the array is empty, it means that the current end user isn't entitled to any App ID that provides the service. In that case, your app shouldn't use the service. If the array has more than one element, it means that the end user is entitled to more than one GD App ID that provides the service (likely a configuration error by the enterprise). Your app would have to pick one of the GD App IDs, or try all of them, or prompt the user to select. In the `GDServiceProvider` object, there is a `serverCluster` attribute. It contains an array of `GDAppServer` objects, each of which tells you the address and port number of a server, and the priority of that instance within the cluster.

Item	Description
Server selection	If the `serverCluster` array has only one element, then server selection is trivial. Use the server address and port number of the first element. If the `serverCluster` array is empty, that indicates an enterprise configuration error. If the `serverCluster` array has more than one element, then you must implement a server selection algorithm. A sample algorithm is given on the `GDAndroid`, `GDiOS`, and `GDMac` pages in the BlackBerry Dynamics API reference, in the `getApplicationConfig` section. The algorithm is the same for the BlackBerry Dynamics SDK for Android and for the BlackBerry Dynamics SDK for iOS. The recommended selection algorithm is as follows. For each priority value in the list, starting with the highest: • Select a server that has that priority, at random. • Attempt to connect to the server. • If the connection succeeds, use that server. • If the connection fails, try another server at the same priority, at random. • If there are no more untried servers at that priority, try the servers at the next lower priority.

Creating wearable apps for Wear OS devices

You can use the BlackBerry Dynamics Wearable Framework that is packaged with the SDK to create secure wearable solutions for Wear OS devices. You can leverage the following features and functionality for your custom wearable apps:

- User authentication
- Secure data storage
- Secure data exchange with BlackBerry Dynamics apps on handheld devices
- Secure command exchange with BlackBerry Dynamics apps on handheld devices (for example, to display an alert on the wearable device)

A wearable app is a component of a BlackBerry Dynamics app that is installed on a handheld device. Once the handheld app is installed, the wearable app can be installed on a paired Wear OS device. The wearable app can only activate and authenticate the user through the handheld BlackBerry Dynamics app.

For more information about the BlackBerry Dynamics Wearable Framework, including platform requirements and programming and activation details, see the BlackBerry Dynamics Wearable Framework appendix in the API reference.

Your organization's administrator can access settings in the BlackBerry UEM or standalone Good Control management console to permit or disallow the activation of wearable apps, to configure a timeout that locks a wearable app after it is disconnected from a handheld device for a certain amount of time, and to configure authentication options.

The BlackBerry Dynamics SDK includes a wearable framework secure store sample app that you can use as a starting point. The sample app also demonstrates the necessary build configuration.

For more information about developing with the BlackBerry Dynamics Wearable Framework, you can watch any of the following video tutorials:

- Introducing the Wearable Framework
- Wearable Framework Demo
- Wearable Framework SDK Structure
- Building and Debugging with Android Studio
- Wearable APIs
- Building your own Wearable app

Implementing SafetyNet attestation for BlackBerry Dynamics apps

BlackBerry UEM version 12.10 and later supports SafetyNet attestation for BlackBerry Dynamics apps. You can use SafetyNet to extend BlackBerry root and exploit detection and to enhance app security and integrity. For more information about SafetyNet attestation, implementation considerations, and instructions for enabling the feature, see the BlackBerry UEM Configuration Guide. This chapter details considerations for developers who want to enable SafetyNet support for their BlackBerry Dynamics apps.

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

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 SafetyNet, add this library to the app project dependencies along with the main GDLibrary.

The GDSafetyNet library includes all of the client-side source code that is required to support SafetyNet. No additional app code is required. The GDSafetyNet library requires Google Play Services 11.0 or later to use device SafetyNet APIs. Verify that your BlackBerry Dynamics app is dependent on only a single version of Google Play Services.

For example, you can add the following to the app's build.gradle file if the app uses play-services modules:

```
api 'com.google.android.gms:play-services-safetynet:16.0.0'
api ('com.blackberry.blackberrydynamics:android_handheld_gd_safetynet:
$DYNAMICS_SDK_VERSION') {
          transitive = false
}
```

Completing SafetyNet registration

You must obtain an API key from Google and add it to the app's AndroidManifest.xml file. The API key has a quota limit of 10,000 requests each day. If necessary, you can use this online form to request an increase to this quota.

For the following sections of the form, select these answers:

- Method of validating SafetyNet Attestation API responses: Server side - by verifying the signing certificate chain.
- Method of retrying in case of errors: Something else (or unknown). In the "Anything else that you want to tell us" field, type the following: If the SafetyNet Service responds to the device with a transient error, then the device will retry with an exponential back-off. If the SafetyNet Service responds with a valid blob, but the blob is found to contain an error value when it is decoded, then attestation is retried after 15 minutes.

In the AndroidManifest.xml file, add the following to the <application> element:

```
<meta-data android:name="com.blackberry.attestation.ApiKey"
 android:value="YOUR_API_KEY" />
```

Updating the BlackBerry Dynamics application policy file

During a SafetyNet 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.

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>SafetyNet 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>Safety Net</label>
            <value>Safety Net</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). 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.

Using the BlackBerry Web Services REST APIs for SafetyNet attestation and status

The BlackBerry Web Services for BlackBerry UEM are a collection of REST APIs that you can use to execute administrative actions in BlackBerry UEM or to retrieve status information about UEM users, groups, devices, and the overall UEM domain. The BlackBerry Web Services version 12.10 and later provide REST APIs that you can use to both initiate and check the status of SafetyNet attestation.

For an introduction to the BlackBerry Web Services REST APIs, see the Getting started section in the REST API reference.

You can use the following APIs to initiate attestation for a specific BlackBerry Dynamics app or for a user's device:

- POST /{tenantGuid}/api/v1/users/{userGuid}/userDevices/{userDeviceGuid}/applications/{appGuid}/commands
- POST /{tenantGuid}/api/v1/users/{userGuid}/userDevices/{userDeviceGuid}/commands

You can use the following APIs to retrieve the attestation status (as well as other status information) for all of a user's devices, for a specific device, or for all of the apps on a specific device:

- GET /{tenantGuid}/api/v1/users/{userGuid}/userDevices
- GET /{tenantGuid}/api/v1/users/{userGuid}/userDevices/{userDeviceGuid}
- GET /{tenantGuid}/api/v1/users/{userGuid}/userDevices/{userDeviceGuid}/applications

The returned status information provides the time that an attestation result was last reported. You can establish a trust window in which an attestation call is not required (for example, four hours). If you do use a REST API for an ATTEST call, you must get the attestation status later (asynchronously), as there is no notification that UEM has completed the attestation process.

You can use the REST APIs for various use cases. For example, if you have a server application that is used by your organization's internal mobile apps, you could have the server application use the REST APIs noted above

to check the app's attestation status (or to initiate an attestation challenge) before releasing data to the app or accepting data from it.

Note the following about the communication channels for the REST APIs:

- The REST APIs that attest the device or get status information for the device communicate with the BlackBerry UEM Client over a direct device channel that doesn't require user authentication.
- The REST APIs that attest a specific BlackBerry Dynamics app remain pending on the UEM server until the app is running on the device. When the app starts and is authenticated, it connects to UEM and the attestation challenge occurs.

Testing the app

After completing the integration tasks, your app is SafetyNet ready and you can proceed with further testing. Verify that the app displays in the list of SafetyNet 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 SafetyNet capable apps, the administrator can then enable the app for SafetyNet. For instructions, see the BlackBerry UEM Configuration Guide. An app that was signed correctly will activate successfully. An app that was not signed correctly will not activate and a SafetyNet validation failure message will display in the app.

Sample apps

You can use the sample apps available in the BlackBerry Dynamics SDK for Android to help you plan the development of your apps. The source code samples demonstrate how to apply BlackBerry Dynamics functionality. The most basic is the skeleton app, which can be used as a starting point for implementing your own projects with BlackBerry Dynamics using Enterprise Simulation Mode.

The samples are located in *<sdk_install_directory>*/samples/, or you can download them from the BlackBerry Developers for Enterprise Apps site.

All sample apps have Android Gradle configurations. You can load any of the apps in Android Studio. When you troubleshoot a BlackBerry Dynamics app, you should enable detailed logging from theconsole or build configuration.

Note: It is recommended to make a copy of the source code in another location before making any changes. Reinstalling or upgrading will overwrite or even remove the sample apps in the default location.

Sample app	Description
Apache HTTP client	Provides an example of how to use BlackBerry Dynamics Secure Communication APIs to access resources behind the enterprise firewall. These secure communication APIs can be used to exchange data between the mobile app on the device and an application server using the secure BlackBerry Dynamics proxy infrastructure. In 4.0.0 and later, this sample supports HTTP PATCH requests.
AppBasedCertImport	Demonstrates how to create an app that can import a user's PKI credentials using the BlackBerry Dynamics Certificate Credential Import API. For more information, see Package com.good.gd.pki and Creating user credential profiles for app-based certificates in the *UEM Administration Guide*.
AppKinetics Shared Services	Provides three examples of how to use the BlackBerry Dynamics Shared Services Framework.
App policy	Provides an example of how to use application-specific policy APIs. App policies control specific features of a single app, compared to built-in policies that apply to all apps. This app also demonstrates how to support Night Mode.
Bypass Unlock	Illustrates how to program for the Bypass Unlock feature. For more information about Bypass Unlock, see User authentication.
Interaction	Provides an example of how to interact with the BlackBerry Dynamics Library and use the Remote Settings API. Displays which BlackBerry Dynamics events are happening. This can be helpful when using BlackBerry Dynamics for the first time. This app also demonstrates the use of the GDAndroid.executeBlock and GDAndroid.executeUnblock APIs that can be used to locally block or unblock a user's access to the UI of a BlackBerry Dynamics app.

Sample app	Description
Greetings client/server	Demonstrates how to write a client and server that use the BlackBerry Dynamics Inter Container Communications (ICC) API. The ICC system exchanges data securely between two BlackBerry Dynamics applications running on the same mobile device so that data is not compromised during the exchange. The Greetings server app demonstrates how to support Night Mode.
Push channel	Demonstrates how to use the BlackBerry Dynamics Push infrastructure, including 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 used to receive real-time notifications from an application server.
Secure copy-cut-paste	Compares the use of secured BlackBerry Dynamics UI text controls (GDTextView, GDEditText, GDAutoCompleteTextView, GDSearchView, GDWebView) and the corresponding default UI text controls. Text data is encrypted or decrypted before copy or paste operations are performed. For more information, see Data Leakage Prevention.
Secure SQL	Demonstrates how to use the secure SQL database, showing how to add, edit, delete, and list contacts. All data stored in the secure SQL database is encrypted on the device by the BlackBerry Dynamics Runtime.
Skeleton	Provides a basic skeleton application that you can use as a starting point for application development.
Wearable framework 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. The secure store app consists of three sub-projects, and is best loaded into Android Studio IDE: • /handheld_app: Code specific to a BlackBerry Dynamics handheld application. • /common: Code built into both the BlackBerry Dynamics handheld and BlackBerry Dynamics wearable applications. • /wearable_app: Code specific to a BlackBerry Dynamics wearable application. **Note:** Importing this sample app into Android Studio causes an error message that can be ignored. The application builds correctly despite the error.

Testing and troubleshooting

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

Implementing automated testing for BlackBerry Dynamics apps

The BlackBerry Dynamics SDK includes the BlackBerry Dynamics Automated Test Support Library (ATSL) to support automated testing for your BlackBerry Dynamics apps. The library is delivered as binary libraries: a Java library (.jar) and an Android library (.aar).

The library includes helper functions for testing common user interactions in BlackBerry Dynamics apps, such as activation and authorization. The configuration and structure of the library is compatible with the native Android Testing Support Library. It makes use of the following components:

- com.android.support.test:rules
- com.android.support.test.uiautomator:uiautomator-v18

For more information about these components and the Android Testing Support Library, see Android Studio: Test your app.

You can use the BlackBerry Dynamics library, the native library components mentioned above, and Gradle and JUnit tools to automate the building, execution, and reporting of your application tests.

Since the BlackBerry Dynamics ATSL is delivered as binary libraries, you cannot make your own changes to it. If you want to review the implementation and customize it, you can see the source in GitHub at https://github.com/blackberry/BlackBerry-Dynamics-Android-Samples/tree/master/AutomatedTestSupportLibrary.

The Java library for the BlackBerry Dynamics ATSL is located in the sub-directory dynamics_sdk/libs/common/atsl. The Android library for the ATSL is located in the sub-directory dynamics_sdk/m2repository/com/blackberry/blackberrydynamics/atsl. If you use the Android SDK Manager to install the BlackBerry Dynamics SDK, the libraries are located under the Android home directory at ANDROID_HOME/extras/good/dynamics_sdk/.

Automated testing with the BlackBerry Dynamics sample apps

The sample apps included in the BlackBerry Dynamics SDK have been integrated with the BlackBerry Dynamics Automated Test Support Library (ATSL). Each sample app includes test code that executes automated tests of processing and behavior.

You can use the integration of the ATSL in any sample app as a guide for integrating the library with your own BlackBerry Dynamics apps. Note the following details about the ATSL integration:

Item	Description
Build target for testing	Each app has an `androidTest` build target that the Android Plugin for Gradle can use. For more information about the Android Plugin for Gradle, see Android Studio: Configure your build.
Code for JUnit tests	The test code is located in the sub-directory *<sample_app>*/tests/*<sample_app_package>*/test/. The code is based on JUnit4. You can run tests with `AndroidJUnitRunner`.

Item	Description

Use of ATSL for interaction with BlackBerry Dynamics screens	The test code uses helper functions in the ATSL. For example, the following code executes the entire activation process for the app:
	`BBDActivationHelper.loginOrActivateApp()`
	The user's email, access key, and password are read from a file in JSON format (see Preparing for automated testing).
	You can run ATSL helper functions in the JUnit assertion programming interface. For example:
	`assertTrue("Failed to provision AppKinetics sample app.", BBDActivationHelper.loginOrActivateApp());`
	This assertion will fail if activation fails.
	Each BlackBerry Dynamics screen is represented by an object that provides convenient methods to interact with the UI elements that it consists of.
	You can use the following snippet to enter user credentials and start the activation process using credentials that are not stored in the JSON file:
	`BBDActivationUI activationScreen = new BBDActivationUI(uiAutomatorUtils.getAppPackageName()); activationScreen.enterUserLogin("email@example.com"); activationScreen.enterKey("12345", "12345", "12345"); activationScreen.clickOK();`

Use of ATSL for general interactions	The test code also uses helper functions in the ATSL for general interactions (for example, checking that an item exists in the user interface and then interacting with that item). See the following example: `uiAutomatorUtils.isTextShown("Retrieving data");` The ATSL functions handle this by using UIAutomator.
Broadcast receiver for authorization events	The test code registers a broadcast receiver that receives authorization events from the BlackBerry Dynamics SDK: `GDAndroid.getInstance().registerReceiver (GDSDKStateReceiver.getInstance(), GDSDKStateReceiver.getInstance().getIntentFilter());` Broadcast receiver registration supports multiple receiver classes and doesn't block `GDStateListener` or `GDAppEventListener` implementations, if any. The broadcast receiver code is in the `GDSDKStateReceiver` class.

Preparing for automated testing

As part of your automated testing you must activate a new installation of your BlackBerry Dynamics app or log in to an already activated app. This requires a user identifier (typically an email address), an access key, and a password. The ATSL can read activation or login credentials from a file in JSON format stored in the assets folder of the app running the tests.

The file does not have any naming restrictions and must contain a single object with the following fields:

- GD_TEST_PROVISION_EMAIL: String that specifies the user's email address
- GD_TEST_PROVISION_ACCESS_KEY: String that specifies the access key
- GD_TEST_PROVISION_PASSWORD: String that specifies the user's password after provisioning or during login
- GD_TEST_UNLOCK_KEY: Specifies the unlock key, if necessary (optional)

For example:

```
{
      "GD_TEST_PROVISION_EMAIL": "user@acme.com",
      "GD_TEST_PROVISION_ ACCESS_KEY": "012345678901234",
      "GD_TEST_PROVISION_PASSWORD": "abcd"
}
```

There are different options to prepare the file:

- Manually generate an access key for a user in the UEM or standalone Good Control management console, then manually edit the file.
- Manually generate and store a number of access keys in advance, then automatically edit the file and consume a key from the store.
- Automatically generate an access key, and optionally a user, with the BlackBerry Web Services APIs, then automatically edit the file.

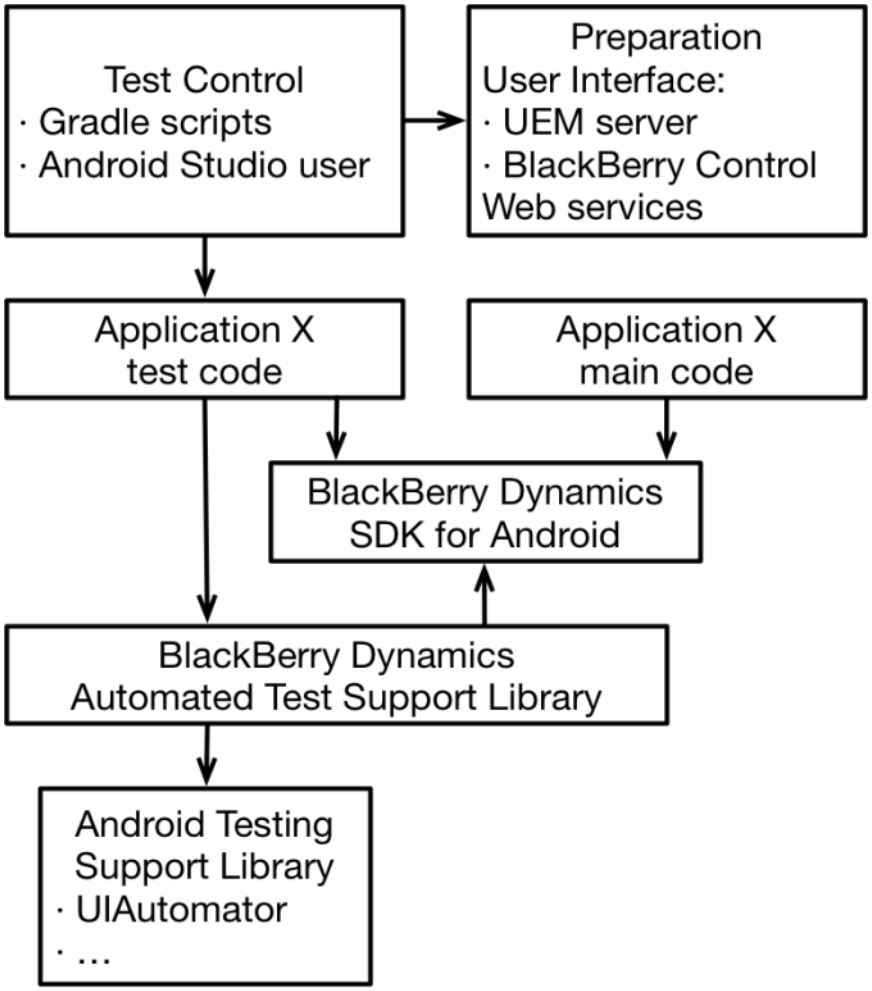
Use the option that best suits your needs, based on your access to the management console and administrator level permissions, your familiarity with the BlackBerry Web Services APIs, and your preference for using a new activation for every test or running subsequent tests on the same app as an upgrade.

Please note the following:

- In the JSON file, the access key and unlock key have to be 15 characters and cannot have dashes.
- Automated testing works in Enterprise Simulation mode, but will have the same restrictions (for example, the app cannot connect to any back-end servers).
- Every BlackBerry Dynamics app needs a one-time initialization to complete activation. To ensure that this is executed first, define a dedicated test method that executes BlackBerry Dynamics activation and have that activation test executed in isolation.

Components of a sample automated testing configuration

The following diagram illustrates the different components in a sample automated testing configuration:



Execute all tests from the command line with Gradle

Follow these instructions to build the application, resolve dependencies, run the tests on a connected Android device, and record the results. This command can be run from a continuous integration system such as Jenkins.

1. Change the directory to samples/<*app_name*>.
2. Run the following command: `./gradlew connectedAppDebugAndroidTest`

Test results are written to the following sub-directory: <*app_name*>/build/reports/androidTests/connected/flavors/APP.

You can save test results using a continuous integration system. You can also use a JUnit plug-in to display the test results in a readable form, to send email messages to project members, or to take other common continuous integration actions.

For more information about running tests using the command line, see Android Studio: Test from the command line.

Execute specific tests from the command line with Gradle

Follow these instructions to build the application, resolve dependencies, run the tests in the specified class on a connected Android device, and record the results.

1. Change the directory to samples/<*app_name*>.
2. Run the following command: `./gradlew connectedAppTestDebugAndroidTest -Pandroid.testInstrumentationRunnerArguments.class=<className>`

   Replace <*classname*> with the name of your class.

You can also run a specific test manually using the following method:

1. Use Gradle to build the `androidTest` target.
2. Use the Android Debug Bridge (ADB) tool to install the app from the resulting .apk file. You can choose to uninstall any existing app first or to install as an upgrade.
3. Execute specific test suites or individual tests using the ADB tool, with a command such as: `adb shell am instrument -w -e debug false -e class <packageName>.<className> <packageName>.test/testRunner`

   Replace <*packagename*> and <*classname*> with values for your app and class.

Test results are written to the following sub-directory: <*app_name*>/build/reports/androidTests/connected/flavors/APPTEST.

You can save test results using a continuous integration system. You can also use a JUnit plug-in to display the test results in a readable form, to send email messages to project members, or to take other common continuous integration actions.

For more information about running tests using the command line, see Android Studio: Test from the command line.

Execute tests from the Android Studio IDE

The tests that you can run from the command line (Execute all tests from the command line with Gradle and Execute specific tests from the command line with Gradle) can also be run from within the IDE. This can be useful for investigating test failures with breakpoints or for running tests as you write the code.

1. In **Run > Edit Configurations**, add a new test configuration.
2. Set the following parameters for the test configuration:
   - Module: <*app_name*>
   - Instrumentation runner: android.support.test.runner.AndroidJUnitRunner
3. Save the configuration.
4. Select the new configuration and Run or Debug.

The tests will run and generate results. The IDE will show progress and results. If Debug was selected, you can insert breakpoints in the code.

Add automated testing to your BlackBerry Dynamics Android app

1. Add the Android Testing Support Library and the BlackBerry Dynamics ATSL to your project. Your project must have a compile SDK level of 23 or higher. You can add the following to the build.gradle file to add both libraries:

```
// Dependencies on the modules of the Android Testing Support Library.
androidTestImplementation 'com.blackberry.blackberrydynamics:atsl:6.0.0.26'
implementation('com.android.support.test.espresso:espresso-contrib:2.2.2') {
    exclude module: 'support-annotations'
}
androidTestImplementation 'com.android.support:support-annotations:24.2.1'
androidTestImplementation 'com.android.support.test:rules:0.4'
androidTestImplementation 'com.android.support.test:runner:0.4'
androidTestImplementation 'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
```

```
// Include tests in androidTest target
androidTest {
    manifest.srcFile 'tests/AndroidManifest.xml'
    java.srcDirs = ['tests']
    assets.srcDirs = ['tests/assets']
}
// Following configuration is required to set the download location of
// dependent projects.
allprojects {
    repositories {
        mavenCentral()
        .....
        maven { url '../../../m2repository' }
    }
}
```

2. Define AndroidJUnitRunner as the runner for test instrumentation. Add the following to the build.gradle file:

```
android {
    .....
    defaultConfig {
        .....
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
}
```

3. Add or write code for your app tests. Use the helper functions in the ATSL in your test code.

You can use the code for the app tests in any of the sample apps as a starting point. For example, in the SecureSQL sample app, the first app test, `test1`, executes BlackBerry Dynamics activation and unlock as an automated test. Put the code in one of the source directories specified in the `androidTest` target. In the above example, only the `tests` sub-directory is specified.

Configure compliance settings so you can debug your app

Compliance profiles in BlackBerry UEM provide the ability to detect when a device OS is rooted and to initiate an enforcement action (this option is disabled by default). This feature extends to deployed BlackBerry Dynamics apps, compiled with SDK version 5.0 or later, where an active debugging tool is detected. Your options for configuring this feature depend on the version of BlackBerry UEM and the BlackBerry Dynamics SDK:

- If your organization uses BlackBerry UEM version 12.11 MR1 or later and the BlackBerry Dynamics SDK version 6.1 or later, when you enable the compliance setting to detect a rooted OS, you can configure the setting “Enable anti-debugging for BlackBerry Dynamics apps”. If enabled, the BlackBerry Dynamics Runtime stops a BlackBerry Dynamics app if it detects an active debugging tool. If disabled, the BlackBerry Dynamics Runtime takes no action when it detects an active debugging tool.
- In UEM versions earlier than 12.11 MR1, the “Enable anti-debugging for BlackBerry Dynamics apps” option is not present and this functionality is enabled by default. If you enable the compliance setting to detect a rooted OS, the BlackBerry Dynamics Runtime stops a BlackBerry Dynamics app when it detects an active debugging tool.

If you want to debug a BlackBerry Dynamics app in an environment where a compliance profile is applied, verify that the compliance settings are configured as required. Alternatively, you can use a non-debug build of your app to test it with the compliance settings enabled.

Emulators and the rooted OS compliance setting

If a compliance profile (BlackBerry UEM) or policy (standalone Good Control) is configured to check for a rooted OS, and the profile or policy is applied to a BlackBerry Dynamics app that is running on a vanilla Android emulator, the emulator will wipe the BlackBerry Dynamics app. The default behavior for the compliance setting is to wipe an app on a rooted device, and the emulator is considered a rooted device. Note that compliance actions related to SafetyNet attestation and hardware certificate attestation will also consider an emulator to be compromised.

This default compliance behavior is best for production service but interferes with development testing.

Consider any of the following recommendations for a development environment:

- Use enterprise simulation mode for basic operations on the emulator. This requires no setup in the management console. For more information, see Using enterprise simulation mode.
- Use a new Android emulator configuration with minimum API Level 26 that includes a Google Play system image. This configuration is not considered a rooted device.
- In the management console, configure and assign a new profile or policy set for development purposes, with the root detection setting disabled.

Using enterprise simulation mode

During the development and testing of your BlackBerry Dynamics app, you can run the app in enterprise simulation mode to verify its execution and functionality. This mode simulates the user authentication process, so there is no direct communication with your organization's UEM or Good Control server. As a result, a valid activation key is not required to activate the app.

Enterprise simulation mode runs the app on an Android virtual device (AVD) that is included in the SDK. Note that even though there is no communication with the management server, communication with the BlackBerry Dynamics NOC still occurs during the initial activation of the app. The BlackBerry Dynamics NOC must be accessible from your testing environment.

You will notice the following differences when you run the app in enterprise simulation mode:

- A [Simulated] label appears in the BlackBerry Dynamics Runtime user interface.
- Any email address and activation key (PIN) is accepted for enterprise activation.
- The provisioning and policy setup flow is only simulated in the UI.
- A hard-coded set of profiles and policies from the management server is applied. Authentication delegation is not supported.

You should also note the following:

- If you run an app that was built for enterprise simulation mode on an actual device and not on an emulator, the app will be wiped.
- If you try to change to simulation mode when the app is already installed on a device, the app will be wiped.
- Lost password recovery is not available.
- Inter-container Communication (ICC) cannot be used; as a result, the Shared Services Framework cannot be used.
- The Secure Storage, Secure Communication, and Push Channel APIs (Android/iOS) are all available in enterprise simulation mode. The communication APIs will not be able to connect to your organization's application servers through the BlackBerry Dynamics proxy infrastructure. You can make connections to your organization's application servers if, for example, the AVD is running on a computer on your organization's LAN or VPN.

Enable enterprise simulation mode

In the **settings.json** file is located in the ../assets/ folder of the app, change `"GDLibraryMode":"GDEnterprise"` to `"GDLibraryMode":"GDEnterpriseSimulation"`.

After you finish: Revert this change when you are ready to compile your final production build.

Troubleshooting common issues

Problem	Possible solution
App crashes	• A crash in the early stages of the app lifecycle is most likely caused by a configuration issue. Review the errors printed in the logs for details. • Initialize the BlackBerry Dynamics libraries before calling any other BlackBerry Dynamics API.
App crashes at startup	This can occur if android:maxSdkVersion is specified in the uses-permission element in AndroidManifest.xml (for example, <uses-permission android:maxSdkVersion="25" android:name="android.permission.WAKE_LOCK"/>). If a permission is defined with this syntax and the app is run on a device with an API level higher than the maxSdkVersion, the app is not granted the permission at startup and crashes. Check the runtime merge process to identify and correct this issue.
Provisioning error	The email address or access key (PIN) has been incorrectly entered, or an old access key has been used. Double-check the credentials and, if necessary, send a new access key from the BlackBerry UEM management console.

Logging and diagnostics

The processing activity of the BlackBerry Dynamics Runtime is logged by the runtime itself. The activity log is written to the BlackBerry Dynamics secure container on the device after deployment. You can configure how your BlackBerry Dynamics apps generate console log information. For more information about console logs controlled by developers and container logs controlled by UEM or Good Control administrators, see the BlackBerry Dynamics Runtime activity log appendix in the API reference (Android/iOS).

If your app uses SDK version 5.0 or later, and the UEM or Good Control administrator has turned off “Enable detailed logging for BlackBerry Dynamics apps” in the BlackBerry Dynamics profile (UEM) or security policy (Good Control), the app does not generate console log information. This provides additional protection against attacks by malicious users. This change has no impact on how container logs are generated.

The “Enable detailed logging for BlackBerry Dynamics apps” setting is off by default.

For BlackBerry Dynamics apps running SDK version 5.0 or later, console logs are generated only if this setting is turned on or if the app is running in enterprise simulation mode.

Configure logging for the Android Debug Bridge console

You can configure the message categories that are printed to the Android Debug Bridge (adb) console. You can configure the logging to be detailed or selective. Changes to the console logging configuration will be applied the next time the target is built and run. Changes made to console logging do not affect the container log.

1. Open the app's **assets/settings.json** file.
2. Perform one of the following actions:

Task	Steps

Configure detailed logging (print all messages)	Add a **GDConsoleLogger** array attribute to the settings object or change the existing attribute to a single string element with a value of "GDFilterNone", as shown below:
	`...`
	`{`
	`"GDLibraryMode": "GDEnterprise",`
	`"GDApplicationID": "com.example.browser",`
	`"GDApplicationVersion": "1.0.0.0",`
	`"GDConsoleLogger": [ "GDFilterNone" ]`
	`}`
	`….`

Configure selective logging	Add a **GDConsoleLogger** array attribute to the settings object or change the existing attribute to add one or more of the logging categories, as shown below:
	`...`
	`{`
	`"GDLibraryMode": "GDEnterprise",`
	`"GDApplicationID": "com.example.browser",`
	`"GDApplicationVersion": "1.0.0.0",`
	`"GDConsoleLogger": [`
	`"GDFilterErrors_",`
	`"GDFilterWarnings_",`
	`"GDFilterInfo",`
	`"GDFilterDetailed"`
	`]`
	`}`
	`...`
	The log categories marked with an underscore (_) will be included and the others excluded. For example, in the example above, info and detailed log entries are filtered out and errors and warnings are included.

Monitoring app log uploads by device users

You can use the GDLogManager class (Android/iOS) to monitor app log file uploads that are initiated by your organization's device users. This class does not manage or display information about log uploads that are initiated by the UEM or standalone Good Control administrator, or by management console profiles or policies.

GDLogManager provides the following information:

- Upload size
- Amount of data uploaded

- Events that indicate the following states:
  - Upload completed
  - Upload abandoned or canceled
  - Upload suspended
  - Upload resumed after suspension
- Actions for managing a log upload (cancel, suspend, resume)
- Enable detailed logging with detailedLoggingFor (Android/iOS)

When detailed logging is disabled by the management server profiles or policies, calling any API in the GDLogManager class has no effect. It is a best practice to check this setting using the getApplicationConfig API (Android/iOS). If detailed logging is enabled, present the log upload progress UI or any other related UI.

Testing connectivity to application servers and diagnostic functions

You can use the GDDiagnostic class to test connectivity to application servers and other diagnostic functions.

The GreetingsClient sample app that is provided with the SDK demonstrates how to use the GDDiagnostic class.

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 or standalone Good Control 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 and for standalone Good Control. 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 • Assigning profiles
Collect activity and compliance violation information for BlackBerry Dynamics apps	See the following topic in the *BlackBerry UEM Administration Guide*: • Activity and compliance violation reports for BlackBerry Dynamics apps

For more information about testing and distributing BlackBerry Dynamics apps using standalone Good Control, see Developer Bootstrap: Good Control Essentials and the Good Control documentation.

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.

In standalone Good Control 5.0, the following options are available:

- Allow all BlackBerry Dynamics library versions: Apps that use any version of the SDK library are allowed. If this option is enabled, the administrator cannot select specific versions to allow or disallow. By default, this option is disabled.
- Allow unlisted BlackBerry Dynamics library versions: Apps that use versions of the SDK library that are newer than the latest version listed in the compliance rule are allowed. The administrator can still allow or disallow specific versions of the library from the version list. By default, this option is enabled.

Consult with the UEM or Good Control administrator to ensure that the compliance rule is configured appropriately.

Implementing a method to back up app data

The default method that the Android OS provides for automatically backing up app data is partially compatible with BlackBerry Dynamics, but has limitations for what can be saved. Also, with the default backup you will require an unlock key after any data restore.

You can modify the automatic backup method to be compatible with BlackBerry Dynamics, or you can turn off the automatic backup and implement a different option for backing up app data. For more information about your options and implementation instructions, see the Android Auto Backup appendix in the BlackBerry Dynamics SDK for Android API Reference.

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.

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 BlackBerry UEM or standalone Good Control 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. The app can use client certificates if:

- The app uses the BlackBerry Dynamics Secure Communication Networking APIs.
- The device user's UEM or standalone Good Control 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 BlackBerry UEM or standalone Good Control 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*

For more information about configuring certificate support in standalone Good Control, see the Good Control and Good Proxy Admin Help.

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 or standalone Good Control) 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*.

For more information about configuring the desired Kerberos implementation in Good Control, see the Good Control and Good Proxy Admin Help and Kerberos Constrained Delegation with Good Control.

Legal notice

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

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

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

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

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

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

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

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

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

Published in Canada