Development Guide

BlackBerry Dynamics SDK for macOS

Version 3.2

Published: 2018-06-19
SWD-20180619164316249

Contents

About this guide

1

This guide is an introduction and overview to the BlackBerry Dynamics software development kit (SDK) for macOS operating system, with a focus on installing the SDK, configuring projects, and other basic information.

The guide is for software developers who already have an understanding of developing software for the macOS platform. It is not a basic tutorial.

BlackBerry Dynamics background

2

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

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

BlackBerry Dynamics API reference

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

You can access the API reference:

- Online at https://community.blackberry.com/view-doc.jspa?fileName=index.html&docType=mac.
- In the installed directories for the BlackBerry Dynamics SDK.

Shared Services Framework

BlackBerry Dynamics-enabled apps can communicate with each other using the Shared Services Framework. There are two kinds of shared services:

- Server-side services
- Client-side services

For a conceptual background, see BlackBerry Dynamics Services Framework.

Support for client certificates

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

- General requirements for working with PKI certs
- Kerberos PKINIT: client certificates in the Kerberos authentication model. (This is not Kerberos Constrained Delegation, or KCD).

Support for "No Password" security policy

The BlackBerry Dynamics Runtime now supports the security policy for not requiring the end user to set an app password.

Security consideration

- Consider the security ramifications in your environment carefully. Enabling "no password" is only one of the design options available. Others include authentication delegation, enabling "no password" on certain devices only, device management, and more. You should consider enabling "no password" for select groups of users whose devices are under tight control via device management profiles or other controls.
- Do not enable "no password" and authentication delegation in the same policy set.
- Enabling "no password" does not permit authentication in the background, because there is no authentication.

Software prerequisites

1. Standalone Good Control 3.0.50.70 or later or BlackBerry UEM 12.7 or later
2. The "No Password" policy must be enabled and applied to the desired users or app group.

User-visible changes and behavior

With a BlackBerry Dynamics app that is protected by a security policy that requires a password, if the administrator changes the security policy to "Do not require a user password":

- The user is shown an informational screen stating that a password is no longer required for the app.
- The user is then in "No Password mode" and is never prompted for password again.

If the user is in "No Password mode" and the administrator changes the security policy to require a password:

- The user is prompted to set a password.
- The user is shown an informational screen stating that a password is now required for the app.

Unsupported or partially supported BlackBerry Dynamics features

Because macOS is a relatively new supported platform for BlackBerry Dynamics, the BlackBerry Dynamics SDK for macOS does not currently support the following features of BlackBerry Dynamics, which are being added to the SDK with upcoming releases::

- Authentication delegation

- Easy Activation
- Enterprise simulation mode
- Mobile application management
- Mobile device management

Locales

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

- English (US)
- Chinese (Simplified)
- Dutch
- French
- Japanese
- Korean
- Swedish

For localization, the bindings rely on the underlying SDKs' supported platforms, Android and iOS.

Requirements

BlackBerry Dynamics software versions

- BlackBerry Dynamics SDK for macOS 3.0.0.227

Compatibility with earlier releases

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

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

Software requirements

macOS development

Item	Requirement
macOS	• 10.10 or later
Xcode	• 6.4 or later
Deployment target	• macOS 10.12 or later

BlackBerry Dynamics entitlement ID and version

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 for your apps, as well as for publishing and service provider registration. The BlackBerry Dynamics entitlement ID was formerly known as the app ID or GD App ID.

The entitlement ID is used in the app, in the BlackBerry UEM or standalone Good Control management console for app management, and in some administrative user interfaces on the application developer portal.

For details about defining the entitlement ID and version, see Set BlackBerry Dynamics entitlement ID and entitlement version.

Note: The entitlement ID and entitlement version are different from the native application ID and native application version. The native application ID is a unique identifier for the app that is used by the OS and associated platforms (for example, the package name for Android or bundle identifier for iOS). The native application version is the app version number that you must change if you want to distribute a new version of an app. You only need to change the entitlement version if the app starts to provide a new shared service or shared service version, or if the app stops providing a shared service or shared service version. For more information about when to change the entitlement ID and entitlement version, see the BlackBerry Dynamics API reference.

Requirements for the entitlement ID and entitlement version

Requirement	Description
Required for apps	You must define both the entitlement ID and the entitlement version for all your BlackBerry Dynamics apps, regardless of whether you use the BlackBerry Dynamics Shared Services Framework. Developers and administrators should ensure that the value specified for the `GDappVersion` key in the app configuration files is the same as the value the administrator specifies in BlackBerry UEM or in standalone Good Control. The entitlement version is independent of any native version identifier. For more information, see Distinction from and use with native language identifiers.
Represent the same app across all platforms	The same entitlement ID must be used to represent the app across all platforms. By default, access to apps varies by the type of app: • By default, all versions of partner or ISV apps are available to all authorized users in any organization that the app has been published to. • By default, each version of a BlackBerry Dynamics app requires that the administrator grant access in BlackBerry UEM or in standalone Good Control before users can run the app on users' devices.
Naming scheme	Develop a naming scheme to meet your needs. For example: • Entitlement ID: com.manufacturingco.gd • Entitlement version: 1.0.0.0 • Native application version: 2.0
Entitlement ID format	• The general form of an entitlement ID is <*company_name*>.<*app_name*>. • The ID must use reverse domain name form, for example, com.company.example. Use a domain name owned by your organization.

Requirement	Description
	• The ID must not begin with com.blackberry, com.good, com.rim, or net.rim. • The ID can contain only lower-case letters, numeric digits, hyphens, and periods. • The string must follow the <subdomain> format defined in section 2.3.1 of RFC 1035, as amended by Section 2.1 of RFC RFC 1123.
Entitlement version value	• The value must use one to four segments of digits, separate by periods (x.x.x.x). • Each segment can be up to three digits and must not use a leading zero (for example, 01.02 is not valid). A segment can use a single 0. • The first release of an app should use the entitlement version 1.0.0.0.

Distinction from and use with native language identifiers

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

- Info.plist
    - `CFBundleIdentifier`
    - `CFBundleVersion`

Unique native identifiers for enterprise apps

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

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

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

Mapping BlackBerry Dynamics entitlement ID to native identifiers

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

Native version identifiers: * wildcard allowed for blocking app

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

- Platform: macOS `CFbundleVersion`
- A series of integers separated by ".". No explicit limit on number of words.
- More information from Apple

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

- Allowed: 2.3.*
- Not allowed: 2.*.3
- 2.* includes 2.*.*

Required build-time declarations: URL type

Your app must declare a URL type so that it can be discovered by other apps on the same user's device.

This enables BlackBerry Dynamics AppKinetics (shared services) communication, which is required for many BlackBerry Dynamics features. In other words, this scheme is used to discover applications on a device.

Table 1: URL type discovery schemes

Format	Description	Example
`com.good.gd.discovery`	Always required	Exactly as shown
`native_application_ identifier``.sc2`	Enables an app to use authentication delegation and is required for all BlackBerry Dynamics apps.	`com.example.gd.myapp.sc2`
`native_application_ identifier``.sc2.`*`GD_application_ version_number`*	Required only if your app provides a service that must be discoverable.	`com.example.gd.myapp.sc2.1.0.0.0`

Supported BlackBerry Dynamics SDK programming constructs or macOS features

You might already familiar with the BlackBerry Dynamics SDK for iOS. Here is a list of some of the constructs that are supported by the BlackBerry Dynamics SDK for macOS to those of the BlackBerry Dynamics SDK for iOS.

- The Push Service API familiar from the BlackBerry Dynamics SDK for iOS is supported. The API functions exactly the same way on both platforms.
- The classes from GDUtility.h are supported, specifically in support of the single sign-on GDAuthToken.
- Tabbed windows on macOS 10.12 are supported.

This is only a partial list. For full details, see the BlackBerry Dynamics API reference.

Unsupported classes compared to BlackBerry Dynamics SDK for iOS

You might already familiar with the BlackBerry Dynamics SDK for iOS. Here is a comparison of some calls in the BlackBerry Dynamics SDK for macOS to those of the BlackBerry Dynamics SDK for iOS.

- `GDHTTPRequest` is not included in the BlackBerry Dynamics SDK for macOS. Use `GDURLProtocol` instead.
- Deprecated `GDFileSystem` is not included in the BlackBerry Dynamics SDK for macOS.. Use `GDFileManger` instead.
- `GDSocket` is not supported.

Unsupported Apple programming constructs or features

The following Apple UI controls and other classes are not supported.

- File Open and Save dialogs
- `NSDrawer`
- `NSURLDownload`: Use `NSURLConnection` instead.
- `NSPopover`
- `NSURLSession`: Use `NSURLConnection` instead.

- Webview/WebFrame: The `loadRequest` instance method on WebFrame is not supported.
- WKWebview: The `loadRequest`instance method alone is not supported. An alternative is to use `NSURLConnection` to make a request and then load the returned data into WebFrame or WebView with the `loadRequest` instance method.

SiriKit not supported

SiriKit is an extension framework by Apple for integrating system voice control services with your application. If the SiriKit framework is integrated, the application's data could be sent out of its private space on the device and even off the device to servers operated by Apple.

Data sent via SiriKit integration cannot be protected by BlackBerry Dynamics . The mechanisms to send the data are private to Apple and cannott be secured by the BlackBerry Dynamics runtime on the device, nor can they be sent by BlackBerry Dynamics secure communication.

For these reasons BlackBerry Dynamics does not support SiriKit.

App user interface security

The BlackBerry Dynamics SDK for macOS takes control of the security of the application user interface (UI) at different times throughout the application lifecycle to enforce the polices applied to the user and maintain application security.

These are programming recommendations for interacting with the SDK's secure control.

BlackBerry Dynamics SDK in front and application in front

The BlackBerry Dynamics SDK is in front when it takes control of application UI, displays its own UI, and removes and stores the application's UI (including windows, alerts and menus). The application can create new UI elements whilst the BlackBerry Dynamics SDK is in front but attempts to display them will be automatically queued until the application is in front once more. The application can remove and destroy existing UI elements in the usual way whilst the BlackBerry Dynamics SDK is in front. When control of application UI is about to be given back to the application, the BlackBerry Dynamics SDK removes its own UI, restores the application UI, and then the application is in front. See discussion in Monitoring control of application UI for examples of how to monitor control of application UI.

Modal sessions cannot be persisted and so are aborted when the BlackBerry Dynamics SDK is in front. For background information, see Apple's NSModalSession class reference

An application should avoid calling the following methods due to the inclusion of a `NSModalSession` instance:

- `NSApp runModalForWindow:`
- `NSApp beginModalSessionForWindow:`
- `NSAlert runModal`
- Deprecated methods:

- `NSApp runModalForWindow:relativeWindow:`
- `NSApp beginModalSessionForWindow:relativeWindow:`

An application can create a new window and display it to the user in a central location as an alternative to creating a modal session:

```
//Strong window reference so that Cocoa Window Manager does not remove the window
_myModalWindow = [self window];
//Center and display the window to the user
[_myModalWindow center]; //Cocoa uses this method to position its own modal windows
[_myModalWindow makeKeyAndOrderFront:self];
```

Application UI behavior

The application should take care when working with UI elements while the BlackBerry Dynamics SDK is in front because Cocoa UI elements can work in unexpected and undocumented ways when not on screen. For example, `NSControl` instances may not return the expected `NSEditor` instance via the `currentEditor` property while off screen. It is recommended that the application use the `NSWindow visible` property or programmatically monitor control of application UI prior to calling these methods - see "Monitoring Control of Application UI" below. The main `NSApplication` instance, `NSApp`, has a number of properties that are altered by the presence of the BlackBerry Dynamics SDK:

- `mainWindow` and `keyWindow`: The application, where possible, should address windows it owns via strongly referenced properties or, if making use of these convenience methods, validate that the returned window is of a known class type before performing actions on it. These methods should not be used while the BlackBerryDynamics SDK is in front.
- `windows`: The array of NSWindow instances returned by this method contains windows owned by the BlackBerry Dynamics SDK and windows owned by the application. The application should address windows it owns via strongly referenced properties rather then performing operations on this array.
- `mainMenu`: The application should avoid traversal of the main menu and should instead address menu items it owns via strongly referenced properties. The contents of the main menu will be replaced when the BlackBerry Dynamics SDK is in front.

Monitoring control of application UI

Current control of application UI can be checked by reading the `GDUserInterfaceState userInterfaceState` property of the GDMac class singleton. The `userInterfaceState` property returns a value of `GDUIStateApplicationInFront` when the application is in front. There are several ways to monitor the change of application UI control, discussed below.

NSNotificationCenter and observing GDStateChangeNotification

```
 //Register an observer
 [[NSNotificationCenter defaultCenter] addObserver:self
                               selector:@selector(gdStateChanged:)
                               name:GDStateChangeNotification
                               object:nil];

 //Method invoked on state change
 - (void)gdStateChanged:(NSNotification*)notification
 {
      NSDictionary* userInfo = [notification userInfo];
       NSString *propertyName = [userInfo objectForKey:GDStateChangeKeyProperty];
       GDState* state = [userInfo objectForKey:GDStateChangeKeyCopy];

       // User Interface state and Current Screen
       if ([propertyName isEqualToString:GDKeyUserInterfaceState])
             {
                   //User interface changed - inspect 'state.userInterfaceState'
             }
       else if ([propertyName isEqualToString:GDKeyCurrentScreen])
             {
             //Current screen changed - inspect 'state.currentScreen'
             }

 //Note: can also be used to receive notification of change in authorization state:
 //GDKeyIsAuthorized - inspect 'state.isAuthorized'
 //GDKeyReasonNotAuthorized - inspect 'state.reasonNotAuthorized'
 }
```

Key-Value Observing (KVO) and adding observers to the GDUserInterfaceState userInterfaceState property of the GDMac class singleton

```
 //Get reference to GDState property from main GDMac instance
 GDState *state = [GDMac sharedInstance].state;

 //Register KVO observers
 NSKeyValueObservingOptions keyValueOptions = (NSKeyValueObservingOptionNew |
NSKeyValueObservingOptionOld);
 [state addObserver:self forKeyPath:GDKeyUserInterfaceState options:keyValueOptions
context:NULL];
 [state addObserver:self forKeyPath:GDKeyCurrentScreen options:keyValueOptions
context:NULL];

 //Note: GDState also allows observation of 'GDKeyIsAuthorized' and
'GDKeyReasonNotAuthorized' key paths

 //Implement KVO method
 - (void)observeValueForKeyPath:(NSString *)keyPath
                                    ofObject:(id)object
                                    change:(NSDictionary *)change
                                    context:(void *)context {

       if ([keyPath isEqualToString:GDKeyUserInterfaceState]) {
             GDUserInterfaceState newState = (GDUserInterfaceState)[[change
objectForKey:NSKeyValueChangeNewKey] integerValue];
       }
       else if ([keyPath isEqualToString:GDKeyCurrentScreen]) {
             GDLibraryScreen newScreen = (GDLibraryScreen)[[change
objectForKey:NSKeyValueChangeNewKey] integerValue];
       }
       else {
             [super observeValueForKeyPath:keyPath ofObject:object change:change
context:context];
       }

 //Note: can also be used to monitor change in authorization state:
 //GDKeyIsAuthorized - BOOL
 //GDKeyReasonNotAuthorized - NSString
 }
```

Automatic Removal and Restoration of Application Windows

Application windows have their delegate property replaced whilst the BlackBerry Dynamics SDK is in front so it is important that the application validate a window's delegate is of a known class before performing an action upon it during this time:

```
 //Check the window delegate is as expected before performing actions upon it
 id<NSWindowDelegate> windowDelegate = [_myWindow delegate];
 if ([GDMac sharedInstance].state == GDUIStateApplicationInFront
     || (windowDelegate && [windowDelegate isKindOfClass:[myWindowDelegate class]]) {
       //Access window delegate as usual here
     }
  }
```

The following methods and notifications may be triggered by the OS while control of application UI is changing:

- `NSWindow` instance methods such as `orderOut:`, `orderWindow:relativeTo:`, `makeFirstResponder`, `becomeFirstResponder` and `makeKeyAndOrderFront`
- `NSWindow` notifications such as `NSWindowDidBecomeKeyNotification` and `NSWindowDidBecomeMainNotification`

Steps to get started with the BlackBerry Dynamics SDK

4

Follow the steps below to start working with the|BlackBerry Dynamics SDK for macOS.

1. Download and install the BlackBerry Dynamics SDK. For information, see Install the BlackBerry Dynamics SDK for macOS .
2. Familiarize yourself with the features of the BlackBerry Dynamics SDK for macOS.
    - For information, see BlackBerry Dynamics background.
3. Understand the requirements and possible constraints on your programming.
4. Start working with the code.
5. Become familiar with available classes and metheds in the BlackBerry Dynamics API Reference.
6. Add your own code or start from scratch. See Setup a new or existing project for BlackBerry Dynamics SDK.
7. Build your application.
8. If necessary, troubleshoot your app. To help you troubleshoot issues, you can set up logging and diagnostics. For more information, see Logging and diagnostics.
9. Set up an uninstaller program for your app to give to end-users. See Template for end-user to uninstall a BlackBerry Dynamics app.
10. Deploy your app. For options, see Ready your application for deployment.

Install the BlackBerry Dynamics SDK for macOS

Here are details about installing the BlackBerry Dynamics SDK for macOS. The SDK comes with three scripts for installing:

- `install_SDK.sh`: Installs only the software.
- `install_Documentation.sh`: Installs only the documentation.
- `install_Extras.sh`: installs the template for application uninstallation.
- `install_All.sh`: Installs all of the above.

1. Download the SDK from the BlackBerry Developer Network.
2. In the macOS UI, double-click the zip package, or in a terminal (shell) window, enter: `unzip` *`name_of_downloaded_file`*`.zip`
3. In a terminal (shell) window, change directory to the unzipped folder.

4. Enter the following command to install both the software and the documentation: `./install_All.sh` 5. If prompted, enter the password.

Location of installed BlackBerry Dynamics SDK for macOS

After installation, the BlackBerry Dynamics SDK for macOS is located as follows:

- The SDK home folder ~/Library/Application Support/BlackBerry/Good.platform/macOS contains files and subfolders for extras and the framework. This folder is referred to as `$GD_SDK_HOME`.
- The installer also creates a symbolic link in /Library/Frameworks to the SDK framework.
- Documentation: ~/Library/Developer/Shared/Documentation/DocSets

Uninstall the BlackBerry Dynamics SDK for macOS

BlackBerry supplies a script to uninstall the SDK, because the various components of the SDK (libraries, documentation, and others) are installed in different locations on your system. To uninstall either the entire SDK or individual components, use the scripts distributed with the SDK or the copies of them installed in `$GD_SDK_HOME`.

Note: For the definition of `$GD_SDK_HOME`, see Location of installed BlackBerry Dynamics SDK for macOS.

1. In a terminal (shell) window, change directory to `$GD_SDK_HOME`.
2. Enter the following command to uninstall the complete SDK: `./uninstall.sh`
3. If prompted, enter your password.
   The BlackBerry Dynamics SDK for macOS is now uninstalled.

Setup a new or existing project for BlackBerry Dynamics SDK

You can use the details here as a starting point for setting up your own BlackBerry Dynamics SDK-enabled secure application.

To setup new project or add BlackBerry Dynamics support to existing project, follow these general steps.

1. Add the BlackBerry Dynamics framework for macOS. See Add BlackBerry Dynamics framework.
2. Define your BlackBerry Dynamics entitlement ID and entitlement version.
3. Optionally set logging filters. See Optionally set logging filters with GDConsoleLogger.
4. Implement a class conforming to the `GDMacDelegate` protocol. See Implement the GDMacDelegate protocol.

Add BlackBerry Dynamics framework

Add the BlackBerry Dynamics SDK for macOS framework to your project target's **General** settings as an **Embeded Binary** and also in **Linked Frameworks and Libraries**, as shown below.

Set BlackBerry Dynamics entitlement ID and entitlement version

In your Info.plist, define the required key/value pairs for BlackBerry Dynamics entitlement ID and BlackBerry Dynamics entitlement Version, described in BlackBerry Dynamics entitlement ID and version, as shown below:

- `GDApplicationID`: string in the form of separated words by dots, usually same as bundle identifier. Example: `com.companyname.gdenabledapp`
- `GDApplicationVersion`: string of application version in the form of numbers separated by dots. Example: `1.0.0.0`

Implement the GDMacDelegate protocol

You need to create a class conforming to the `GDMacDelegate` protocol, which enables your application to receive events and to be a fully functional BlackBerry Dynamics application.

The conforming class should implement the `GDMacDelegate` protocol and its required method:

```
-(void)handleEvent:(GDAppEvent*)anEvent;
```

Also consider the following alternatives:

- If your class also implements the `NSApplicationDelegate` protocol and is in fact the application delegate then the BlackBerry Dynamics user authentication process starts automatically, and the BlackBerry Dynamics Runtime takes control of the user inferface to display the standard BlackBerry Dynamics provisioning window.
- If your class does not implement `NSApplicationDelegate`, you must manually call the `[[GDMac sharedInstance] authorize:gdMacDelegate]` method to start the provisioning process.

Template for end-user to uninstall a BlackBerry Dynamics app

Most users typically uninstall macOS applications by deleting the app icon (which is the app bundle) from the Applications folder or by dragging it to the Trash. This practice does not actually uninstall the application but merely removes only one component of it. If the user later installs another copy of the application, the application essentially carries on from where it left off.

BlackBerry Dynamics applications are no different in this respect. Many apps provide an uninstall script for the user to run, which typically does the following:

- Deletes app data stored in the file system
- Deletes app entries (passwords) from the keychain
- Deletes the app bundle (the actual app itself)

If a user wants to completely uninstall the app, all three components must be deleted, and the easiest way to accomplish this is with an uninstall script. BlackBerry Dynamics SDK for macOS includes a template uninstaller script that you should customize and include in your BlackBerry Dynamics-based application's installation folder. This allows your customers to uninstall your applications cleanly and thoroughly.

The template is located in $GD_SDK_HOME/Extras and is named:

```
GD_app_uninstaller_template.tool
```

Usage guidelines:

- Make a copy of this template and name the copy appropriately for your Good-based application. For example, if your application is named "Secure Widgetizer", you could name your copy of the uninstaller template uninstall_SecureWidgetizer.tool.
- The template has two variables whose values you must set; these are the same values you set in your application's Info.plist:
    - `APP_BUNDLE_ID`: The native bundle ID for your application
    - `APP_BUNDLE_NAME`: The native bundle name of your application
- In the documentation for your application that you deliver to customers, provide instructions (such as a ReadMe file) for using your customized uninstaller script, such as the following instruction:

  ```
  You need only double-click your uninstaller to run it; no shell is required.
  ```
- Test your customized uninstaller along with your other tests before you release your application.

Testing and troubleshooting

5

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 might be asked to provide the log file to the BlackBerry Dynamics technical support team, but this is typically only necessary for complex support issues.

Log message categories

Messages in the activity log are assigned to one of four categories:

- Errors: critical failures
- Warnings: failures that arise but from which the BlackBerry Dynamics Runtime has recovered
- Info: normal operational activity
- Detailed: additional diagnostic information used for troubleshooting complex problems

You can configure the logging system to filter some or all messages. By default, only messages that belong to the Errors, Warnings, and Info categories are printed.

The logging locations on the IDE console and device container are configured differently and independently. In principle, you control the console log and the BlackBerry UEM administrator controls the container log.

Optionally set logging filters with GDConsoleLogger

For information about how to configure logging filters using the `GDConsoleLogger` key in the Info.plist file, see the following resources:

- Android: BlackBerry Dynamics runtime activity log
- iOS: BlackBerry Dynamics runtime activity log

Configure detailed logging for the Xcode console

Messages printed to the Xcode console are configured in the build targets of the app project. Different targets can have different activity logging configurations. You can set logging to be detailed or selective. Changes to the console logging configuration take effect 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 `Info.plist` file.
2. Add a row with `GDConsoleLogger` as its key.
3. Set `String` as the type of the new row.
4. Set `GDFilterNone` as the value of the new row.

Configure selective logging for the Xcode console

If there are multiple `Info.plist` files, check that you are editing the correct one by opening the Info tab of the app target being built to make sure the new settings are there.

As with detailed logging, the console will only include log messages that are not in any of the categories specified in the `Info.plist`.

To set the target's activity logging configuration to print a selection of message categories, do the following:

1. Open the app `Info.plist` file.
2. Add a row with GDConsoleLogger as its key or change the existing row as follows:
    a. Set `Array` as the type of the logger row.
    b. For each category you don't want to include, add a row under the logger row as an item in the logger's array.
    c. Set the item to type `String`.
    d. Set the value to one of the following:
        - GDFilterDetailed
        - GDFilterInfo
        - GDFilterWarnings
        - GDFilterErrors

Configure logging in Good Control

You can specify the message categories that are recorded in the container log file in the Good Control management console. To configure logging, you must be a Good Control administrator. Container logging can be set for a particular installation of the app provisioned for a specific user. Selective logging for the container is not supported.

Changes that you make to container logging are effective immediately if the app is running and connected to the BlackBerry Dynamics infrastructure. Otherwise, changes are effective as soon as app does connect.

The app can export or upload the container log file using the BlackBerry Dynamics API. For more information, see the function definitions in the `GDFileSystem` class reference in SDK programming reference.

GDLogManager class for log uploading

You can use the `GDLogManager` class that is included with the BlackBerry Dynamics SDK to monitor app log file uploads that are initiated by end users. This class does not manage or display information about log uploads initiated by the BlackBerry UEM administrator's policies or explicit actions.

`GDLogManager`'s displayed information and functions include the following:

- Size of whole upload
- Amount of data uploaded so far
- 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 upload
    - Suspend upload
    - Resume upload

For more detail, see `GDLogManager` in the SDK programming reference.

When detailed logging is disabled by policy, calling any API in the `GDLogManager` class has no effect. You are encouraged to first check the setting of this policy in the app configuration, using the `getappConfig` API, and if detailed logging is enabled, then present the log upload progress UI or any other related UI.

GDDiagnostic API

The BlackBerry Dynamics SDK includes an API that you can use to test connectivity to application servers and other diagnostic functions. See the discussion of the `GDDiagnostic` class in the BlackBerry Dynamics API reference.

The following samples that are delivered with the SDK illustrate how to use `GDDiagnostic`:

- GreetingsClient

Readying your app for deployment: server setup

6

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

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

BlackBerry UEM preferred

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

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

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

For information on how to use BlackBerry UEM to manage BlackBerry Dynamics apps, see Getting Started with BlackBerry UEM and BlackBerry Dynamics. For more information on the benefits of using BlackBerry UEM, see Benefits of upgrading from Good Control to BlackBerry UEM.

Older Good Control

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

Details of support for client certificates

7

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

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

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

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

Setup in Good Control

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

To deploy Personal Information Exchange files with BlackBerry Dynamics apps, the following steps must be taken to configure the GC and employee's account. These generalized steps are extracted from the Good Control online help topic "PKCS12 Certificate Management for Email and Client Authentication", which has the exact steps.

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

Behavior of personal certificates in the app

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

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

1. Only client certificates suitable for SSL/TLS client authentication are eligible for sending to the server. That is, certificates that have no Key Usage and Extended Key Usage, or Key Usage contains “Digital Signature" or "Key Agreement”, or Extended Key Usage contain "TLS Web Client Authentication”, and those whose Key Usages and Extended Key Usages do not contradict allowances for SSL/TLS client auth.
2. If the server advertises the client certificate authority in the SSL/TLS handshake, only client certificates issued by these authorities will be considered
3. Only current client certificates will be considered (that is, certificates that have not expired or are not yet valid)

Usually this is sufficient to identify the correct client certificate, but if there is still more than one certificate meeting all of the above criteria then the first one is used. If the certificate chosen is not the desired one, the administrator or employee can manage this by removing the undesired client certificate from Good Control. The administrator can also increase the chance of success by ensuring the server is configured to advertise the client certificate authority in the SSL/TLS handshake.

Certificate requirements and troubleshooting

Make sure your certificates conform to these requirements:

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

There are many sources of certificates:

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

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

Beware of weak ciphers from export

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

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

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

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

Kerberos PKINIT: User authentication with PKI certificates

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

No programming is required to use Kerberos PKINIT.

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

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

Key requirements for PKINIT

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

Servers

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

Client certificates

- The minimum keylength for the certificates must be 2,048 bytes.
- Client certificates must include the User Principal Name (UPN) (for example, user@domain.com) in the Subject Alternative Name (SAN) of object ID (OID) szOID_NT_PRINCIPAL_NAME 1.3.6.1.4.1.311.20.2.3, as specified by Microsoft at https://support.microsoft.com/en-us/kb/287547.
- The domain of the UPN must match the name of the realm of the Windows Key Distribution Center (KDC) service.
- The Extended Key Usage (EKU) property of the certificate must be Microsoft Smart Card logon (1.3.6.1.4.1.311.20.2.2). For information about certificate requirements for smart card logon, see https://technet.microsoft.com/en-us/library/ff404293 (v=ws.10).aspx.
- Certificates must be valid. Validate them against the servers listed above.

Client applications

- In BlackBerry Work, to allow the use of client certificates, you must enable the `useEASAuthCert` setting.
- Apps must not send any password in the HTTP/HTTPS request.
- Apps must either set the HTTP/HTTPS header `WWW-Authenticate: Negotiate` or not set any authorization method in the HTTP or HTTPS request, to which the server has responded with 401 `WWWAuthenticate: Negotiate`, as detailed in https://www.ietf.org/rfc/rfc4559.txt.

Key points

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

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

Background on PKINIT, with FAQ

Consider the interactions in this drawing: http://www.ibm.com/developerworks/ibmi/library/i-sso/figure1.jpg

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

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

In a large organization users and resources might belong to various realms and there may be many KDCs, so how does BlackBerry Dynamics find the right one?

1. How does the client locate the user’s KDC Authentication Service when initializing the user’s session?

    - Password-based authentication

      The realm in the user name must contain the host name of the KDC AS. For example:

      User: `user@MY.REALM.COM`

      Password: `myPassword`

    - Certificate-based authentication: This is PKINIT.

      The realm in the UPN of the user’s certificate must contain the host name of the KDC AS. For example:

      `UPN (OID 1.3.6.1.4.1.311.20.2.3): user@MY.REALM.COM`

2. How does the client locate the resource's KDC Ticket-Granting Service (TGS) when retrieving the resource?

BlackBerry Dynamics attempts to obtain a TGS from the host in the domain of the resources URL. For example:

URL: http://resource.myrealm.com/index.html

The client will connect to KDC TGS running on host myrealm.com on TCP port 88.

Legal notice

8

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

BlackBerry UK Limited
200 Bath Road
Slough, Berkshire SL1 3XE
United Kingdom

Published in Canada