BlackBerry Dynamics Shared Services Framework

Last updated: Wednesday, December 21, 2016
Version: BlackBerry Dynamics SDK 3.0.xxxx

Contents

Revision history

Shared Services Framework

Date	Description
2016-12-19	Version numbers updated for latest release; no content changes.
2016-09-27	• Added cross-references for syntax details to some of the more importnat classes and methods for Shared Services: Programming shared services with the BlackBerry Dynamics SDK: API reference • Clarified some cross-references
2016-08-18	Added a general approach to Discovering the BlackBerry Enterprise Mobility Server doc services
2016-08-03	Corrected the name of an API in Comparison of APIs and protocols , which is **`getServiceProvidersFor`**.
2016-06-29	Version numbers updated for latest release; no content changes.
2016-04-13	Added important note about Same user, same GC: provisioning shared services apps .
2016-03-10	Truncated revision history to reduce bulk.
2016-01-15	Version numbers updated for latest release; no content changes.
2015-12-23	Version numbers updated for latest release; no content changes.
2015-11-30	Now includes important information About BlackBerry Dynamics entitlement ID and version

Overview to shared services framework

The Shared Services Framework is a system for collaboration, which is defined as two-sided: a service provided by one side and consumed by the other. These two sides can be any of the following:

- Developers of different mobile BlackBerry Dynamics applications, through application-based services
  - Partner with in-house app.
  - In-house app with in-house app.
  - Partner with Partner.
- Developers of mobile BlackBerry Dynamics applications and developers of application servers, through server-based services

Audience

This content is intended for readers familiar with BlackBerry Dynamics products and application programming, including data communications protocols and other mobile application programming concepts.

Other documentation

This content relies on reference documentation for many different BlackBerry Dynamics products that can be used under the Shared Services Framework. Documentation for these products and other resources referred to are listed Reference documentation and related resources.

About BlackBerry Dynamics software version numbers

The cover of this document shows the base or major version number of the product, but not the full, exact version number (which includes "point releases"), which can change over time while the major version number remains the same. The document, however, is always current with the latest release.

Product	Version
Good Control	3.0.56.70
Good Proxy	3.0.56.32
BlackBerry Dynamics SDK for Android	3.0.0.3015
BlackBerry Dynamics SDK for iOS	3.0.0.6008
BlackBerry Dynamics SDK for macOS	3.0.0.227
BlackBerry Dynamics SDK for Universal Windows Platform	3.0.0.805
BlackBerry Launcher Library for Android	2.5.0.175
BlackBerry Launcher Library for iOS	2.5.0.88

Product	Version
BlackBerry Dynamics Cordova (formerly also called "PhoneGap")	3.0.0.106
BlackBerry Dynamics Bindings for Xamarin.Android and for for Xamarin.iOS	3.0
Digital Authentication Framework (DAF) • Android • iOS	 • 3.0.0.13 • 2.0.0.12

If in doubt about the exact version number of a product, check the BlackBerry Developer Network for the latest release.

About Entitlement ID and version

About BlackBerry Dynamics entitlement ID and version

In the Good Control console and the BlackBerry Developer Network, BlackBerry Dynamics-based applications are identified by a *BlackBerry Dynamics entitlement ID* and *entitlement Version*. A primary purpose of the BlackBerry Dynamics entitlement ID and entitlement Versions is for you to manage end-user entitlement to your BlackBerry-provided applications; in this context you might hear the BlackBerry Dynamics entitlement ID referred to as "entitlement ID"; for BlackBerry Dynamics-based applications, the terms are equivalent.

A single BlackBerry Dynamics entitlement ID must be used to represent the same application across all platforms. Other restrictions also apply.

By default, access to applications varies by type of application:

- All versions of Partner/ISV applications are by default permitted to all to authorized users of any organization to which the application has been published.
- Each version of a BlackBerry Dynamics-based application by default requires the BlackBerry Dynamics administrator's explicit granting of access on the GC console to run.

BlackBerry recommends that you devise a naming scheme to meet your needs. Use these guidelines to help you formulate that naming scheme.

A simple example: assume we have a BlackBerry Dynamics-based application from a company called Acme, Inc. The native version number is completely independent of the BlackBerry Dynamics entitlement version.

- BlackBerry Dynamics entitlement ID: com.acme.gd
- BlackBerry Dynamics entitlement version: 1.0.0.0
- Native version number: 2.0

Other variations on naming schemes for BlackBerry Dynamics entitlement ID and entitlement Versions are also possible, but keep these details in mind when you devise your own BlackBerry Dynamics entitlement ID naming scheme.

BlackBerry Dynamics entitlement and entitlement version both required for all BlackBerry Dynamics-based apps

You need to define both the BlackBerry Dynamics entitlement ID and the entitlement Version for all your BlackBerry Dynamics-based applications, regardless of whether or not you use the BlackBerry Dynamics Shared Services Framework. Developers and administrators should ensure that the value specified for the GDApplicationVersion key in an app's application configuration files is the same as the value the administrator specifies in Good Control.

The entitlement Version is independent of any native version identifier; see more information in Distinction from and use with native language identifiers .

When to change the BlackBerry Dynamics entitlement version?

The BlackBerry Dynamics entitlement Version is distinct from any visible version number you might use for your application. For example, your BlackBerry Dynamics entitlement Version might be "1.0.0.0" while at the same time you publicly show a native version number "2.1".

Because each new BlackBerry Dynamics entitlement Version of your BlackBerry Dynamics-based application requires “publishing” it to your existing customers, it is recommended to change the BlackBerry Dynamics entitlement version number as infrequently as possible. There are three primary reasons to change the BlackBerry Dynamics entitlement version number:

1. To provide early access, beta, or limited access to a new version for specific customers.
2. For Partners/ISVs, to monetize new functionality differently from your existing version.
3. To represent large level differences in BlackBerry Dynamics functionality (not your own functionality). For example, you might update a service definition, that is, publish a service update that is not supported on an older entitlement Version.

When a new version is to be made available per above (which is usually rare), ensure that the new version is listed on the Marketplace by a partner or on the GC console for custom applications *well before* an application reporting that BlackBerry Dynamics version is ever available in the App Store, Play Store or elsewhere. If the new version of the application is downloaded to a device before the version is published on GDN or in GC, the application is blocked. You should never delist a version unless it is to enforce payment, force end-of-life, or remove a version with a fatal security issue. If a BlackBerry Dynamics entitlement ID or entitlement Version is ever unpublished or an end-user unentitled from an a previously entitled application, the container is wiped from end-user devices for all end-users who installed the application.

Format of BlackBerry Dynamics entitlement ID and version values

The general form of a BlackBerry Dynamics entitlement ID is:

`your_company_name.your_application`

The value of your BlackBerry Dynamics entitlement IDs must follow these rules:

- Must be in reverse domain name form, like `com.yourcompany.something`.
- Must not begin with any of the following:
  - **`com.blackberry`**
  - **`com.good`**
  - **`com.rim`**
  - **`net.rim`**
- No uppercase letters.
- In addition, the string must conform to the **`<subdomain>`** format defined in section 2.3.1 of RFC 1035, as amended by Section 2.1 of RFC 1123.

Note: In the BlackBerry Dynamics SDK for Microsoft Windows 8.1, the value of BlackBerry Dynamics entitlement ID (Application ID) cannot be longer than 35 characters. This does not apply to the BlackBerry Dynamics SDK for UWP.

The value of your entitlement Versions must follow these rules:

- From one to four segments of digits, separated by periods, like **`100`** or **`1.2.3.4`**.
- No leading zeroes in the numeric segments. For example, these are *not* allowed: **`0100`** or **`01.02.03.04`**.
- The length of the numeric segments can be from one to three characters. This is an allowable example: **`100.200.300.400`**.

Distinction from and use with native language identifiers

The BlackBerry Dynamics Entitlement ID and Entitlement Version are Good-specific metadata and are independent of the identifiers needed by the application platforms themselves. The key point is that the BlackBerry 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 BlackBerry Dynamics Entitlement ID and version are stored.

Platform	Location	Platform-specific Names
Android	**`Manifest.xml`**	• **`packageName`** • **`packageVersion`**
iOS	**`Info.plist`**	• **`CFBundleIdentifier`** • **`CFBundleVersion`**
macOS	**`Info.plist`**	• **`CFBundleIdentifier`** • **`CFBundleVersion`**
Universal Windows Platform (UWP)	**`Package.appxmanifest`**	For Windows 10/UWP, the BlackBerry Dynamics SDK relies on Package Family Name, which is not explicitly set but is generated by Visual Studio and is displayed in the GUI editor of the package manifest, as shown below.

Mapping BlackBerry Dynamics entitlement ID to native identifiers

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

- In BlackBerry Client SDK for Android, the Native Bundle ID is the package name in your app's AndroidManifest.xml file.
- In BlackBerry Client SDK for iOS, the Native Bundle ID is the CFBundleIdentifier in your app's plist file.
- In BlackBerry Client SDK for macOS, the Native Bundle ID is the CFBundleIdentifier in your app's plist file.
- This same Native Bundle ID must be registered with BlackBerry to match the app's specific GDs App ID. Without this mapping your app cannot take advantage of Easy Activation.

  Contact your BlackBerry Dynamics administrator to have this mapping recorded in the GC console or in GDN. In the GC console, the steps are as follows. For each application that requires the native Bundle ID:

- Go to Manage Applications.
- Click the name of the application.
- Go to the Advanced tab. (The Advanced tab is available only for custom applications developed by an organization or to Independent Software Vendors (ISVs).)
- Set the identifier for the appropriate devices.

Native version identifiers: * wildcard allowed for blocking app

The BlackBerry Dynamics 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 Good Control to deny apps by native version.

Platform	Definition	Reference
Android **`packageVersion`**	A string of the format *major·minor·point* with no explicit requirement to use integers, although this is implied and followed by convention.	More information from Google
iOS **`CFbundleVersion`**	A series of integers separated by ".". No explicit limit on number of words.	More information from Apple
macOS **`CFbundleVersion`**	A series of integers separated by ".". No explicit limit on number of words.	More information from Apple
UWP **`/Package/Identity/@Version`**	A string in quad notation, "*Major·Minor·Build·Revision*"	More information from

Platform	Definition	Reference
		Microsoft

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.*.*

About 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.

Enforcement of BlackBerry Dynamics entitlement ID and version in Good Control

The following are the basic rules that application developers must comply with. In this discussion, the terms "BundleIdentifier" and "BundleVersion" are used to cover all similar platform-specific identifiers, such as package name or Application ID.

1. Application name is unique with in the organization.
2. Bundle Version, Bundle Identifier combination is unique for a platform.
3. Change in BlackBerry Dynamics Version enforces change in Bundle Version. The other way round is not true.
4. An application (family of binaries) is either BlackBerry Dynamics (all binaries under it are BlackBerry Dynamics) or non-BlackBerry Dynamics (all binaries under it are non-BlackBerry Dynamics). This rule derives from that entitlement ID is locked at the time of creation. The entitlement ID is the BlackBerry Dynamics entitlement ID if the application is a BlackBerry Dynamics-enabled app.
5. BlackBerry Dynamics entitlement ID is unique throughout the system.
6. Bundle Identifier for a platform is unique for a BlackBerry Dynamics entitlement ID and vice versa. Therefore, a change in BlackBerry Dynamics entitlement ID requires a change in Bundle Identifier, and vice versa.
7. Non-BlackBerry Dynamics and BlackBerry Dynamics versions of same binary have different Bundle Identifiers.

Common errors

The following are errors in usage of the BlackBerry Dynamics entitlement ID and Entitlement Version that are checked by Good Control when BlackBerry Dynamics-based applications are added.

Use Case	Explanation of Error
Administrator submits an app with an existing BlackBerry Dynamics entitlement ID	The BlackBerry Dynamics entitlement ID must

Use Case	Explanation of Error
for an app with some other org.	be unique across organizations.
Administrator submits an app with an existing BlackBerry Dynamics entitlement ID, Bundle Identifier, Bundle Version but different BlackBerry Dynamics Entitlement Version.	The Bundle Version must be changed when there is a change in BlackBerry Dynamics version.
Administrator submits an app with an existing Bundle Identifier and Bundle Version but different BlackBerry Dynamics entitlement ID.	The Bundle Identifier should be different for different BlackBerry Dynamics entitlement ID.
Administrator submits an app with an existing BlackBerry Dynamics entitlement ID, but different Bundle Identifier for an existing platform.	The Bundle Identifier for the same platform should be unique within a BlackBerry Dynamics App.
Administrator submits a BlackBerry Dynamics-enabled app with same Bundle Identifier as an existing non-BlackBerry Dynamics app	Upgrading a non-BlackBerry Dynamics app to a BlackBerry Dynamics app binary requires a change in Bundle Identifier.
Administrator submits a non-BlackBerry Dynamics enabled app with same Bundle Identifier as an existing BlackBerry Dynamics app	Downgrading a non-BlackBerry Dynamics app to a BlackBerry Dynamics app requires a change in Bundle Identifier

Application-based and server-based providers and consumers

The provider of an application-based service can be any of the following:

- A BlackBerry Dynamics application, running on the same device as the consumer.
- The consumer communicates with the provider by way of AppKinetics, which uses the proprietary BlackBerry Inter-Container Communication (ICC) protocol.

The provider of a server-based service is any of the following:

- An application server running on a computer that can be located behind the enterprise firewall.
- The consumer communicates with the provider using HTTP or any other protocol that can be conveyed over TCP sockets.

The consumer in either case is a BlackBerry Dynamics application that uses a sequence of steps to discover and communicate with the provider.

High-level sequence in shared services

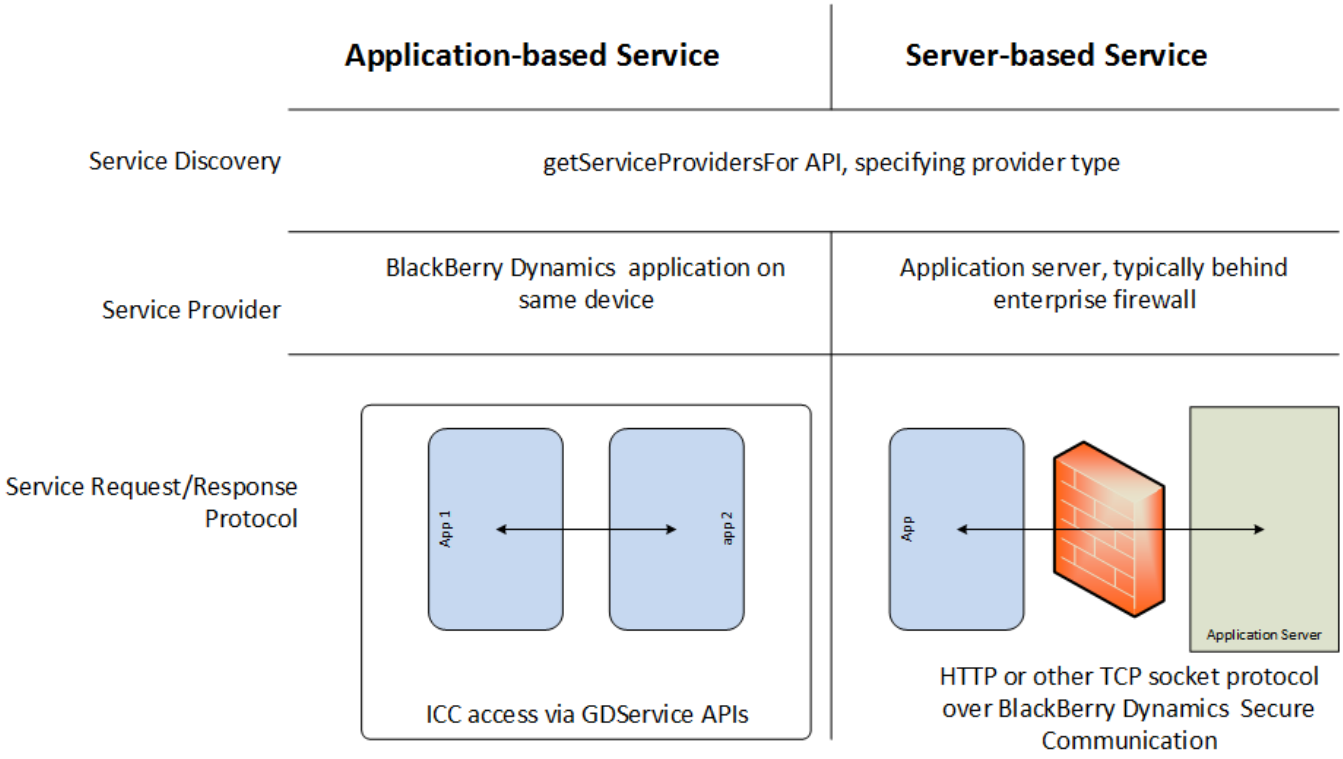
This run-time sequence of steps applies to both application-based and server-based services (although different APIs are used).

1. Service discovery: Query for service providers issued by the service consumer.
2. Service provider selection by the consumer, if there is more than one available service provider.

3. Service request: Sent by the consumer to the selected provider.
4. Service response: Received from the provider, if that is part of the service.

Comparison of APIs and protocols

The diagram below compares the APIs, protocols, and other aspects between application-based and server-based services.



1. Service discovery uses the BlackBerry Dynamics API **`getServiceProvidersFor`**, which takes a parameter for the type of service host: Application or Server.
2. Service provider selection, if necessary, is handled by the application code without reference to BlackBerry Dynamics.
3. Service requests:
   a. Application-based services use the **`GDServiceClient`** API.
   b. Server-based services use TCP sockets or HTTP over BlackBerry Dynamics secure communication.
4. Service responses:
   - Application-based services use the **`GDService`** API, including the **`setServiceListener`** method (and others) in the BlackBerry Dynamics SDK for Android. In the BlackBerry Dynamics SDK for iOS and macOS, you need to set the delegate property of the **`GDService`** instance.
   - Server-based services use BlackBerry Dynamics secure communication, just as with service requests.

Feature comparison of application-based and server-based services

The application-based and server-based service mechanisms share certain features.

Application-based Services	Server-based Services
Can be used offline. Whether this is useful depends on the service.	Server-based services could be provided by a clustered application server.
Application-based services require a platform-specific "footprint" on the device	
End-user entitlement is enforced by both mechanisms.	
Both mechanisms can support conversational services: • Application-based by having a paired service provided by the initial consumer. • For example, see the Edit File and Save Edited File pair of services, with service definitions available at https://community.good.com/marketplace-services.jspa.	

Choosing the mechanism for implementation of your service

Some guidelines for choosing between the mechanisms:

- If the service requires specific user interaction, or is exposing a capability of an existing substantial application, choose application-based.
  - Avoid duplication of effort.
- If the server software already exists outside BlackBerry Dynamics, choose server-based:
  - No need to develop and deploy a mobile application.
  - Cross-platform.
- If there are multiple services in the product and they have different profiles, make the application a dual-provider product, as described in Dual-provider products.

Same user, same GC: provisioning shared services apps

A security feature of shared-services-based applications that are intended to communicate among themselves is that these applications must be provisioned for the same user from the same Good Control server (or cluster).

You cannot entitle a user to an application from one Good Control server and communicate with an application from a different Good Control server with the same user. These "users" might be the same human person but they are distinct users. Likewise these two Good Control servers are distinct.

For more about application entitlement, provisioning and related information, see the Good Control online help at the link listed in BlackBerry Dynamics documentation .

Relationship to Cloud GC: feature supported

The feature, service, server type, or software described here is fully supported on and compatible with Good Control Cloud.

Programming shared services with the BlackBerry Dynamics SDK: API reference

The BlackBerry Dynamics SDK has several classes with callable methods for working with Shared Services. Some of the main classes or methods are listed below. See the documentation for your particular platform in BlackBerry Dynamics documentation .

Android

- **`GDAndroid.getServiceProvidersFor`**
- **`GDServiceProvider`**
- **`GDServiceDetail`**

iOS

- **`GDiOS.getServiceProvidersFor`**
- **`GDServiceProvider`**
- **`GDServiceDetails`**

macOS

Note: The BlackBerry Dynamics SDK for macOS does not support application-based services.

- **`GDMac.getServiceProvidersFor`**
- **`GDServiceProvider`**
- **`GDServiceDetails`**

Server-based services with the Push Channel

The BlackBerry Dynamics Push Channel is a mechanism for sending notifications from an application server to a mobile BlackBerry Dynamics application. Documentation for the Push Channel API is listed in Resources. The channel is end-to-end secure at the same level as BlackBerry Dynamics secure communication. The BlackBerry Dynamics application thus does not need poll the application server, which decreases load on both the application and the application server. Any application server that is a service provider can use the Push Channel.

General use of the Push Channel is as follows:

1. BlackBerry Dynamics application requests a Push Channel token from the BlackBerry Dynamics Network Operation Center (NOC).
2. BlackBerry Dynamics application sends the token to an application server.
3. The application server can then send a notification back to the BlackBerry Dynamics application via the Good Proxy (GP).

Specific example

The mobile BlackBerry Dynamics application (which in this context is a service consumer) requests a Push Channel token from the NOC, before sending service request. The token is then sent to the application server as part of the service request, conforming to the service's protocol. The application server can then send a Push Channel notification later back to the BlackBerry Dynamics mobile application.

For example, the Push Channel can be used to notify an end user:

- When a file on a sharing service is updated.
- When a particular contact signs in to Instant Messaging.
- When a share price or interest rate changes.

Service definitions

Every service requires an API; otherwise, the developer of a consumer cannot determine the parameters the service requires. This is known as a service definition, which specifies the properties (fields) or parameters of the API.

For application-based services there is an interface definition language (IDL) for service definitions specified by BlackBerry. See Reference documentation and related resources for location.

Server-based services can use any format to define the API. A typical choice is the Web Services Definition Language (WSDL).

> **Note:** The service definition merely details the parameters of the API for communication between the server and the application. It does not define or enforce any business logic you might require. The application or service itself must enforce this logic, because this logic is outside the scope of BlackBerry Dynamics.

Setting up service providers

> **Note:** The BlackBerry Dynamics SDK for macOS does not support application-based services.

Setting up an application-based service provider requires the following high-level steps:

- Creation of mobile BlackBerry Dynamics applications.
  - On Android, you can make your service "findable" by setting a **`<meta-data>`** tag in the **`Android Manifest.xml`**.

In **AndroidManifest.xml** for an application that provides a service, the BlackBerry Dynamics Application Version is called out in a **<meta-data>** tag in the **<application>** block, as shown in the following snippet:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.good.gd.example.appkinetics
       [...]
    <application
         [...]
         <meta-data android:name="GDApplicationVersion" android:value ="1.0.0.0"/>
    </application>
</manifest>
```

- Registration of these applications as service providers using:
  - The enterprise Good Control (GC) console, if in-house.
  - The GDN website, if Partner.
- Installation of these applications by end users, i.e. roll-out.

Setting up a server-based service provider requires these general steps:

1. Creation of application server software.
2. Installation of the server by the enterprise.
3. Configuration in the enterprise GC console.

Dual-provider products

An interesting and useful design approach is to create an application/service that relies on both an application-based service and a server-based service. For example, a developer could supply a single product that included both:

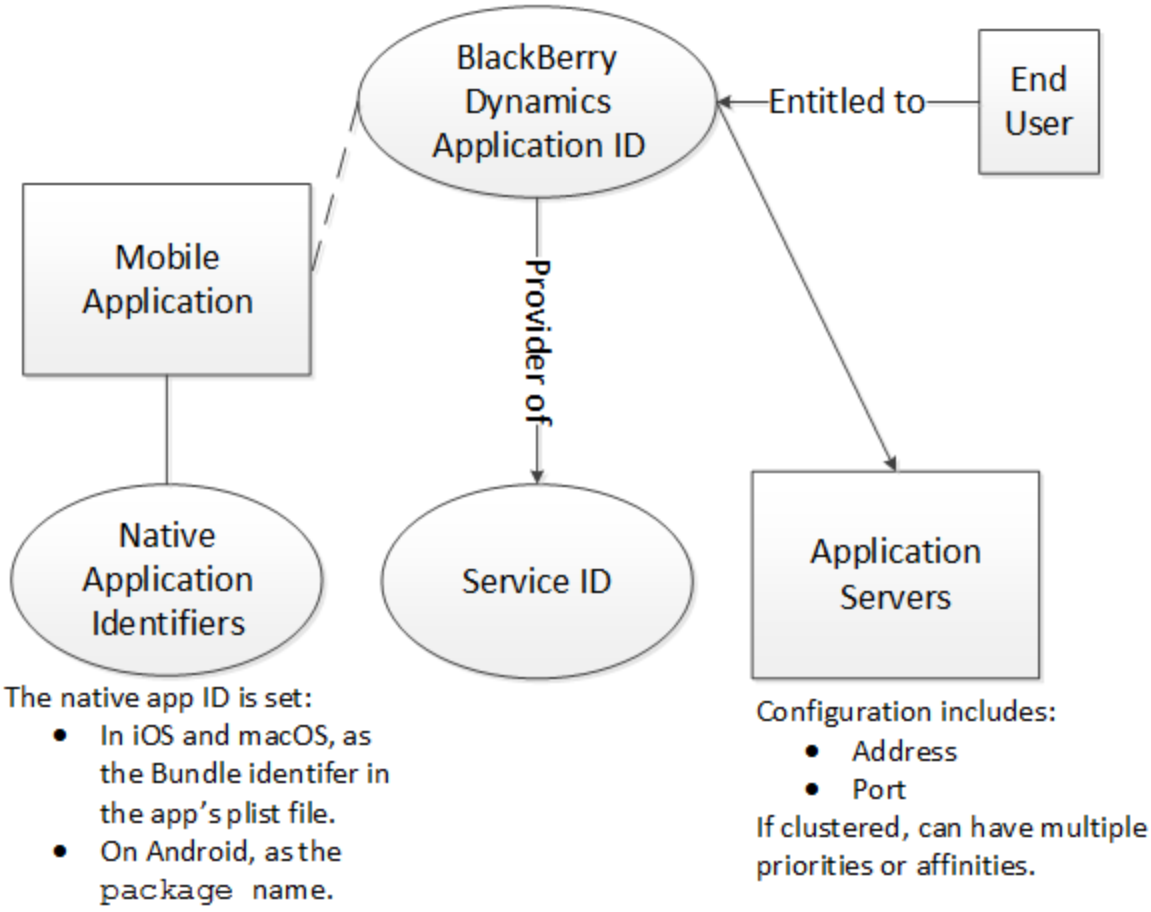
- A mobile BlackBerry Dynamics application that provides one or more application-based services.
- An application server that provides one or more server-based services.

Another developer might program a BlackBerry Dynamics mobile application to consume either or both, depending on what the services offered, for example.

If a customer installed only the application server part of the product, the customer would make use of only the server-based services.

Configuration of server-based service providers

The configuration of a server-based service provider at an enterprise is always with a BlackBerry Dynamics Product Identifier (BlackBerry Dynamics Application ID). In the case of a dual-provider product, this can be an ID of a mobile BlackBerry Dynamics application; otherwise, the ID is merely a placeholder. The relation of the BlackBerry Dynamics Application ID to other identifiers is shown below.



Note: Not all applications require all identifiers. For instance, only an application-based service requires a Native Application Identifier. However, the best practice is to always set all identifiers. This way, if a service needs a particular identifier, it is available.

To set the identifiers, the GC administrator:

- Adds the ID using the application management screens in the GC console, as for an in-house mobile BlackBerry Dynamics application.
- Registers the ID as the provider of the service, as if it were an application-based service provider.

Configuration of server details and end user entitlement

The server address and port number at which the provider is accessible are configured in the same way as the application server assigned to a mobile BlackBerry Dynamics application: the GC administrator enters the values in the GC console. For information about registering a new service, managing a service, binding a service and deleting a service, see the GC online help.

End user entitlement is managed the same way as for a mobile BlackBerry Dynamics application.

Server-based services with BlackBerry Dynamics server clustering and affinities

The provider of a server-based service is an application server like any other in a BlackBerry Dynamics deployment. Thus, BlackBerry Dynamics Server Clustering can be used to deploy multiple instances of the provider of a server-based service.

Likewise, affinities can be used to allocate Good Proxy resources.

The consumer of a server-based service that could be provided by a clustered server must implement a server selection algorithm. See Reference documentation and related resources for more details on BlackBerry Dynamics Server Clustering and Affinities for developers.

In summary: priority must be observed, and random selection within priority is recommended.

Programming with BlackBerry Enterprise Mobility Server services

Described here is a general process for programming with BlackBerry Enterprise Mobility Server services. This task can be divided into the following parts:

- Programming a "consumer" application to interact with the desired BlackBerry Enterprise Mobility Server services.
- Entitling users to the necessary applications.

BlackBerry Enterprise Mobility Server services conform to the BlackBerry Dynamics shared services framework. As such, a service consists of two applications:

1. A program that provides the service.
2. An application that consumes the service.

In the case of BlackBerry Enterprise Mobility Server, the first program, the service provider, is already supplied by BlackBerry but must be configured for use in Good Control.

The second application, the consumer, you yourself must write.

Descriptions of BlackBerry Enterprise Mobility Server services APIs and more

All BlackBerry Enterprise Mobility Server services are detailed at this link: https://community.good.com/community/gdn/features/mobile-services. Of particular interest are the Presence service and the Docs service.

The descriptions include a high-level overview, API details, and sample applications.

Programming your service consumer application

Here is a general approach to programming your service consumer application.

You must define a unique BlackBerry Dynamics App ID for your application. This and other requirements, including details on adding your application to Good Control, are discussed in BlackBerry Device and Application Management.

The BlackBerry Dynamics SDK has functions to discover services, and each BlackBerry Enterprise Mobility Server service has specific programming interfaces.

To discover the BlackBerry Enterprise Mobility Server services, use **`getServiceProvidersFor`**. This API and other APIs for shared services are described in other sections of this guide and in the API References.

After your consumer discovers the service, the exact details of communicating with the service depend on the service definition.

Important: Most BlackBerry Enterprise Mobility Server services run over SSL (HTTPS) on port 8443. Be sure your consumer application connects to the correct server and port.

Configuring BlackBerry Enterprise Mobility Server services in Good Control

Configuring Good Control for BlackBerry Enterprise Mobility Server services has several parts:

- Associating your BlackBerry Enterprise Mobility Server servers with the BlackBerry Enterprise Mobility Server service application.
- Entitling your end-users to the BlackBerry Enterprise Mobility Serverservice application and your consumer application.

Prerequisite

Verify that your Good Control server has the desired BlackBerry Enterprise Mobility Server services:

1. Navigate to **Manage Services**.
2. In the list, find the desired BlackBerry Enterprise Mobility Serverservice.

If you do not see the desired service already predefined, consult the documentation linked in Descriptions of BlackBerry Enterprise Mobility Server services APIs and more .

To associate BlackBerry Enterprise Mobility Server servers with BlackBerry Enterprise Mobility Server services in Good Control:

1. Navigate to **Manage Apps**.
2. Find the desired BlackBerry Enterprise Mobility Server service application and click its name.
3. Click the **BlackBerry Dynamics** tab.
4. For the Server block, click **Edit**.
5. Enter the fully qualified domain name or IP address of your BlackBerry Enterprise Mobility Serverserver and its port (which is 8443 by default).
6. Click **Save** to save your changes or **Cancel** to discard them.

Entitling end-users

The easiest way to entitle your end-users is to use the **Everyone** application group. If you have finer-grained groups, then use them instead.

To entitle end-users to the service application in Good Control:

1. Navigate to **App Groups**.
2. Click the desired group name, such as **Everyone**.
3. Under **Entitled Enterprise Apps**, click **Add More**.
4. In the displayed list, click the checkbox next to the name of the desired GEMS service application, such as **BlackBerry Enterprise Services** (BlackBerry Dynamics APP ID com.good.gdservice-entitlement.enterprise).
5. Click **OK**.

To entitle end-users to your service consumer application, repeat these steps but substitute your own application name.

Discovering the BlackBerry Enterprise Mobility Server doc 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 BlackBerry Enterprise Mobility Server installation.

1. Service identifier

First thing is to know the service identifier and version. Find these details in the following document on the Good Developer Network (GDN):

https://community.good.com/marketplace-services.jspa

Click the service you want to use, in this case Docs Service.

The page shows that the identifier is "com.good.gdservice.enterprise.docs" and the version is "1.0.0.0".

2. Service discovery

Next thing is to code a service discovery query in your application program.

The API for that is getServiceProvidersFor in the GDAndroid, GDiOS, and GDMac classes.

The parameters needed are the service identifier and version from above, and the service provider type specifier for server.

3. Server cluster

The result of the service discovery query is an array of GDServiceProvider objects. Each object corresponds to a BlackBerry Dynamics Application Identifier (also historically known as "GD App ID" and GD entitlement identifier) 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 application 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, which is probably a configuration error by the enterprise. Your application 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.

4. 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 API Reference, in the getApplicationConfig section. The algorithm is the same for Android and for iOS. The recommended selection algorithm is as follows.

For each priority value in the list, starting with the highest:

1. Select a server that has that priority, at random.
2. Attempt to connect to the server.
3. If connection succeeds, use that server.
4. If connection fails, try another server at the same priority, at random.
5. If there are no more untried servers at that priority, try the servers at the next lower priority.

Reference documentation and related resources

BlackBerry Dynamics SDK for Android

- Definition and consumption of services in the BlackBerry Dynamics SDK for Android: https://community.good.com/view-doc.jspa?fileName=namespacecom_1_1good_1_1gd_1_1icc.html&docType=android
- Discovery of services in the BlackBerry Dynamics SDK for Android, including getServiceProvidersFor: https://community.good.com/view-doc.jspa?fileName=classcom_1_1good_1_1gd_1_1_g_d_android.html&docType=android#a0158488867e2fa2ae153c60e3fb42b10
- Push Channel API in the BlackBerry Dynamics SDK for Android:https://community.good.com/view-doc.jspa?fileName=interface_g_d_push_channel.html&docType=api

BlackBerry Dynamics SDK for iOS

- Definition and consumption of services in the BlackBerry Dynamics SDK for iOS: https://community.good.com/view-doc.jspa?fileName=interface_g_d_service.html&docType=api
- Discovery of services in the BlackBerry Dynamics SDK for iOS, inclluding getServiceProvidersFor: https://community.good.com/view-doc.jspa?fileName=interface_g_di_o_s.html&docType=api#a0655c21561a251fa30a61d06de70ebc2
- Push Channel API in the BlackBerry Dynamics SDK for iOS: https://community.good.com/view-doc.jspa?fileName=interface_g_d_push_channel.html&docType=api

BlackBerry Dynamics SDK for macOS

- Consumption of services in BlackBerry Dynamics SDK for macOS: https://community.good.com/view-doc.jspa?fileName=interface_g_d_service_provider.html&docType=mac
- Discovery of services in BlackBerry Dynamics SDK for macOS: https://community.good.com/view-doc.jspa?fileName=interface_g_d_mac.html&docType=mac#a3cd518e9d516fcb9790d795240f63bcd
- Push Channel API in the BlackBerry Dynamics SDK for macOS: https://community.good.com/view-doc.jspa?fileName=interface_g_d_push_channel.html&docType=mac

Service definitions and descriptions

- Documentation of IDL for Application-Based Services (the service definition is the same for both platforms, but the hyperlinks are separate.):
  - iOS: https://community.good.com/view-doc.jspa?fileName=_i_c_c_service_definition.html&docType=api
  - Android: https://community.good.com/view-doc.jspa?fileName=_i_c_c_service_definition.html&docType=android
- Definitions and Descriptions of Published Services: https://community.good.com/marketplace-services.jspa

BlackBerry Dynamics documentation

Category	Title	Description
Cross-platform	• *Getting Started Guide for Marketplace Partners* • *Good Control/Good Proxy Platform Overview for Administrators and Developers*	Overviews of the BlackBerry Dynamics system
	• *Good Control Device and Application Management* • *Good Control Device Management Enrollment: Good Agent for iOS* • *Good Control Device Management Enrollment: Good Agent for Android*	Device and application management on Good Control, including app distribution, with client-side device enrollment details
Security	*BlackBerry Dynamics Security White Paper*	Description of the security aspects of BlackBerry Dynamics
	BlackBerry Dynamics and Fingerprint Authentication	Discussion of the implementation of BlackBerry security with fingerprint recognition systems: Apple Touch ID and Android Fingerprint
Servers	• *BlackBerry Secure Enterprise Planning Guide* • *BlackBerry Secure Servers Compatibility Matrix* • *BlackBerry Performance Calculator*	Guidelines and tools for planning your BlackBerry Secure Enterprise deployment
	Good Control/Good Proxy Server Preinstallation Checklist	Same checklist extracted from the installation guide below
	Good Control/Good Proxy Server Installation	Details on installing Good Control, Good Proxy, and the GC database
	Kerberos Constrained Delegation for Good Control	Configuration details for integrating the Kerberos authentication system with BlackBerry Dynamics
	Direct Connect for Good Control	Configuring BlackBerry Dynamics servers to securely access internal resources from the external Internet
	Good Control Easy Activation Overview	A look at the Easy Activation feature
	Good Control/Good Proxy Server Backup and Restore	Minimal steps for backing up and restoring the BlackBerry Dynamics system
	Good Control Online Help	Printable copy of the GC console online help
	PKI Cert Creation via Good Control: Reference Implementation	A reference implementation in Java for creating end-user PKI certificates via Good Control and a Certificate Authority (CA)

Category	Title	Description
	Good Control Cloud Online Help	Printable copy of the Cloud GC console online help
	Technical Brief: BlackBerry Dynamics Application Policies	Description of formatting application policies for use in Good Control, with examples.
	Good Control Web Services	Programmatic interfaces on Good Control • Basic control and application management: SOAP over HTTPS. Documentation is in the WSDL files included with GC. • Device management: HTTP API (with JSON) for device management. Zipfile of API reference.
Software Development	*Developer Bootstrap: Good Control Essentials*	Bare minimum of information that a developer of BlackBerry Dynamics applications needs to get started with the Good Control server to test applications.
	BlackBerry Dynamics Shared Services Framework	Description of the BlackBerry Dynamics shared services framework for software developers
Android	• *Getting Started with the BlackBerry Dynamics SDK for Android* • *API Reference for Android*	Working with the BlackBerry Dynamics SDK for Android and the essential reference for developers
iOS	• *Getting Started with the BlackBerry Dynamics SDK for iOS* • *API Reference for iOS*	Working with the BlackBerry Dynamics SDK for iOS and the essential reference for developers
macOS	• *Getting Started with the BlackBerry Dynamics SDK for macOS* • *API Reference for macOS*	Working with the BlackBerry Dynamics SDK for macOS and the essential reference for developers
Windows	• *Getting Started with the BlackBerry Dynamics SDK for Universal Windows Platform (UWP)* • *API Reference for UWP*	Working with the BlackBerry Dynamics SDK for Universal Windows Platform (UWP) and the essential reference for developers
iOS, Android	*BlackBerry Launcher Library*	Source code and header files for implementing the popular BlackBerry Launcher interface
Cross-platform	*BlackBerry Dynamics Cordova for iOS and Android*	Working with the BlackBerry Dynamics SDK and the Cordova plugins
	BlackBerry Dynamics Secure HTML5 Bundle Getting Started Guide for Developers	Working with the BlackBerry Dynamics SDK and the secure HTML5 bundle
	BlackBerry Dynamics Bindings for Xamarin for Android and for iOS and the *API Reference for Xamarin.iOS*	Working with the BlackBerry Dynamics SDK and the Xamarin cross-platform integrated development environment

Category	Title	Description
		For Xamarin.Android, no separate API reference is needed; see the standard BlackBerry Dynamics SDK *API Reference for Android*