



Blackberry Web Services legacy SOAP API .NET Development Guide

12.10

Contents

- What are the BlackBerry Web Services?.....4**
 - Benefits of the BlackBerry Web Services SOAP APIs..... 5
 - Capabilities in comparison to the management console..... 6
- Architecture: BlackBerry Web Services..... 8**
- System requirements: Developing apps to use the BlackBerry Web Services SOAP APIs.....10**
- Configuring BlackBerry UEM for development.....11**
 - Add the BlackBerry UEM domain as a trusted authority..... 11
 - Creating administrator accounts that your applications can use..... 12
 - Create a BlackBerry UEM administrator account.....12
- Generating the client proxy files..... 13**
 - Generate the proxy files for the BWS and BWSUtil web services..... 13
- Configuring your development environment.....14**
 - Create a project..... 14
 - Import the BlackBerry Web Services proxy files to your project..... 14
- Using BlackBerry Web Services APIs..... 15**
 - Administrative roles required for using SOAP APIs..... 15
 - Sample walkthrough: Creating a user account.....17
 - Initializing and authenticating with the BlackBerry Web Services SOAP APIs..... 18
 - Creating a user account..... 21
 - Sample walkthrough: Authenticating with the BlackBerry Web Services..... 25
- Legal notice..... 32**

What are the BlackBerry Web Services?

BlackBerry UEM offers a collection of REST APIs and SOAP APIs that you can use to create apps to customize how your organization monitors and manages a BlackBerry UEM domain. You can use the APIs to automate many tasks that administrators typically perform using the UEM management console. For example, you can create an app that automates the process of creating user accounts, adds users to multiple groups, and manages users' devices. Both API collections are installed with BlackBerry UEM.

Note:

The BlackBerry Web Services SOAP APIs are still supported and released with every version of BlackBerry UEM, but are officially in maintenance mode. See the [BlackBerry Web Services 12.9 SOAP API reference](#) for complete information about the supported legacy SOAP APIs. For information about the BlackBerry Dynamics SOAP APIs that are compatible with the BlackBerry Web Services SOAP APIs, see [BlackBerry UEM compatibility with the BlackBerry Dynamics SOAP APIs](#).

If your organization uses the BlackBerry Web Services SOAP APIs, going forward BlackBerry recommends transitioning to the [BlackBerry Web Services REST APIs](#). The REST APIs are updated with new functionality in every BlackBerry UEM release.

APIs	Description
BlackBerry Web Services SOAP APIs	A collection of SOAP web services supported by BlackBerry UEM version 12.4 and later. The SOAP APIs provide your custom apps with access to a variety of UEM management features, including the ability to add and activate user accounts, assign profiles and IT policies, send commands to devices, and so on. The SOAP APIs also provide compatibility with key BlackBerry Dynamics SOAP APIs (GC SOAP and CAP SOAP) for UEM environments that have been integrated with a standalone Good Control server (for more information, see the Compatibility with BlackBerry Dynamics SOAP APIs Reference Guide).
BlackBerry UEM REST APIs	A collection of REST APIs supported by BlackBerry UEM version 12.6 and later. The REST APIs offer custom apps access to a growing list of UEM management features using a RESTful endpoint structure that is accessed using HTTP. You can use REST APIs to manage user accounts, apps, activation passwords, email templates, profiles, and more. The list of available REST APIs will continue to grow with each UEM release. For more information about the REST APIs, see "Getting Started with REST" on the Inside BlackBerry Developer Blog. The REST APIs offer improved performance compared to SOAP and provide full support for JSON objects (requests and returns are formatted in JSON). Many of the EMM functions offered by Good Control are now available as REST calls.

Note: The rest of this guide takes you through the set up and use of the BlackBerry Web Services SOAP APIs. The information you need to get started with the REST APIs can be found in the [BlackBerry UEM REST API Reference](#). You can also see the ["Getting Started Guide for making web services calls" on the Inside BlackBerry Developer Blog](#).

To use the REST APIs or SOAP APIs, you should be proficient in one of the supported programming languages and related concepts. For the SOAP APIs, you should be familiar with the use of SOAP, XML, and WSDL, and for the REST APIs you should be familiar with REST calls and JSON. You should also be familiar with the configuration and administration of UEM, including the management of user accounts, groups, IT policies, profiles, and security settings.

Benefits of the BlackBerry Web Services SOAP APIs

Benefit	Description
Programmatic access to common management tasks	You can use the BlackBerry Web Services to develop applications that manage the BlackBerry UEM domain, user accounts, and all supported devices. You can develop applications that automate and combine several tasks that administrators would typically perform using the management console. The BlackBerry Web Services use SOAP to communicate with authenticated applications. SOAP is platform-independent, which makes it easier to integrate your new applications with existing applications. A request may require authentication from an administrator account before it can be completed. The roles and permissions that are associated with the administrator account determine what APIs the application can use, and what management tasks the application can perform. For example, if you create an application to add user accounts to BlackBerry UEM, the administrator account that the application uses must have permission to create users. To run a BlackBerry Web Services app, computers must have authenticated access to the BlackBerry UEM domain.
Backward and forward compatibility using abstracted data objects	The BlackBerry Web Services use abstracted data objects. Abstracted data objects make application development easier because they separate the implementation of data elements from the interface. The benefit to using these objects is that your apps don't require any changes when you install a new version of BlackBerry UEM.
Security	The BlackBerry Web Services is accessible using the HTTPS protocol. HTTPS provides secure communication, which helps protect your organization's environment from passive and active network attacks while you perform administrative tasks.
Logging	The BlackBerry Web Services write information about API activity to log files. You can use these log files to troubleshoot and debug any issues your app has with requests. Event logs are stored with the core BlackBerry UEM logs, the location of which is configured during installation. For more information about logs, see the BlackBerry UEM Administration Guide.

Capabilities in comparison to the management console

The BlackBerry Web Services SOAP APIs provide access to a subset of the common tasks that administrators can perform using the management console. The sections below describe some of the management tasks that you can and cannot perform using BlackBerry Web Services.

For detailed information about all management tasks, see the [BlackBerry UEM Administration Guide](#).

Actions supported by BlackBerry Web Services SOAP APIs

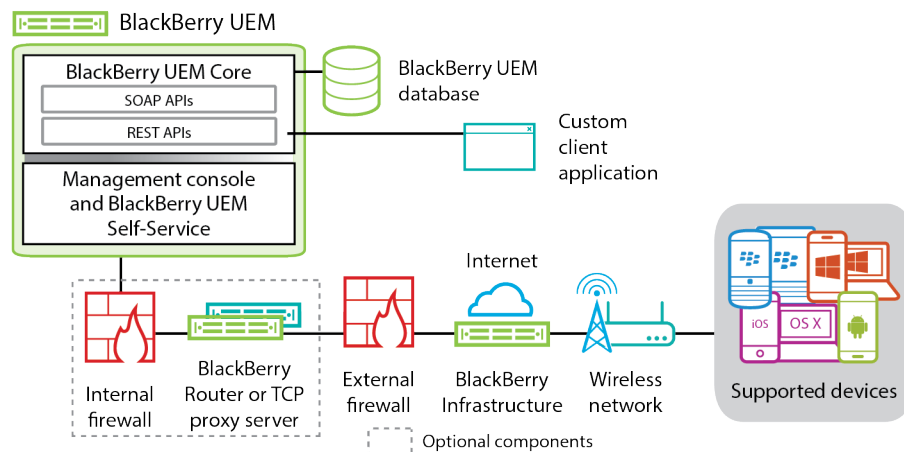
Users and roles • Create and delete users and administrators • Search for users and user info • Look up administrative roles and retrieve information about roles • Assign a role to an administrator account (during user creation) • Retrieve info about administrator permissions	Groups • Create, delete, and look up groups • Retrieve detailed group information • Assign users or remove users from a group • Assign groups or remove groups from a group	Applications • Search for applications that can be distributed to users • Retrieve a list of the applications that are installed on devices • Resend applications to devices
Device activation and control • Set or clear a user's activation password • Search for information about device activations • Retrieve device details • Lock and set a new password for a device • Lock and set a new password for the work space on a BlackBerry device • Delete all device data, or work data only, from a device • Enable or disable the work space for iOS and Android devices • Request that devices update their info with the server	Software configurations • Look up software configurations • Assign a software configuration or remove a software configuration from a group • Assign a software configuration or remove a software configuration from users	Server management • Look up server instances • Retrieve detailed information about the server instances • Look up high availability pools

IT Policies • Look up IT policies • Retrieve information about IT policy rules • Assign an IT policy or remove an IT policy to a group • Assign an IT policy or remove an IT policy from users • Resend an IT policy to users	VPN profiles • Look up VPN profiles • Assign a VPN profile or remove a VPN profile from a group	Wi-Fi profiles • Look up Wi-Fi profiles • Assign a Wi-Fi profile or remove a Wi-Fi profile from a group
Email profiles • Look up email profiles • Assign an email profile to a user and define email settings • Remove an email profile from a user		

Actions not supported by BlackBerry Web Services

Users and roles • Create, change, or delete an administrative role • Assign or remove a role from an existing administrator account	Applications • Create, change, or delete a software configuration • Make applications available to distribute to users • Configure how applications are distributed to devices	Device activation and control • Activate a device • Configure the work space for BlackBerry devices • Create, change, or delete IT policies or work space IT policies
Wi-Fi profiles • Create, change, or delete a Wi-Fi profile • Assign or remove a Wi-Fi profile from users	VPN profiles • Create, change, or delete a VPN profile • Associate a VPN profile with a Wi-Fi profile • Assign or remove a VPN profile from users	Other profiles • Create, change, or delete an email profile • Perform any actions with SCEP or proxy profiles

Architecture: BlackBerry Web Services



SOAP APIs and REST APIs

BlackBerry UEM offers a collection of SOAP APIs and REST APIs that you can use to create apps to customize how your organization monitors and manages a BlackBerry UEM domain. You can use the SOAP APIs or REST APIs to automate many tasks that administrators typically perform using the UEM management console. For example, you can create an app that automates the process of creating user accounts, adds users to multiple groups, and manages users' devices. Both API collections are installed with BlackBerry UEM.

Client application

A custom app that you can develop and integrate with the available SOAP APIs or REST APIs to manage user accounts and devices that are associated with BlackBerry UEM. Your apps use the credentials of an administrator account to authenticate with BlackBerry UEM.

BlackBerry UEM Core

The BlackBerry UEM Core is the central component of the BlackBerry UEM architecture and consists of several subcomponents that are responsible for:

- Logging, monitoring, reporting, and management functions
- Authentication and authorization services for the BlackBerry UEM Core local directory and company directories
- Scheduling and sending commands, policies, and profiles to the devices

If there are multiple BlackBerry UEM instances in the domain, all the BlackBerry UEM Core instances are active and each of them can connect to the BlackBerry Infrastructure and process traffic.

Management console

The management console is a web-based UI that you can use to:

- Complete post-installation configuration settings
- View and manage users, devices, policies, profiles, and apps

- View and manage system settings, including customizing the activation email message or adding an APNs certificate
- Move IT policies, profiles, groups, and users to BlackBerry UEM

BlackBerry Router or TCP proxy

The BlackBerry Router or a TCP proxy server is an optional component that acts as a proxy server for connections over the BlackBerry Infrastructure between BlackBerry UEM and devices.

BlackBerry Infrastructure

The BlackBerry Infrastructure registers user information for device activation and validates licensing information for BlackBerry UEM. Communication between the BlackBerry Infrastructure and BlackBerry UEM is authenticated and encrypted to provide a secure communication channel into your organization for devices outside the firewall.

System requirements: Developing apps to use the BlackBerry Web Services SOAP APIs

To develop applications to use the BlackBerry Web Services SOAP APIs, verify that the following software is installed. Though other software might be supported, it's a best practice to follow the requirements listed below.

Item	Requirement
Operating system	Windows 7 or later
Software development kit (SDK)	Windows SDK The Windows SDK includes all of the utilities that are required to work with web services. The setup application for Microsoft Visual Studio automatically installs the required version of the Windows SDK.
Integrated development environment (IDE)	Microsoft Visual Studio 2005 - 2010
Web service framework	Use the following web service framework to bind web service requests and to generate the required client proxy files: Microsoft .NET Framework 2.0 or later The setup application for Microsoft Visual Studio automatically installs the required version of the Microsoft .NET Framework.

Configuring BlackBerry UEM for development

This section will take you through configuration tasks that will allow your custom apps to access the BlackBerry Web Services SOAP APIs.

You should install one or more instances of BlackBerry UEM to use specifically for testing and debugging your applications. Using a test environment can prevent accidental changes to your organization's production environment. The version of the test UEM instance should match the version used in your production environment, to ensure that the features and functionality of the BlackBerry Web Services SOAP APIs remain the same.

Verify that your development computer has network access to the computers that host the BlackBerry UEM components.

When you are ready to implement your applications in your organization's production environment, consider using a trusted certificate that is signed by a certification authority.

For more information about installing and configuring BlackBerry UEM, see the [BlackBerry UEM Installation and Upgrade Guide](#) and the [BlackBerry UEM Administration Guide](#).

Add the BlackBerry UEM domain as a trusted authority

You must add the SSL certificate of the BlackBerry UEM domain to the Trusted Root Certification Authorities certificate store on your computer. The BlackBerry UEM setup application creates a self-signed SSL certificate during installation. The administrator can replace the self-signed certificate at any time with a trusted certificate signed by a CA.

Your app uses the SSL certificate to authenticate with the BlackBerry Web Services SOAP APIs. You can obtain the certificate from the management console.

Microsoft Visual Studio can access the self-signed certificate in the certificate store that Internet Explorer uses.

Note:

- The following steps can be completed using Internet Explorer 11. Depending on the version of the browser that you're using, the steps might differ. For information about adding certificates using other browsers or versions, see the help for that browser.
- In previous releases of BlackBerry UEM and BES12, the BlackBerry Web Services certificate included the domain name in the Common Name. This practice is no longer supported by certain browsers. For BlackBerry UEM 12.6 MR3 and later, the domain name is now in the SAN of the BlackBerry Web Services certificate. If you upgraded to 12.6 MR3 and your organization has apps that use the BlackBerry Web Services SOAP APIs, you must add the certificate to the trust store again.

1. Run Internet Explorer as an administrator.
2. In the address bar, type the URL for the server where the BlackBerry UEM Core is installed. The address is `https://<BlackBerry_UEM_Core_server_name>:<port>/enterprise/admin/util/ws?wsdl`, where `<server_name>` is the FQDN of the computer that hosts the BlackBerry UEM Core. The default port value is 18084. To check the ports that the BlackBerry UEM setup application assigned, see the [UEM Installation and upgrade content](#).
3. Click **Continue to the website (not recommended)**.
4. On the address bar, click **Certificate error**.
5. Click **View Certificates**.
6. Click **Certification Path**.
7. Click on the root certificate.

8. Click **View Certificate**.
9. Click **Install Certificate**.
10. Click **Next**.
11. Select **Place all certificates in the following store**. Click **Browse**.
12. Click **Trusted Root Certification Authorities**. Click **OK**.
13. Click **Next**.
14. Click **Finish**.
15. Click **Yes** to install the certificate.

Creating administrator accounts that your applications can use

When your app makes calls to the BlackBerry Web Services SOAP APIs, the app must use the login information of a BlackBerry UEM administrator account to authenticate with UEM and authorize its use of the API. You can create an administrator account that is reserved specifically for your custom applications, or you can use an existing account.

Determine the administrative tasks that you want your application to perform, and identify the SOAP APIs that you want your application to use. After you identify the tasks, you can determine the appropriate roles for the administrator account that your application will use. For example, if you want your application to create user accounts, the application needs to use an administrator account that has a role with the Create a user permission. Select a predefined role with the required permissions, or create and assign a custom role. For the security and stability of your domain, you should use a role that is limited to the required permissions. When you develop and test your application, you can modify the role as necessary. To review the list of permissions for predefined roles, see the [BlackBerry UEM Administration Guide](#).

Create a BlackBerry UEM administrator account

Follow these steps in the BlackBerry UEM management console to create a new administrator account.

Before you begin:

- Only a Security Administrator can create an administrator account.. Ask your organization's BlackBerry UEM administrator to perform this task, or give you access to a Security Administrator account.
 - Verify that you have a user account that you want to assign an administrator role to. The user account must have an email address associated with it.
1. Log in to the management console using an administrator account that has the Security Administrator role.
 2. On the menu bar, click **Settings > Administrators**.
 3. Click **Users**.
 4. Click .
 5. Search for and select the user account that you want to make an administrator.
 6. In the **Role** drop-down list, click the role that you want to assign.
 7. Click **Save**.

BlackBerry UEM sends a message to the email address associated with the user account containing the username and a link to the management console. BlackBerry UEM also sends a separate message with the password for the management console. If the user account does not have a console password, BlackBerry UEM generates a temporary password.

After you finish: If necessary, create a custom role and assign the custom role to the administrator. For more information about the roles required to use different SOAP APIs, see [Administrative roles required for using SOAP APIs](#).

Generating the client proxy files

The BlackBerry Web Services SOAP APIs use WSDL files to describe the classes that they expose. To integrate your applications with the BlackBerry Web Services, you must use a proxy generator to generate the client proxy files for the BWS and BWSUtil interfaces.

The BWS and BWSUtil proxy files must be stored together as a combined set of proxy classes in a single source file.

Each release of the BlackBerry Web Services SOAP APIs may introduce new features and functionality, improvements to existing features, and bug fixes. You should generate and use a new set of proxy files whenever your organization implements a new version of BlackBerry UEM, so that your application can leverage the most recent improvements and fixes.

Generate the proxy files for the BWS and BWSUtil web services

To avoid duplication type compiler errors, merge all of the generated proxy files for the BWS and BWSUtil web services into a single proxy file.

Before you begin: Create a folder to store the merged proxy file (for example, C:\Temp\BWSproxy).

1. Run the command prompt as an administrator.
2. Type `cd <file_path>`, where `<file_path>` is the path of the bin folder for your Microsoft SDKs. For example:

```
cd C:\Program Files\Microsoft SDKs\Windows\v7.0A\bin
```

3. Press ENTER.
4. Type `wsdl /sharetypes /o:<proxy_path>\BWSService.cs https://<server_name>:<port>/enterprise/admin/ws?wsdl https://<server_name>:<port>/enterprise/admin/util/ws?wsdl`, where `<proxy_path>` is the path of the proxy files folder, `<server_name>` is the FQDN of the computer that hosts the BlackBerry UEM, and `<port>` is the BlackBerry Web Services port (default 18084).

For example:

```
wsdl /sharetypes /o:C:\Temp\BWSproxy\BWSService.cs https://  
bes_server1.test.rim.net:18084/enterprise/admin/ws?wsdl https://  
bes_server1.test.rim.net:18084/enterprise/admin/util/ws?wsdl
```

5. Press ENTER.

After you finish: New versions of the BlackBerry Web Services SOAP APIs are included with each release of BlackBerry UEM. If your organization's administrator upgrades BlackBerry UEM, repeat the steps above to generate an updated set of proxy files. You can add the new proxy files to a different file path and configure your development environment to use the new proxy files, or you can add the proxy files to the same file path to overwrite the previous set of proxy files.

Configuring your development environment

This section describes how to configure your development environment so that you can integrate your applications with the BlackBerry Web Services SOAP APIs.

Create a project

1. In Microsoft Visual Studio, on the **File** menu, click **New > Project**.
2. In the **New Project** dialog box, select the **Visual C#** project type and the **Console Application** template.
3. In the **Name** field, type a name for the project.
4. Click **OK**.

Import the BlackBerry Web Services proxy files to your project

Import the proxy files for the BWS and BWSUtil web services to make them available for use in your applications.

1. In Microsoft Visual Studio, in the **Solution Explorer** pane, right-click the project and click **Add > Existing Item**.
2. Navigate to the proxy file that you generated (for example, C:\Temp\BWSproxy\BWSService.cs).
3. Click **Add**.

Using BlackBerry Web Services APIs

The [BlackBerry Web Services SOAP API reference](#) describes the interfaces, classes, methods, and data types of the BlackBerry Web Services SOAP APIs. Navigate to the BWS and BWSUtil pages to view the details for each API.

To see some examples of how to use the APIs, visit <https://github.com/blackberry/BWS-Samples> to download sample apps for JavaC# and PowerShell. To read walkthroughs for the samples, see [Sample walkthrough: Creating a user account](#) and [Sample walkthrough: Authenticating with the BlackBerry Web Services](#).

Administrative roles required for using SOAP APIs

An API request can only be completed if the application uses an administrator account with the required permissions. The following tables indicate which preconfigured roles have the permissions that are required for each API. These results have been tested and verified with BlackBerry UEM.

For more information about permissions for preconfigured roles, see the [BlackBerry UEM Administration Guide](#).

BWS interface

API	Security	Enterprise	Senior Helpdesk	Junior Helpdesk	Self service	User with no role
assignEmailProfilesToUser	✓	✓				
assignGroupsToGroup	✓	✓	✓			
assignSWConfigsToGroup	✓	✓	✓			
assignSWConfigsToUser	✓	✓	✓			
assignUsersToGroup	✓	✓	✓			
assignVPNConfigsToGroup	✓	✓	✓			
assignWLANConfigsToGroup	✓	✓	✓			
clearGroupsITPolicy	✓	✓	✓			
clearUsersITPolicy	✓	✓	✓			
createGroups	✓	✓	✓			
createUserEmailProfiles	✓	✓				✓
createUsers - create a user account	✓	✓	✓			
createUsers - create a user account with any available activation type	✓	✓	✓			

API	Security	Enterprise	Senior Helpdesk	Junior Helpdesk	Self service	User with no role
createUsers - create a local user account (not integrated with the user directory)	✓	✓	✓			
createUsers - create an administrator account	✓					
deleteGroups	✓	✓				
deleteUserEmailProfiles	✓	✓				
deleteUsers	✓	✓	✓			
echo	✓	✓	✓	✓	✓	✓
getBESHAPOols	✓	✓				
getDevicesDetail	✓	✓	✓	✓		
getEmailProfiles	✓	✓	✓	✓		
getGroups	✓	✓	✓	✓		
getGroupsDetail	✓	✓	✓	✓		
getGroupsDetail - verbose information	✓	✓	✓	✓		
getITPolicies	✓	✓	✓	✓		
getMailStoreUsers	✓	✓	✓	✓		
getRoles	✓	✓	✓			
getRolesDetail	✓					
getServers	✓	✓				
getServersDetail	✓	✓				
getSWConfigApplications	✓	✓	✓	✓		
getSWConfigs	✓	✓	✓	✓		
getSystemInfo	✓	✓	✓	✓	✓	✓
getUserActivations	✓	✓	✓	✓		
getUsers	✓	✓	✓	✓		

API	Security	Enterprise	Senior Helpdesk	Junior Helpdesk	Self service	User with no role
getUsersDetail	✓	✓	✓	✓		
getVPNConfigs	✓	✓	✓	✓		
getWLANConfigs	✓	✓	✓	✓		
setDevicesLock	✓	✓	✓	✓		
setDevicesPassword	✓	✓	✓	✓		
setDevicesWipe	✓	✓	✓	✓		
setDevicesWorkSpaceState	✓	✓	✓	✓		
setGroupsITPolicy	✓	✓	✓			
setUsersActivationPassword	✓	✓	✓	✓		
setUsersITPolicy	✓	✓	✓			
setUsersResendITPolicy	✓	✓	✓	✓		
setUsersServer	✓	✓				
unassignEmailProfilesFromUser	✓	✓				
unassignSWConfigsFromGroup	✓	✓	✓			
unassignSWConfigsFromUser	✓	✓	✓			
unassignUsersFromGroup	✓	✓	✓			
unassignVPNConfigsFromGroup	✓	✓	✓			
unassignWLANConfigsFromGroup	✓	✓	✓			

BWSUtil interface

All requests in BWSUtil (`getAuthenticators()`, `getEncodedUsername()`, and `getLocales()`) are unauthenticated, so they do not require an administrative role or any administrative permissions.

Sample walkthrough: Creating a user account

The following section looks at parts of the SampleBwsClient.cs application available at <https://github.com/blackberry/BWS-Samples>. This application performs the following tasks:

- Initializes the BWS and BWSUtil web services
- Authenticates the application with the BlackBerry UEM domain

- Collects and displays system information
- Creates a specified user account
- Displays the details for a specified user account

Initializing and authenticating with the BlackBerry Web Services SOAP APIs

Before an application can make calls to the BlackBerry Web Services SOAP APIs, the application must initialize the BWS and BWSUtil web services.

When the BWS and BWSUtil web services are initialized, they accept subsequent API calls from the application. If initialization or authentication of a request are not successful, the application throws an exception. The exception contains a simple text message property that your application can access for more information.

Define metadata

Each method call contains a metadata object that specifies locale, client version, and organization ID data. The inclusion of this metadata helps supports compatibility with different versions of BlackBerry Web Services. To learn more about metadata such as the `ClientVersion`, `Locale`, and `OrgUid`, see the [API Reference](#).

Here's an example of how to create the `RequestMetadata` object and declare the variables used for authentication:

```
// The request Metadata information.
// This is the version of the WSDL used to generate the proxy, not the version of
// the server.
private const string ClientVersion = "12.0.1";

// To use a different locale, call getLocales() in the BWSUtilService web service
// to see which locales are supported.
private const string Locale = "en_US";
private const string OrgUid = "0";
private static readonly RequestMetadata Metadata = new RequestMetadata();

// Authentication type name.
private const string AuthenticatorName = "BlackBerry Administration Service";

// Hostname to use when connecting to web service. Includes port.
private static string BWSHostName = null; // e.g. BWSHostName =
"server01.yourcompany.net:18084"
private static string Username = null; // e.g. Username = "admin"
private static string Password = null; // e.g. Password = "password"
```

To send requests, you must provide authentication info, including the host name, user name, and password values.

```
// Hostname to use when connecting to web service.
BWSHostName = "<BWSHostName>"; // e.g. BWSHostName = "server01.yourcompany.net".
Username = "<username>"; // e.g. Username = "admin".
Password = "<password>"; // e.g. Password = "password".
```

Assign values to the Metadata object

Here's how to set the client version, locale, and organization ID of the metadata object that was previously created:

```
Metadata.clientVersion = ClientVersion;
Metadata.locale = Locale;
Metadata.organizationUid = OrgUid;
```

Initialize and set the URL properties of the web services

Next, you initialize and set the values for the URL properties of the web services so that the application can connect to the BlackBerry Web Services.

```
logMessage("Initializing BWS web service stub");
bwsService = new BWSService();
logMessage("BWS web service stub initialized");
logMessage("Initializing BWSUtil web service stub");
bwsUtilService = new BWSUtilService();
logMessage("BWSUtil web service stub initialized");
// These are the URLs that point to the web services used for all calls.
bwsService.Url = "https://" + BWSHostName + "/enterprise/admin/ws";
bwsUtilService.Url = "https://" + BWSHostName + "/enterprise/admin/util/ws";

// Set the connection timeout to 60 seconds.
bwsService.Timeout = 60000;
bwsUtilService.Timeout = 60000;
```

Define the authenticator object

The following code defines the Authenticator object that the application requires for the initialization and authentication process. In the sections that follow, the application uses the authenticator object to collect the login information and the encoded user name that the application uses to authenticate with the BlackBerry Web Services.

```
Authenticator authenticator = GetAuthenticator(AuthenticatorName);
if (authenticator != null)
{
    string encodedUsername = GetEncodedUserName(Username, authenticator);
    if (!string.IsNullOrEmpty(encodedUsername))
    {
        /*
        * Set the HTTP basic authentication on the BWS service.
        * BWSUtilService is a utility web service that does not require
        * authentication.
        */
        bwsService.Credentials = new NetworkCredential(encodedUsername, Password);

        /*
        * Send an HTTP Authorization header with requests after authentication
        * has taken place.
        */
        bwsService.PreAuthenticate = true;
        returnValue = true;
    }
}
```

```

    }
    else
    {
        logMessage("'encodedUsername' is null or empty");
    }
}
else
{
    logMessage("'authenticator' is null");
}

```

Authenticate with the BlackBerry Web Services

After the Authenticator object is created, here's how to retrieve the encoded login information for the administrator account that the app uses, and authenticate the app with the BlackBerry Web Services.

```

public static string GetEncodedUserName(string username, Authenticator
    authenticator)
{
    const string methodName = "GetEncodedUserName()";
    const string bwsApiName = "bwsUtilService.getEncodedUsername()";
    logMessage("Entering {0}", methodName);
    string returnValue = null;

    GetEncodedUsernameRequest request = new GetEncodedUsernameRequest();
    request.metadata = Metadata;
    request.username = username;
    request.orgUid = Metadata.organizationUid;
    request.authenticator = authenticator;

    CredentialType credentialType = new CredentialType();
    credentialType.PASSWORD = true;
    credentialType.value = "PASSWORD";
    request.credentialType = credentialType;

    GetEncodedUsernameResponse response = null;

    try
    {
        logRequest(bwsApiName);
        response = bwsUtilService.getEncodedUsername(request);
        logResponse(bwsApiName, response.returnStatus.code, response.metadata);
    }
    catch (WebException e)
    {
        // Log and re-throw exception.
        logMessage("Exiting {0} with exception \"{1}\"", methodName, e.Message);
        throw e;
    }

    if (response.returnStatus.code.Equals("SUCCESS"))
    {
        returnValue = response.encodedUsername;
    }
    else
    {
        logMessage("Error Message: \"{0}\"", response.returnStatus.message);
    }
}

```

```

    try
    {
        logMessage("Decoded value of encoded username \"{0}\"",
            Encoding.Default.GetString(Convert.FromBase64String(returnValue)));
    }
    catch (Exception)
    {
        // Not actually base64 encoded. Show actual value
        logMessage("Value of encoded username \"{0}\"", returnValue);
    }
    logMessage("Exiting {0}", methodName);
    return returnValue;
}

```

When the app completes the initialization and authentication process, the BlackBerry Web Services are ready to accept API calls from the application.

Creating a user account

Now that initialization is complete, the app can send a request to create a new user account.

These sections explain the types of user accounts that you can create using the BWS.createUsers API, and highlight the section of the code sample that is used to create a new user account.

Specify the email address for the new user

The first step to create a new user is to specify the email address to associate with the user.

```

/*
 * Email address used to create a new user with the createUsers() API call.
 * This value must exactly match the full string value in the directory for
 * successful user creation.
 */
CreateNewUserEmail = "user02@example.net\";

```

Create the user

Next, you create a CreateUsersRequest object to send the request to the BlackBerry Web Services, and assign the Metadata object that you previously created to its metadata property. The NewUser object that represents the user and the AccountAttributes object contains information about the user, like the email address

```

// Create the request object.
CreateUsersRequest createUsersRequest = new CreateUsersRequest();
createUsersRequest.metadata = Metadata;

NewUser newUser = new NewUser();

// To create an administrator user, create and set the "UserAttributes" and the
// roleUid field.
AccountAttributes accountAttributes = new AccountAttributes();

logMessage("Email address set to \"{0}\"", CreateNewUserEmail);

// Value of the variable "CreateNewUserEmail" is used to create the user.

```

```

accountAttributes.emailAddress = CreateNewUserEmail;

newUser.accountAttributes = accountAttributes;

// Server value is validated and then ignored.
newUser.server = null;

```

Prepare and send the API call

Lastly, you prepare and send the request. The app adds the user to the `CreateUsersRequest`, `BWS.createUsersRequest()` is called to send the request, and the result of the request is verified and output to the console.

```

List<NewUser> newUsers = new List<NewUser>();
newUsers.Add(newUser);
createUsersRequest.newUsers = newUsers.ToArray();

CreateUsersResponse response = null;

try
{
    logRequest(bwsApiName);
    response = bwsService.createUsers(createUsersRequest);
    logResponse(bwsApiName, response.returnStatus.code, response.metadata);
}
catch (WebException e)
{
    // Log and re-throw exception.
    logMessage("Exiting {0} with exception \"{1}\"", methodName, e.Message);
    throw e;
}

if (response.returnStatus.code.Equals("SUCCESS"))
{
    if (response.individualResponses != null)
    {
        foreach (IndividualResponse individualResponse in response.individualResponses)
        {
            displayResult("User created with UID \"{0}\"", individualResponse.uid);
            displayResult("Email address used in creation is \"{0}\"",
                accountAttributes.emailAddress);
        }

        returnValue = true;
    }
}
else
{
    logMessage("Error Message: \"{0}\"", response.returnStatus.message);
    if (response.individualResponses != null)
    {
        foreach (IndividualResponse individualResponse in response.individualResponses)
        {
            logMessage("Individual Response - Code: \"{0}\"", Message: \"{1}\"",
                individualResponse.returnStatus.code,
                individualResponse.returnStatus.message);
        }
    }
}

```

```
}
```

Types of user accounts

The table below describes the types of user accounts that you can create. The sample application creates a directory user account. For more information about creating user accounts, see `BWS.createUsers()` in the [API Reference](#).

The `BWS.createUsers()` API passes a **CreateUsersRequest** object in its request.

The **CreateUsersRequest** object contains the metadata for the request, and **NewUser** objects that represent the user accounts that you want to create. **NewUser** contains the following properties:

- **AccountAttributes**: The account attributes that distinguish the user, such as the user's email address.
- **DeviceActivationType**: Not currently supported.
- **UserAttributes**: Contains the authentication information for administrator accounts or local user accounts.
- **Server**: Not currently supported. Should be set to null.

User type	Configuration of NewUser in CreateUsersRequest
Directory user • A standard user that does not require login information for the administration console. • An administrator can assign a device to the user, or the user can activate a device.	AccountAttributes Specify any of the following fields: • A valid email address for emailAddress • An identifier from your organization's directory (Microsoft Active Directory or LDAP) for externalUserId • A valid distinguished name for userDistinguishedName DeviceActivationType • Not currently supported. UserAttributes • Null Server • Not currently supported.
Local user • A local user account that is not integrated with the directory. • An administrator can assign a device to the user, or the user can activate a device.	AccountAttributes • Set localUser to true • Optional: A valid email address for emailAddress DeviceActivationType • Not currently supported. UserAttributes • For authenticator, set authenticatorType to internal (local user accounts don't support directory authentication). • Specify loginName, loginPassword, and displayName. Server • Not currently supported.

User type	Configuration of NewUser in CreateUsersRequest
Administrator account - An administrator user that is assigned login information and an administrative role. - An administrator can assign a device to the user.	AccountAttributes Specify any of the following fields: - A valid email address for emailAddress - An identifier from your organization's directory (Microsoft Active Directory or LDAP) for externalUserId - A valid distinguished name for userDistinguishedName DeviceActivationType - Not currently supported. UserAttributes - Specify authenticator to select the type of authentication. You can retrieve a list of supported authenticators using the <code>BWSUtil.getAuthenticators</code> API. - For local user authentication, specify loginName, loginPassword, and displayName. - For Microsoft Active Directory authentication, specify loginName, domain, and displayName. - For LDAP authentication, specify loginName and displayName. - Specify roleUid to select the administrative role. You can retrieve a list of available roles using <code>BWS.getRoles()</code>. Server - Not currently supported.
Administrator account with permissions granted by group membership - A user with login information, but no administrative role. - Create this type of user only if you want to give the user administrative permissions by adding the user to a group with an administrative role.	AccountAttributes - Null DeviceActivationType - Not currently supported. UserAttributes - Specify authenticator to select the type of authentication. You can retrieve a list of supported authenticators using <code>BWSUtil.getAuthenticators()</code>. - For local user authentication, specify loginName, loginPassword, and displayName. - For Microsoft Active Directory authentication, specify loginName, domain, and displayName. - For LDAP authentication, specify loginName and displayName. - Do not specify roleUid. Server - Not currently supported.

Sample walkthrough: Authenticating with the BlackBerry Web Services

The following section examines the AuthenticationSample.cs application available at <https://github.com/blackberry/BWS-Samples>. This application demonstrates the different methods for authenticating with BlackBerry UEM.

Before an application can make calls to the BlackBerry Web Services, the application must initialize the BWS and BWSUtil web services and authenticate with the web services using the login information of an administrator account. When the BWS and BWSUtil web services are initialized, they accept subsequent API calls from the application.

Administrator accounts can use one of the following authentication methods:

- Local user authentication (that is, non-directory users)
- Microsoft Active Directory authentication
- LDAP authentication

Note: Although the sample application also includes an option for SSO authentication, this feature is not supported in BlackBerry UEM and is only included to support BlackBerry Enterprise Service 10.

Administrators select the authentication method when they create new administrator accounts. Single sign-on authentication requires an administrator to complete additional configuration steps. For more information about creating an administrator account, the different authentication types, and configuring single sign-on authentication, see the [BlackBerry UEM Documentation](#).

Define metadata

Like any BlackBerry Web Services request, you must first define the metadata that describes the application's requests. Each metadata object specifies the locale, client version, and organization ID data. The inclusion of this metadata supports forward and backward compatibility with different versions of BlackBerry Web Services. To learn more about metadata values such as the ClientVersion, Locale, and OrgUid, see the [API Reference overview](#).

Here's an example of how to create the RequestMetadata object and declare the variables used for authentication:

```
// The request Metadata information.
// This is the version of the WSDL used to generate the proxy, not the version of
// the server.
private const string CLIENT_VERSION = "<Client Version>"; // e.g. CLIENT_VERSION =
"12.0."

// The enum used to determine the current server type.
private enum ServerType { Unknown, BDS, UDS, BES12 } ;

// Enum used to determine the server used in this execution
private static ServerType serverType = ServerType.Unknown;

/*
 * To use a different locale, call getLocales() in the BWSUtilService web service
 * to see which locales are supported.
 */
private const string LOCALE = "en_US";
private const string ORG_UID = "0";
private static readonly RequestMetadata REQUEST_METADATA = new RequestMetadata();
```

Get the login information

Next, the app retrieves the encoded login information and domain data (if necessary) for the administrator account that the application uses to authenticate with the BlackBerry Web Services, and defines the possible log data and status messages.

```
public static string getEncodedUserName(String username, Authenticator
    authenticator,
    CredentialType credentialType, String domain)
{
    const string METHOD_NAME = "getEncodedUserName()";
    const string BWS_API_NAME = "bwsUtilService.getEncodedUsername()";
    logMessage("Entering {0}", METHOD_NAME);
    string returnValue = null;

    GetEncodedUsernameRequest request = new GetEncodedUsernameRequest();
    request.metadata = REQUEST_METADATA;
    request.username = username;
    request.orgUid = REQUEST_METADATA.organizationUid;
    request.authenticator = authenticator;

    request.credentialType = credentialType;
    request.domain = domain;

    GetEncodedUsernameResponse response = null;
    try
    {
        logRequest(BWS_API_NAME);
        response = bwsUtilService.getEncodedUsername(request);
        logResponse(BWS_API_NAME, response.returnStatus.code, response.metadata);
    }
    catch (WebException e)
    {
        // Log and re-throw exception.
        logMessage("Exiting {0} with exception \"{1}\"", METHOD_NAME, e.Message);
        throw e;
    }

    if (response.returnStatus.code.Equals("SUCCESS"))
    {
        returnValue = response.encodedUsername;
    }
    else
    {
        logMessage("Error Message: \"{0}\"", response.returnStatus.message);
    }

    // BES12 returns a Base64 encoded string while BES10 does not. As a result, a
    // BES10 encoded username
    // throws an exception when an attempt is made to decode from base64. The try
    // block will log the decoded
    // username if run against a BES12 system and the catch then logs the actual
    // value returned by BES10.
    try
    {
        logMessage("Decoded value of encoded username \"{0}\"",
            Encoding.Default.GetString(Convert.FromBase64String(returnValue)));
    }
    catch (Exception)
    {
    }
}
```

```

        // Not actually base64 encoded. Show actual value
        logMessage("Value of encoded username \"{0}\"", returnValue);
    }
    logMessage("Exiting {0}", METHOD_NAME);
    return returnValue;
}

```

Perform an echo call

To verify that the app can call and get a response from the BlackBerry Web Services, you can call `BWS.echo()`. The following code demonstrates how to call `echo()` and handle the return value.

```

private static bool echo()
{
    const string METHOD_NAME = "echo()";
    const string BWS_API_NAME = "bwsService.echo()";
    logMessage("Entering {0}", METHOD_NAME);

    bool returnValue = true;

    EchoRequest request = new EchoRequest();
    EchoResponse response = null;

    request.metadata = REQUEST_METADATA;
    request.text = "Hello World!";

    try
    {
        logRequest(BWS_API_NAME);
        response = bwsService.echo(request);
        logResponse(BWS_API_NAME, response.returnStatus.code,
            response.metadata);
    }
    catch (WebException e)
    {
        if (e.Status == WebExceptionStatus.ProtocolError)
        {
            HttpResponseMessage httpResponse = (HttpResponseMessage)
                e.Response;
            if (httpResponse != null
                && httpResponse.StatusCode ==
                HttpStatusCode.Unauthorized)
            {
                returnValue = false;
                logMessage("Failed to authenticate with the
                    BWS web service");
                logMessage("Exiting {0} with value \"{1}\"",
                    METHOD_NAME, returnValue);
                return returnValue;
            }
        }

        // Log and re-throw exception.
        logMessage("Exiting {0} with exception \"{1}\"", METHOD_NAME,
            e.Message);
        throw e;
    }
}

```

```

        logMessage("Exiting {0} with value \"{1}\"", METHOD_NAME, returnValue);
        return returnValue;
    }

```

Provide the server name and port number

To authenticate with BlackBerry Web Services you must first specify the FQDN of the computer that hosts the BlackBerry UEM along with the port number.

```

static int Main(string[] args)
{
    startTime.Start();

    // Return codes.
    const int SUCCESS = 0;
    const int FAILURE = 1;

    int returnCode = SUCCESS;

    // Hostname to use when connecting to web service. Must contain the fully
    // qualified domain name.
    String bwsHostname = "<bwsHostname>"; // e.g. bwsHostname =
    "server01.example.net"

    // Port to use when connecting to web service. The same port is used to access
    // the
    // webconsole prior to BES12. Default ports: BES10 BDS=38443, BES10 UDS=18082,
    // BES12=18084
    String bwsPort = "<bwsPort>"; // e.g. bwsPort = "18084"

```

Next, you can select the authentication method that you want to use. In this example, local user authentication is always called, in addition to the directory method that you select below. To select a method, set the option to **true** and the other options to **false**. For example, for LDAP sign-on authentication, set `useLDAP = true` and `useAD = false`.

```

// Select which authentication methods you would like to test by setting the
// variables to true
bool useAD = false; // Active Directory
bool useLDAP = true; // LDAP

```

The following sections describe each of the different authentication methods described above. You can configure the section that corresponds to your selected authentication method with your own credentials.

Local user authentication

The following code defines the login information for local users (otherwise known as non-directory users). Specify the `<username>` and `<password>` values for the administrator account.

```

// The BlackBerry Administration Service Credentials to use
String username = "<username>"; // e.g. username = "admin".
String password = "<password>"; // e.g. password = "password".
String authenticatorName = "BlackBerry Administration Service";
String domain = null; // not needed

```

Microsoft Active Directory authentication

The following code defines the login information for Microsoft Active Directory authentication. Specify the `<username>`, `<password>`, and `<domain>` values.

```
// The Active Directory Credentials to use
String username = "<username>"; // e.g. username = "admin"
String password = "<password>"; // e.g. password = "password"
String authenticatorName = "Active Directory";
String activeDirectoryDomain = null;
// Only BDS requires domain for authentication
if (serverType == ServerType.BDS)
{
    activeDirectoryDomain = "<domain>"; // e.g. activeDirectoryDomain = "example.net"
```

LDAP authentication

The following code defines the login information for LDAP authentication. Specify the `<username>` and `<password>` values.

```
// The LDAP Credentials to use
String username = "<username>"; // e.g. username = "admin"
String password = "<password>"; // e.g. password = "password"
String authenticatorName = "LDAP";
String domain = null; // not needed
```

Authenticate with the BlackBerry Web Services

The following code tests whether the application can authenticate with the BlackBerry Web Services using the login information that you specified.

```
private static bool demonstrateBwsSetupAndAuthenticatedCall(String bwsHostname,
    String bwsPort,
    String username, String password, String domain, String authenticatorName,
    CredentialType credentialType)
{
    bool returnCode = false;
    logMessage("Initializing web services...");
    if (setup(bwsHostname, bwsPort, username, password, authenticatorName,
        credentialType, domain))
    {
        /*
         * It is anticipated that the first time through this method, _serverType
         * will be unknown.
         * So getSystemInfo() will populate this value, which will be used in the
         * subsequent
         * demonstrate calls if required.
         */
        if (serverType == ServerType.Unknown)
        {
            GetSystemInfo();
        }
        /*
         * Demonstrate authenticated call to bwsService.echo() API.
         */
        logMessage("Attempting authenticated BWS call to echo()...");
```

```

        if (echo())
        {
            logMessage("Authenticated call succeeded!");
            returnCode = true;
        }
        else
        {
            logMessage("Authenticated call failed!");
        }
    }
    else
    {
        logMessage("Error: setup() failed");
    }
    return returnCode;
}

```

Assign values to the Metadata global object

The following code assigns the metadata values to the Metadata object.

```

private static bool setup(String hostname, String bwsPort, String username, String
password,
    String authenticatorName, CredentialType credentialType, String domain)
{
    const string METHOD_NAME = "setup()";
    logMessage("Entering {0}", METHOD_NAME);
    bool returnValue = false;

    REQUEST_METADATA.clientVersion = CLIENT_VERSION;
    REQUEST_METADATA.locale = LOCALE;
    REQUEST_METADATA.organizationUid = ORG_UID;
}

```

Initialize and set the URL properties of the web services

The following code initializes and sets the values for the URL properties of the web services so that the application can connect to the BlackBerry Web Services.

```

logMessage("Initializing BWS web service stub");
bwsService = new BWSService();
logMessage("BWS web service stub initialized");
logMessage("Initializing BWSUtil web service stub");
bwsUtilService = new BWSUtilService();
logMessage("BWSUtil web service stub initialized");
// These are the URLs that point to the web services used for all calls.
// e.g. with no port:
// https://server01.example.net/enterprise/admin/ws
// e.g. with port:
// https://server01.example.net:18084/enterprise/admin/ws

String port = "";

if (bwsPort != null)
{
    port = ":" + bwsPort;
}

```

```

bwsService.Url = "https://" + hostname + port + "/enterprise/admin/ws";
bwsUtilService.Url = "https://" + hostname + port + "/enterprise/admin/util/ws";

// Set the connection timeout to 60 seconds.
bwsService.Timeout = 60000;
bwsUtilService.Timeout = 60000;

```

Define the authenticator object

The following code defines the Authenticator object that the application requires for the overall authentication and initialization process.

```

Authenticator authenticator = getAuthenticator(authenticatorName);
if (authenticator != null)
{
    string encodedUsername = getEncodedUserName(username, authenticator,
credentialType, domain);
    if (!string.IsNullOrEmpty(encodedUsername))
    {
        /*
        * Set the HTTP basic authentication on the BWS service.
        * BWSUtilService is a utility web service that does not require
        * authentication.
        */
        bwsService.Credentials = new NetworkCredential(encodedUsername, password);

        /*
        * Send an HTTP Authorization header with requests after authentication
        * has taken place.
        */
        bwsService.PreAuthenticate = true;
        returnValue = true;
    }
    else
    {
        logMessage("'encodedUsername' is null or empty");
    }
}
else
{
    logMessage("'authenticator' is null");
}

logMessage("Exiting {0} with value \"{1}\"", METHOD_NAME, returnValue);
return returnValue;

```

Legal notice

©2018 BlackBerry Limited. Trademarks, including but not limited to BLACKBERRY, BBM, BES, EMBLEM Design, ATHOC, MOVIRTU and SECUSMART are the trademarks or registered trademarks of BlackBerry Limited, its subsidiaries and/or affiliates, used under license, and the exclusive rights to such trademarks are expressly reserved. All other trademarks are the property of their respective owners.

Android is a trademark of Google Inc. iOS is a trademark of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries. iOS® is used under license by Apple Inc. Java is a trademark of Oracle and/or its affiliates. Microsoft, Windows, Windows Internet Explorer, Microsoft .NET Framework, Microsoft Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners.

This documentation including all documentation incorporated by reference herein such as documentation provided or made available on the BlackBerry website provided or made accessible "AS IS" and "AS AVAILABLE" and without condition, endorsement, guarantee, representation, or warranty of any kind by BlackBerry Limited and its affiliated companies ("BlackBerry") and BlackBerry assumes no responsibility for any typographical, technical, or other inaccuracies, errors, or omissions in this documentation. In order to protect BlackBerry proprietary and confidential information and/or trade secrets, this documentation may describe some aspects of BlackBerry technology in generalized terms. BlackBerry reserves the right to periodically change information that is contained in this documentation; however, BlackBerry makes no commitment to provide any such changes, updates, enhancements, or other additions to this documentation to you in a timely manner or at all.

This documentation might contain references to third-party sources of information, hardware or software, products or services including components and content such as content protected by copyright and/or third-party websites (collectively the "Third Party Products and Services"). BlackBerry does not control, and is not responsible for, any Third Party Products and Services including, without limitation the content, accuracy, copyright compliance, compatibility, performance, trustworthiness, legality, decency, links, or any other aspect of Third Party Products and Services. The inclusion of a reference to Third Party Products and Services in this documentation does not imply endorsement by BlackBerry of the Third Party Products and Services or the third party in any way.

EXCEPT TO THE EXTENT SPECIFICALLY PROHIBITED BY APPLICABLE LAW IN YOUR JURISDICTION, ALL CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS, OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS OR WARRANTIES OF DURABILITY, FITNESS FOR A PARTICULAR PURPOSE OR USE, MERCHANTABILITY, MERCHANTABLE QUALITY, NON-INFRINGEMENT, SATISFACTORY QUALITY, OR TITLE, OR ARISING FROM A STATUTE OR CUSTOM OR A COURSE OF DEALING OR USAGE OF TRADE, OR RELATED TO THE DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE, HARDWARE, SERVICE, OR ANY THIRD PARTY PRODUCTS AND SERVICES REFERENCED HEREIN, ARE HEREBY EXCLUDED. YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY BY STATE OR PROVINCE. SOME JURISDICTIONS MAY NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES AND CONDITIONS. TO THE EXTENT PERMITTED BY LAW, ANY IMPLIED WARRANTIES OR CONDITIONS RELATING TO THE DOCUMENTATION TO THE EXTENT THEY CANNOT BE EXCLUDED AS SET OUT ABOVE, BUT CAN BE LIMITED, ARE HEREBY LIMITED TO NINETY (90) DAYS FROM THE DATE YOU FIRST ACQUIRED THE DOCUMENTATION OR THE ITEM THAT IS THE SUBJECT OF THE CLAIM.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, IN NO EVENT SHALL BLACKBERRY BE LIABLE FOR ANY TYPE OF DAMAGES RELATED TO THIS DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE, HARDWARE, SERVICE, OR ANY THIRD PARTY PRODUCTS AND SERVICES REFERENCED HEREIN INCLUDING WITHOUT LIMITATION ANY OF THE FOLLOWING DAMAGES: DIRECT, CONSEQUENTIAL, EXEMPLARY, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR AGGRAVATED DAMAGES, DAMAGES FOR LOSS OF PROFITS OR REVENUES, FAILURE TO REALIZE ANY EXPECTED SAVINGS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF BUSINESS OPPORTUNITY, OR CORRUPTION OR LOSS OF DATA, FAILURES TO TRANSMIT OR RECEIVE ANY DATA,

PROBLEMS ASSOCIATED WITH ANY APPLICATIONS USED IN CONJUNCTION WITH BLACKBERRY PRODUCTS OR SERVICES, DOWNTIME COSTS, LOSS OF THE USE OF BLACKBERRY PRODUCTS OR SERVICES OR ANY PORTION THEREOF OR OF ANY AIRTIME SERVICES, COST OF SUBSTITUTE GOODS, COSTS OF COVER, FACILITIES OR SERVICES, COST OF CAPITAL, OR OTHER SIMILAR PECUNIARY LOSSES, WHETHER OR NOT SUCH DAMAGES WERE FORESEEN OR UNFORESEEN, AND EVEN IF BLACKBERRY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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

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

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

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

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

BlackBerry 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