



Blackberry Web Services legacy SOAP API Java 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
Creating a keystore for your applications	13
Download the SSL certificate of the BlackBerry UEM domain.....	13
Generate a keystore and import the SSL certificate into the keystore.....	13
Configure the proxy generator to access the keystore.....	14
Generating the client proxy files	15
Generate the proxy files for the BWS and BWSUtil web services.....	15
Configuring your development environment	17
Create a project.....	17
Configure the VM arguments in your project.....	17
Import Apache CXF libraries to your Eclipse project.....	18
Import the BlackBerry Web Services proxy files to your project.....	18
Using BlackBerry Web Services APIs	19
Administrative roles required for using SOAP APIs.....	19
Sample walkthrough: Creating a user account.....	21
Initializing and authenticating with the BlackBerry Web Services SOAP APIs.....	22
Creating a user account.....	25
Sample walkthrough: Authenticating with the BlackBerry Web Services.....	30
Legal notice	37

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	<p>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.</p> <p>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).</p>
BlackBerry UEM REST APIs	<p>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.</p> <p>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.</p> <p>The REST APIs offer improved performance compared to SOAP and provide full support for JSON objects (requests and returns are formatted in JSON).</p> <p>Many of the EMM functions offered by Good Control are now available as REST calls.</p>

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
<p>Programmatic access to common management tasks</p>	<p>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.</p> <p>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.</p> <p>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.</p> <p>To run a BlackBerry Web Services app, computers must have authenticated access to the BlackBerry UEM domain.</p>
<p>Backward and forward compatibility using abstracted data objects</p>	<p>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.</p>
<p>Security</p>	<p>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.</p>
<p>Logging</p>	<p>The BlackBerry Web Services write information about API activity to log files. You can use these log files to troubleshoot and debug any issues you app has with requests.</p> <p>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.</p>

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

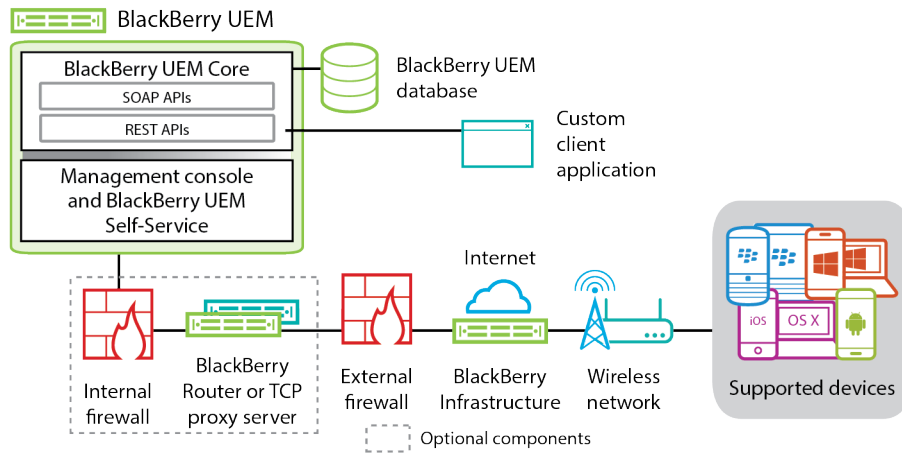
Users and roles <ul style="list-style-type: none">• Create and delete users and administrators• Search for users and user info• Look up administrative roles and retrieve information about roles• Assign a role to an administrator account (during user creation)• Retrieve info about administrator permissions	Groups <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• Look up server instances• Retrieve detailed information about the server instances• Look up high availability pools

<p>IT Policies</p> <ul style="list-style-type: none"> • 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 	<p>VPN profiles</p> <ul style="list-style-type: none"> • Look up VPN profiles • Assign a VPN profile or remove a VPN profile from a group 	<p>Wi-Fi profiles</p> <ul style="list-style-type: none"> • Look up Wi-Fi profiles • Assign a Wi-Fi profile or remove a Wi-Fi profile from a group
<p>Email profiles</p> <ul style="list-style-type: none"> • 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

<p>Users and roles</p> <ul style="list-style-type: none"> • Create, change, or delete an administrative role • Assign or remove a role from an existing administrator account 	<p>Applications</p> <ul style="list-style-type: none"> • Create, change, or delete a software configuration • Make applications available to distribute to users • Configure how applications are distributed to devices 	<p>Device activation and control</p> <ul style="list-style-type: none"> • Activate a device • Configure the work space for BlackBerry devices • Create, change, or delete IT policies or work space IT policies
<p>Wi-Fi profiles</p> <ul style="list-style-type: none"> • Create, change, or delete a Wi-Fi profile • Assign or remove a Wi-Fi profile from users 	<p>VPN profiles</p> <ul style="list-style-type: none"> • Create, change, or delete a VPN profile • Associate a VPN profile with a Wi-Fi profile • Assign or remove a VPN profile from users 	<p>Other profiles</p> <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none">Windows 7 or later
Software development kit (SDK)	<ul style="list-style-type: none">Java SE Development Kit 8 <p>The Java SE Development Kit includes the Java Platform, Standard Edition (Java SE), the Java Runtime Environment (JRE), and the Java keytool (keytool.exe), which you use to create the required keystore.</p> <p>Visit www.oracle.com to download the Java SE Development Kit. Verify that your computer satisfies the installation prerequisites or system requirements.</p>
Integrated development environment (IDE)	<ul style="list-style-type: none">Latest version of the Eclipse IDE for Java Developers <p>Visit www.eclipse.org/downloads/ to download the Eclipse IDE for Java Developers. Verify that your computer satisfies the installation prerequisites or system requirements.</p>
Web service framework	<p>Use one of the following web service frameworks to bind web service requests and to generate the required client proxy files:</p> <ul style="list-style-type: none">Apache CXF 3.1.7 or later (binary distribution)Apache Axis2 1.5.1 (binary distribution) <p>Visit http://cxf.apache.org/ to download Apache CXF, or visit http://axis.apache.org/ to download Apache Axis2. Verify that your computer satisfies the installation prerequisites or system requirements.</p> <p>The BlackBerry Web Services might support other web services frameworks, but additional configuration or resource libraries might be required.</p>

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.

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](#).

Creating a keystore for your applications

To generate the client proxy files that your applications require to use the BlackBerry Web Services SOAP APIs, and to enable your applications to make SSL connections to BlackBerry UEM, you must complete the following tasks:

- [Download the SSL certificate of the BlackBerry UEM domain](#)
- [Generate a keystore and import the SSL certificate into the keystore](#)
- [Configure the proxy generator to access the keystore](#)

Download the SSL certificate of the BlackBerry UEM domain

You need the SSL certificate of the BlackBerry UEM domain to authenticate with the BlackBerry Web Services SOAP APIs. You can download the certificate from the BlackBerry UEM management console.

Note: The following steps can be completed using Internet Explorer 11. Depending on the browser that you are using, the steps might differ. For information about downloading certificates using other browsers or versions, see the help for that browser.

1. Create a folder to temporarily store the SSL certificate on your computer (for example, C:\Temp\BWS\).
2. Run Internet Explorer as an administrator.
3. In the address bar, type the URL for the server where the BlackBerry UEM Core is installed. The address is `https://<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 [Installation and upgrade content](#).
4. Click **Continue to the website (not recommended)**.
5. On the address bar, click **Certificate error**.
6. Click **View Certificates**.
7. On the **Details** tab, click **Copy to File**.
8. Click **Next**.
9. Select the **DER encoded binary X.509 (.CER)** option.
10. Click **Next**.
11. Click **Browse** and navigate to the folder that you created in step 1. In the **File name** field, type a name for the certificate (for example, uemcert).
12. Click **Save**.
13. Click **Next**.
14. Click **Finish**.

After you finish: [Generate a keystore and import the SSL certificate into the keystore](#).

Generate a keystore and import the SSL certificate into the keystore

You can use the keytool utility to generate a keystore for your applications, and to import the SSL certificates of the BlackBerry UEM domain into the keystore. Your applications use the certificates to make SSL connections to the BlackBerry Web Services SOAP APIs. Ensure that you include the entire chain in the keystore. For example: root, intermediate, and the key.

1. Run the command prompt as an administrator.

2. Navigate to the bin folder of the active JRE. For example:

```
cd C:\Program Files\Java\jre8\bin
```

3. **Note:** If you upgraded BES12 version 12.3 to BES12 version 12.5 or BlackBerry UEM version 12.6 or later, and you previously downloaded and installed the SSL certificate into the keystore, you must delete the existing certificate before you perform this step.

Type `keytool -import -trustcacerts -file <cert_file_path> -keystore <keystore_name> -storepass <password> -alias <cert_alias>`, where `<cert_file_path>` is the full path where you stored the SSL certificate, `<keystore_name>` is the name you want to specify for the keystore, `<password>` is the password you want to specify for the keystore, and `<cert_alias>` is an alias name you want to specify for the certificate. For example:

```
keytool -import -trustcacerts -file C:\Temp\BWS\bescert.cer -keystore bes.keystore -storepass password -alias bes
```

Repeat this step for each certificate in the keychain as needed, and give each certificate a different alias.

4. Press ENTER.
5. When asked if you want to trust this certificate, type **y**. Press ENTER.
A message confirms that the certificate was added to the keystore.

After you finish: [Configure the proxy generator to access the keystore.](#)

Configure the proxy generator to access the keystore

To generate the proxy files that your applications require to use the BlackBerry Web Services SOAP APIs, you must configure the proxy generator (`wSDL2java.bat`) to access the keystore that you created.

1. Navigate to the bin folder of the Apache CXF installation (for example, `C:\Program Files\Apache CXF\apache-cxf-<version>\bin`).
2. In a text editor, open `wSDL2java.bat`.
3. After `"%JAVA_HOME%\bin\java"`, type the following: -
`Djavax.net.ssl.trustStorePassword=<password>` -
`Djavax.net.ssl.trustStore="<keystore_path>"`, where `<password>` is the keystore password that you specified, and `<keystore_path>` is the full path of the keystore that you created. For example:

```
"%JAVA_HOME%\bin\java" -Djavax.net.ssl.trustStorePassword=password -Djavax.net.ssl.trustStore="C:\Program Files\Java\jre8\bin\bes.keystore" -Xmx128M -Djava.endorsed.dirs="%CXF_HOME%\lib\endorsed" -cp "%CXF_JAR%;%TOOLS_JAR%;%CLASSPATH%" -Djava.util.logging.config.file="%CXF_HOME%\etc\logging.properties" org.apache.cxf.tools.wSDLto.WSDLToJava %*
```

4. Save and close the file.

After you finish: [Generate the client proxy files for the BWS and BWSUtil web services.](#)

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 are stored in one folder.

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, store all of the generated proxy files for the BWS and BWSUtil web services in one folder.

Before you begin: Create a folder to store the proxy files in (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 Apache CXF installation. For example:

```
cd C:\Program Files\Apache CXF\apache-cxf-3.1.7\bin
```

3. Press ENTER.
4. To generate the proxy files for the BWS web service, type `wsdl2java.bat -wv 1.1 -d <proxy_path> https://<server_name>:<port>/enterprise/admin/ws?wsdl`, where `<proxy_path>` is the path for 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:

```
wsdl2java.bat -wv 1.1 -d C:\Temp\BWSproxy https://bes_server1.test.rim.net:18084/enterprise/admin/ws?wsdl
```

5. Press ENTER.
6. Type `cd <file_path>`, where `<file_path>` is the path of the bin folder of your Apache CXF installation. For example:

```
cd C:\Program Files\Apache CXF\apache-cxf-3.1.7\bin
```

7. To generate the proxy files for the BWSUtil web service, type `wsdl2java.bat -wv 1.1 -d <proxy_path> https://<server_name>:<port>/enterprise/admin/util/ws?wsdl`, using the same values that you used in step 4.

For example:

```
wsdl2java.bat -wv 1.1 -d C:\Temp\BWSproxy https://bes_server1.test.rim.net:18084/enterprise/admin/util/ws?wsdl
```

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. The instructions may vary depending on the version of Eclipse IDE for Java Developers that you are using.

Create a project

1. Open a workspace in Eclipse.
2. On the **File** menu, click **New > Java Project**.
3. Type a name for the project. Click **Finish**.
4. In the **Package Explorer** pane, expand your project folder.
5. Right-click the **src** folder. Click **New > Class**.
6. Type a package name for the code sample (for example, `com.rim.enterprise.admin.HelloWorld`).
7. Type a class name for the code sample (for example, `HelloWorld`).
8. Click **Finish**.

Configure the VM arguments in your project

To save data about your development environment, including information about the BlackBerry UEM domain and the administrator accounts that you want your applications to use, you can configure environment variables as VM arguments. VM arguments allow your application to access these variables by using `System.getProperty(<property name>)` method calls. For example, if you want to set a variable called `besUrl`, make the following method call: `besUrl=System.getProperty("besurl")`.

1. On the **Run** menu, click **Run Configurations**.
2. In the left pane, select the class that you created (for example, `HelloWorld`).
3. On the **Arguments** tab, in the **VM arguments** field, type the following argument for the FQDN of the BlackBerry UEM: `-Dbesurl="<server_name>".` For example:

```
-Dbesurl="bes_server1.test.rim.net"
```

4. On the next line, type the following argument for the user name of the BlackBerry UEM administrator account: `-Dusername="<user_name>".` For example:

```
-Dusername="admin"
```

5. On the next line, type the following argument for the password of the administrator account: `-Dpassword="<password>".` For example:

```
-Dpassword="password"
```

6. On the next line, type the following argument for the location and password of the keystore that you created: `-Djavax.net.ssl.trustStore="<keystore_path>" -Djavax.net.ssl.trustStorePassword="<keystore_password>".` For example:

```
-Djavax.net.ssl.trustStore="C:\Program Files\Java\jre8\bin\bes.keystore"  
-Djavax.net.ssl.trustStorePassword="password"
```

7. Click **Apply**.
8. Click **Close**.

Import Apache CXF libraries to your Eclipse project

Import the Apache CXF libraries to make them available for use in your applications.

1. In Eclipse, in the **Package Explorer** pane, right-click your project.
2. Click **Properties**.
3. In the left pane, click **Java Build Path**.
4. On the **Libraries** tab, click **Add External JARs**.
5. Navigate to the **lib** folder of your Apache CXF installation (for example, C:\Program Files\Apache CXF\apache-cxf-3.1.7\lib).
6. Select all of the .jar files.
7. Click **Open**.
8. 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 Eclipse, in the **Package Explorer** pane, right-click your project.
2. Click **Properties**.
3. In the left pane, click **Java Build Path**.
4. On the **Source** tab, click **Add Folder**.
5. Click **Create New Folder**.
6. Type a name for the new source folder (for example, proxy).
7. Click **Finish**.
8. Click **OK** as required to save the changes and close the properties window.
9. In Windows Explorer, navigate to the folder that contains the client proxy files (for example, C:\Temp\BWSproxy\).
10. Drag the **com** folder into the new source folder you created in Eclipse. If you are given the option to copy the files or to create static links, choose to copy the files.

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.java` 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 final static String CLIENT_VERSION = "12.6.0";

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

// Authentication type name.
private final static String AUTHENTICATOR_NAME = "BlackBerry Administration
Service";

// Hostname to use when connecting to web service. Includes port
private static String BWS_HOST_NAME = null; // e.g. BWS_HOST_NAME =
"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.
BWS_HOST_NAME = "<BWSHostName>"; // e.g. BWS_HOST_NAME =
"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:

```
REQUEST_METADATA.setClientVersion(CLIENT_VERSION);
REQUEST_METADATA.setLocale(LOCALE);
REQUEST_METADATA.setOrganizationUid(ORG_UID);
```

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.

```
URL bwsServiceUrl = null;
URL bwsUtilServiceUrl = null;

try
{
    // These are the URLs that point to the web services used for all calls.
    bwsServiceUrl = new URL("https://" + BWS_HOST_NAME + "/enterprise/admin/ws");
    bwsUtilServiceUrl = new URL("https://" + BWS_HOST_NAME + "/enterprise/admin/
util/ws");
}
catch (MalformedURLException e)
{
    logMessage("Cannot initialize web service URLs");
    logMessage("Exiting %s with value \"%s\"", METHOD_NAME, returnValue);
    return returnValue;
}

// Initialize the BWS web service stubs that will be used for all calls.
logMessage("Initializing BWS web service stub");
QName serviceBWS = new QName("http://ws.rim.com/enterprise/admin", "BWSService");
QName portBWS = new QName("http://ws.rim.com/enterprise/admin", "BWS");
_bwsService = new BWSService(null, serviceBWS);
_bwsService.addPort(portBWS, "http://schemas.xmlsoap.org/soap/",
    bwsServiceUrl.toString());
_bws = _bwsService.getPort(portBWS, BWS.class);
logMessage("BWS web service stub initialized");

logMessage("Initializing BWSUtil web service stub");
QName serviceUtil = new QName("http://ws.rim.com/enterprise/admin",
    "BWSUtilService");
QName portUtil = new QName("http://ws.rim.com/enterprise/admin", "BWSUtil");
_bwsUtilService = new BWSUtilService(null, serviceUtil);
_bwsUtilService.addPort(portUtil, "http://schemas.xmlsoap.org/soap/",
    bwsUtilServiceUrl.toString());
_bwsUtil = _bwsUtilService.getPort(portUtil, BWSUtil.class);
logMessage("BWSUtil web service stub initialized");

// Set the connection timeout to 60 seconds.
HTTPClientPolicy httpClientPolicy = new HTTPClientPolicy();
httpClientPolicy.setConnectionTimeout(60000);

httpClientPolicy.setAllowChunking(false);
httpClientPolicy.setReceiveTimeout(60000);
```

```

Client client = ClientProxy.getClient(_bws);
HTTPConduit http = (HTTPConduit) client.getConduit();
http.setClient(httpClientPolicy);

client = ClientProxy.getClient(_bwsUtil);
http = (HTTPConduit) client.getConduit();
http.setClient(httpClientPolicy);

```

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(AUTHENTICATOR_NAME);
if (authenticator != null)
{
    String encodedUsername = getEncodedUserName(USERNAME, authenticator);
    if (encodedUsername != null && !encodedUsername.isEmpty())
    {
        /*
        * Set the HTTP basic authentication on the BWS service.
        * BWSUtilService is a utility web service that does not require
        * authentication.
        */
        BindingProvider bp = (BindingProvider) _bws;
        bp.getRequestContext().put(BindingProvider.USERNAME_PROPERTY,
encodedUsername);
        bp.getRequestContext().put(BindingProvider.PASSWORD_PROPERTY, PASSWORD);

        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)
{
    final String METHOD_NAME = "getEncodedUserName()";
    final String BWS_API_NAME = "_bwsUtil.getEncodedUsername()";
    logMessage("Entering %s", METHOD_NAME);

```



```

String returnValue = null;

GetEncodedUsernameRequest request = new GetEncodedUsernameRequest();
request.setMetadata(REQUEST_METADATA);
request.setUsername(username);
request.setOrgUid(REQUEST_METADATA.getOrganizationUid());
request.setAuthenticator(authenticator);

CredentialType credentialType = new CredentialType();
credentialType.setPASSWORD(true);
credentialType.setValue("PASSWORD");
request.setCredentialType(credentialType);

GetEncodedUsernameResponse response=null;
try
{
    logRequest(BWS_API_NAME);
    response = _bwsUtil.getEncodedUsername(request);
    logResponse(BWS_API_NAME, response.getReturnStatus().getCode(),
response.getMetadata());
}
catch (WebServiceException e)
{
    //Log and re-throw exception.
    logMessage("Exiting %s with exception \"%s\"", METHOD_NAME,
e.getMessage());
    throw e;
}

if (response.getReturnStatus().getCode().equals("SUCCESS"))
{
    returnValue = response.getEncodedUsername();
}
else
{
    logMessage("Error Message: \"%s\"",
response.getReturnStatus().getMessage());
}

if (Base64.isBase64(returnValue))
{
    logMessage("Decoded value of encoded username \"%s\"",
StringUtils.newStringUtf8(Base64.decodeBase64(returnValue)));
}
else
{
    logMessage("Value of encoded username \"%s\"", returnValue);
}
logMessage("Exiting %s", METHOD_NAME);
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.
 */
CREATE_NEW_USER_EMAIL = "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.setMetadata(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 \"%s\"", CREATE_NEW_USER_EMAIL);

// Value of the variable "CREATE_NEW_USER_EMAIL" is used to create the user.
accountAttributes.setEmailAddress(CREATE_NEW_USER_EMAIL);

newUser.setAccountAttributes(accountAttributes);

// Server value is validated and then ignored.
newUser.setServer(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.

```
createUsersRequest.getNewUsers().add(newUser);
CreateUsersResponse response=null;
try
{
    logRequest(BWS_API_NAME);
    response = _bws.createUsers(createUsersRequest);
}
```

```

    logResponse(BWS_API_NAME, response.getReturnStatus().getCode(),
    response.getMetadata());
}
catch (WebServiceException e)
{
    // Log and re-throw exception.
    logMessage("Exiting %s with exception \"%s\"", METHOD_NAME, e.getMessage());
    throw e;
}

if (response.getReturnStatus().getCode().equals("SUCCESS"))
{
    if (response.getIndividualResponses() != null)
    {
        for (IndividualResponse individualResponse : response.getIndividualResponses())
        {
            displayResult("User created with UID \"%s\"",
            individualResponse.getUid());
            displayResult("Email address used in creation is \"%s\"",
            accountAttributes.getEmailAddress());
        }

        returnValue = true;
    }
}
else
{
    logMessage( "Error Message: \"%s\"", response.getReturnStatus().getMessage());
    if (response.getIndividualResponses() != null)
    {
        for (IndividualResponse individualResponse : response.getIndividualResponses())
        {
            logMessage( "Individual Response - Code: \"%s\"", Message: \"%s\"",
            individualResponse.getReturnStatus().getCode(),
            individualResponse.getReturnStatus().getMessage());
        }
    }
}
}

```

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
<p>Directory user</p> <ul style="list-style-type: none"> • 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. 	<p>AccountAttributes</p> <p>Specify any of the following fields:</p> <ul style="list-style-type: none"> • A valid email address for emailAddress • An identifier from your organization's directory (Microsoft Active Directory or LDAP) for externalUserUid • A valid distinguished name for userDistinguishedName <p>DeviceActivationType</p> <ul style="list-style-type: none"> • Not currently supported. <p>UserAttributes</p> <ul style="list-style-type: none"> • Null <p>Server</p> <ul style="list-style-type: none"> • Not currently supported.
<p>Local user</p> <ul style="list-style-type: none"> • 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. 	<p>AccountAttributes</p> <ul style="list-style-type: none"> • Set localUser to true • Optional: A valid email address for emailAddress <p>DeviceActivationType</p> <ul style="list-style-type: none"> • Not currently supported. <p>UserAttributes</p> <ul style="list-style-type: none"> • For authenticator, set authenticatorType to internal (local user accounts don't support directory authentication). • Specify loginName, loginPassword, and displayName. <p>Server</p> <ul style="list-style-type: none"> • Not currently supported.

User type	Configuration of NewUser in CreateUsersRequest
<p>Administrator account</p> <ul style="list-style-type: none"> An administrator user that is assigned login information and an administrative role. An administrator can assign a device to the user. 	<p>AccountAttributes</p> <p>Specify any of the following fields:</p> <ul style="list-style-type: none"> 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 <p>DeviceActivationType</p> <ul style="list-style-type: none"> Not currently supported. <p>UserAttributes</p> <ul style="list-style-type: none"> 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>. <p>Server</p> <ul style="list-style-type: none"> Not currently supported.
<p>Administrator account with permissions granted by group membership</p> <ul style="list-style-type: none"> 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. 	<p>AccountAttributes</p> <ul style="list-style-type: none"> Null <p>DeviceActivationType</p> <ul style="list-style-type: none"> Not currently supported. <p>UserAttributes</p> <ul style="list-style-type: none"> 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. <p>Server</p> <ul style="list-style-type: none"> Not currently supported.

Sample walkthrough: Authenticating with the BlackBerry Web Services

The following section examines the AuthenticationSample.java 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 final static String CLIENT_VERSION = "<Client Version>"; // e.g.
CLIENT_VERSION = "12.6.0"

// 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 final static String LOCALE = "en_US";
private final static String ORG_UID = "0";
private final static 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) throws WebServiceException {

    final String METHOD_NAME = "getEncodedUserName()";
    final String BWS_API_NAME = "_bwsUtil.getEncodedUsername()";
    logMessage("Entering %s", METHOD_NAME);
    String returnValue = null;

    GetEncodedUsernameRequest request = new GetEncodedUsernameRequest();
    request.setMetadata(REQUEST_METADATA);
    request.setUsername(username);
    request.setOrgUid(REQUEST_METADATA.getOrganizationUid());
    request.setAuthenticator(authenticator);

    request.setCredentialType(credentialType);
    request.setDomain(domain);

    GetEncodedUsernameResponse response = null;
    try {
        logRequest(BWS_API_NAME);
        response = _bwsUtil.getEncodedUsername(request);
        logResponse(BWS_API_NAME,
            response.getReturnStatus().getCode(), response.getMetadata());
    } catch (WebServiceException e) {
        // Log and re-throw exception.
        logMessage("Exiting %s with exception \"%s\"", METHOD_NAME,
            e.getMessage());
        throw e;
    }

    if (response.getReturnStatus().getCode().equals("SUCCESS")) {
        returnValue = response.getEncodedUsername();
    } else {
        logMessage("Error Message: \"%s\"",
            response.getReturnStatus().getMessage());
    }

    if (Base64.isBase64(returnValue))
    {
        logMessage("Decoded value of encoded username \"%s\"",
            StringUtils.newStringUtf8(Base64.decodeBase64(returnValue)));
    }
    else
    {
        logMessage("Value of encoded username \"%s\"", returnValue);
    }
    logMessage("Exiting %s", 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.

```
public static boolean echo() throws WebServiceException {
    final String METHOD_NAME = "echo()";
    final String BWS_API_NAME = "_bws.echo()";
    logMessage("Entering %s", METHOD_NAME);

    boolean returnValue = true;

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

    request.setMetadata(REQUEST_METADATA);
    request.setText("Hello World!");

    try {
        logRequest(BWS_API_NAME);
        response = _bws.echo(request);
        logResponse(BWS_API_NAME, response.getReturnStatus().getCode(),
            response.getMetadata());
    } catch (WebServiceException e) {
        if (e.getCause() instanceof HTTPException) {
            HTTPException httpException = (HTTPException) e.getCause();
            // Handle authentication failure.
            if (httpException != null && httpException.getResponseCode()
                == HttpURLConnection.HTTP_UNAUTHORIZED) {
                returnValue = false;
                logMessage("Failed to authenticate with the BWS
                    web service");
                logMessage("Exiting %s with value \"%s\"", METHOD_NAME,
                    returnValue);
                return returnValue;
            }
        }

        // Log and re-throw exception.
        logMessage("Exiting %s with exception \"%s\"", METHOD_NAME,
            e.getMessage());
        throw e;
    }

    logMessage("Exiting %s with value \"%s\"", 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.

```
public static void main(String[] args) throws IOException {
    // Return codes
    final int SUCCESS = 0;
    final 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
// management console 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
boolean useAD = false; // Active Directory
boolean 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";
// Only BDS requires domain for authentication
String activeDirectoryDomain = null;
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 boolean demonstrateBwsSetupAndAuthenticatedCall(String bwsHostname,
String bwsPort,
    String username, String password, String domain, String authenticatorName,
    CredentialType credentialType) throws WebServiceException {
    boolean 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 _bws.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 boolean setup(String hostname, String bwsPort, String username,
    String password,
    String authenticatorName, CredentialType credentialType, String domain) {
    final String METHOD_NAME = "setup()";
    logMessage("Entering %s", METHOD_NAME);
    boolean returnValue = false;
    REQUEST_METADATA.setClientVersion(CLIENT_VERSION);
    REQUEST_METADATA.setLocale(LOCALE);
    REQUEST_METADATA.setOrganizationUid(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.

```
URL bwsServiceUrl = null;
URL bwsUtilServiceUrl = null;

try {
    // 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;
    }

    bwsServiceUrl = new URL("https://" + hostname + port + "/enterprise/admin/
ws");
    bwsUtilServiceUrl = new URL("https://" + hostname + port + "/enterprise/admin/
util/ws");

} catch (MalformedURLException e) {
    logMessage("Cannot initialize web service URLs");
    logMessage("Exiting %s with value \"%s\"", METHOD_NAME, returnValue);
    return returnValue;
}

// Initialize the BWS web service stubs that will be used for all calls.
logMessage("Initializing BWS web service stub");
QName serviceBWS = new QName("http://ws.rim.com/enterprise/admin", "BWSService");
QName portBWS = new QName("http://ws.rim.com/enterprise/admin", "BWS");
_wsService = new BWSService(null, serviceBWS);
_wsService.addPort(portBWS, "http://schemas.xmlsoap.org/soap/",
    bwsServiceUrl.toString());
_ws = _wsService.getPort(portBWS, BWS.class);
logMessage("BWS web service stub initialized");

logMessage("Initializing BWSUtil web service stub");
QName serviceUtil = new QName("http://ws.rim.com/enterprise/admin",
    "BWSUtilService");
```

```

QName portUtil = new QName("http://ws.rim.com/enterprise/admin", "BWSUtil");
_bwsUtilService = new BWSUtilService(null, serviceUtil);
_bwsUtilService.addPort(portUtil, "http://schemas.xmlsoap.org/soap/",
    bwsUtilServiceUrl.toString());
_bwsUtil = _bwsUtilService.getPort(portUtil, BWSUtil.class);
logMessage("BWSUtil web service stub initialized");

// Set the connection timeout to 60 seconds.
HTTPClientPolicy httpClientPolicy = new HTTPClientPolicy();
httpClientPolicy.setConnectionTimeout(60000);

httpClientPolicy.setAllowChunking(false);
httpClientPolicy.setReceiveTimeout(60000);

Client client = ClientProxy.getClient(_bws);
HTTPConduit http = (HTTPConduit) client.getConduit();
http.setClient(httpClientPolicy);

client = ClientProxy.getClient(_bwsUtil);
http = (HTTPConduit) client.getConduit();
http.setClient(httpClientPolicy);

```

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 (encodedUsername != null && !encodedUsername.isEmpty()) {
        /*
         * Set the HTTP basic authentication on the BWS service. BWSUtilService is
         a utility web service that
         * does not require authentication.
         */
        BindingProvider bp = (BindingProvider) _bws;
        bp.getRequestContext().put(BindingProvider.USERNAME_PROPERTY,
            encodedUsername);
        bp.getRequestContext().put(BindingProvider.PASSWORD_PROPERTY, password);

        returnValue = true;
    } else {
        logMessage("\"encodedUsername\" is null or empty");
    }
} else {
    logMessage("\"authenticator\" is null");
}

logMessage("Exiting %s with value \"%s\"", 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.

Apache CXF is a trademark of The Apache Software Foundation. Android is a trademark of Google Inc. Eclipse is a trademark of Eclipse Foundation, 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 and JRE are trademarks of Oracle and/or its affiliates. Microsoft, Windows, Microsoft Active Directory, and Windows Internet Explorer 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