Push notifications of application updates

Overview

You can configure the Application Center client so that push notifications are sent to users when an update is available for an application in the store.

The Application Center administrator uses push notifications to send notification automatically, to any iOS or Android device. Notifications are sent for updates to favorite applications and of new applications that are deployed on the Application Center server and that are marked as recommended.

Push notification process

Push notifications are sent to a device if the following conditions are met:

- The device has Application Center installed and started it at least one time.
- The user has not disabled push notification for this device for the Application Center in the **Settings → Notifications** interface.
- The user is allowed to install the application. Such permissions are controlled through the Application Center access rights.
- The application is marked as recommended, or is marked as preferred for the user who is using Application Center on this device. Those flags are set automatically when the user installs an application through Application Center. You can see which applications are marked as preferred by looking at the Application Center **Favorites** tab on the device.
- The application is not installed on the device or a more recent version is available than the version that is installed on the device.

The first time that the Application Center client starts on a device, the user might be asked whether to accept incoming push notifications. This is the case for iOS mobile devices. The push notification feature does not work when the service is disabled on the mobile device.

iOS and modern Android operating system versions offer a way to switch this service on or off on a per application basis.

Refer to your device vendor to learn how to configure your mobile device for push notifications.

Jump to

- Configuring push notifications for application updates
- Configuring the Application Center server for connection to Google Cloud Messaging
- Configuring the Application Center server for connection to Apple Push Notification Services
- Building a version of the mobile client that does not depend on the GCM API

Configuring push notifications for application updates

You must configure the credentials or certificates of Application Center services to be able to communicate with third-party push notification servers.

Configuring the server scheduler of the Application Center

The server scheduler is a background service that automatically starts and stops with the server. This scheduler is used to empty at regular intervals a stack that is automatically filled by administrator actions with push update messages to be sent. The default interval between sending two batches of push update messages is twelve hours. If this default value does not suit you, you can modify it by using the **ibm.appcenter.push.schedule.period.amount** and **ibm.appcenter.push.schedule.period.unit** server environment variables.

The value of **ibm.appcenter.push.schedule.period.amount** is an integer. The value of **ibm.appcenter.push.schedule.period.unit** can be seconds, minutes, or hours. If the unit is not specified, the amount is an interval that is expressed in hours. These variables are used to define the elapsed time between two batches of push messages.

Use JNDI properties to define these variables.

> **Important:** In production, avoid setting the unit to seconds. The shorter the elapsed time, the higher the load on the server. The unit expressed in seconds is implemented only for testing and evaluation purposes. For example, when the elapsed time is set to 10 seconds, push messages are sent almost immediately.

See JNDI properties for Application Center (../../installation-configuration/production/appcenter/#jndi-properties-for-application-center) for a complete list of properties that you can set.

Example for Apache Tomcat server

Define these variables with JNDI properties in the server.xml file:

```
<Environment name="ibm.appcenter.push.schedule.period.unit" override="false" type="java.lang.String" value="hours"/>
<Environment name="ibm.appcenter.push.schedule.period.amount" override="false" type="java.lang.String" value="2"/>
```

WebSphere® Application Server v8.5

To configure JNDI variables for WebSphere Application Server v8.5, proceed as follows:

1. Click **Applications → Application Types → Websphere enterprise applications**.
2. Select the Application Center Services application.
3. Click **Web Module Properties → Environment entries for Web modules** .
4. Edit the string in the **Value** column.

WebSphere Application Server Liberty profile

For information about how to configure JNDI variables for WebSphere Application Server Liberty profile, see Using JNDI binding for constants from the server configuration files (http://ibm.biz/knowctr#SSAW57_8.5.5/com.ibm.websphere.wlp.nd.doc/ae/twlp_dep_jndi.html).

The remaining actions for setting up the push notification service depend on the vendor of the device where the target application is installed.

Configuring the Application Center server for connection to Google Cloud Messaging

To enable Google Cloud Messaging (GCM) for an application, you must attach the GCM services to a developer Google account with the Google API enabled. See Getting Started with GCM (http://developer.android.com/google/gcm/gs.html) for details.

> Important: The Application Center client without Google Cloud Messaging: The Application Center relies on the availability of the Google Cloud Messaging (GCM) API. This API might not be available on devices in some territories such as China. To support those territories, you can build a version of the Application Center client that does not depend on the GCM API. The push notification feature does not work on that version of the Application Center client. See Building a version of the mobile client that does not depend on the GCM APIfor details.

1. If you do not have the appropriate Google account, go to Create a Google account (https://mail.google.com/mail/signup) and create one for the Application Center client.
2. Register this account by using the Google API in the Google API console (https://code.google.com/apis/console/). Registration creates a new default project that you can rename. The name you give to this GCM project is not related to your Android application package name. When the project is created, a GCM project ID is appended to the end of the project URL. You should record this trailing number as your project ID for future reference.
3. Enable the GCM service for your project; in the Google API console, click the **Services** tab on the left and enable the "Google Cloud Messaging for Android" service in the list of services.
4. Make sure that a Simple API Access Server key is available for your application communications.
   - Click the **API Access** vertical tab on the left of the console.
   - Create a Simple API Access Server key or, if a default key is already created, note the details of the default key. Two other kinds of key exist that are not of interest at this time.
   - Save the Simple API Access Server key for future use in your application communications through GCM. The key is about 40 characters long and is referred to as the Google API key that you will need later on the server side.
5. Enter the GCM project ID as a string resource property in the JavaScript project of the Application Center Android client; in the **IBMAppCenter/apps/AppCenter/common/js/appcenter/config.json** template file, modify this line with your own value:

   ```
   gcmProjectId:""// Google API project (project name = com.ibm.appcenter) ID needed for Android push.
   // example : 123456789012
   ```

6. Register the Google API key as a JNDI property for the Application Center server. The key name is : **ibm.appcenter.gcm.signature.googleapikey**. For example, you can configure this key for an Apache Tomcat server as a JNDI property in the **server.xml** file:

   ```
   <Context docBase="AppCenterServices" path="/applicationcenter" reloadable="true" source="org.eclipse.jst.jee.server:AppCenterServices">
     <Environment name="ibm.appcenter.gcm.signature.googleapikey" override="false" type="java.lang.String"
     value="AIxaScCHg0VSGdgfOZKtzDJ44-oi0muUasMZvAs"/>
   </Context>
   ```

   The JNDI property must be defined in accordance with your application server requirements.
   See JNDI properties for Application Center (../../installation-configuration/production/appcenter/#jndi-properties-for-application-center) for a complete list of properties that you can set.

Important:

- If you use GCM with earlier versions of Android, you might need to pair your device with an existing Google account for GCM to work effectively. See GCM service (http://developer.android.com/google/gcm/gcm.html): "It uses an existing connection for Google services. For pre-3.0 devices, this requires users to set up their Google account on their mobile devices. A Google account is not a requirement on devices running Android 4.0.4 or higher."
- You must also ensure that your firewall accepts outgoing connections to android.googleapis.com on port 443 for push notifications to work.

Configuring the Application Center server for connection to Apple Push Notification Services

Configure your iOS project for Apple Push Notification Services (APNs). Ensure that the following servers are accessible from Application Center server.

Sandbox servers
gateway.sandbox.push.apple.com:2195 feedback.sandbox.push.apple.com:2196

Production servers
gateway.push.apple.com:2195 feedback.push.apple.com:2196

You must be a registered Apple developer to successfully configure your iOS project with Apple Push Notification Services (APNs). In the company, the administrative role responsible for Apple development requests APNs enablement. The response to this request should provide you with an APNs-enabled provisioning profile for your iOS application bundle; that is, a string value that is defined in the configuration page of your Xcode project. This provisioning profile is used to generate a signature certificate file. Two kinds of provisioning profile exist: development and production profiles, which address development and production environments respectively. Development profiles address Apple development APNs servers exclusively. Production profiles address Apple production APNs servers exclusively. These kinds of servers do not offer the same quality of service.

Note: Devices that are connected to a company wifi behind a firewall are only able to receive push notifications if connection to the following type of address is not blocked by the firewall.

`x-courier.sandbox.push.apple.com:5223`
Where x is an integer.

1. Obtain the APNs-enabled provisioning profile for the Application Center Xcode project. The result of your administrator's APNs enablement request is shown as a list accessible from https://developer.apple.com/ios/my/bundles/index.action (https://developer.apple.com/ios/my/bundles/index.action). Each item in the list shows whether or not the profile has APNs capabilities. When you have the profile, you can download and install it in the Application Center client Xcode project directory by double-clicking the profile. The profile is then automatically installed in your keystore and Xcode project.
2. If you want to test or debug the Application Center on a device by launching it directly from XCode, in the "Xcode Organizer" window, go to the "Provisioning Profiles" section and install the profile on your mobile device.
3. Create a signature certificate used by the Application Center services to secure communication with the APNs server. This server will use the certificate for purposes of signing each and every push request to the APNs server. This signature certificate is produced from your provisioning profile.

- Open the "Keychain Access" utility and click the **My Certificates** category in the left pane.
- Find the certificate you want to install and disclose its contents. You see both a certificate and a private key; for the Application Center, the certificate line contains the Application Center application bundle **com.ibm.imf.AppCenter**.
- Select **File → Export Items** to select both the certificate and the key and export them as a Personal Information Exchange (.p12) file. This .p12 file contains the private key required when the secure handshaking protocol is involved to communicate with the APNs server.
- Copy the .p12 certificate to the computer responsible for running the Application Center services and install it in the appropriate place. Both the certificate file and its password are needed to create the secure tunneling with the APNs server. You also require some information that indicates whether a development certificate or a production certificate is in play. A development provisioning profile produces a development certificate and a production profile gives a production certificate. The Application Center services web application uses JNDI properties to reference this secure data

The examples in the table show how the JNDI properties are defined in the server.xml file of the Apache Tomcat server.

JNDI Property	Type and description	Example for Apache Tomcat server
ibm.appcenter.apns.p12.certificate.location	A string value that defines the full path to the .p12 certificate.	`<Environment name="ibm.appcenter.apns.p12.certificate.location" override="false" type="java.lang.String" value="/Users/someUser/someDirectory/apache-tomcat/conf/AppCenter_apns_dev_cert.p12"/>`
ibm.appcenter.apns.p12.certificate.password	A string value that defines the password needed to access the certificate.	`<Environment name="ibm.appcenter.apns.p12.certificate.password" override="false" type="java.lang.String" value="this_is_a_secure_password"/>`
ibm.appcenter.apns.p12.certificate.isDevelopmentCertificate	A boolean value (identified as true or false) that defines whether or not the provisioning profile used to generate the authentication certificate was a development certificate.	`<Environment name="ibm.appcenter.apns.p12.certificate.isDevelopmentCertificate" override="false" type="java.lang.String" value="true"/>`

See JNDI properties for Application Center (../../installation-configuration/production/appcenter/#jndi-properties-for-application-center) for a complete list of JNDI properties that you can set.

Building a version of the mobile client that does not depend on the GCM API

You can remove the dependency on Google Cloud Messaging (GCM) API from the Android version of the client to comply with constraints in some territories. Push notifications do not work on this version of the client.

The Application Center relies on the availability of the Google Cloud Messaging (GCM) API. This API might not be available on devices in some territories such as China. To support those territories, you can build a version of the Application Center client that does not depend on the GCM API. The push notification feature does not work on that version of the Application Center client.

1. Check that push notifications are disabled by checking that the **IBMAppCenter/apps/AppCenter/common/js/appcenter/config.json** file contains this line: `"gcmProjectId": "" ,`.
2. Remove from two places in the **IBMAppCenter/apps/AppCenter/android/native/AndroidManifest.xml** file all the lines that are located between these comments: `<!-- AppCenter Push configuration -->` and `<!-- end of AppCenter Push configuration -->`.
3. Delete the **IBMAppCenter/apps/AppCenter/android/native/src/com/ibm/appcenter/GCMIntenteService.java** class.
4. In Eclipse, run "Build Android Environment" in the IBMAppCenter/apps/AppCenter/android folder.
5. Delete the **IBMAppCenter/apps/AppCenter/android/native/libs/gcm.jar** file that was created by the MobileFirst plug-in when you ran the previous

"Build Android Environment" command.

6. Refresh the newly created IBMAppCenterAppCenterAndroid project, so that the removal of the GCM library is taken into account.
7. Build the .apk file of the Application Center.

The **gcm.jar** library is automatically added by the MobileFirst Eclipse plug-in each time that the Android environment is built. Therefore, this java archive file must be deleted from the **IBMAppCenter**/**apps**/**AppCenter**/**android**/**native**/**libs**/ directory each time that the MobileFirst Android build process is run. Otherwise, the **gcm.jar** library is present in the resulting **appcenter.apk** file.