Troubleshooting Push Notifications

Overview

Find information to help resolve issues that you might encounter when you use the MobileFirst Foundation Push service.

How does the Push service treats various “failed to deliver” notification situations?

GCM

- If the response from the GCM is "internal server error" or "GCM service is unavialable" then an attempt is made to resend the notification. The frequncy of attempts is determined based on the value of the "Retry-After".
- GCM is not reachable - an error is printed in the **trace.log** file and the message sending will not be re-sent.
- GCM is reachable but failures were received
    - Not registered / invalid id / mismatch id / registration missing - likely to have been caused by an invalid usage of the device ID or registration of the app in GCM. The davice ID is deleted from the database in order to avoid stale data in the database. A notification is not resent.
    - The message is too big - the message sending is not retryed and a log is recorded in the **trace.log** file.
    - Error code 401 - likley due to authentication error, the message sending is not retryed and a log is recorded in the **trace.log** file.
    - Error codes 400 or 403 - the message sending is not retryed and a log is recorded in the **trace.log** file.

APNS

For APNS, a retry attempt is made if the APNS connection is closed. There will be three attempts to establish a connection with APNS. For other cases no retry attempt is made.

I get errors related to "apns-environment" in Xcode

Verify the provisioning profile used to sign the application has the Push capability enabled. This can be verified in the App ID's settings page in the Apple Developer Portal.

There are Java socket exceptions in the logs when dispatching an APNS notifications and the notification never reaches the device

APNS requires persistent socket connections between the MobileFirst Server and the APNS service. The Push service assumes there is an open socket and tries to send the notification out. Many times though, a customer's firewall may be configured to close unused sockets (push might not be frequently used). Such abrupt socket closures cannot be found by either end point – because normal Socket closures happen with either end point sending the signal and the other acknowledging. When the Push service dispatch is attempted over a closed socket, the logs will show socket exceptions.

To avoid, either ensure a firewall rules do not close APNS sockets, or use the `push.apns.connectionIdleTimeout` JNDI property of the Push service (../../installation-configuration/production/server-configuration/#list-of-jndi-properties-for-mobilefirst-server-push-service). By configuring this property, the server will gracefully close the socket used for APNS push notifications within a given timeout (less than the firewall timeout for sockets). This way, a customer can close sockets before it is abruptly shut down by the firewall.

There are SOCKS-related errors when sending a push notification to iOS devices

For example:

> java.net.SocketException: Malformed reply from SOCKS serverat
> java.net.SocksSocketImpl.readSocksReply(SocksSocketImpl.java

APNS notifications are sent over TCP sockets. This brings in the restriction that the proxy used for APNS notifications should support TCP sockets. A normal HTTP proxy (which works for GCM) does not suffice here. For this reason, the only proxy supported in case of APNS notifications is a SOCKS proxy. SOCKS protocol version 4 or 5 is supported. The exception is thrown when JNDI properties are configured to direct APNS notifications via a SOCKS proxy, but the proxy is either not configured, offline / not available, or blocks traffic. A customer should verify their SOCKS proxy is available and working.

A notification was sent, but never reached the device

Notifications can be instantaneous or may be delayed. The delay might be a few seconds to a few minutes. There are various reasons that can be accounted for:

- MobileFirst Server has no control over the notification once it has reached the mediator service.
- The device’s network or online status (device on /off) needs to be accounted for.
- Check if firewalls or proxies are used and if used, that they are not configured to block the communication to the mediator (and back).
- Do not selectively whitelist IPs in your firewall for the GCM/APNS/WNS mediators instead of using the actual mediators URLs. This can lead to issues, as the mediator URL may resolve to any IP. Customers should allow access to the URL and not the IP. Note that ensuring mediator connectivity by telnetting to the mediator URL does not always provide the complete picture. Specifically for iOS, this only proves outbound connectivity. The actual sending is done over TCP sockets which telnet does not guarantee. by allowing only specific IP address the following may occur, for example For GCM:

> Caused by:
> java.net.UnknownHostException:android.googleapis.com:android.googleapis.com:
> Name or service not known.

In iOS, some notifications arrive to the device but some don't

Apple's QoS defines what is called **coalescing notifications**. What this means is that the APNS server will only maintain 1 notification in their server if it cannot be immediately delivered to a device (identified via a token). For example, if 5 notifications are dispatched one after the other. If the device is on a shaky network, then if the first one was delivered and the second one is stored by APNS server temporarily. By then the 3rd notification has been dispatched and reaches APNS server. Now, the earlier (undelivered) 2nd notification is discarded and the 3rd (and latest one) is stored. To the end user this manifests as missing notifications.

In Android, notifications are available only if the app is running and in the foreground

... If the application is not running, or when the application is in the background, tapping on the notification in the notification shade does not launch the application. This may be because the **app_name** field in the **strings.xml** file was changed to a custom name. This causes a mismatch in the application name and the intent action defined in the **AndroidManifest.xml** file. If a different name for the application is required, the **app_label** field should be used instead, or use custom definitions in the **strings.xml** file.

The Push service is shown as Inactive in the MobileFirst Operations Console

The Push service is shown as Inactive despite its .war file deployed and the `mfp.admin.push.url`, `mfp.push.authorization.server.url`, and `secret` properties are correctly configured in the **server.xml** file.

Verify that the server's JNDI properties are set correctly for the MFP Admin service. It should contain the following as an example:

```
<jndiEntry jndiName="mfpadmin/mfp.admin.push.url" value="'http://localhost:9080/imfpush'"/>
<jndiEntry jndiName="mfpadmin/mfp.admin.authorization.server.url" value="'http://localhost:9080/mfp'"/>
<jndiEntry jndiName="mfpadmin/mfp.push.authorization.client.id" value="'push-client-id'"/>
<jndiEntry jndiName="mfpadmin/mfp.push.authorization.client.secret" value="'pushSecret'"/>
<jndiEntry jndiName="mfpadmin/mfp.admin.authorization.client.id" value="'admin-client-id'"/>
<jndiEntry jndiName="mfpadmin/mfp.admin.authorization.client.secret" value="'adminSecret'"/>
<jndiEntry jndiName="mfpadmin/mfp.config.service.password" value="'{xor}DCs+LStubWw='"/>
<jndiEntry jndiName="mfpadmin/mfp.config.service.user" value="'configUser'"/>
```

Last modified on