Tigase Push Component

Tigase Team
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Tigase Push Component ..................................................................</td>
<td>1</td>
</tr>
<tr>
<td>Workflow</td>
<td>1</td>
</tr>
<tr>
<td>Enabling notifications</td>
<td>1</td>
</tr>
<tr>
<td>Receiving notifications</td>
<td>1</td>
</tr>
<tr>
<td>2. Configuration</td>
<td>2</td>
</tr>
<tr>
<td>Enabling component</td>
<td>2</td>
</tr>
<tr>
<td>3. Usage</td>
<td>3</td>
</tr>
<tr>
<td>Sending notifications</td>
<td>3</td>
</tr>
<tr>
<td>Registering device</td>
<td>3</td>
</tr>
<tr>
<td>Unregistering device</td>
<td>3</td>
</tr>
<tr>
<td>4. Providers</td>
<td>4</td>
</tr>
<tr>
<td>Tigase Push Component - FCM provider</td>
<td>4</td>
</tr>
<tr>
<td>Overview</td>
<td>4</td>
</tr>
<tr>
<td>Configuration</td>
<td>4</td>
</tr>
<tr>
<td>Enabling provider</td>
<td>4</td>
</tr>
<tr>
<td>Setting FCM credentials</td>
<td>4</td>
</tr>
<tr>
<td>Connection pool</td>
<td>5</td>
</tr>
<tr>
<td>Tigase Push Component - APNs provider</td>
<td>5</td>
</tr>
<tr>
<td>Overview</td>
<td>5</td>
</tr>
<tr>
<td>Configuration</td>
<td>5</td>
</tr>
<tr>
<td>Enabling provider</td>
<td>5</td>
</tr>
<tr>
<td>Setting APNs credentials</td>
<td>5</td>
</tr>
</tbody>
</table>
Welcome to Tigase Push component guide
Chapter 1. Tigase Push Component

Tigase Push component is a Push notifications component implementing XEP-0357: Push Notifications [https://xmpp.org/extensions/xep-0357.html]. It is a gateway between Push Notification services and XMPP servers. It is configured by default to run under name of push.

**Note**

Tigase Push component requires at the minimum version 8.0.0 of Tigase XMPP Server.

Push notifications enable messages and pertinent information for clients, even if they are offline as long as they are registered with the push service. Tigase Messenger for iOS and Tigase Messenger for Android both have support for this feature.

**Workflow**

The workflow for enabling and using push notifications works as follows:

**Enabling notifications**

In order to receive notifications, clients will require registration with a push service. Although this process is mainly invisible to the user, the steps in registration are listed here:

- The client registers and bootstraps its associated push service. This is done automatically.
- The client registers itself with the push service server which then will dedicate a node for the device.
- Node information is passed back to the client and is shared with necessary components.

**Receiving notifications**

Notifications sent from the server are received the following (simplified) way:

- A message is published on the XMPP node which is then sent to the push service on the same server.
- The push service will then inform the user agent (an application on the device running in the background) that a push notification has been sent.
- The user agent will then publish the notification to the client for a user to see, waking up or turning on the client if it is not running or suspended.
Chapter 2. Configuration

Enabling component

Push notifications may be sent by Tigase XMPP Server with or without use of Push component. Push Component is only required if you have your own application for mobile devices for which you want to send push notifications.

This component is not loaded and enabled by default as it requires implementations of Push notifications providers and additional configuration (including credentials required to authorize to push services). Following entries will activate component:

```plaintext
push () {
}
```

**Note**

You need to enable and configure push providers implementations before it will be possible to send push notifications. For more details about this process, please check documentations of push service provider projects.
Chapter 3. Usage

Sending notifications

When you will register a device for a Push Notifications, you will receive name of the PubSub node where you should publish items. Publishing items to this node, as specified in XEP-0357: Push Notifications [https://xmpp.org/extensions/xep-0357.html] will result in push notifications being delivered to the registered device.

Registering device

To register a device you need to execute the adhoc command register-device available at Push Notification component. This command will return a form which needs to be filled.

Form consists of following fields:

- provider: ID of a provider for which you want to register a device. It contains a list of available providers and you need to select a proper one.
- device-token: Unique token which your application retrieved from a device or client library and which should be used to identify device you want to register for push notifications.

When you submit this form, it will be processed and will respond with a result type form. Within this form you will find a node field which will contain a PubSub node name created by the Push Notifications component, to which you should publish notification items. This returned node with jid of the Push Notifications Component should be passed to your XMPP server as the address of the XMPP Push Service.

Unregistering device

To unregister a device, you need to execute the adhoc command unregister-device available within the Push Notification component. This command will return a form which needs to be filled out.

This form consists of the following fields:

- provider: ID of a provider for which your devices was registered.
- device-token: Unique token which your application retrieved from a device or client library and was registered at this push notifications component.

When you submit this form, it will be processed and will respond with a result form to notify you that device was successfully unregistered from the push notifications component.
Chapter 4. Providers

Providers availability depends on the deployed binaries, by default Tigase includes following providers:

Tigase Push Component - FCM provider

Overview

Tigase Push Component - FCM provider is an implementation of FCM provider for Tigase Push Component. It allows Tigase Push Component to connect to Firebase Cloud Messaging and send notifications using this service.

Configuration

Enabling provider

To enable this provider, you need to enable fcm-xmpp-api bean within push component configuration scope.

Example.

```
push () {
    -'fcm-xmpp-api' () {
        # FCM configuration here
    }
}
```

Note

You need to pass FCM configuration parameters to make it work, see below.

Setting FCM credentials

FCM XMPP API provider will not work properly without API key and project id as this values are required for authorization by FCM. You need to get information from FCM account.

When you have this data, you need to pass sender id as sender-id property and server key as server-key property.

Example.

```
push () {
    -'fcm-xmpp-api' () {
        -'sender-id' = -'your-sender-id'
        -'server-key' = -'your-server-key'
    }
}
```
Connection pool

By default this provider uses single client to server connection to FCM for sending notifications. If in your use case it is too small (as you need better performance), you should adjust value of pool-size configuration property. Setting it to 5 will open five connections to FCM for better performance.

Example.

```python
push () {
    -'fcm-xmpp-api' () {
        -'pool-size' = 5
    }
}
```

Tigase Push Component - APNs provider

Overview

Tigase Push Component - APNs provider is an implementation of APNs provider for Tigase Push Component. It allows Tigase Push Component to connect to Apple Push Notification service and send notifications using this service.

Configuration

Enabling provider

To enable this provider, you need to enable apns-binary-api bean within push component configuration scope.

Example.

```python
push () {
    -'apns-binary-api' () {
        # APNs configuration here
    }
}
```

Note

You need to pass APNs configuration parameters to make it work, see below.

Setting APNs credentials

APNs binary API provider will not work properly without certificate file required for authorization by APNs and password to decrypt this certificate file. You need to get certificate using Apple Developer Account.

When you have this certificate, you need to pass path to certificate file as cert-file property, password as cert-password and APNS topic (bundle id) as apns-topic.

Example for /etc/apns-cert.p12, Pa$$word and com.bundle.id.
push () {
    -'apns-binary-api' () {
        -'cert-file' = -'/etc/apns-cert.p12'
        -'cert-password' = -'Pa$$w0rd'
        -'apns-topic' = -'com.bundle.id'
    }
}