How to support push notifications for new messages on phpFox Mobile apps

This tutorial will guide you step by step to enable push notifications for new messages on phpFox Mobile apps.

As you may know, Instant Messaging app supports 2 options of Chat Server: **NodeJS** and **Google Firebase**.

Basing on the chosen chat server, you can follow our detailed instructions below as they are totally different.

Requirements:

- Mobile API 4.2.1 or newer

If Chat server is using NodeJS

1. Configure Firebase Send ID

1. Go to AdminCP > Apps > Installed > Mobile API > Settings
2. Configure ***Firebase Sender ID*** setting with the server Sender Id of your Firebase project. You can get the sender id by going to **Firebase App console,** then go to **Project Settings > Cloud Messaging**
    - In case you are using our IM Hosting service, you can configure ***904356105667*** to ***Firebase Sender ID*** setting

2. If NodeJS server set up on your self-hosting

Open ssh and go to Apps/core-im/server, run the following command to update dependencies.

```
npm install
```

Edit `Apps/core-im/server/config.js` file and add a new config with configured values of [server_key] and [sender_id]

```
firebase: {serverKey: "[server_key]", senderId: "[sender_id]"}
```

Restart index.js with the command:

```
forever restartall
```

Now users can get notifications on Mobile App when setting up IM with NodeJS.

If Chat server is using Google Firebase

1. Set up Node.js and the Firebase CLI

You'll need a Node.js environment to write functions, and the Firebase CLI (which also requires Node.js and npm) to deploy functions to the Cloud Functions runtime.

From among the supported Node.js versions available for Cloud Functions, we **strongly recommend Node.js 8 for getting started**.

Once you have Node.js and npm installed, install the Firebase CLI via npm:

```
npm install -g firebase-tools
```

2. Initialize Firebase SDK for Cloud Functions

Run command **`firebase login`** in the terminal. The browser will be opened automatically in order for you to log in and authenticate the firebase tool.

```
firebase login
```

Then, create a new folder for Google Firebase project on your server. If the folder already exists, you can just go to your Firebase project directory

```
mkdir myproject
cd myproject
```

Run **`firebase init functions`**. The tool gives you an option to install dependencies with npm. If you want to manage dependencies in another way, just decline.

```
firebase init functions
```

Select **Javascript** for language support. Structure of your project will be as below

```
myproject
 +- .firebaserc    # Hidden file that helps you quickly switch between
 |                 # projects with `firebase use`
 |
 +- firebase.json  # Describes properties for your project
 |
 +- functions/     # Directory containing all your functions code
      |
      +- .eslintrc.json  # Optional file containing rules for JavaScript linting.
      |
      +- package.json  # npm package file describing your Cloud Functions code
      |
      +- index.js      # main source file for your Cloud Functions code
      |
      +- node_modules/ # directory where your dependencies (declared in
                       # package.json) are installed
```

Edit **package.json** file, add 2 new lines for ***"request": "^2.88.0"*** *and* ***"emojione": "^4.5.0"*** to **dependencies** section as below

```
{
  "name": "functions",
  ....
  "engines": {
    "node": "16"
  },
  "dependencies": {
    .....
    "request": "^2.88.0",
    "emojione": "^4.5.0"
  },
  ....
}
```

Then run **cd functions/ && npm install** (from project folder) to update project modules.

```
cd functions/ && npm install
```

Edit file **functions/index.js** and replace the content with following source codes

```
const functions = require('firebase-functions')
const admin = require('firebase-admin')
const request = require('request')
const emojione = require('emojione/lib/js/emojione')

admin.initializeApp()

const firestore = admin.firestore()

exports.sendPushNotification = functions.firestore.document(
  'rooms/{roomId}/messages/{messageId}').onCreate(event => {
  const writeData = event.data()
  const sender = writeData.sender
  const recipient = writeData.receiver
  const threadId = writeData.thread_id
  if (!writeData.sender_id || !writeData.server_key) {
    return false
  }
  const notifyTo = {
    recipient: recipient,
    sender_id: writeData.sender_id,
    server_key: writeData.server_key
  }
  firestore.doc('users/' + recipient).get().then(doc => {
    const senderData = doc.data()
    if (senderData.rooms) {
      let roomList = [], rooms = senderData.rooms, roomListId = []
      for (let i in rooms) {
        let room = rooms[i]
        if (room.active && room.messages && room.id !== threadId) {
            roomList.push(room)
        }
      }
      roomList.sort(dynamicSort('-last_update'))

      for (let i = 0; i < roomList.length; i++) {
        roomListId.push(roomList[i].id)
      }
      let roomListIds = [], chunk = 10;
      for (let i = 0; i < roomListId.length; i += chunk) {
        if (roomListIds.length < 10) {
          roomListIds.push(roomListId.slice(i, i + chunk))
        } else {
          break
        }
      }
      return processNotificationBadge(roomListIds, sender, recipient, writeData, notifyTo)
    } else {
      return prepareSendNotification(sender, writeData, notifyTo, 1)
    }
  }).catch(e => console.warn(e))
  return true
})

dynamicSort = (property) => {
  let sortOrder = 1
  if (property[0] === "-") {
    sortOrder = -1
    property = property.substr(1)
  }
  return (a, b) => {
    let result = (a[property] < b[property]) ? -1 : (a[property] > b[property]) ? 1 : 0
    return result * sortOrder
  }
}
```

```
processNotificationBadge = (roomListId, sender, recipient, writeData, notifyTo) => {
  let badge = 1
  let processRooms = (i, size, listIds) => {
    if (size === 0 || typeof listIds[i] === "undefined") {
      return prepareSendNotification(sender, writeData, notifyTo, badge)
    }
    firestore.collection('rooms').
      where('id', 'in', listIds[i]).get().then(snapShot => {
      if (snapShot.size) {
        snapShot.forEach(doc => {
          let roomData = doc.data()
          if (roomData && roomData['users'][recipient] &&
            roomData['users'][recipient] > 0) {
            badge++
          }
        })
      }
      if (i === (size - 1)) {
        return prepareSendNotification(sender, writeData, notifyTo, badge)
      } else {
        return processRooms(i + 1, size, listIds)
      }
    }).catch(e => console.warn(e))

    return true
  }
  processRooms(0, roomListId.length, roomListId)
}

prepareSendNotification = (sender, writeData, notifyTo, badge) => {
  let payload = {}
  firestore.doc('users/' + sender).get().then(doc => {
    const senderData = doc.data()
    payload = {
      notification: {
        title: senderData.name,
        body: writeData.text
          ? emojione.shortnameToUnicode(writeData.text)
          : '[FILE]',
        sound: 'default',
        badge: badge > 99 ? '99+' : badge.toString()
      },
      data: {
        resource_link: 'chat/' + senderData.id,
        web_link: 'chat/' + senderData.id
      }
    }
    return getTokenToSend(payload, notifyTo)
  }).catch(e => console.warn(e))
}

getTokenToSend = (payload, notifyTo) => {
  const recipientId = Buffer.from(notifyTo.recipient, 'base64').
    toString('ascii')
  const notificationKeyName = 'user-' + recipientId
  return getNotificationKey({
    senderId: notifyTo.sender_id,
    serverKey: notifyTo.server_key
  }, notificationKeyName).
    then(notificationKey => admin.messaging().
      sendToDeviceGroup(notificationKey, payload))
}

getNotificationKey = (options, notificationTokenName) => {
  return new Promise((resolve, reject) => {
    request({
      url: 'https://fcm.googleapis.com/fcm/notification',
      method: 'GET',
      json: true,
      headers: {
        Authorization: 'key=' + options.serverKey,
```

```
        project_id: options.senderId,
        'Content-Type': 'application/json'
      },
      qs: { notification_key_name: notificationTokenName }
    }, (error, httpResponse, body) => {
      if (!error && body.notification_key) {
        resolve(body.notification_key)
      } else {
        reject(error || body)
      }
    })
  })
}
```

Run **firebase deploy --only functions** from the project folder to deploy.

```
firebase deploy --only functions
```

Now users can get notifications on Mobile App when chatting with Firebase.