Holla Messenger

Quickly send and receive Holla Messenger messages from your computer.

Created: 1/11/2021
Latest update: 1/01/2022
BY: Mert Çetin
Support Platform : https://github.com/meforce/holla-messenger/issues
Requirements : -Node JS: v14+ | -Npm: v6+ | ionic 5+ | capacitorjs 3+

✓ J Join our Official Github issues for more discussion and hints, bugs, fixes, and anything related to Mobile applications.

Holla Messenger is a FREE messaging app available for Android and other smartphones.

Holla Messenger performs server storage and some functions without the need for a server. Does not use server-based database.

Data transfer processes are instantaneously end-to-end encrypted.

For the transactions to take place; The sender and receiver must be active.

Holla Messenger uses your phone's Internet connection (4G/3G/2G/EDGE or Wi-Fi if available) to let you send and receive messages to family and friends.

Holla Messenger aims to provide instant communication.

So your information is safe. The data is encrypted end-to-end between the two parties and delivered to the receiver. It is not stored on the server. or not stored.

New generation secure communication technology

RTCMultiConnection.js

WebRTC JavaScript library for peer-to-peer applications (screen sharing, audio/video conferencing, file sharing, media streaming etc.)

Holla Messenger uses rtcmulticonnection infrastructure.

In this way, we get rid of a lot of workload.

Ionic 5+ is a system built on capacitor 3+. rtcmulticonnection plugin is included in the system.

Features

- Send Message
- Video or audio call
- Sound recording or sending photos
- End-to-end encryption
- Chat theme customization
- Dark Mode
- Delete message
- Adding a contact
- Holla Messenger WEB

Future Update Features

- Create a group
- Group Chat
- Group voice and video calling
- Gif and stickers
- Location sharing
- User blocking
- More efficient user search feature
- And more...

Mobile, PWA About Access Permissions

-Storage Permit

To store Correspondence, Received Voice Recordings and Photos Files, your phone must have access to the storage partition.

-Camera and microphone Access

Requires this access to video chat or voice chat or send audio recordings

Note that the Development Process is still ongoing.

Your ideas are very important to us! If you have feedback, questions or concerns, please email:

hollatechnology@yandex.com

or follow us on ig:

https://www.instagram.com/hollamessenger/

@hollamessenger

Installation Guide

The installation is pretty easy, please follow the steps below:

- Holla Messenger Script, you can get it from
- Download Node.js - Install Node.js Here
- Download ionic - Install ionic https://ionicframework.com/docs/intro/cli

- Unzip the Holla archive, extract it to new folder, and then open the folder.
- In the main folder you will find the solution.
- Extract holla-messenger.zip to your workspace

Recommended versions

nodejs   v14.17.3

npm 6.14.13

@ionic/cli: ^6.14.1

@capacitor/core: ^3.1.2

install node.js

First, we install node.js on our computer. For installation, you can download the version suitable for your computer from the link below.

https://nodejs.org/en/download

With the installation of Node.js, the npm command will be activated in your computer's command client. To check, you can check it by typing `npm -v` in the command prompt.

install ionic

Before proceeding, make sure your computer has Node.js installed. See these instructions to set up an environment for Ionic.

Install the Ionic CLI with npm:

```
$ npm install -g @ionic/cli
```

If there was a previous installation of the Ionic CLI, it will need to be uninstalled due to a change in package name.

```
$ npm uninstall -g ionic

$ npm install -g @ionic/cli
```

package.json Editing

Open to edit package.json;
If automatic updating will be enabled, it is necessary to connect a github repo.
(The github repo will only contain all packaged applications.)

```
{
  "name": "ion-holla-messenger",
  "version": "1.0.0",
  "author": "meforce",
  "license": "MIT",
  "description": "Holla Messenger",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "keywords": [
    "ionic",
    "ionic starter",
    "ionic template",
    "ionic social network",
    "social network",
    "holla",
    "holla messenger",
    "messenger"
  ],
  "bugs": {
    "url": "https://github.com/meforce/holla-messenger/issues"
  },
  "homepage": "https://github.com/meforce/holla-messenger#readme",
```

Download and initialize HollaMessenger source code with these commands:

- Open command prompt (CMD), PowerShell or terminal
- cd holla-messenger -> Get the directory path from which we extracted the project and run it
- npm install -> npm include in the Project
- npm install -g @ionic/cli

Usage Codes

```
npx cap open android   ->  Android Studio Starts

npx cap open ios            ->  xcode Starts
npx cap add android   ->  Builds Android from Scratch

npx cap add ios           -> Builds ios from Scratch


npx cap sync web       ->  Copy the Capacitor runtime bundle (capacitor.js) into your web assets directory

npx cap sync               ->  Builds and Imports the Created Project

npx cap copy               ->     "        "         "     Copy


ionic serve     ->   Starts the Project

ionic build --prod --release    ->  ionic packaging
```

Installing Capacitor

This guide will help you install Capacitor into an existing frontend web app.

If starting a new app, we recommend using the documentation from your JavaScript framework of choice and then following this guide to integrate Capacitor.

You can also create a new basic app with npm init @capacitor/app.

Capacitor provides a native mobile runtime and API layer for web apps. It does not come with any specific set of UI controls. We recommend you use a mobile component framework (such as Ionic Framework).

Before you start

Make sure your environment is set up for the platforms you will be building for.

Project Requirements

Capacitor was designed to drop into any modern JavaScript web app. Projects must meet the following requirements:

Must have a package.json file.

Must have a separate directory for web assets.

Must have an index.html file with a <head> tag in the root of the web assets directory.

Adding Capacitor to your app

In the root of your app, install Capacitor:

```
npm install @capacitor/core
```

```
npm install @capacitor/cli --save-dev
```

Then, initialize Capacitor using the CLI questionnaire:

```
npx cap init
```

The CLI will ask you a few questions, starting with your app name, and the package id you would like to use for your app.

The npx cap command is how Capacitor is executed locally on the command-line in your project. Learn more about the Capacitor CLI.

Files Structure

- node_modules - All dependencies included in this folder (ionic).
- src folder - all project files are included.
- src\app\pages - Pages used in the project
- src\app\providers - functions used in the project
- src\app\services - external page and commands used in the project
- src\app\shared - external page and link commands used in the project
- src\assets - image, sound, language and data files used in the project
- www folder – contains the packaged files.
- package.json - NodeJS application and package details.

Initial Arrangement

1. Folder Go : src\assets\data
2. Open : config.json

```
{
"alllist": [
{
```

"get": {

"version": "1.0.0",   -> App Version Number

"socketurl": "https://rtcmulticonnection.herokuapp.com:443/",

-> Server socket url used

The default url demo rtcmulticonnection.org is the provided server. You can use it. It is useful to set up your own socket url

Socket server setup https://github.com/muaz-khan/RTCMultiConnection-Server

"server_id": "hollamessenger-server-id-random-code",

-> Server code. must be tall and unique. EG: appname-server-id-45ASD4da43f445d3324/*fsa/

"server_password": "server-pass-random-code",

-> Server Pass. must be tall and unique. EG: appname-pass-sad4dsad_5d3324/*fsa324/as

"user_encryption_token": "token_random_code",

-> Encryption of data is also an intermediary keyword

"pwamobile_encryption_token": "token_crpty_random_code",

-> PWA Encryption of data is also an intermediary keyword

-> Privacy and terms of use url (App not yet added)

"registerTersm": "https://domain.com/terms",

"registerPrivacy": "https://domain.com/privacy",

-> Mail Settings. pwa is used to log in and verify the accuracy of the mail.

```
"mail_securetoken": "868bf7bb-7a54-4964-991b-0e1d768b6e2f",

"mail_host": "smtp.domain.com",

"mail_username": "username@domain.com",

"mail_password": "pass",

"mail_from": "from@domain.com"


}

}

]

}
```

Language Editing

1. Folder Go : src\assets\i18n
2. tr, en, ru
3. copy and paste one to duplicate. replace filename with country code. then open the file and complete the translation process
4. For the added language to appear;

open folder: src\app\pages\auth\language

open: language.page.ts

line: 41

```
this.langdata = [

{ id: "1", orgintitle: "English", title: "English", code: "en", lock: false  },
{ id: "2", orgintitle: "Русский", title: "Russian", code: "ru", lock: false  },

{ id: "3", orgintitle: "Türkçe", title: "Turkish", code: "tr", lock: false  },

{ id: "4", orgintitle: "Azərbaycan", title: "Azerbaijan", code: "az", lock: true  },

{ id: "5", orgintitle: "Deutsche", title: "German", code: "de", lock: true  },

{ id: "6", orgintitle: "Português", title: "Portuguese", code: "pt", lock: true  },

{ id: "7", orgintitle: "简体中文", title: "Chinese Simplified", code: "zh", lock: true  },

{ id: "8", orgintitle: "日本語", title: "Japanese", code: "ja", lock: true  }

];
```

lock: false -> **Available**   lock: true -> **Not used**

Logo Used in the Application

Go Folder: src\assets\logo

Edit: favicon.png

Using Capacitor in a Web Project

Capacitor fully supports traditional web and Progressive Web Apps. In fact, using Capacitor makes it easy to ship a PWA version of your iOS and Android app store apps with minimal work.

Browser Support

Capacitor core and plugins build for ES2017. This newer JavaScript syntax is supported in all modern browsers (including those that power PWAs on iOS and Android), but will not work in IE11 without additional JavaScript transformations, e.g. with Babel.

Plugins with web support will perform feature detection and throw exceptions if a browser does not support a particular Web API.

Installation

If you’re already building with Capacitor for iOS or Android, there are no additional installation steps!

Otherwise, see the Installation guide before continuing.

Using Capacitor as a Module

Most commonly, apps will be using a framework with a build system that supports importing JavaScript modules. By importing from @capacitor/core, or by importing a plugin, the Capacitor JavaScript runtime will be loaded with your app.

Using Capacitor as a Script Include

To use the Capacitor runtime in a web app that is not using a build system or bundler/module loader, do the following:

Set **bundledWebRuntime** to **true** in the Capacitor configuration file

```
"bundledWebRuntime": true
```

Copy the Capacitor runtime bundle (capacitor.js) into your web assets directory

```
npx cap sync web
```

Import capacitor.js in index.html before other JavaScript

```
<script src="capacitor.js"></script> (ADD build folder www index.html)

<script src="your/app.js"></script>
```

Going Live

When you’re ready to publish your Progressive Web App and share it with the world, just upload the contents of your web assets directory.

That will contain everything you need to run your app!