# Electron Build Process (For Mac, Linux and Windows)

Prerequisite

node --version

12.13.0

npm install -g @ionic/cli

npm install -g @angular/cli

Step 1 → Clone project and run these commands:

git clone https://isthiaq@bitbucket.org/tigerit/communicator-desktop-pwa.git

cd communicator-desktop-pwa

git checkout dev

git pull origin dev

npm i

ionic build

npx cap add @capacitor-community/electron

cd electron

npm install electron-dl node-machine-id electron-is-dev

Step 2 → Update package.json file

In package.json update the followings keys values

  "name": "commchatbeta",

  "version": "1.1.40-internal",

  "description": "CommChat (Beta) is an end to end encrypted messaging application.",

"scripts": {

    "build": "tsc",

    "electron:start": "npm run build && electron ./",

    "electron:pack": "npm run build && electron-builder build --dir",

    "electron:build-windows": "npm run build && electron-builder build --windows",

    "electron:build-mac": "npm run build && electron-builder build --mac",

    "electron:build-linux": "npm run build && electron-builder build --linux"

  },

"build": {

    "appId": "commchatbeta.desktop.pwa",

    "productName": "CommChat (Beta)",

    "nsis": {

      "deleteAppDataOnUninstall": true

    },

    "files": [

      "assets/\*",

      "build/\*",

      "preloader.js",

      "plugins/\*",

      "capacitor.config.json",

      "app/\*\*"

    ],

    "mac": {

      "category": "public.app-category.developer-tools",

      "target": "dmg",

      "icon": "assets/logo\_fx.png"

    },

    "win": {

      "target": "nsis",

      "icon": "assets/logo\_fx.png"

    },

    "linux": {

      "target": "snap",

      "icon": "assets/logo\_fx.png"

    }

  }

Step 3 → Update electron/src/index.ts file

Replace the the code in electron/src/index.ts with the following code

import { app, dialog, ipcMain } from "electron";

import { createCapacitorElectronApp } from "@capacitor-community/electron";

import \* as path from "path";

const { download } = require("electron-dl");

const os = require("os");

import { machineId, machineIdSync } from "node-machine-id";

const myCapacitorApp = createCapacitorElectronApp({

  splashScreen: {

    useSplashScreen: false,

    splashOptions: {

      imageFilePath: path.join(app.getAppPath(), "assets", "logo\_fx.png")

    }

  },

  mainWindow: {

    windowOptions: {

      icon: path.join(app.getAppPath(), "assets", "logo\_fx.png"),

      width: 1150,

      height: 892,

      minHeight: 600,

      minWidth: 930

    }

  },

  applicationMenuTemplate: [

    {

      role: "options",

      submenu: [

        { role: "about", label: "About CommChat (Beta)" },

        { type: "separator" },

        { label: "Undo", accelerator: "CmdOrCtrl+Z", selector: "undo:" },

        { label: "Redo", accelerator: "Shift+CmdOrCtrl+Z", selector: "redo:" },

        { type: "separator" },

        { label: "Cut", accelerator: "CmdOrCtrl+X", selector: "cut:" },

        { label: "Copy", accelerator: "CmdOrCtrl+C", selector: "copy:" },

        { label: "Paste", accelerator: "CmdOrCtrl+V", selector: "paste:" },

        {

          label: "Select All",

          accelerator: "CmdOrCtrl+A",

          selector: "selectAll:"

        },

        { type: "separator" },

        {

          label: "Exit CommChat (Beta)",

          accelerator: "CmdOrCtrl+Q",

          click() {

            app.exit();

          }

        }

      ]

    }

  ]

});

app.setAppUserModelId("CommChat (Beta)");

app.on("ready", () => {

  myCapacitorApp.init();

  myCapacitorApp.getMainWindow().setMenuBarVisibility(false);

  // myCapacitorApp.getMainWindow().webContents.openDevTools(); // for debug build uncomment this line

  ipcMain.on("download", (event, info) => {

    let path = app.getPath("downloads");

    download(

      myCapacitorApp.getMainWindow(),

      info.url,

      info.properties

    ).then(dl =>

      myCapacitorApp

        .getMainWindow()

        .webContents.send("download complete", dl.getSavePath())

    );

  });

  ipcMain.on("device-id", (event, info) => {

    myCapacitorApp.getMainWindow().webContents.send("device-id", deviceId);

  });

});

let deviceId;

machineId().then(id => {

  deviceId = id;

});

app.on("browser-window-blur", () => {

  myCapacitorApp

    .getMainWindow()

    .webContents.send("asynchronous-message", "blur");

});

app.on("browser-window-focus", () => {

  myCapacitorApp

    .getMainWindow()

    .webContents.send("asynchronous-message", "focus");

  myCapacitorApp

    .getMainWindow()

    .webContents.send("current-platform", os.platform());

  myCapacitorApp

    .getMainWindow()

    .webContents.send("app-version", { version: app.getVersion() });

});

app.on("window-all-closed", function (event) {

  if (process.platform !== "darwin") {

    app.quit();

  }

});

app.on("activate", function () {

  if (myCapacitorApp.getMainWindow().isDestroyed()) myCapacitorApp.init();

});

Step 4 → Copy the logo of the application

Copy the logo of the application from location electron/app/assets/logo\_fx.png to

electron/assets/

make sure you are in electron directory and execute this command:

cp app/assets/logo\_fx.png assets/

Step 5 → Now run build commands in project/electron directory

First, run

ionic cap sync electron

(Inorder to reflect code changes in electron app always run this command before running build command)

For Production please use this command.

ionic cap sync electron --prod

Then,

For Mac build run:

npm run electron:build-mac

For Linux build run:

npm run electron:build-linux

For Windows build run:

 npm run electron:build-windows

For Errors

If running these build commands throws an error then run:

  npm install @types/node

then run the build command

To fix npm ERR! namespace 'fs' has no exported member 'RmOptions' , add

"typeRoots": [

      "./functions/node\_modules/@types"

    ],

in **/electron/tsconfig.json** bellow “target”:

## (Updated index.ts for windows and linux)

*import* { app, dialog, ipcMain, Menu } *from* "electron";

*import* { createCapacitorElectronApp } *from* "@capacitor-community/electron";

*import* \* *as* path *from* "path";

const { download } = require("electron-dl");

const isDev = require('electron-is-dev');

const os = require("os");

*import* { machineId, machineIdSync } *from* "node-machine-id";

let window: any;

let deviceId: any;

let willQuitApp = false;

const myCapacitorApp = createCapacitorElectronApp({

  splashScreen: {

    useSplashScreen: false,

    splashOptions: {

      imageFilePath: path.join(app.getAppPath(), "assets", "logo\_fx.png")

    }

  },

  mainWindow: {

    windowOptions: {

      icon: path.join(app.getAppPath(), "assets", "logo\_fx.png"),

      width: 1150,

      height: 892,

      minHeight: 600,

      minWidth: 930

    }

  },

  applicationMenuTemplate: [

    {

      role: "options",

      submenu: [

        { role: "about", label: "About CommChat (Beta)" },

        { type: "separator" },

        { label: "Undo", accelerator: "CmdOrCtrl+Z", selector: "undo:" },

        { label: "Redo", accelerator: "Shift+CmdOrCtrl+Z", selector: "redo:" },

        { type: "separator" },

        { label: "Cut", accelerator: "CmdOrCtrl+X", selector: "cut:" },

        { label: "Copy", accelerator: "CmdOrCtrl+C", selector: "copy:" },

        { label: "Paste", accelerator: "CmdOrCtrl+V", selector: "paste:" },

        {

          label: "Select All",

          accelerator: "CmdOrCtrl+A",

          selector: "selectAll:"

        },

        { type: "separator" },

        {

          label: 'developer tool',

          accelerator: 'CmdOrCtrl+Shift+D',

          click() {

            myCapacitorApp.getMainWindow().webContents.openDevTools();

          },

        },

        {

          label: "Exit CommChat (Beta)",

          accelerator: "CmdOrCtrl+Q",

          click() {

            app.exit();

          }

        }

      ]

    }

  ]

});

const menuTemplateDev = [

  {

    label: 'Help',

    submenu: [

      {

        label: 'Troubleshooting',

        submenu: [

          {

            label: 'Open developer tools',

            accelerator: 'CmdOrCtrl+Shift+D',

            click() {

              myCapacitorApp.getMainWindow().webContents.openDevTools();

            },

          },

        ],

      },

    ],

  },

];

app.setAppUserModelId("CommChat (Beta)");

machineId().then(id => {

  deviceId = id;

});

const gotTheLock = app.requestSingleInstanceLock();

*if* (!gotTheLock) {

  app.quit();

} *else* {

  app.on("second-instance", (event, commandLine, workingDirectory) => {

*if* (myCapacitorApp.getMainWindow()) {

*if* (myCapacitorApp.getMainWindow().isMinimized())

        myCapacitorApp.getMainWindow().restore();

      myCapacitorApp.getMainWindow().focus();

    }

  });

}

app.on("ready", () => {

  myCapacitorApp.init();

  myCapacitorApp.getMainWindow().setMenuBarVisibility(true);

  myCapacitorApp.getMainWindow().setMenu(Menu.buildFromTemplate(menuTemplateDev));

*if* (isDev) {

    myCapacitorApp.getMainWindow().webContents.openDevTools();

  }

  window = myCapacitorApp.getMainWindow();

  window.on("close", e => {

*if* (!willQuitApp) {

      e.preventDefault();

      myCapacitorApp

        .getMainWindow()

        .webContents.send("all-window-closed", true);

    }

*else* {

      window = null;

    }

  });

  ipcMain.on("all-interval-cleared", (event, info) => {

    console.log("all-interval-cleared", info);

*if* (process.platform !== "darwin" && info.cleared) {

      willQuitApp = true

      app.quit();

    } *else* {

      app.hide();

    }

  });

  ipcMain.on("download", (event, info) => {

    console.log(info);

    let path = app.getPath("downloads");

    download(

      myCapacitorApp.getMainWindow(),

      info.url,

      info.properties

    ).then(dl =>

      myCapacitorApp

        .getMainWindow()

        .webContents.send("download complete", dl.getSavePath())

    );

  });

  ipcMain.on("device-id", (event, info) => {

    myCapacitorApp.getMainWindow().webContents.send("device-id", deviceId);

  });

  ipcMain.on('notification-click', (event, info) => {

    console.log("notification-click");

    app.focus();

*if* (myCapacitorApp.getMainWindow()) {

      myCapacitorApp.getMainWindow().focus();

    }

  });

});

app.on("browser-window-blur", () => {

  console.log("browser-window-blur");

  myCapacitorApp

    .getMainWindow()

    .webContents.send("asynchronous-message", "blur");

});

app.on("browser-window-focus", () => {

  console.log("browser-window-focus");

  myCapacitorApp

    .getMainWindow()

    .webContents.send("asynchronous-message", "focus");

  myCapacitorApp

    .getMainWindow()

    .webContents.send("current-platform", os.platform());

  myCapacitorApp

    .getMainWindow()

    .webContents.send("app-version", { version: app.getVersion() });

});

app.on("window-all-closed", function (event) {

*// if (process.platform !== "darwin") {*

*//   app.quit();*

*// }*

});

app.on("activate", function () {

  app.show();

*// if (myCapacitorApp.getMainWindow().isDestroyed()) myCapacitorApp.init();*

});

## (Updated index.ts for Mac)

*import* { app, dialog, ipcMain } *from* "electron";

*import* { createCapacitorElectronApp } *from* "@capacitor-community/electron";

*import* \* *as* path *from* "path";

const { download } = require("electron-dl");

const os = require("os");

*import* { machineId, machineIdSync } *from* "node-machine-id";

const { Menu, MenuItem } = require('electron')

let window: any;

let deviceId: any;

let willQuitApp = false;

let appQuitted = false;

let forceQuit = false;

const myCapacitorApp = createCapacitorElectronApp({

  splashScreen: {

    useSplashScreen: false,

    splashOptions: {

      imageFilePath: path.join(app.getAppPath(), "assets", "logo\_fx.png")

    }

  },

  mainWindow: {

    windowOptions: {

      icon: path.join(app.getAppPath(), "assets", "logo\_fx.png"),

      width: 1150,

      height: 892,

      minHeight: 600,

      minWidth: 930

    }

  }

});

app.setAppUserModelId("CommChat (Beta)");

machineId().then(id => {

  deviceId = id;

});

const gotTheLock = app.requestSingleInstanceLock();

*if* (!gotTheLock) {

  app.quit();

} *else* {

  app.on("second-instance", (event, commandLine, workingDirectory) => {

*if* (myCapacitorApp.getMainWindow()) {

*if* (myCapacitorApp.getMainWindow().isMinimized())

        myCapacitorApp.getMainWindow().restore();

      myCapacitorApp.getMainWindow().focus();

    }

  });

  app.on("ready", () => {

    myCapacitorApp.init();

    Menu.setApplicationMenu(menu)

    // myCapacitorApp.getMainWindow().setMenuBarVisibility(false);

    // myCapacitorApp.getMainWindow().webContents.openDevTools();

    window = myCapacitorApp.getMainWindow();

    window.on("close", e => {

*if* (process.platform === "darwin" && forceQuit) {

        console.log("inside force quit");

        appQuitted = true;

        window = null;

      } *else* {

*if* (!willQuitApp) {

          e.preventDefault();

          myCapacitorApp

            .getMainWindow()

            .webContents.send("all-window-closed", true);

        } *else* {

          window = null;

        }

      }

    });

    ipcMain.on("all-interval-cleared", (event, info) => {

      console.log("all-interval-cleared", info);

*if* (process.platform !== "darwin" && info.cleared) {

        willQuitApp = true;

        app.quit();

      } *else* {

        app.hide();

      }

    });

    ipcMain.on("download", (event, info) => {

      console.log(info);

      let path = app.getPath("downloads");

      download(

        myCapacitorApp.getMainWindow(),

        info.url,

        info.properties

      ).then(dl =>

        myCapacitorApp

          .getMainWindow()

          .webContents.send("download complete", dl.getSavePath())

      );

    });

    ipcMain.on("device-id", (event, info) => {

      myCapacitorApp.getMainWindow().webContents.send("device-id", deviceId);

    });

    ipcMain.on('notification-click', (event, info) => {

      console.log("notification-click");

      app.focus();

*if* (myCapacitorApp.getMainWindow()) {

        myCapacitorApp.getMainWindow().focus();

      }

    });

  });

  app.on("browser-window-blur", () => {

    console.log("browser-window-blur");

*if* (!appQuitted) {

      myCapacitorApp

        .getMainWindow()

        .webContents.send("asynchronous-message", "blur");

    }

  });

  app.on("browser-window-focus", () => {

    console.log("browser-window-focus");

    myCapacitorApp

      .getMainWindow()

      .webContents.send("asynchronous-message", "focus");

    myCapacitorApp

      .getMainWindow()

      .webContents.send("current-platfrom", os.platform());

    myCapacitorApp

      .getMainWindow()

      .webContents.send("app-version", { version: app.getVersion() });

  });

  app.on("window-all-closed", function(event) {

*// if (process.platform !== "darwin") {*

*//   app.quit();*

*// }*

    console.log("inside window-all-closed");

  });

  app.on("before-quit", function(event) {

    console.log("inside before-quit");

*if* (!forceQuit) {

      event.preventDefault();

      forceQuit = true;

      app.quit();

    }

  });

  app.on("activate", function() {

    app.show();

*// if (myCapacitorApp.getMainWindow().isDestroyed()) myCapacitorApp.init();*

  });

}

const template:any[] = [

  {

    role: "options",

    submenu: [

      { role: "about", label: "About CommChat (Beta)" },

      { type: "separator" },

      { role: 'hide', label:'Hide' },

      { role: 'hideOthers' },

      { role: 'unhide' },

      { type: "separator" },

      {

        label: "Exit CommChat (Beta)",

        accelerator: "CmdOrCtrl+Q",

        click() {

          app.exit();

        }

      },

    ]

  },

  {

    label: 'File',

*// role: 'window',*

    submenu: [

       {

          role: 'close'

       }

    ]

 },

 {

   label:"Edit",

   role:'options',

   submenu:[

    { label: "Undo", accelerator: "CmdOrCtrl+Z", selector: "undo:" },

    { label: "Redo", accelerator: "Shift+CmdOrCtrl+Z", selector: "redo:" },

    { type: "separator" },

    { label: "Cut", accelerator: "CmdOrCtrl+X", selector: "cut:" },

    { label: "Copy", accelerator: "CmdOrCtrl+C", selector: "copy:" },

    { label: "Paste", accelerator: "CmdOrCtrl+V", selector: "paste:" },

    {

      label: "Select All",

      accelerator: "CmdOrCtrl+A",

      selector: "selectAll:"

    },

    { type: "separator" },

   ]

 },

{

  label: 'Help',

  submenu: [

    {

      label: 'Troubleshooting',

      submenu: [

        {

          label: 'Open developer tools',

          accelerator: 'CmdOrCtrl+Shift+D',

          click() {

            myCapacitorApp.getMainWindow().webContents.openDevTools();

          },

        },

      ],

    },

  ],

}

];

const menu = Menu.buildFromTemplate(template)