NEO Mail Bot Framework White Paper

Email was the killer app of the early days of the Internet, it connect the people and also connect the business, but now days IM and various rich functions messaging applications dominate the related market, its fate of future is not looking good.

But even without the fancy functions of “modern” messaging functions, Email is not dying, and in fact a huge amount of people still keep actively using it on a daily basis, and still many big names actively put lots of resource to provide free email service to the public.

Why? Because it’s a much less intrusive way of communication, it doesn’t force you to instantly reply to someone’s message. And for marketing message delivery, Email also brings a big advantage (or disvantage, depend on you side), once you got someone’s Email address, you can send him/her a message, but for other messaging applications, you will not get the chance, you should first let others to add you to a ‘friends’ list.

But unfortunately (for marketing/sales guys, of cause not for others), these days, people don’t waste time to read mail from strangers, and mail service provider also actively throw these mails into junk folder.

Consider the huge amount of Email users, it’s a big pity to not be able make some money from them. So what could bring light to these poor marketer of spammer?

Now the saver is coming, block chain and crypto currency has the potential to make the Email based marketing business sexy again.

People don’t like marketing message, but if it bring money, if it even bring cool money (those money no one knows where the value will be at next second), they will reading these lovely marketing message carefully and reply them really carefully to avoid missing the lovely money.

Good story, right? So let’s make it happen!

Marketing is a complex task, to drive people into the desired way of thinking is really hard, so the marketer need the helps of powerful tools.

Now days, which kind of tools are most powerful? Of cause robot!

So dear reader, if you agree to the above reasoning, you will arrive the same conclusion as me: mail bots which could connect the huge amount of mail users to the wonderful block chain world will be a big thing.

This project is an attempt to make this dream picture realized more easily.

The target of this project is to let people only focus on the incentive logic, throw all the dirty work to the framework implementation.

Five parts form the framework：

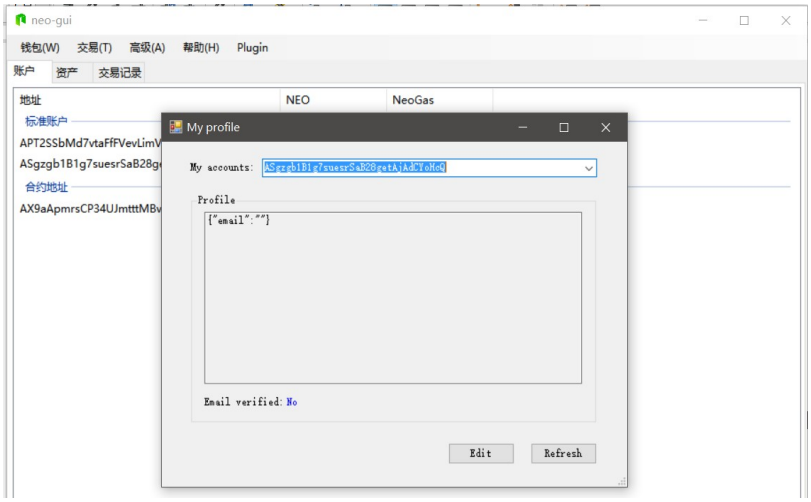
1. Mail api server: responsible for sending and receiving mails, I use Postal (https://github.com/atech/postal)
2. NEO node and rpc server: provide the connection with the NEO network, I implemented an enhanced neo-cli (https://github.com/gotmyname2018/neo-gui-nel/tree/master/neo-cli)

3. neo-mail-bot: nodejs server which provide both email / web api to the user side (https://github.com/gotmyname2018/crypto-mail-bot).
4. Account profile smart contract: deploy to the test net at (0xd39b50214e6020f324a3cbf026a575a7b2b6bec5), the code (https://github.com/gotmyname2018/crypto-mail-bot), it provide email address ↔ NEO account address two way binding.
5. GUI for interop with account profile smart contract: I implemented an enhanced neo-gui (https://github.com/gotmyname2018/neo-gui-nel/tree/master/neo-gui) and account profile plugin (https://github.com/gotmyname2018/neo-gui-nel/tree/master/plugin_profile).

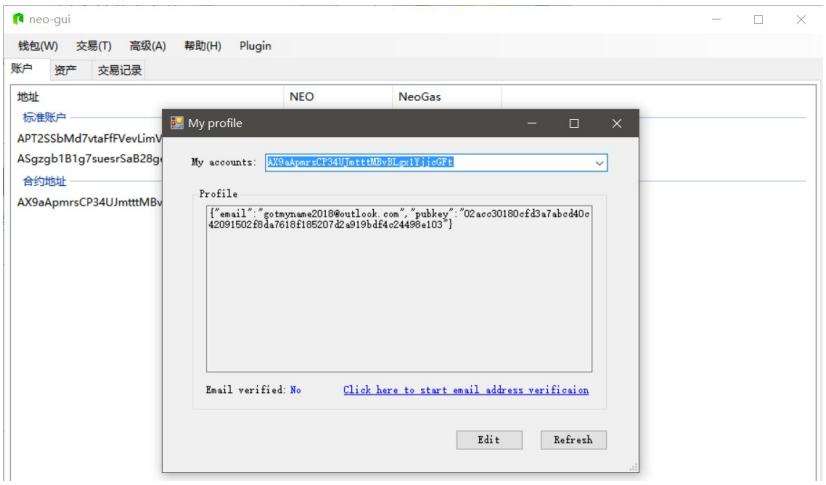
On the bot side, neo-mail-bot provide profile binding service through web api, it call the account profile smart contract functions through neo-cli RPC to do the work.

To achieve an email address ↔ NEO account address two way binding requires following steps:

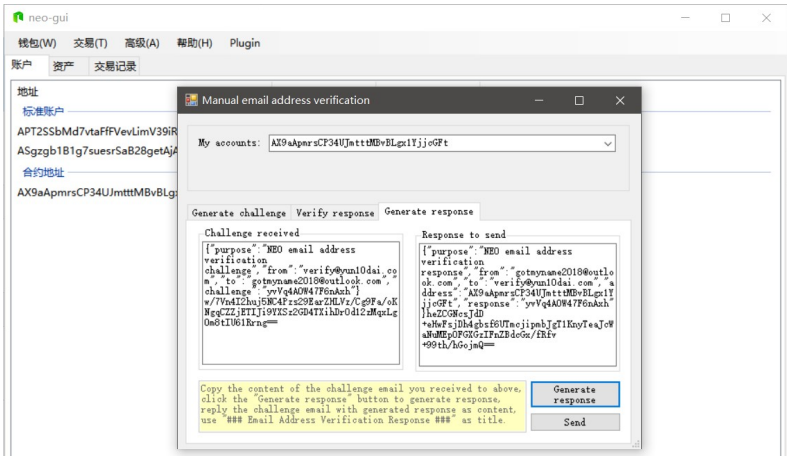
1. On the user side, user use account profile plugin of neo-gui to send a request to neo-mail-bot to register an NEO address → profile (email address included) one way binding.



2. Neo-mail-bot received the request and call account profile smart contract to register the above one way binding information on NEO block chain.
3. On the user side, user use the “my profile” function of account profile plugin to send a request to neo-mail-bot to start to verify the email address bind is valid (i.e to check the email address ownership).



4. Neo-mail-bot received the request and call Postal send mail api to send a challenge mail to the mail address to be verified.
5. On the user side, user check his/her mail box to get the challenge mail and then copy the body of the mail (the challenge message), and use this challenge as input and use the manual verify function of the account profile plugin to generate a response and send to the neo-mail-bot to verify.

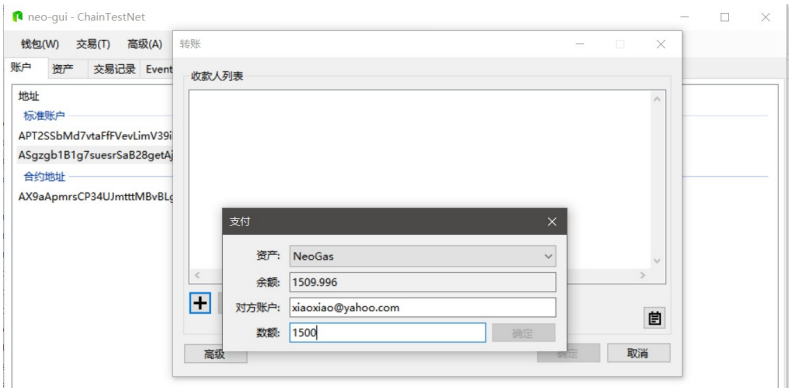


6. Neo-mail-bot receive the response and check its validity against original challenge message, if valid then call the account profile smart contract to grant the email to the corresponding NEO account address. After this, anyone who make on chain query about this user's profile by using this user's NEO account address or email address as input will return this user's verified profile information (currently includes email address and public key)

The above procedure required a central authority to perform the email address ownership checking. But if user insists to double check the ownership (after the central authority already make the checking), he/she could use the manual verify function of profile plugin to generate the challenge, send the challenge mail, and wait the other side to generate the response and send back through mail, and verify the validity of response.

A manual grant email ownership function was provided to make testing the verification process) more easy (could perform this process purely on user side, bypass the complex server side setup).

The neo-gui also got a function enhancement by utilizing these on chain two way binding information: when transfer asset to other user, you could use either that user's email address or account address as destination.



Once got the two way binding functions, and the five ready to use build blocks for crypto mail bot, interesting email based dApps could be fast and easily developed.