Dash Evolution

Evolution Prototype

Design Overview

Rev 1

Evan Duffield - evan@dash.org

1 Planning and Execution

This should document all of the changes we're going to make in the first implementation of Dash Evolution. Eventually this document can be released with v13 as it should exactly describe the new features. The evolution prototype is non-secure and is just a proof-of-concept of the overall design of the user friendly features we plan on making available eventually.

The implementation of this includes dashd (dapi, users and simple queries), dash-t2 (websockets and http) and a javascript based wallet (friends, hd wallet, chat, payments). This is intended to be quite simple and extensible for v13.

1.1 Daemon / Websockets / DashPay Setup

Local testing can be done by setting up the electrum evolution client at https://github.com/evan82/electrum-dash/tree/v13-evo-demo, along with the second tier server at https://github.com/evan82/dash-2t and a local daemon. To configure the local daemon, see the documentation in the dash-docs folder of the local daemon.

2. Client Documentation

2.1 Introduction

For the eventual release of the first Evolution wallet users will have the ability to login to the system using a simple username and a password. However, in the current version of the software, we have a simple login interface built into the installation wizard of the electrum client. The addresses will be derived from an 8-24 word phrase that seeds the HD wallet implementation (e.g. Trezor, BIP44).

2.2 Working Repository

If you would like to help us flesh this wallet out, feel free to help. Code is located at:
https://github.com/evan82/electrum-dash/tree/v13-evo-demo

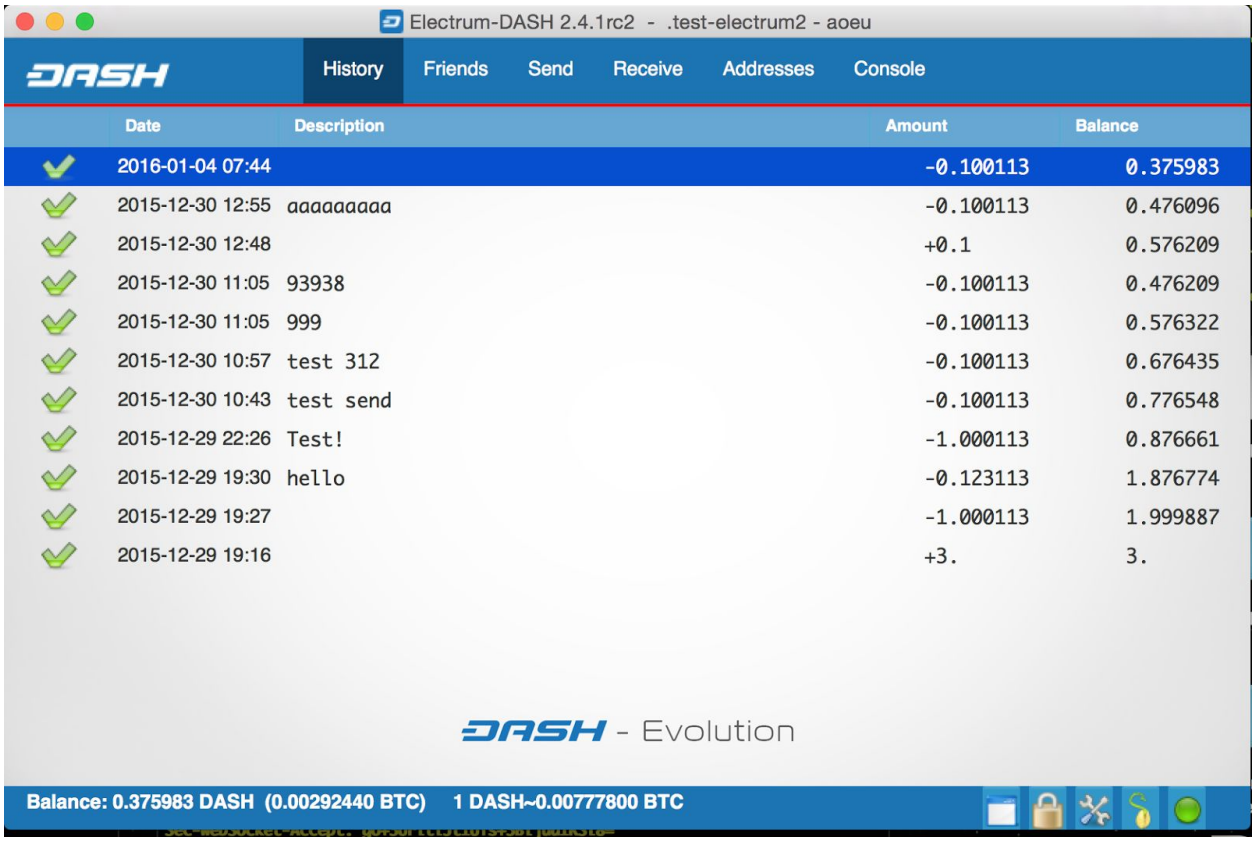
2.3 Public Metadata

UID = Hashed(Username)
FirstName (optional)

LastName (optional)
UserName
Email (optional)
AvatarURL

3. GUI - DashPay (Electrum Based)

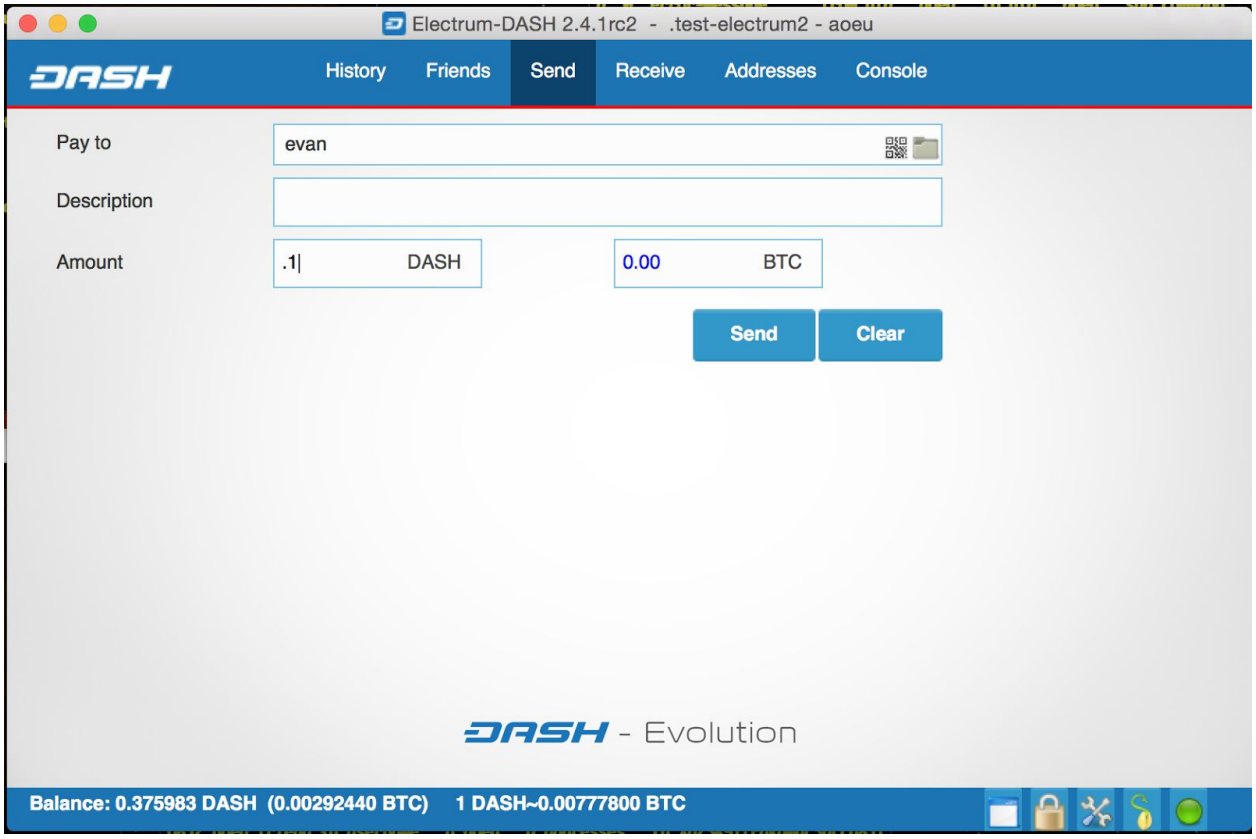
We will stick with an extremely easy and understandable client for the initial prototype and the best client to base our work from is probably the electrum client, which is already supported by the Dash network.



3.2 Pay-To-Contact

The prototype will support paying users by their alias and storing addresses for all of your “friends” in your wallet.

Paying contacts by name:



Tab: Friends

Show the “Friends” of the user. This should be queried from the system using the user’s profile.

Tab: Messaging / Console / Notifications

We want to create a very simple chat interface using the private messaging system for chatting with your friends. Any more advanced functionality (ratings, multisig accounts, etc) will be implemented after the Miami conference.

4 Functionality and API

Simply send requests through the websockets connection to execute commands on the network.

//GET USER PROFILE DATA

```
{
   "object" : "dapi_command",
   "data" : {
```

```
        “command” : ”get-profile”,
        “from_uid” : INT64,
        “to_uid” : INT64,
        “signature” : ‘’,
        “fields” : [“fname”, “lname”]
    }
}
```

//SET USER PROFILE DATA

```
{
    "object" : "dapi_command",
    "data" : {
        "command" : "set-profile",
        "from_uid" : INT256,
        "to_uid" : INT256,
        "signature" = ""
    }
}
```

//SET PRIVATE DATA

```
{
    "object" : "dapi_command",
    "data" : {
        "command" : "set-private-data",
        "from_uid" : INT256,
        "to_uid" : INT256,
        "signature" : "SIGNATURE",
        "slot" : 1,
        "payload" : JSON_WEB_ENCRYPTION
    }
}
```

//GET PRIVATE DATA

```
{
    "object" : "dapi_command",
    "data" : {
        "command" : "get-private-data",
        "from_uid" : UID,
        "to_uid" : UID,
        "signature" : ‘’,
        "slot" : 1
    }
}
```

//SEND MESSAGE

```
{
   "object" : "dapi_command",
   "data" : {
      "command" : "message",
      "subcommand" : "(addr-request,addr,tx-desc,payment-request)",
      "from_uid" : UID,
      "to_uid" : UID,
      "signature" : ‘’,
      "payload" : PAYLOAD
   }
}
```

- PAYLOAD will be encrypted in the future, but not in the v1 prototype
- Lookup target user, see if declared command is accepted and if you’re a friend
- Send if command is accepted or friend

//SEND NEXT-PUBKEY

```
{
   "object" : "dapi_command",
   "data" : {
      "command" : "message",
      "subcommand" : "addr",
      "from_uid" : UID,
      "to_uid" : UID,
      "signature" : "",
      "pubkey" : "BASE58_ENCODED_ADDRESS"
   }
}
```

//BROADCAST ANYTHING

```
{
   "object" : "dapi_command",
   "data" : {
      "command" : "broadcast",
      "subcommand" : "tx", //can support multiple message commands
      "from_uid" : UID,
      "to_uid" : UID,
      "signature" : ‘’,
      "payload" : SERIALIZED_BASE64_ENCODED
   }
}
```

//SEND FRIEND REQUEST

```
{
   "object" : "dapi_command",
   "data" : {
      "command" : “message”,
      "subcommand" : “friend-request”, //Dash network is agnostic to these messages
      "from_uid" : UID,
      "to_uid" : UID,
      "signature" : ‘’,
      "sign_message" : ‘@user1,@user2 friendship’
   }
}
```

- target user will return the signed message, then @user1 can publish the proof to the network which will add the friend

4.1 WebSockets Communication Protocol

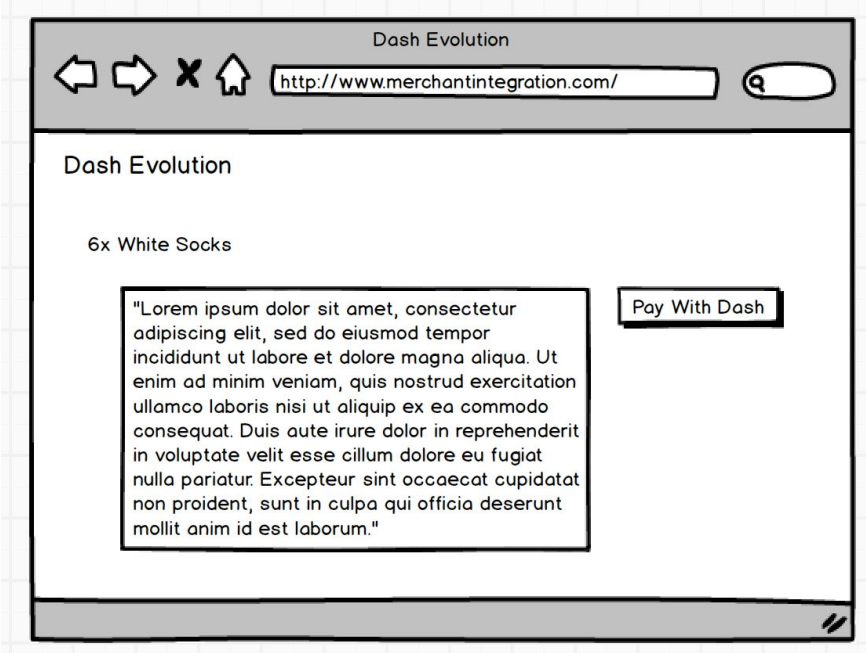
The client will be required to receive information from the network when events happen concerning them, e.g. User1 sends User2 a message. By connecting to port 5000 on dapi.dash.org, a user can maintain an open connection to the network and subscribe for events such as transactions, friend requests and private messages.

4.2 Private Data Encryption

As previously mentioned the prototype is non-secure and uses no encryption. Eventual encrypt of data will be done client side and stored in DashDrive. The prototype should not be used for anything sensitive.

5 Merchant Integration

Merchants will be integrated using the websockets communication protocol and a plugin. Upon visiting a website, the user’s browser will tell the website “My username on the Dash network is ‘Evan’”. The website will open a temporary session with the user over websockets and can send “payment-request” private message.



Merchant Integration Requirements:

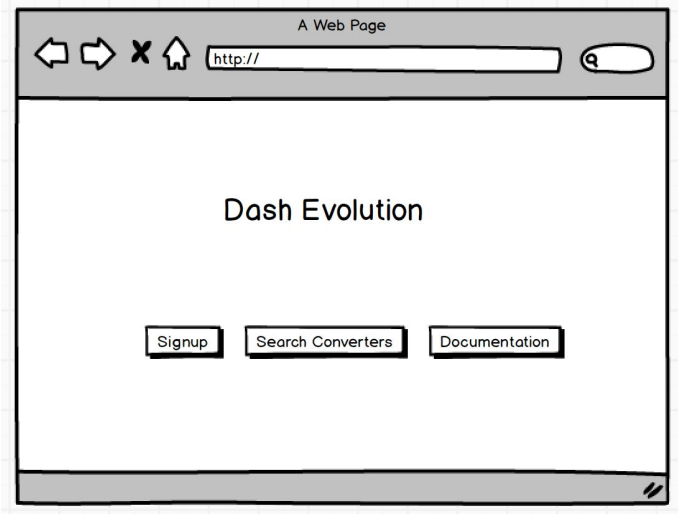
Merchants will simply redirect clients to the second tier, via a URL like this:

https://dapi.dash.org/api/payment-request

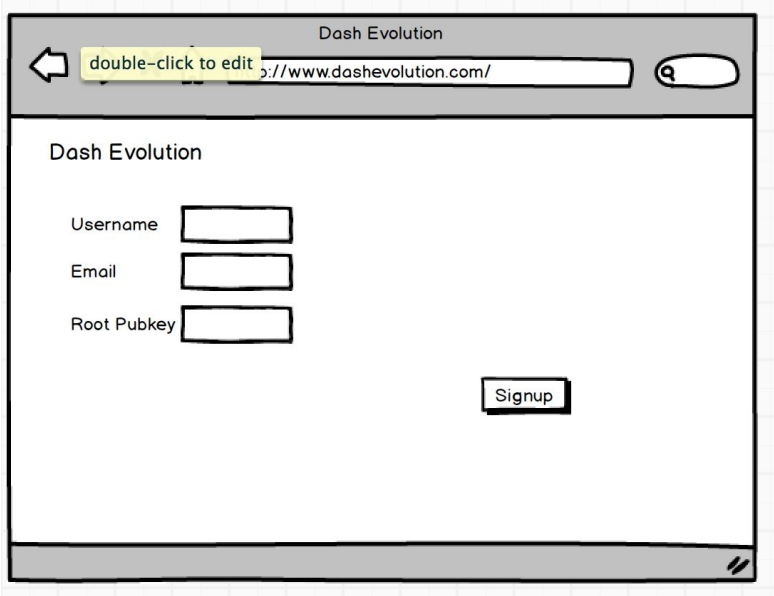
GET PARAMETERS:

```
username_merchant: The merchant`s username on the network
username_client: The client`s username on the network
amount: The cost of the item or service in satoshis
address: The address the merchant would like payment on
description: Description of item, such as “6x White Socks”
callback_url: The URL the network will callback when a status is known

Callback will include

error_id : TBD
error_description : TBD
transaction_hash : The transaction the user sent to pay for the item
in question
```

6 DashEvolution.com



Registration and signup for DashEvolution will be done through DashEvolution.com, which is a website that uses only decentralized commands through websockets. DashEvolution will be given it's own user on the network, which has a large amount of invitations to allow the network to grow and bootstrap itself. It will also send out email to verify accounts. Upon signing up, a user will be sent an email with a link such as "Click Here To Validate Account". This link will include a unique identifier given to DashEvolution by the network. When a user clicks the link the network will use this unique validator in order to prove the user checked their email.



//SEND INVITATION TO USER

```
{
   "object" : "dapi_command",
   "data" : {
      "command" : “invite_user”,
      "from_uid" : UID,
      "to_uid" : ENTERED_USERNAME,
      "to_name" : ENTERED_FULLNAME,
      "to_email" : ENTERED_EMAIL,
      "to_pubkey" : ENTERED_PUBKEY,
      "signature" = “”
   }
}
```

//RESULTING JSON

```
{
   "object" : "dapi_result",
   "data" : {
      "command" : “invite_user”,
      "from_uid" : UID,
      "to_uid" : ENTERED_USERNAME,
      "to_email" : ENTERED_EMAIL,
      "to_pubkey" : ENTERED_PUBKEY,
      "to_challenge_code" : RANDOMLY_GENERATED,
      "signature" : “”
   }
}
```

“Click Here To Validate Account” should contain “target_challenge_code”.

Upon clicking the link, the user will be directed to DashEvolution.com, the site will then construct a message to the network to validate the account.

//VALIDATE ACCOUNT

```
{
   "object" : "dapi_command",
   "data" : {
      "command" : “validate_account”,
      "from_uid" : UID,
      "to_uid" : ENTERED_USERNAME,
      "to_challenge_code" : RANDOMLY_GENERATED,
      "signature" : “”
   }
}
```

//RESULTING JSON

```
{
   "object" : "dapi_result",
   "data" : {
      "command" : “validate_account”,
      "from_uid" : UID,
      "to_uid" : ENTERED_USERNAME,
      "signature" : “”,
      “error_id” : 1000,
      “error_description” : “none”
   }
}
```