

# NEO Dapp - "I made This" - NEO eSignature (<http://esignature.network>)



---

## Online demo


Online demo can be viewed at <http://esignature.network>

Source code - <https://github.com/ANDIT-Solutions/neo-esignature>

---



### Sign

Address 

Choose File

Title (optional)

Description (optional)

Firstname (optional)

Lastname (optional)

Submit

### Verify

Address

Select File

Select Signature

Submit

I MADE THIS

## Idea

**Identity** is big part of NEO Smart economy, and for **digital identity online** electronic document signing is big part of it.

**Blockchain** has been around 10 years now and providing option to store something online that no single entity can remove is ideal solution to prove digital document creation.

To prove that I created document at specific time I don't need to store all file in blockchain all i need is to store file hash. But it's not enough, we need prove time when I created this document, that's why we need to store this document in blockchain. When we store it, it is stored at specific block height at blockchain. Each block has known time when it was created and information.

## Use cases

- Designer came up with new Logo idea, hi then signed this logo image online.
- Engineer created new formula for efficient rocket fuel, he created digital document and signed this document online with his NEO wallet.

- Developer came up with groundbreaking storage algorithm, he signed code library online.
- Payment provider signs his incoming payments online automatically, so he can prove that these are the amounts he have processed and transactions are not tampered with.
- System sign important event logs online, so owner can prove them later that there was no tampering with logs.

## What is eSignature.network?

**eSignature** is tool that provides option for you to attach your online identity to digital documents.

By combining NEO identity with NEO smart contract esignature.network provides tool for fast and easy way to sign and verify digital document online.

NEO identity (Wallet) + Timestamp (smart contract) = eSignature



**Safety** - Safety of blockchain means no one can delete or change this information. Documents are signed only and there is no private information available about documents. Your secrets are safe.



**Mobility** - No papers means its easy to store lot of documents digitally. Digital identity and document availability are global.



**Green** - No papers for document signing.



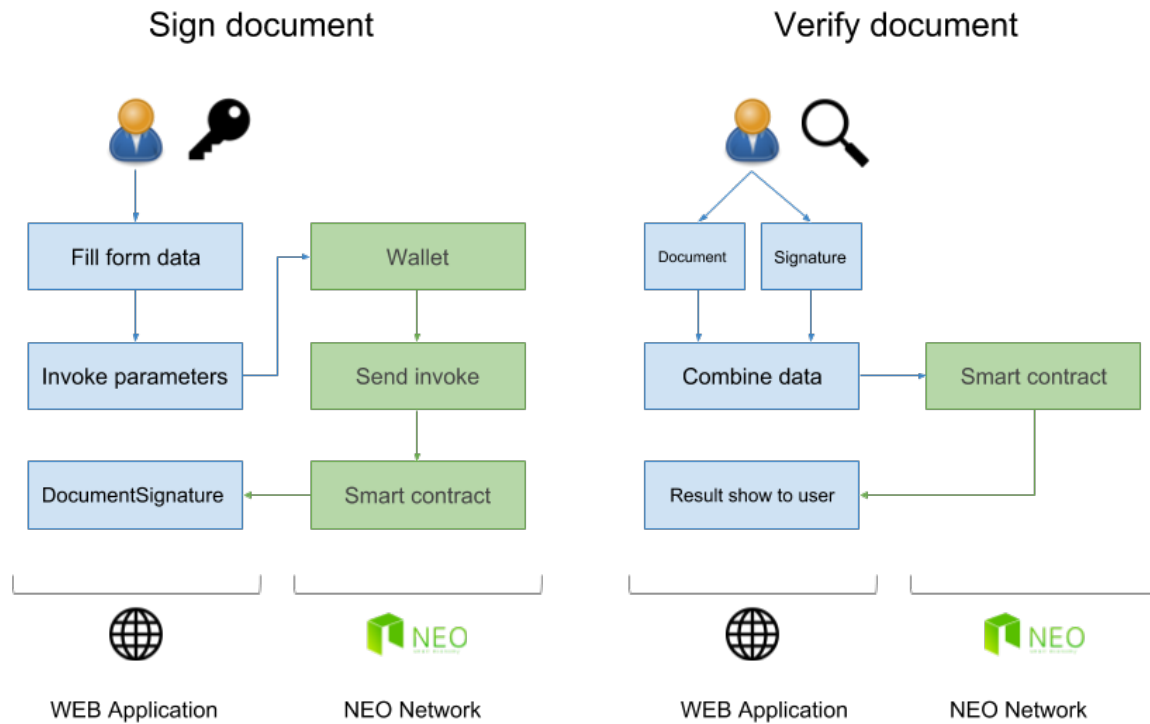
**Ease of usage** - It is easy to use form, signing up, no tokens, upload document, verify and downlaod signature file.



**Affordable and fast** - Its cost only 0.001 gas (\$0.025) to digitally sign document, thats x25 times cheaper than in Latvian (LV) e-signature service.

## Technical process overview

---



## What happens in smart contract?

User sends invocation with parameters from his wallet.

Smart contracts creates hash from these parameters and store it in blockchain together with current block height.

Signing function:

```

1 def Sign(address, document, metadata):
2     #create key id from address and document hash
3     idConcat = concat(address, document)
4     docKey = sha1(idConcat)
5     #create hash from all parameters
6     data = concat(docKey, metadata)
7     hash = sha256(data)
8     #check if there is document with this key already stored
9     if not checkIfKeyExists(docKey):
10        #store document hash
11        StoreKey(docKey, hash)
12        #store document timestamp
13        StoreTimestamp(hash)
14        return hash
15    return False

```

1 Verification functions:

2

```


3 #function SignOnly
4 def SignOnly(address, document, metadata):
5     #create key id from address and document hash
6     idConcat = concat(address,document)
7     docKey = sha1(idConcat)
8     #create hash from all parameters
9     data = concat(docKey,metadata)
10    hash = sha256(data)
11    #return hash for verification
12    return hash
13
14 #operation getdoctimestamp
15 elif action == "getdoctimestamp":
16     address = parameters[0]
17     document = parameters[1]
18     metadata = parameters[2]
19     dochash = SignOnly(address, document, metadata)
20     docKey = sha1(dochash)
21     return GetKey(docKey)

```

## Step 1 - Upload document and fill web form

### Sign

Address

AHE9UGQWgA9hVumTc9DU1avCK1iT5uDqb8 

---

Choose File

Screen Shot 2018-03-10 at 12.14.28.png

Title (optional)

My new Logo

---

Description (optional)

This is logo i created today

---

Firstname (optional)

Eduards

---

Lastname (optional)

Martesteris

---

Submit

## Step 2 - Invoke smart contract with parameters generated in application



## Sign

Address  
AK2nJJ

Choose  
12.14.28

Title (optional)  
asdff

Description  
asdfadf

Firstname  
asdfaf

Lastname  
asdff



Please invoke contract address: **3196c8523664004204c506cb27ccf9022bc4878c**  
With parameters:

1. **sign**

2.

[

**23ba2703c53263e8d6e522dc32203339dcd8eee9**

**a6c04a5b89c4f98faef00695e214dd3a2657195e68be735ca10feabc2d25ea33**

**c7c47c647108a0fae9b35deae9da367fef915c2c031553599bc0bf21317919e9**

]

The contract must be invoked within 10 minutes Time left: 09:41

Submit

I MADE THIS

**Step 3 - Wait while application detects your invoke and download document signature file**



## Sign

## Verify

Add

Address

12

12

Title

My

Des

Th

Firs

Ed

Lastname (optional)

Martesteris

Submit



Succesfully uploaded file to blockchain

[This is your signature file:](#)

[ee354b4f5177660b0935484bff1a148e7c1acb4606f988da5595a3b13136a4ea](#)

**Step 4 - To verify document uplaod original file and signature to verify form.**

## Sign

Address

Choose File

Title (optional)

Description (optional)

Firstname (optional)

Lastname (optional)

Submit

## Verify

Address

AJTfddKqQhr2FRjR9mQ7kJ1FHH7EYVUXf

Select File random.txt

Select Signature signature (1).json

Submit

Verification Successful

Signature Parameters

```
key:
3901a378ba0512985ac10f142dfdaf9b681fbfa4
addressScriptHash:
1d76e3a6d1b5698a9395a025245b5dc2afa8fe61
contractScriptHash:
3196c8523664004204c506cb27ccf9022bc4878c
signatureHash:
d0a4338fae2c43e93a93e3d4ded0f94c4f156d09
7101b1666946f4bf91220920
fileHash:
```

**Step 5 - If document was not the same as original file, signature verification will fail.**

## Sign

Address

Choose File

Title (optional)

Description (optional)

Firstname (optional)

Lastname (optional)

Submit

## Verify

Address

AJTfddKqQhr2FRjR9mQ7kJ1FHH7EYVUXf

Select File random.txt

Select Signature signature.json

Submit

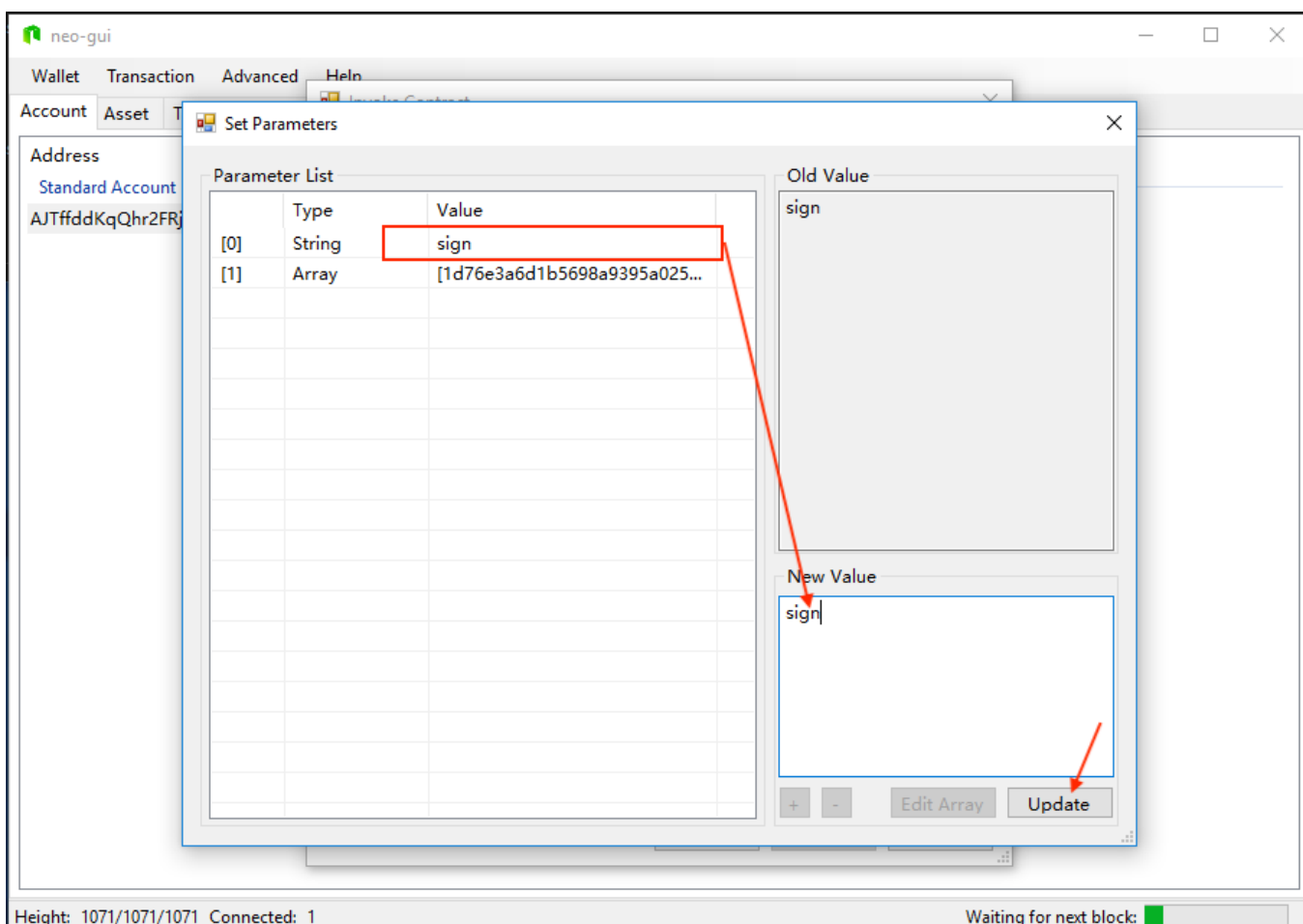
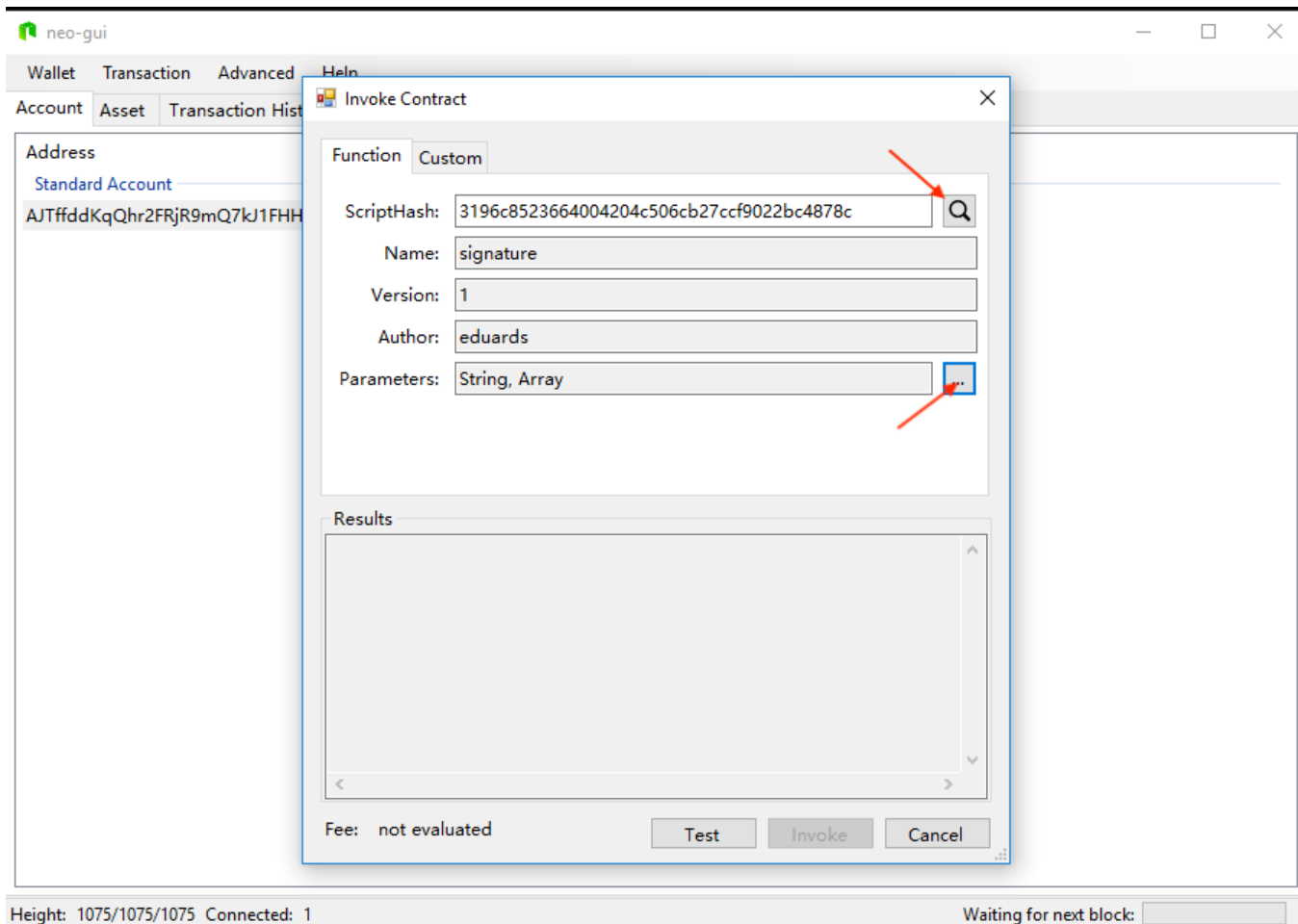
Verification Failed

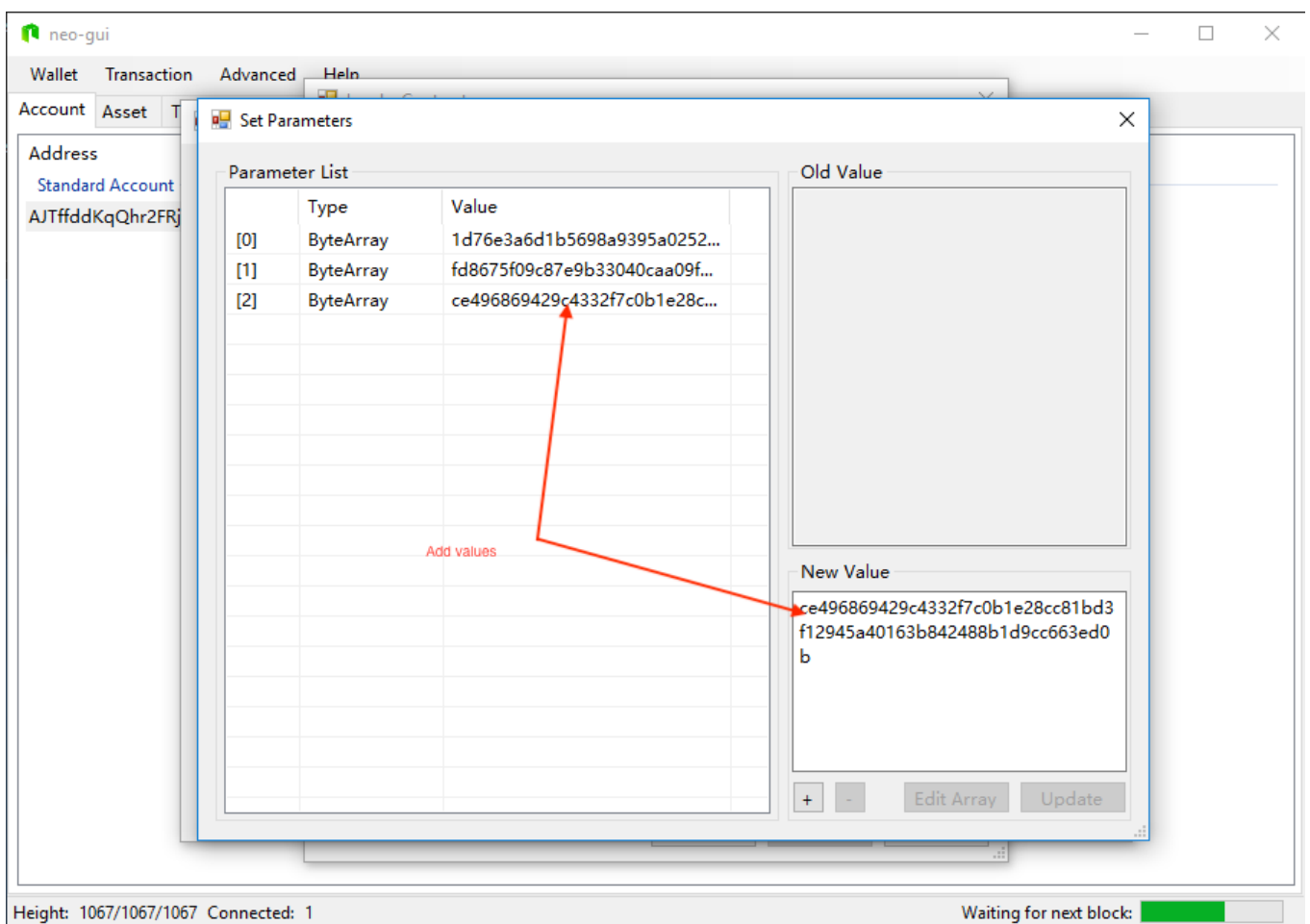
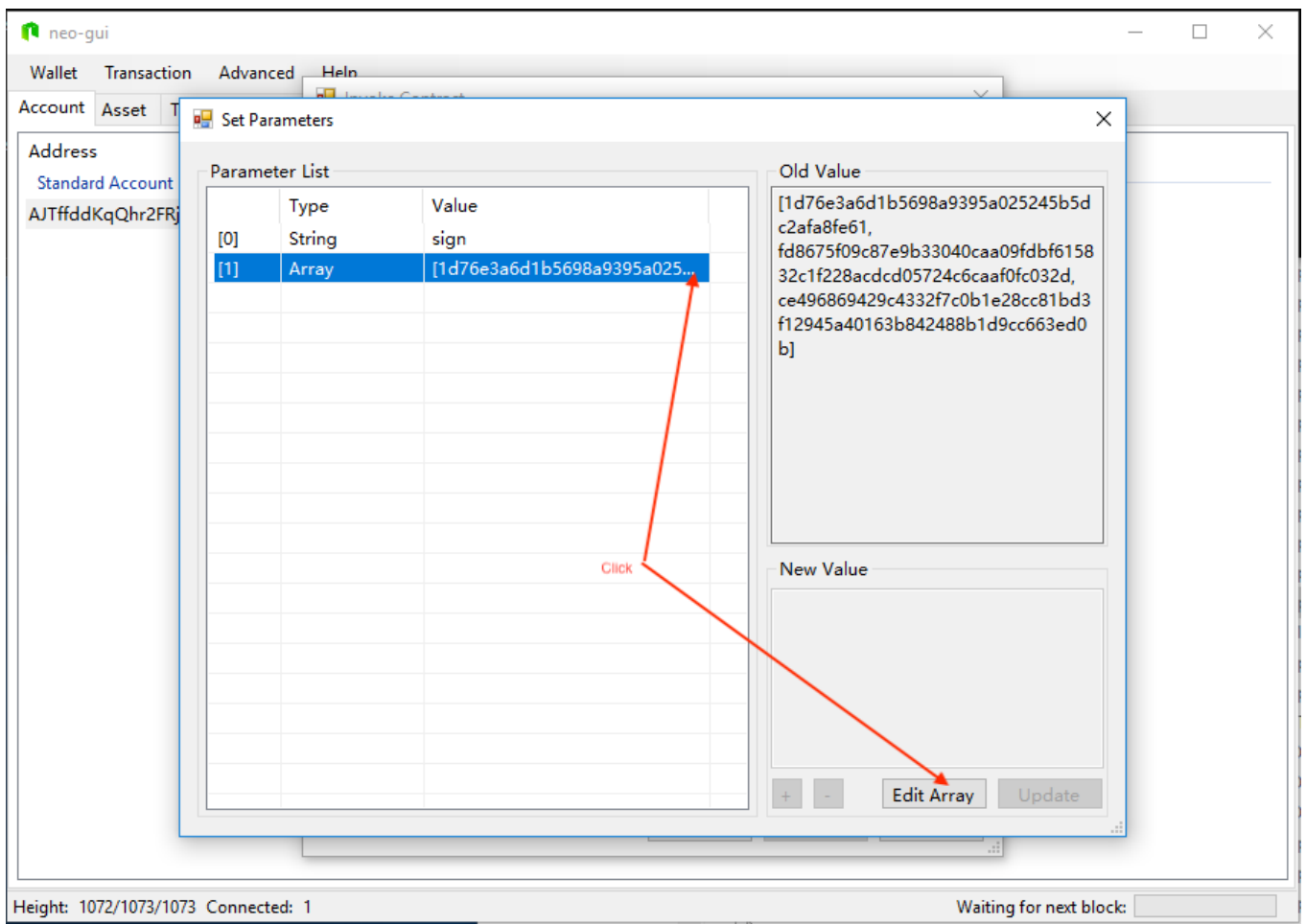
Signature Parameters

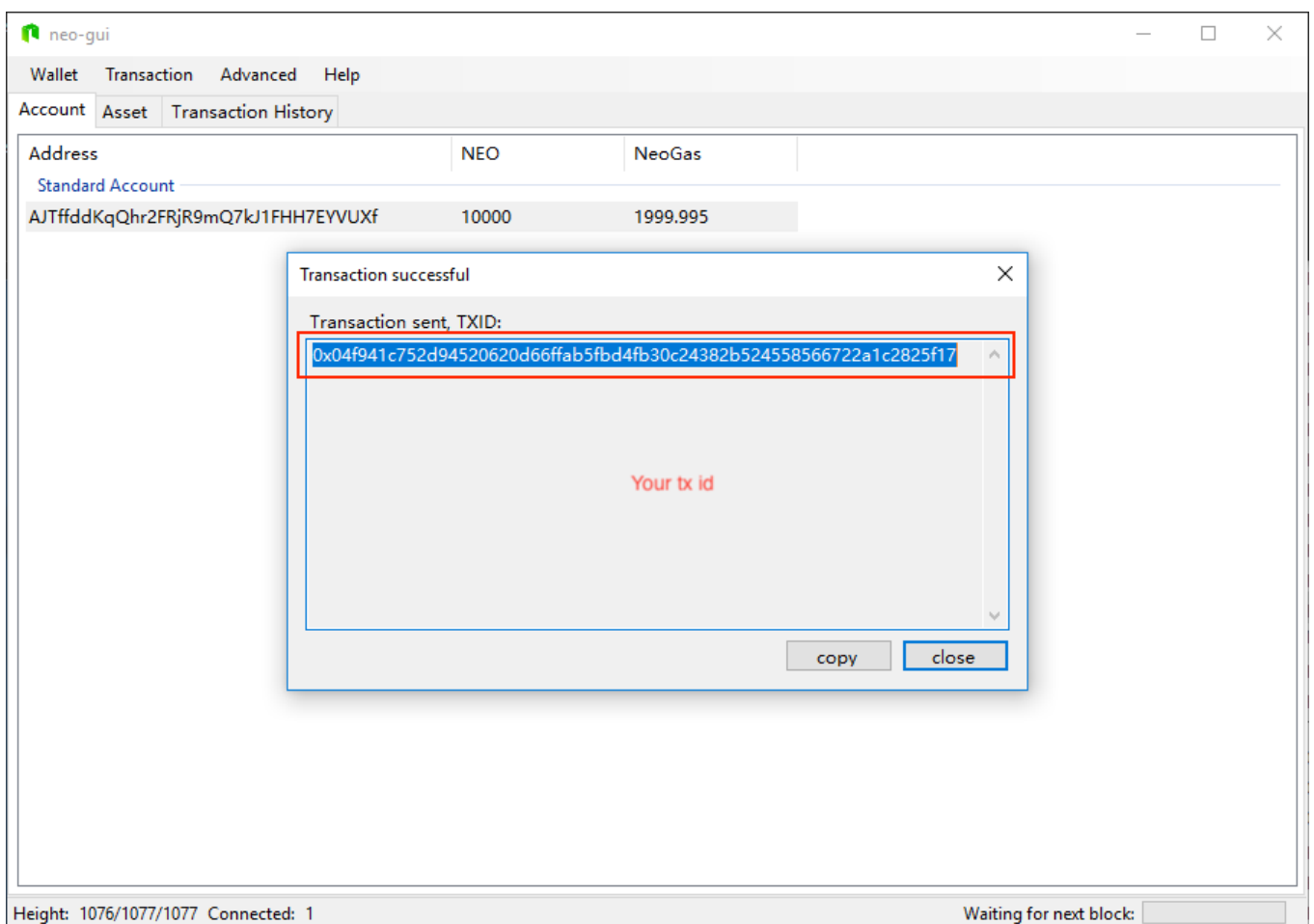
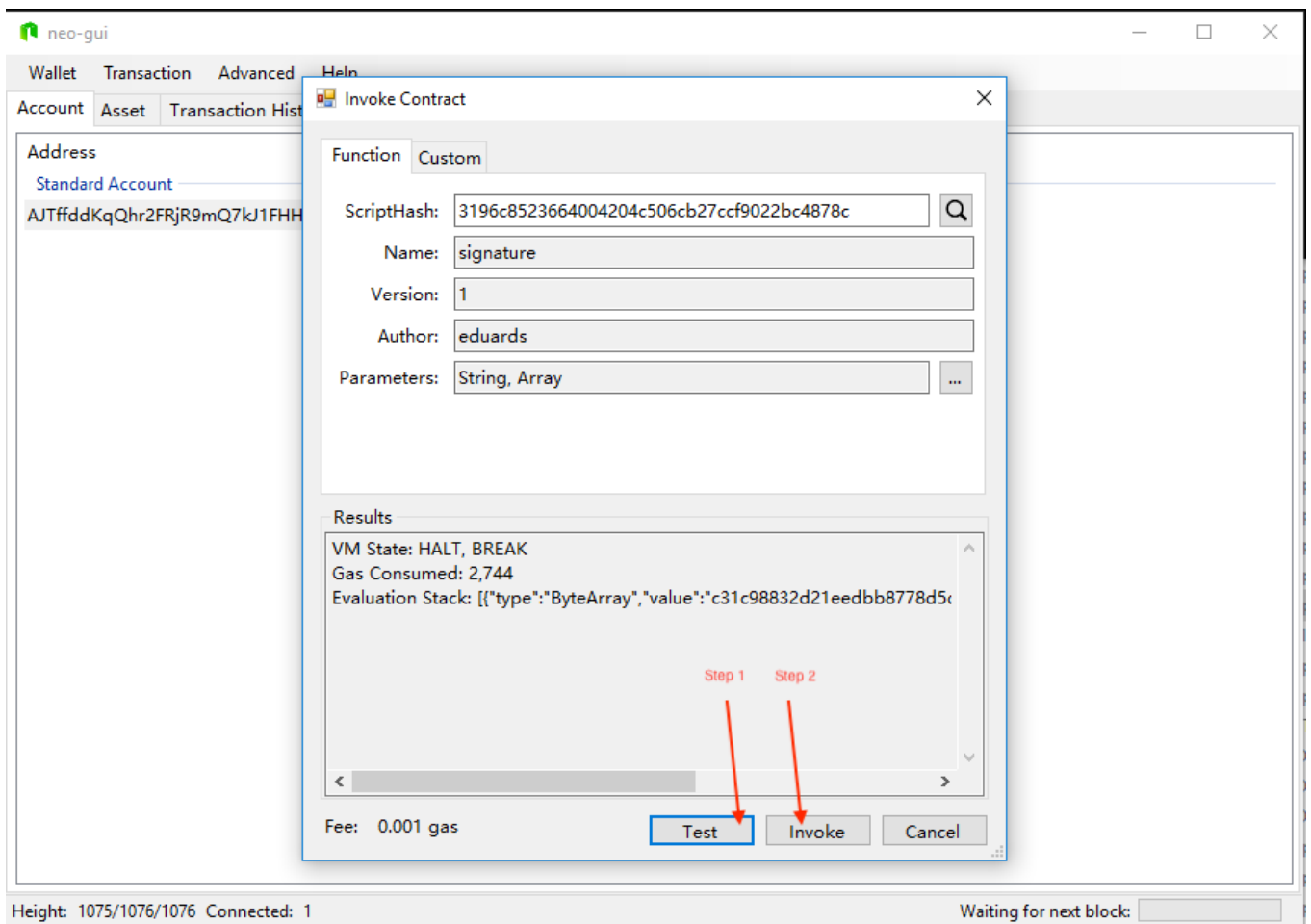
```
1c6fbd614d204591c67fd3aeb8671f35249b2b2fa
51e888b71fedfc15dbb7913
docHash:
b24bbf533824140306f4bf7601176cfec5680f03d1
cae79d7c05893c48670b46
metaData: xxx
metaData: xxx
metaData: xxx
metaData: xxx
timestamp: 1712
```

**How to invoke smart contract from NEO-GUI**









## Requesting document timestamp proof through NEO rpc-api

This is how other persons can verify document not using our created web tool, this is important for decentralised applications.

**getdoctimestamp** - function that queries blockchain and get requested document timestamp, if any of parameters would be different output would be empty.

**value = 1404** - this is block height this document was signed.

Request:

```
1 {
2   "jsonrpc": "2.0",
3   "method": "invoke",
4   "params": [
5     "3196c8523664004204c506cb27ccf9022bc4878c",
6     [
7       {
8         "type": "String",
9         "value": "getdoctimestamp"
10      },
11      {
12        "type": "Array",
13        "value": [
14          {
15            "type": "ByteArray",
16            "value": "0ff0264fe001bc65624aa5333dbcf903b60856b8"
17          },
18          {
19            "type": "ByteArray",
20            "value": "e6604b6cfc82e437aa43efd0a61797d3fde0d239cd8fa746b6e04435f77848b3"
21          },
22          {
23            "type": "ByteArray",
24            "value": "7e3975e060bc507e1015ae0eae647497e7290a8c4d7ed0ff9b8f7c45794839ea"
25          }
26        ]
27      }
28    ]
29  ],
30  "id": 1
31 }
```

Response:

```
1 {
2   "jsonrpc": "2.0",
3   "id": 1,
4   "result": {
5     "script":
6       "207e3975e060bc507e1015ae0eae647497e7290a8c4d7ed0ff9b8f7c45794839ea20e6604b6cfc82e437aa43efd0a61797d3fde0d239cd8fa746b6e04435f77848b3140ff0264fe001bc65624aa5333dbcf903b60856b853c10f676574646f6374696d657374616d70678c87c42b02f9cc27cb06c5044200643652c89631",
7     "state": "HALT, BREAK",
8     "gas_consumed": "0.405",
9   }
10 }
```

```

8     "stack": [
9         {
10            "type": "ByteArray",
11            "value": "1404"
12        }
13    ],
14    "tx":
    "d1017e207e3975e060bc507e1015ae0eae647497e7290a8c4d7ed0ff9b8f7c45794839ea20e6604b6cfc82e437a
    a43efd0a61797d3fde0d239cd8fa746b6e04435f77848b3140ff0264fe001bc65624aa5333dbcf903b60856b853c
    10f676574646f6374696d657374616d70678c87c42b02f9cc27cb06c5044200643652c896310000000000000000
    00000000"
15 }
16 }

```

## Signature file format

```

1 {
2     "key": "f489a6e8e0c680127f4abc3c4f19ffc89493239e",
3     "addressScriptHash": "1d76e3a6d1b5698a9395a025245b5dc2afa8fe61",
4     "contractScriptHash": "3196c8523664004204c506cb27ccf9022bc4878c",
5     "signatureHash": "62ac56265247453a74ece9e4a6ac9f98e46a301614529ab9797a7b6072d788bb",
6     "fileHash": "f65412cb711a6e91a597da118cceb552081aa7d1f0c9091c5e760fa94c9ed6fd",
7     "metaDataHash": "adee37b20de2e31608b35e084aa7daa40df938de32ef142324cf9b815a4b6af6",
8     "docHash": "90b06ed07bf18204561621e35064f08e3079718e5c3621daa38874995c2bbfa2",
9     "metaData": {
10         "title": "Title",
11         "description": "this is my description",
12         "firstname": "Johnny",
13         "lastname": "Bravo"
14     },
15     "timestamp": 1456
16 }

```

## Technology used for this project

- Angular.js
- Node.js
- Neon-js
- Python Boa
- Neo-python

## Future vision

Since NEO Auth and NEO identity still is under heavy development there is lot of room for improvements.

1. eSignature project should be integrated with NeaAuth for correct address identification.
2. We would suggest for future to create NEO link standart for browser plugins where you can send parameters to wallet, and you can invoke contracts very easy.

Example:

```

neoinvoke:3196c8523664004204c506cb27ccf9022bc4878c?
param1=functionname&param2=hexstring&gas=0.002&neo=1

```

3. Create offline web only version of signature.network where you can safely add your private key to sign transactions
4. Provide option for users to store their source file and signature in cloud.
5. NeoAuth + NEO eSignature very close integration
6. eSignature integration in wallets.
7. Sign only option, that is not using NEO key/value storage for scalability
8. API endpoint for machine to machine data signing (Identity is not only for humans :) )
9. Go beyond single blockchain helping NeoX verify entities.

---

I MADE THIS

By ANDIT Developer Team