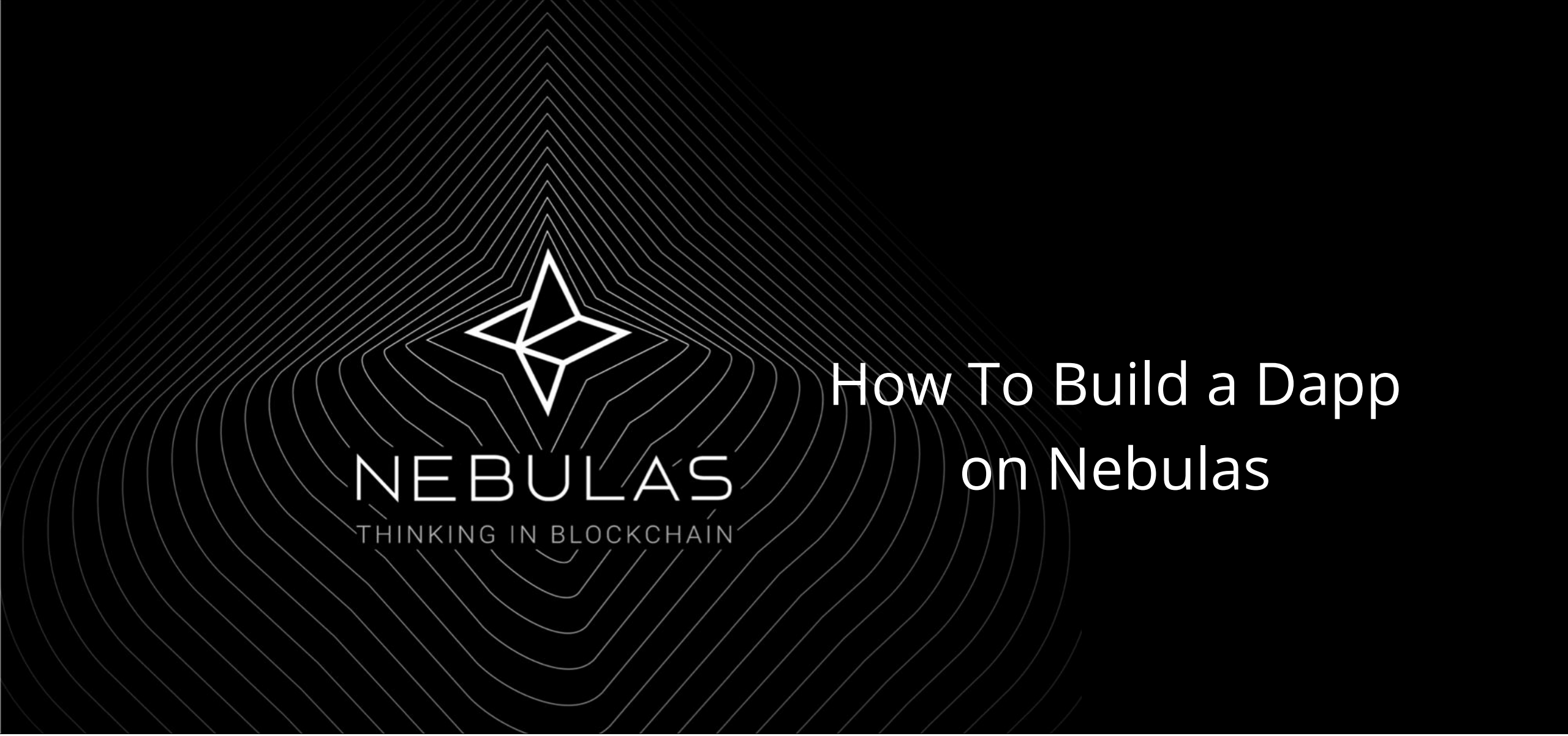


How To Build a Dapp on Nebulas

Nebulas.io

Nebulas
Incentive Program

Mainnet Deployment overview on June 6th, 2018

$5,000,000

460,000 NAS Coins

DApps on Nebulas：**3,563**

(Nearly twice as many as Ethereum)

Registered Nebulas user accounts：**52,304**

9.40% daily increase.

Nebulas smart contracts：**6,623**

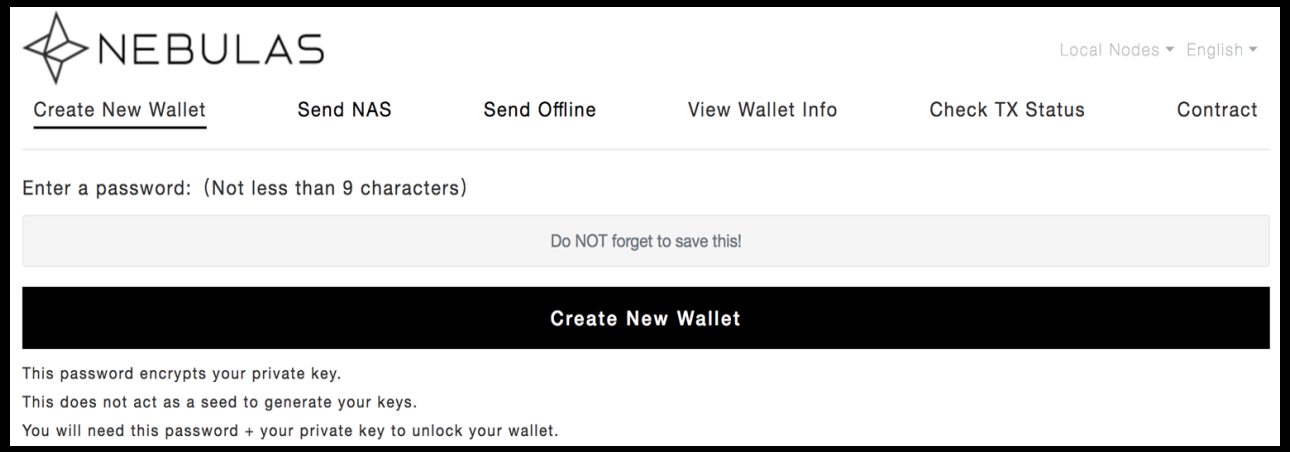
201 more than the day before.

Value based on NAS price during the launch of the Incentive Program

01

Web-Wallet Interaction

Create Wallet



The official Nebulas Web Wallet makes it easy to create your wallet, send transactions and deploy and interact with Smart Contracts.

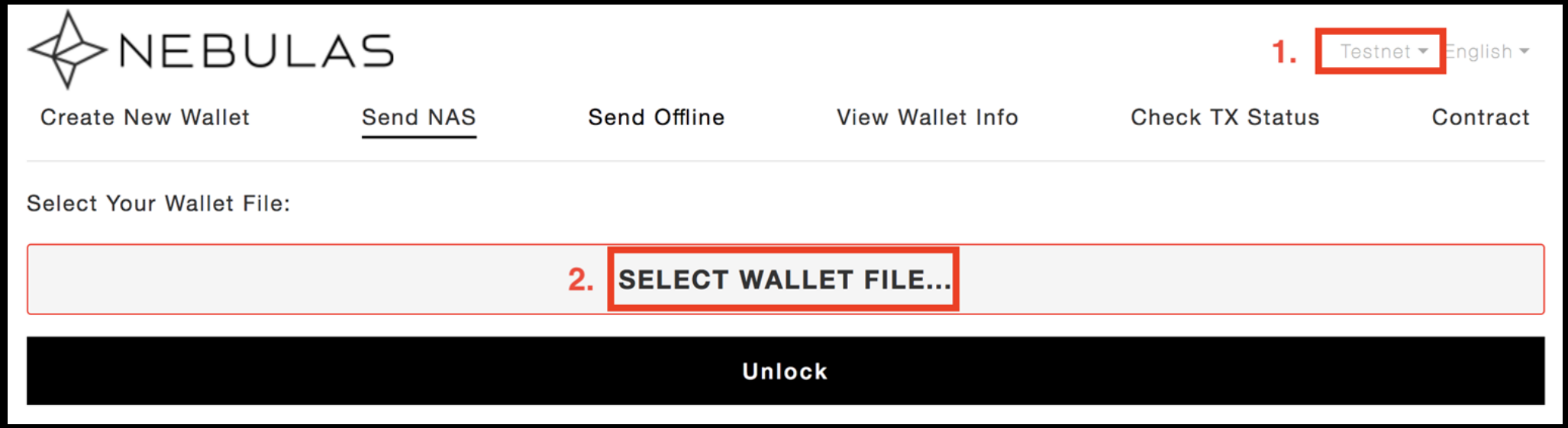
The creation of a wallet will generate a keystore file which contains your address, encrypted private key and additional details about your wallet generation.

Wallets *can* be used for localnet, mainnet and testnet.


```
{
  "version":4,
  "id":"ed0d201b-f421-4e49-a03f-e759e50048fb",
  "address":"n1Ni6YPvAj7YXmChKwA3kHWzNsHEsp1F4ta",
  "crypto":{
    "ciphertext":"e2083cbe5ba03f7b6a3f3ce……0135603287cb85",
    "cipherparams":{
      "iv":"ff7e9d417c8e0edc3aaa3dd8961c20c1"
    },
    "cipher":"aes-128-ctr",
    "kdf":"scrypt",
    "kdfparams":{
      "dklen":32,
      "salt":"a5a601df63f04f3c05cb584d146…….8aa394559e8802",
      "n":4096,
      "r":8,
      "p":1
    },
    "mac":"bdb04873edd4463f80231d61d79c ……. 73a791861e",
    "machash":"sha3256"
  }
}
```

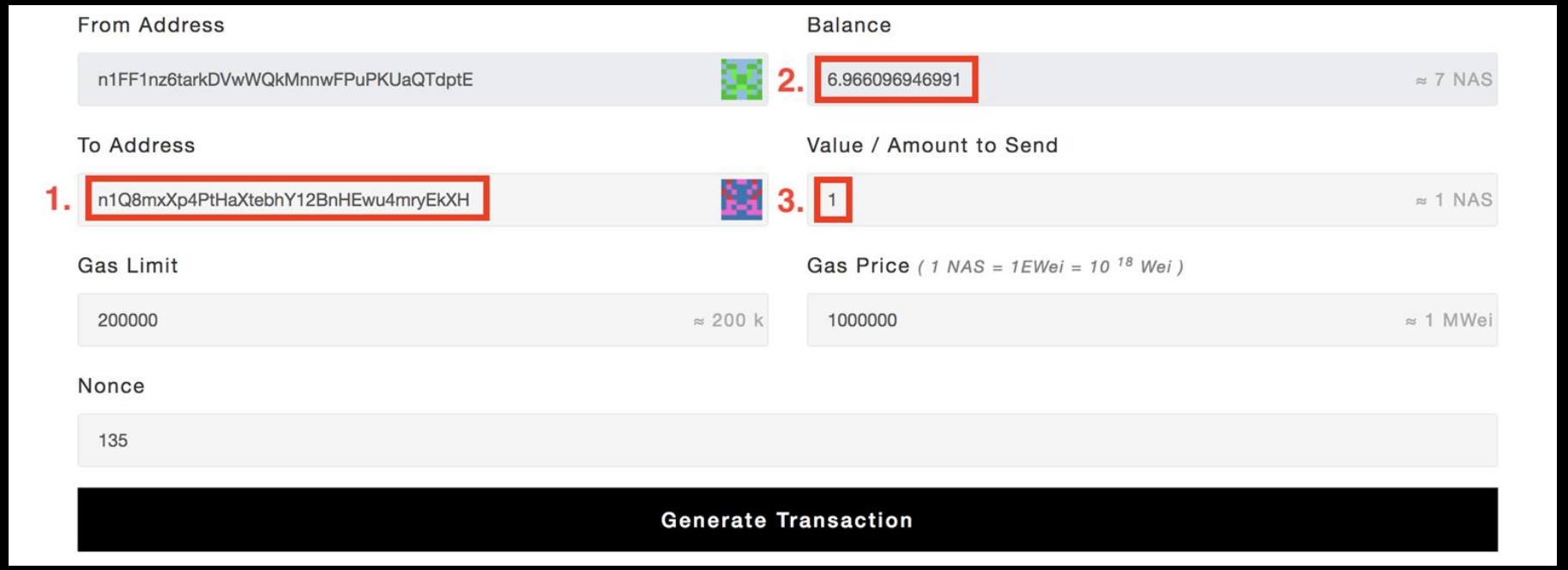

The above is a sample keystore file

Unlock Wallet



1. Network Types： Mainnet, Testnet or LocalNet
2. Select wallet keystore file and enter password to unlock

Signing Transactions



1. After unlocking, enter the To Address.
2. Once unlocked, your Balance will be shown.
3. Enter the Amount of funds you want to send.

Nebulas coins (NAS) are divisible by up to 18 decimal points (zeros). This mean the minimum amount of NAS you can send is: 0.000000000000000001

Gas Price and Gas Limit:

Gas is how transactions are paid. Depending on the transaction, the gas fee varies based on size of TX and interaction/deployment of contract.

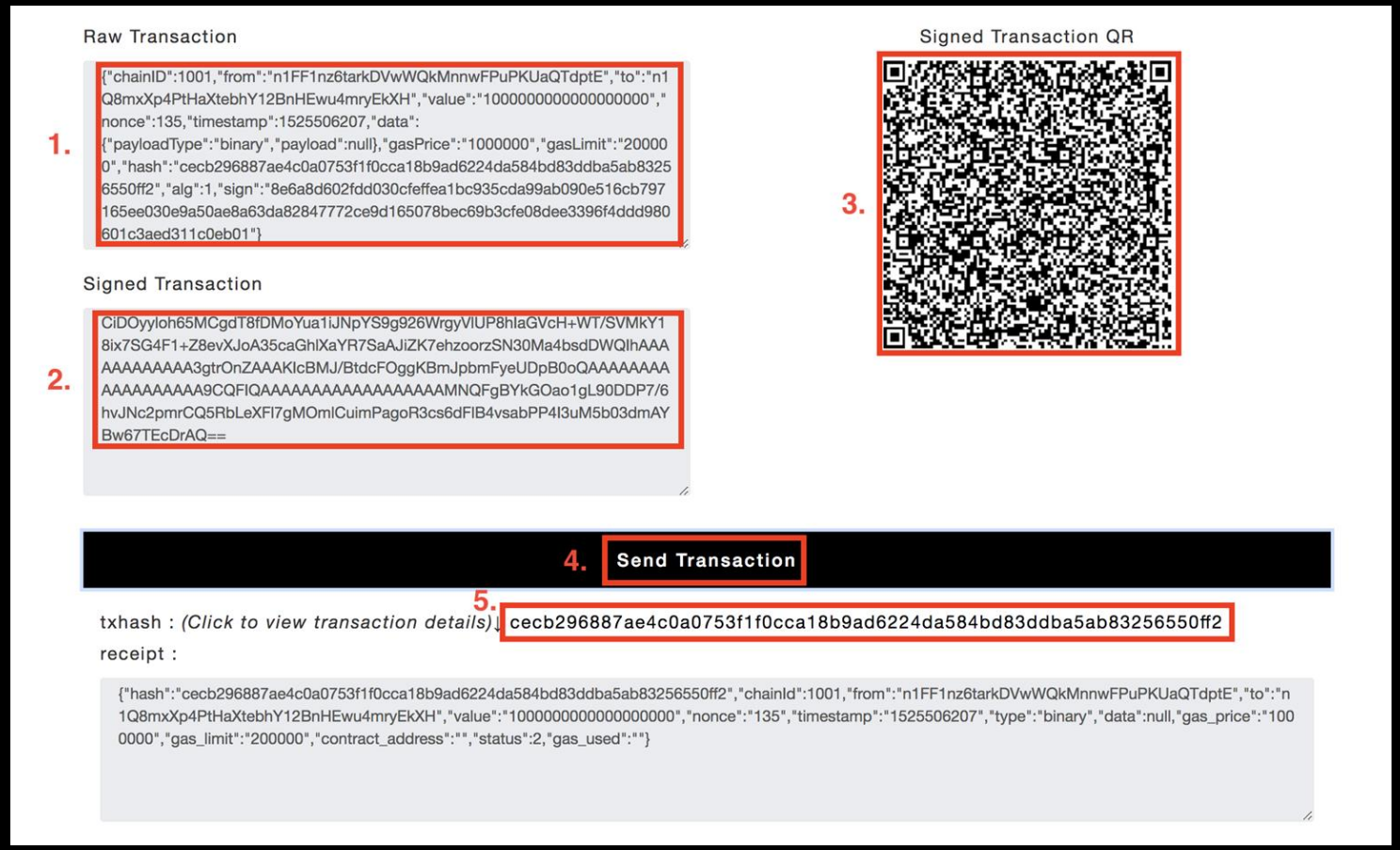
Nonce Height:

Nonce is how many transactions have already occurred on this address. The nonce height must be entered correctly otherwise your transaction will not be accepted into the blockchain. The Nonce will be automatically updated in the web wallet.

Generate Transaction:

Once you enter the" To Address", the value you want to send and the gas fees (usually default), press the Generate Transaction button.

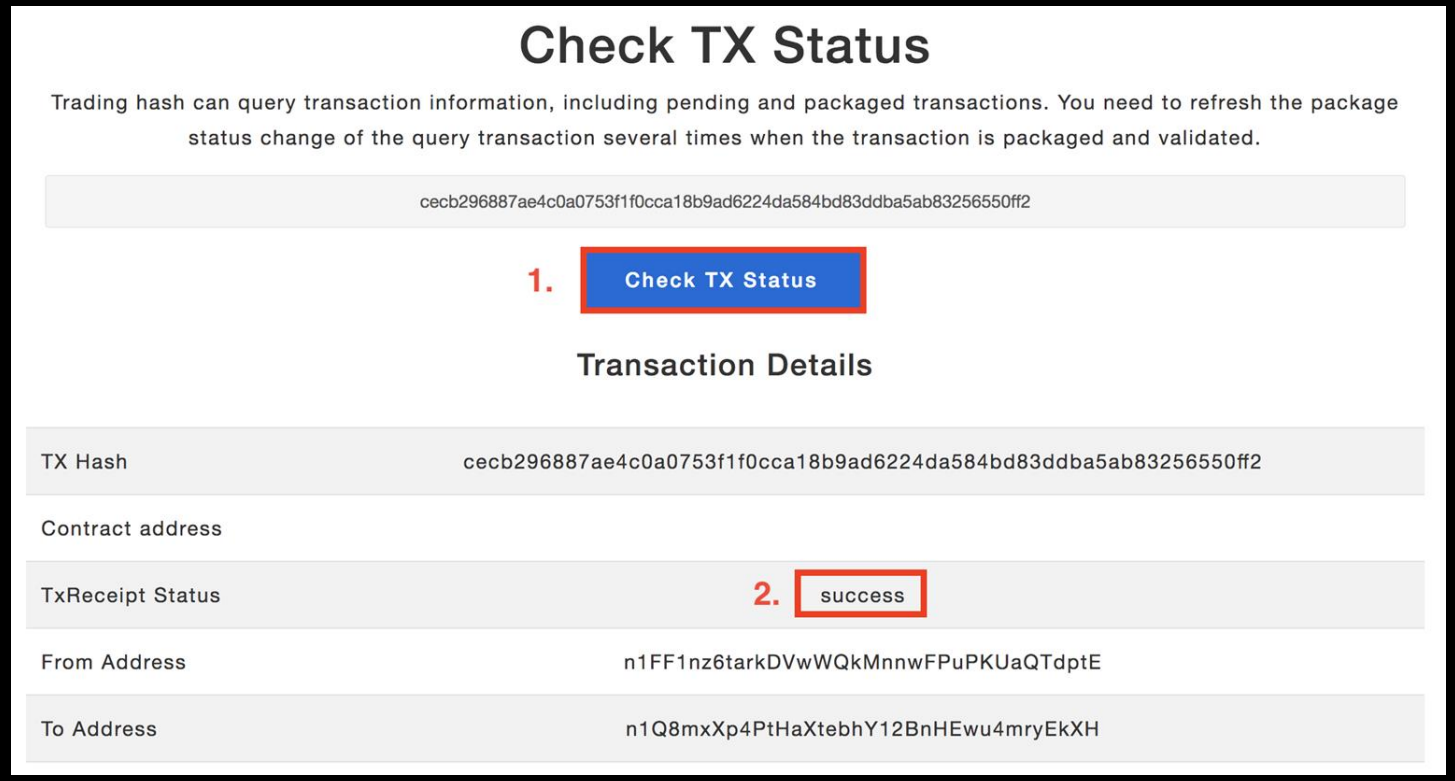
Submitting Transactions



1. Raw Transaction Information
2. Signed TX Data （base64 encoded)
3. Generated QR Code
4. Once you click Send Transaction, your transaction will be sent to a node. Transactions usually complete within 15 seconds after submission.
5. Once your Transaction is submitted, you will receive a Transaction Hash (txhash) which you can use to verify your transaction. This acts as your receipt.

- After you receive your txhash, you can click on it to review the status of the transaction and to verify if your transaction has been successfully included into the blockchain.
- Nebulas mints blocks every 15 seconds.
- All data stored on the chain is base64 encoded.

Reviewing Transaction



1. To review the status of the transaction, Click "Check TX Status".
2. Transaction status will usually become successful within 15 seconds after submission.

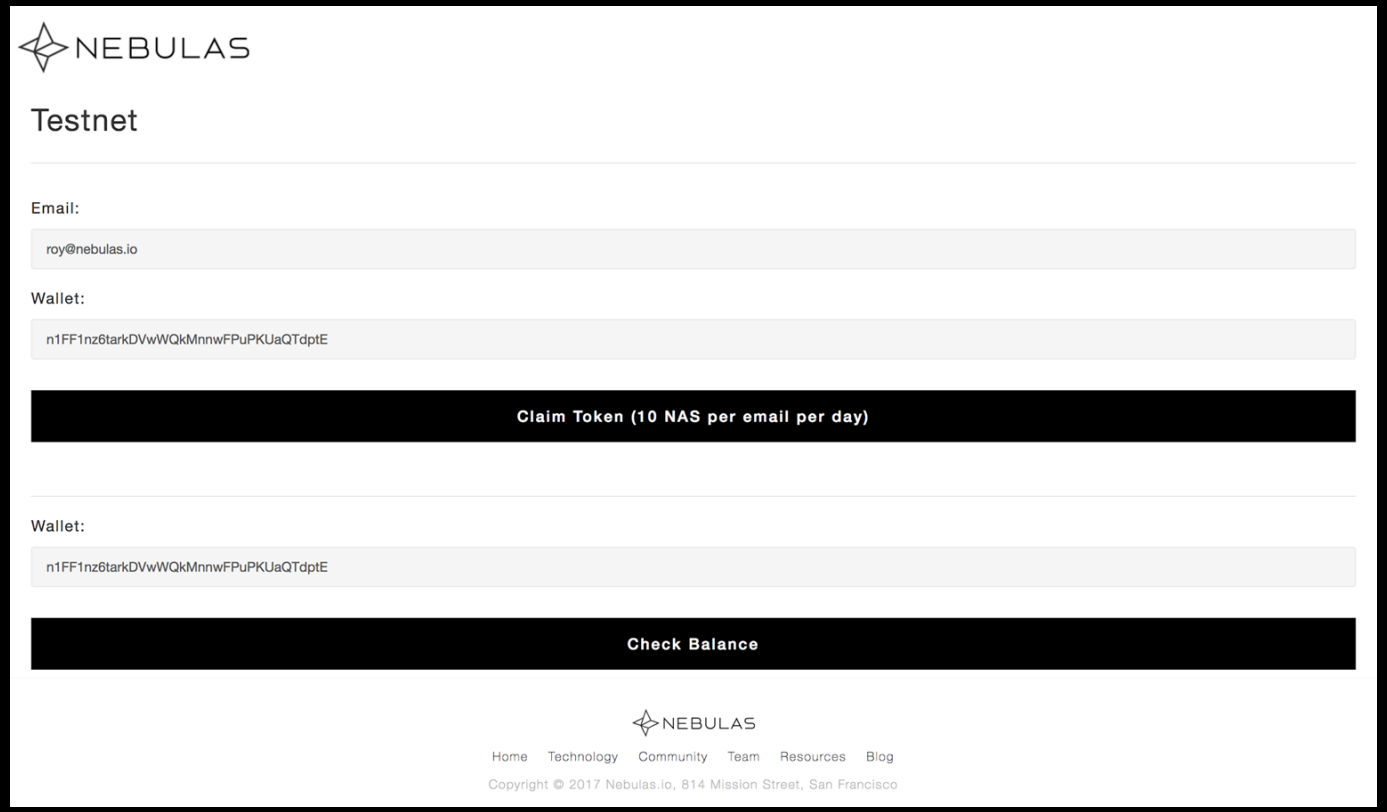
If your transaction remains as pending, verify you have enough funds to cover the amount being sent plus the gas fee.

Transaction status can be one of the following:

- Pending – Your transaction is awaiting inclusion into a minted block.
- Success – Your transactions has been included into a minted block.
- Failed – Your transaction failed. This may be due to insufficient gas fee or improper data submission.

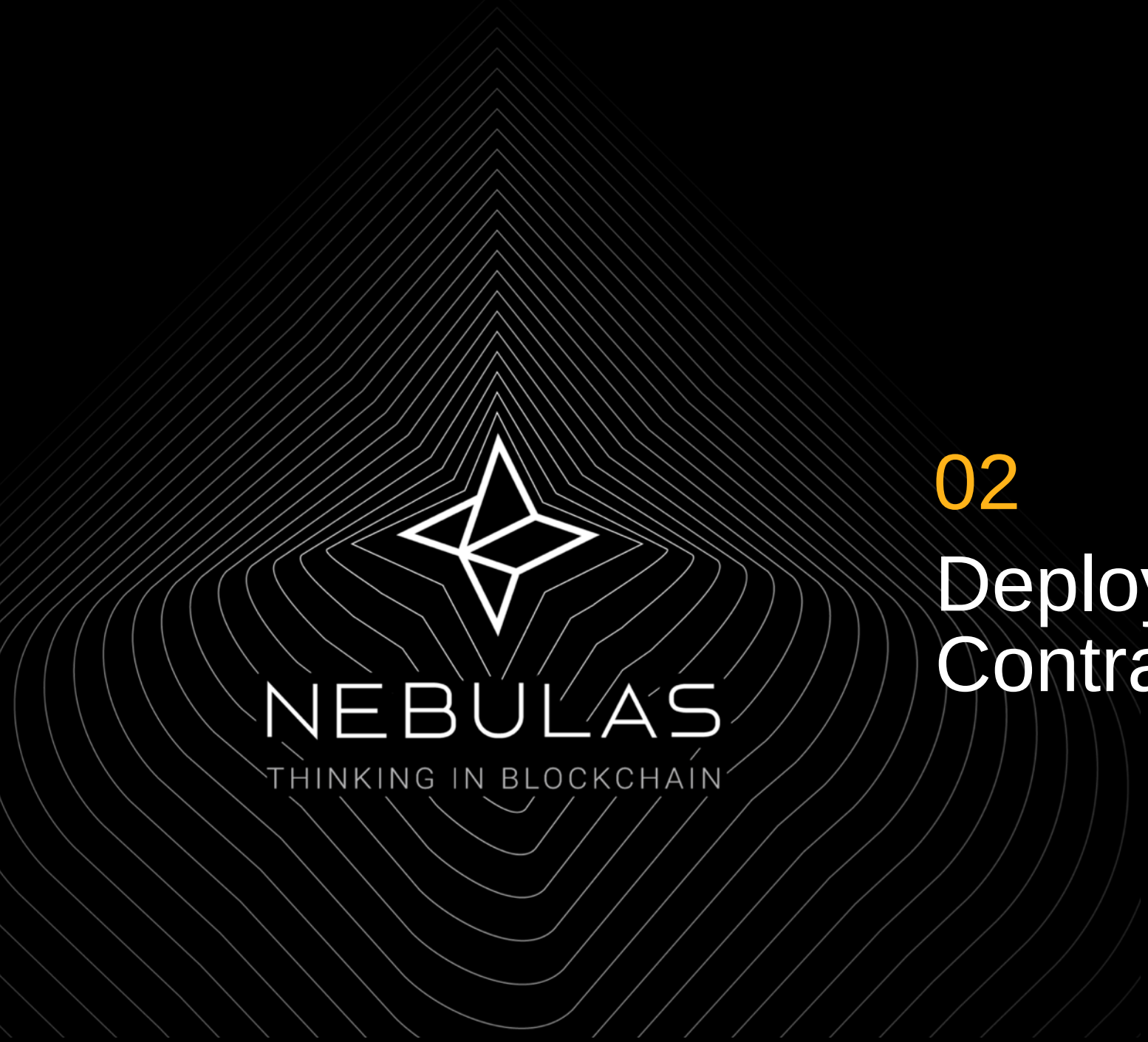
Testnet Faucet

To claim testnet tokens, visit:
https://testnet.nebulas.io/claim/



The purpose of testnet tokens is to test functionality on the test version of Nebulas.

Testnet tokens are good only for the testnet and hold no value on the mainnet.



02

Deploying Smart Contracts

NEBULAS

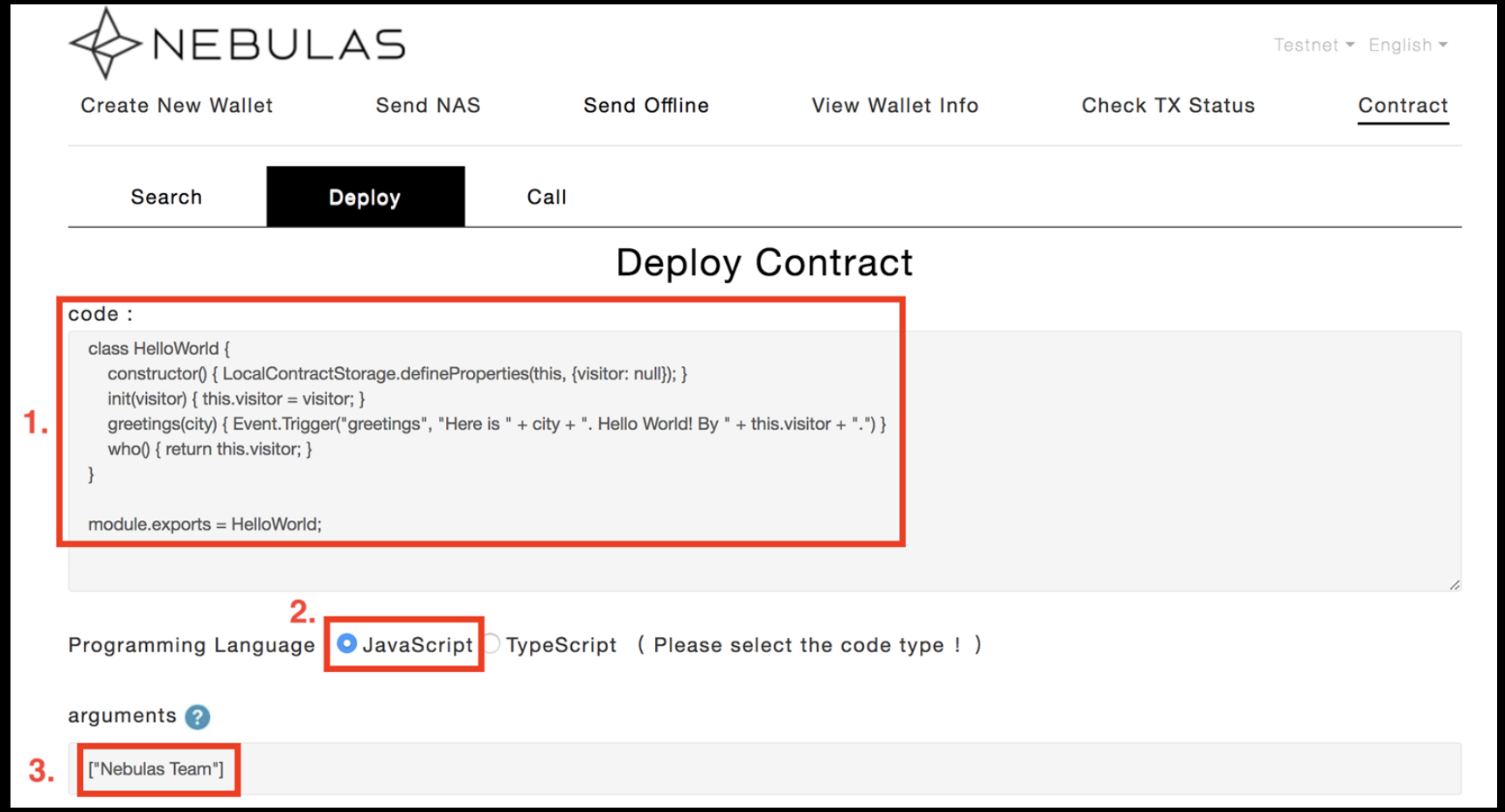
Hello World – Contract Breakdown

This and more are available for you to review at https://github.com/Nebulas-Learning/

```
1  //Nebulas Official Hello Worlds Script
2  class HelloWorld {//Define the class object of our script
3      constructor() {// The constructor method is a special method for creating and initializing an object created within a class.
4          LocalContractStorage.defineProperties(this, {//LocalContractStorage is a built in function of Nebulas
5              visitor: null
6          });
7           /*
8           A list of built infunctions are available at: https://github.com/nebulasio/wiki/blob/master/smart_contract.md
9           */
10     }
11     init(visitor) {//All Nebulas smart contracts must contain the init class. Any Arguments entered during deployment will be attached to the init function and the variables (e.g. visitor).
12         this.visitor = visitor;//Defining visitor variable as this.visitor
13     }
14     greetings(city) {//creating the greetings class/function that can be called by the user.
15         Event.Trigger("greetings", "Here is " + city + ". Hello World! By " + this.visitor + ".")//This is a string that we will be printing (hello world).
16     }
17     who() {//creating the who class/function that can be called by the user.
18         return this.visitor;//This will return the name of the visitor which is defined during deployment.
19     }
20 }
21 module.exports = HelloWorld;//All contracts must use module.exports. It must be the class/protocol object defined in the script. In our example, it is "HelloWorld"
```

NEBULAS

Hello World – Writing a Contract

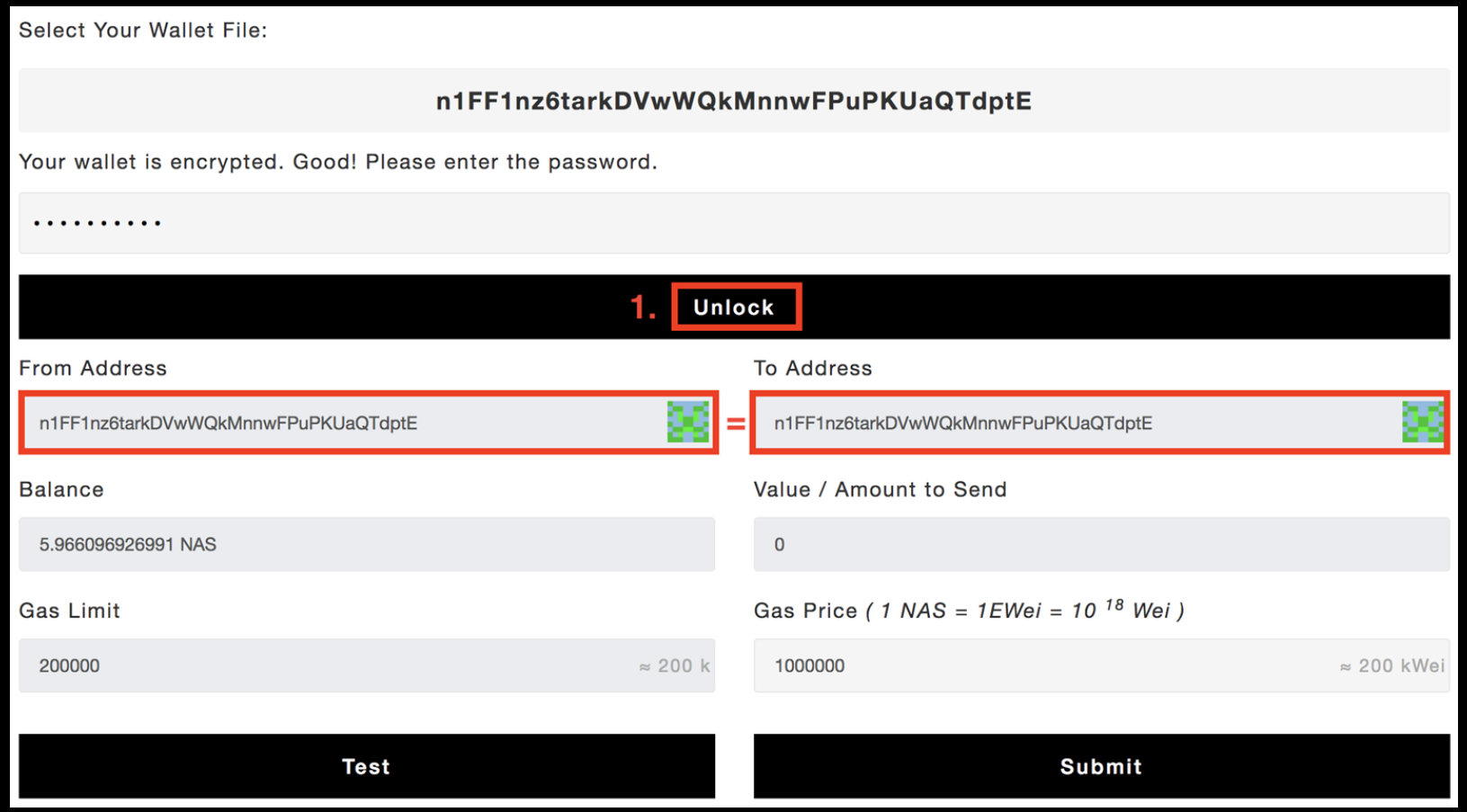


1. Sample Smart Contract
2. Select Contract Type
3. Enter Any Arguments

- Contracts must be either a class or a prototype object.
- All smart contracts must have the init function.
- The “module.exports” function must be called at the end of the script and must contain the class/object name.
- Nebulas currently supports Smart Contracts written in Javascript and Typescript.
- Arguments must be entered in JSON array format.

NEBULAS

Hello World – Unlocking Your Wallet



- When deploying a smart contract, the "To Address" must be the same as the From Address. Keep "Value/Amount To Send" at 0.
- Depending on the size of your smart contract and the current gas cost, you may need to increase your gas limit.

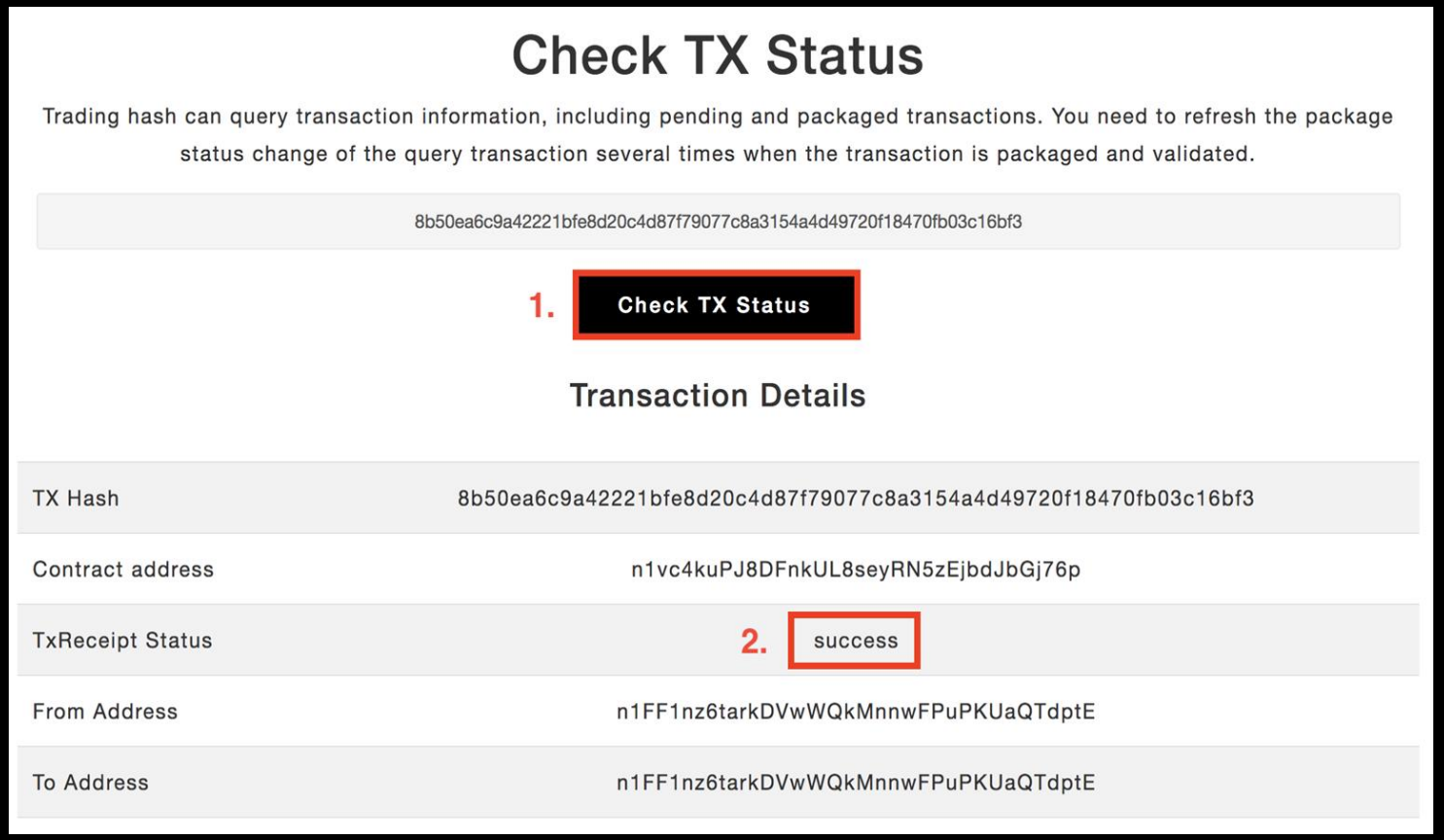
NEBULAS

Hello World – Deployment

1. Upon pressing “Test”, your contract is executed and tested. If it does not pass testing, you will be presented an error message.
2. Upon submission, your Smart Contract will be deployed to the Nebulas Blockchain.
3. Your txhash will be presented after submission. The txhash is how you can locate your smart contract in the future.
4. Upon deployment, you will receive a contract address. This address is required to interact with your contract.

NEBULAS

Hello World – Verifying successful deployment



1. Click "Check TX Status"
2. Transaction status will usually become successful within 15 seconds.

If your contract deployment remains "Pending" for a extended period of time, confirm that you have enough gas to pay the fee. Also make sure the contract test was successful.

If your transaction fails, verify the testing of the smart contract in the previous step.

Our TX Hash: 8b50ea6c9a42221bfe8d20c4d87f79077c8a3154a4d49720f18470fb03c16bf3

NEBULAS

Hello World –View code

1. Enter the deployment transaction hash to view the contract.
2. The function we will first call in this smart contract is “greetings”. It will be expecting one argument for the city variable.

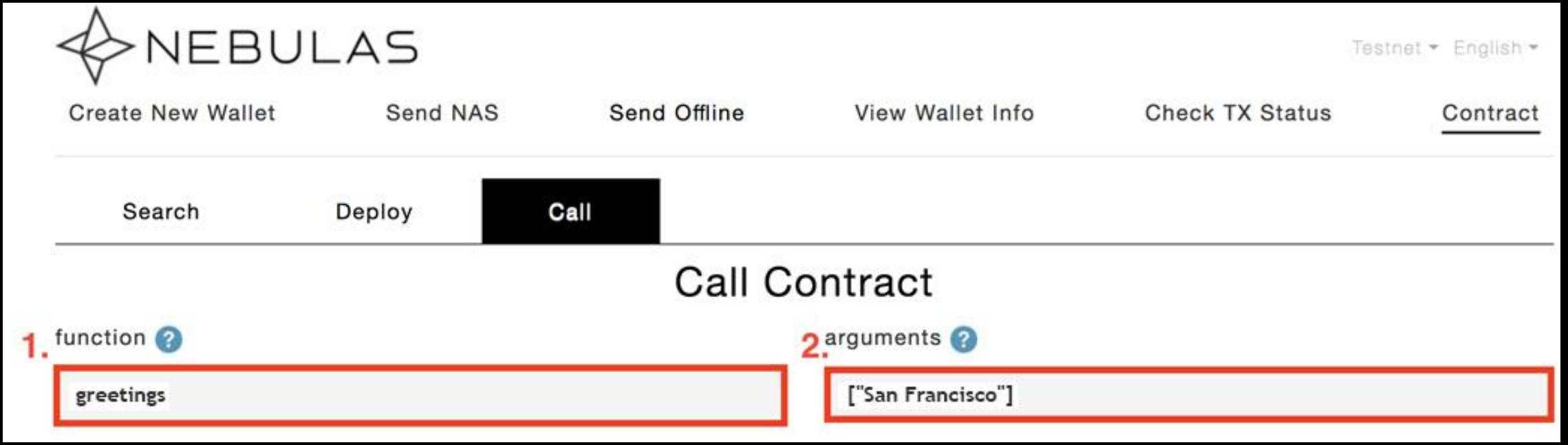
To view a deployed contract, you must have the deployment transaction hash.

To locate the txhash of a deployed smart contract, use the RPC request “GetTransactionByContract”.

curl -i -H 'Content-Type: application/json' -X POST http://localhost:8685/v1/user/getTransactionByContract -d '{"address":"n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p"}'

NEBULAS

Hello World – Calling the Contract



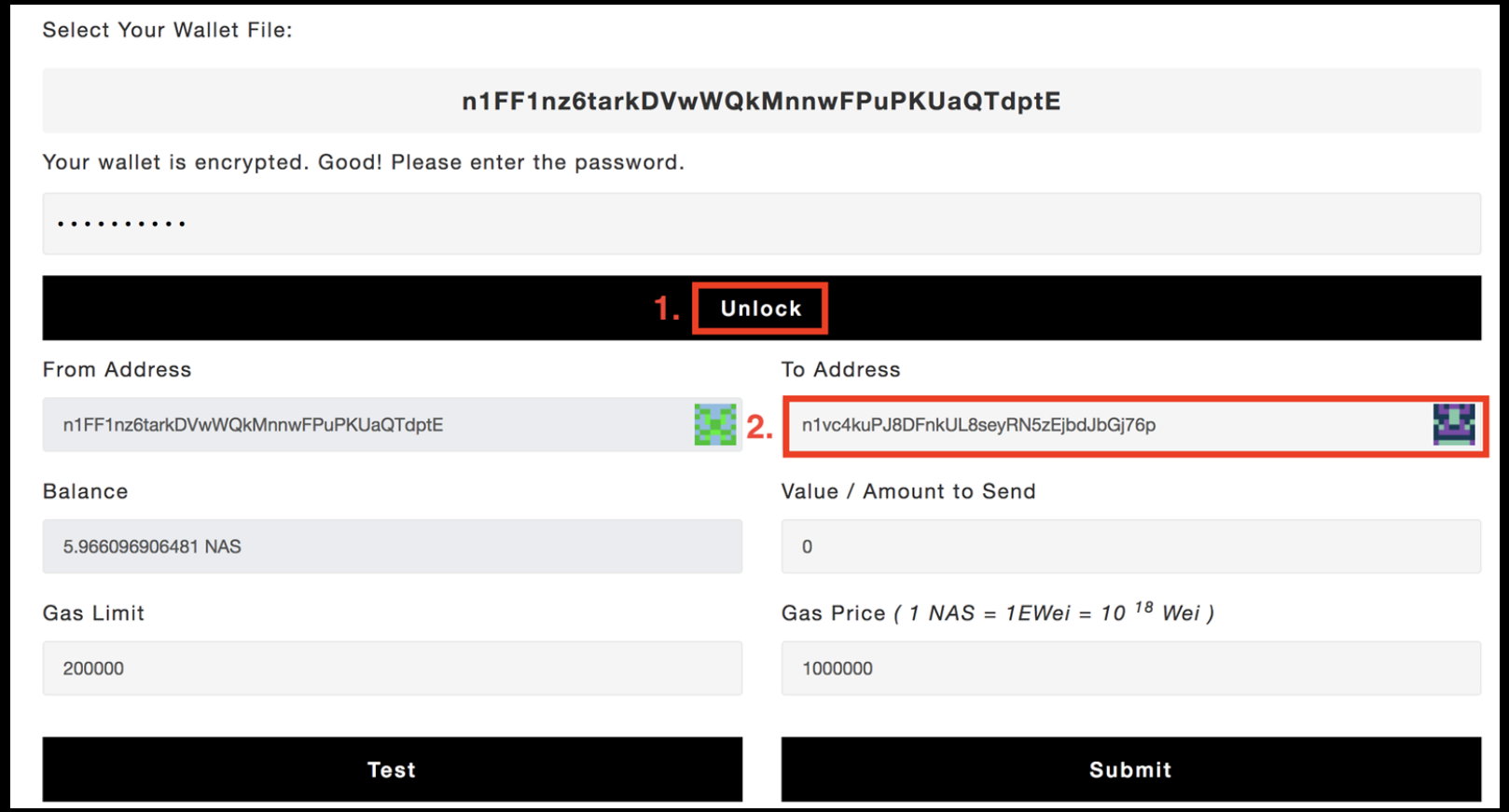
1. Enter the name of the function you want to call. In this example, we are calling the function “greetings”.
2. Enter any arguments that are required for the function. This is where we enter the “city” name. In our example, it is San Francisco.

If your contract has multiple functions, you can enter the different name of each.

All arguments must be entered in JSON array format.

NEBULAS

Hello World – Unlock wallet and enter Contract Address

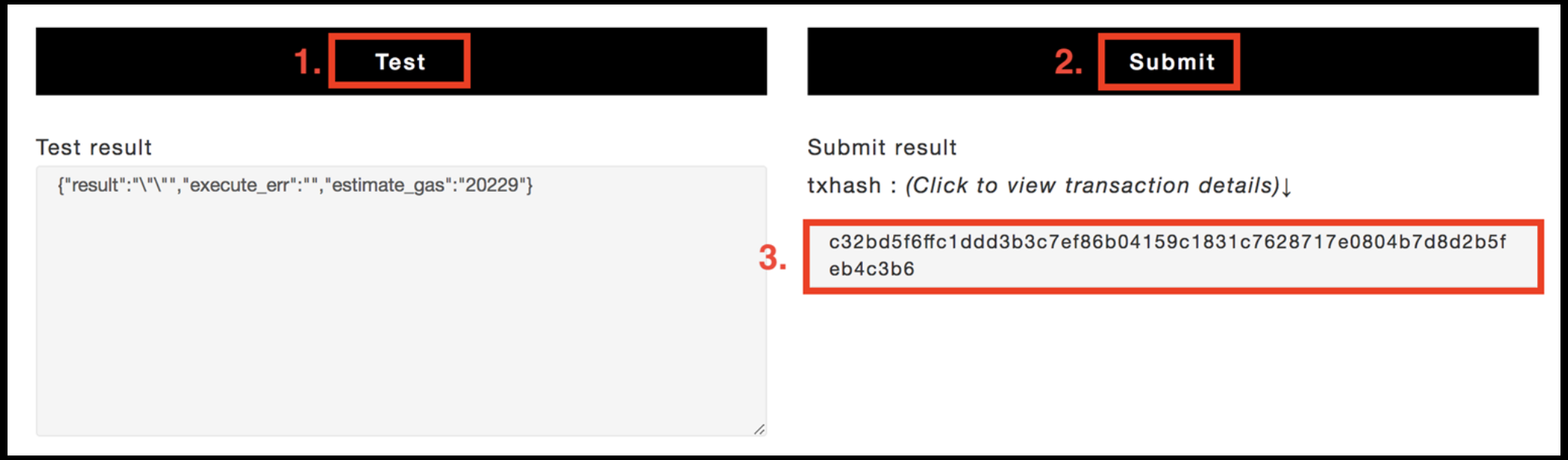


1. The calling of the smart contract is similar to the deployment process.
2. Enter the smart contract address that we were given earlier during the initial deployment.

Our deployed contract address: n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p

NEBULAS

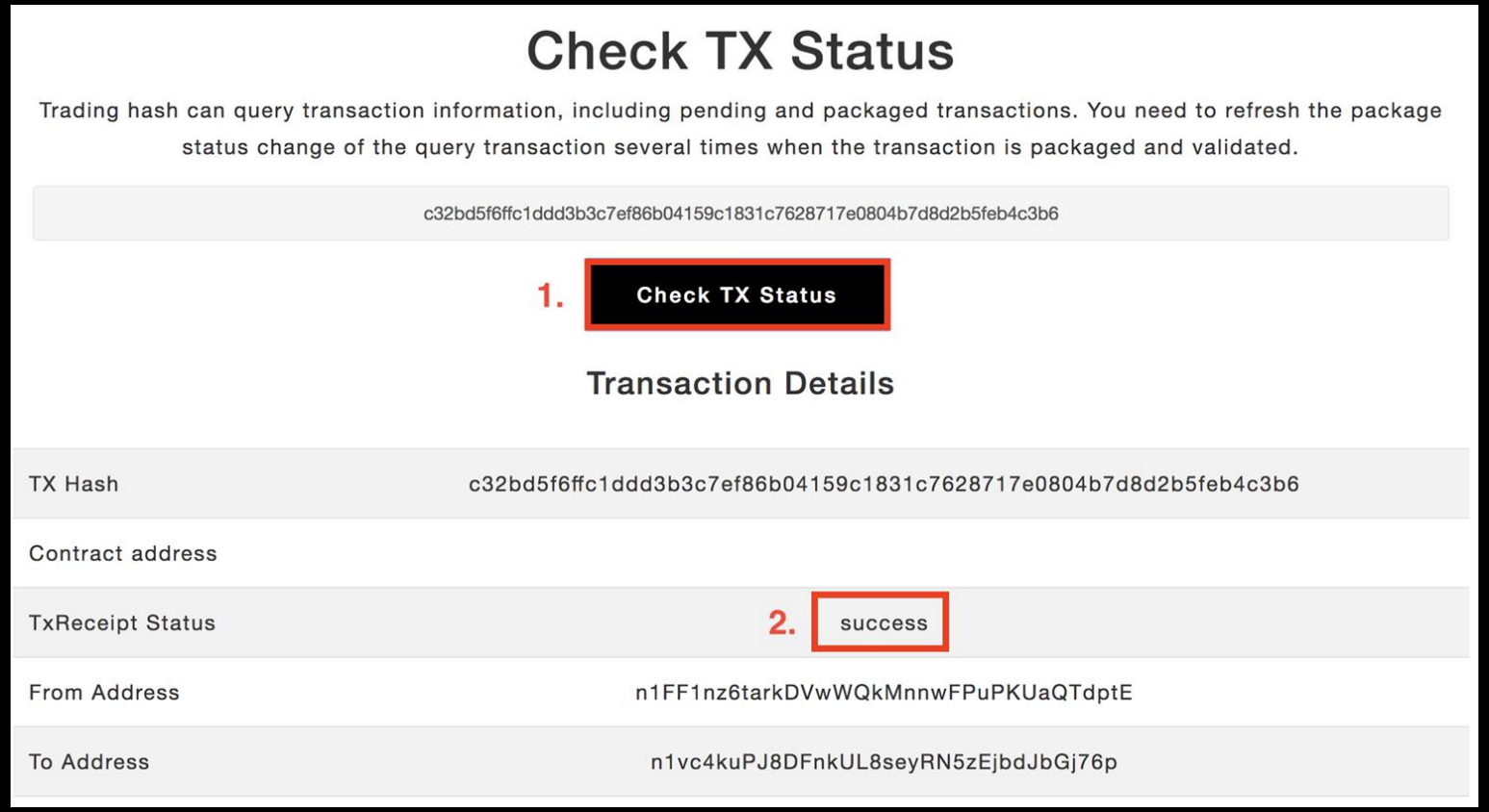
Hello World – Submit transaction to execute contract function



1. Test simulates the execution of the greetings function based on the arguments entered and verifies the successful completion of the function.
2. By submitting the transaction, the request is submitted and executed on the Nebulas Blockchain.
3. The user will be presented with a transaction hash as confirmation. The user can click on the txhash to view the current status of the transaction.

NEBULAS

Hello World – Query the status of the transaction



1. Click “Check TX Status”.
2. Transactions usually become successful within 15 seconds.

If your contract deployment remains “Pending” for a extended period of time, confirm that you have enough gas to pay the fee. Also make sure the contract test was successful.

If your transaction fails, verify the testing of the smart contract in the previous step.

Our transaction hash: c32bd5f6ffc1ddd3b3c7ef86b04159c1831c7628717e0804b7d8d2b5feb4c3b6

Hello World – Query the result of the executed function

RPC Request

```
› curl -i -H 'Content-Type: application/json' -X POST
  https://testnet.nebulas.io/v1/user/getEventsByHash
  -d '{"hash":"c32b … c3b6"}'
```

RPC Result


```
{
   "result":{
      "events":[
         {
            "topic": "chain.contract.greetings",
            "data":"\"Here is San Francisco. Hello World! By Nebulas Team.\""
         }
         ,{
            "topic":"chain.transactionResult",
            "data":"{\"hash\":\”c32b…c3b6\",\"status\":1,\"gas_used\":\"20229\",\"error\":\"\"}"
         }
      ]
   }
}
```


NEBULAS

THINKING IN BLOCKCHAIN

All RPC requests containing data must be submitted in JSON format.

All returned data will be in JSON format.

NEBULAS

Hello World –View code, prepare to query contract information

1. View contract code based on transaction hash.
2. Next, we will call the who function.

The who function will return the visitor passed during the initialization/deployment process earlier on. During deployment, we entered “Team Nebulas” as an argument.

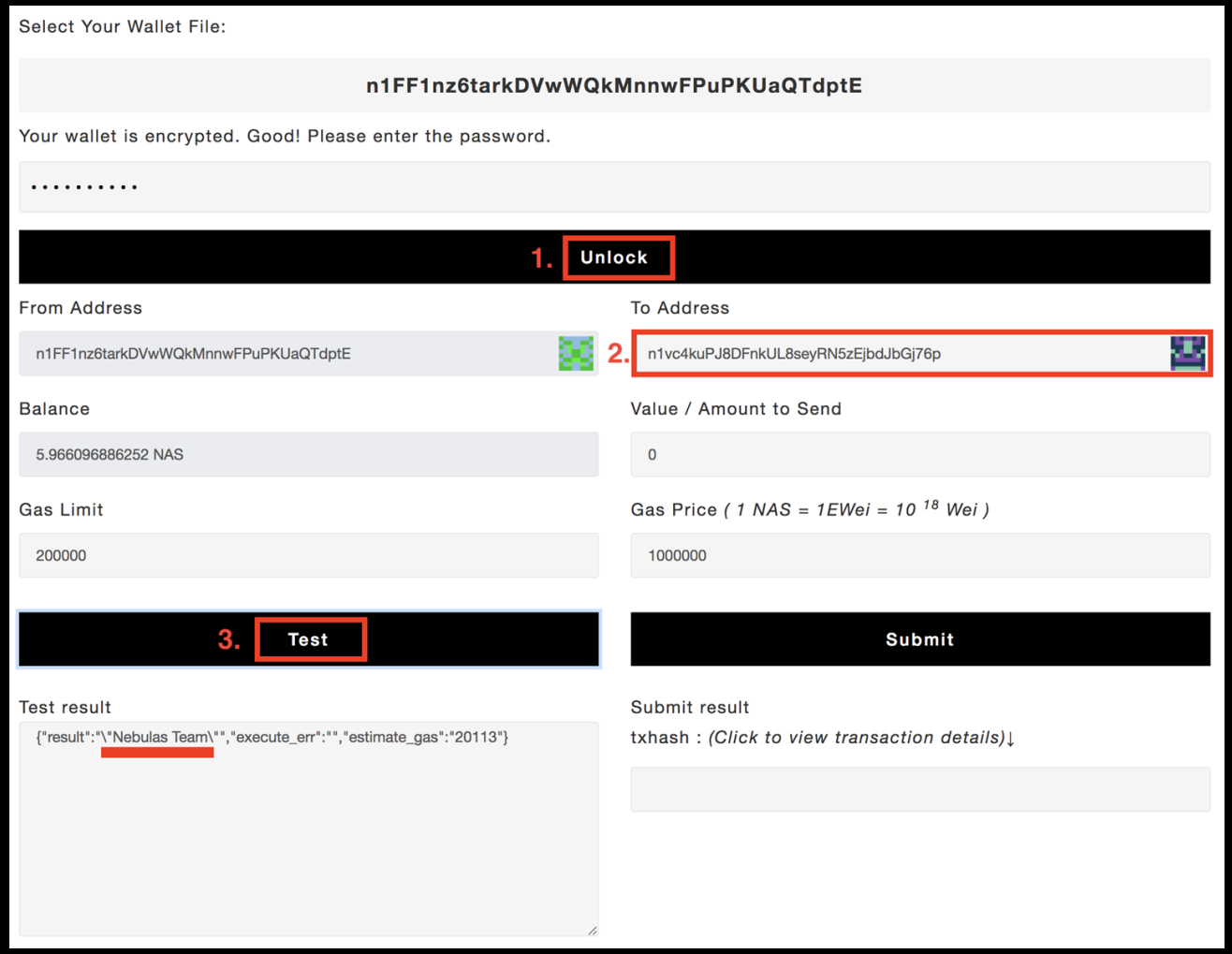
NEBULAS

Hello World – Fill in the parameters for obtaining contract information

1. Fill in the name of the contract function to be executed. In our example, we will execute the function “who”.
2. Fill in function parameters in array format. If you do not have any arguments, enter “[]” for empty/no arguments.

NEBULAS

Hello World –Get contract information



1. Unlock the wallet.
2. The destination contract address to call.
3. Clicking test will call the contract function and return the result.

Notice how we do not need to submit the transaction to receive the result since we are querying for a data result.

If we are pushing new data to the blockchain, the transaction must be submitted.

Our Deployed Smart Contract Address: n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p

RPC Requests and Interactions

Full documentation available at:

https://github.com/nebulasio/wiki/blob/master/rpc.md

https://github.com/nebulasio/wiki/blob/master/rpc_admin.md

Nebulas has two RPC request types.

- Public (user) - /v1/user/
- Permissioned (admin) - /v1/admin/

The public methods can be run locally and remotely. They do not require a password and do not interact with the wallet.

The permissioned methods need to be run on the local machine and are used to interact with a wallet and the local node.

Some requests use GET but most use POST.

Remote Procedure Calls (RPCs) provide a useful abstraction for building distributed applications and services.

RPC can be used instead of the web wallet. This is useful when building your DApp and the server side needs to interact with the blockchain.

In the following examples, we will use RESTful HTTP requests via CURL.

- The data payload of all requests must be JSON format.
- The response from the server is always JSON format.

For instructions on how to build/compile Nebulas, visit https://github.com/nebulasio/go-nebulas

Hello World – RPC Interactions

Sign Transaction / Deploy Smart Contract (step 1)

```
curl -i -H 'Accept: application/json' -X POST http://localhost:8685/v1/admin/sign -H 'Content-Type:
application/json' -d '{"transaction": {"from":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE",
"to":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE", "value":"0","nonce":8, gasPrice":"1000000",
"gasLimit":"2000000","contract":{"source":"class HelloWorld…..module.exports = HelloWorld;","sourceType":"js",
"args":"[\"Nebulas Team\"]"}}, "passphrase": "passphrase"}'
```

JSON Request


```
{
  "transaction":{
    "from":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE",
    "to":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE",
    "value":"0",
    "nonce":8,
    "gasPrice":"1000000",
    "gasLimit":"2000000",
    "contract":{
      "source":"Y2xhc3M.....gSGVBQi",
      "sourceType":"js",
      "args":"[\"Nebulas Team\"]"
    }
  },
  "passphrase":"passphrase"
}
```


Result of Request


```
{
  "result":{
    "data":"CiBhknd4Fr001ynq……kUOoAx8F4ng36Phikz=="
  }
}
```

Hello World – RPC Interactions

Sign Transaction / Deploy Smart Contract (step 2)

curl -i -H 'Accept: application/json' -X POST http://localhost:8685/v1/user/rawtransaction -H 'Content-Type: application/json' -d '{" data" : "iCrHtxyyIJks2/RErvBBA862…….D6iwAaGQ9OK1NisSGAuTBIYGiY1"}'

JSON Request

```
{
  "transaction":{
    "from":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE",
    "to":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE",
    "value":"0",
    "nonce":8,
    "gasPrice":"1000000",
    "gasLimit":"2000000",
    "contract":{
      "source":"Y2xhc3M.....gSGVBQi",
      "sourceType":"js",
      "args":"[\"Nebulas Team\"]"
    }
  },
  "passphrase":"passphrase"
}
```

Result of Request

```
{
  "result":{
    "txhash":"f37acdf9300182d8…..5c77a2ad3746b01c3b52“,
    "contract_address": "n1FF1nz6tarkDVwWQkMn….PKUaQTdptE"
  }
}
```

Hello World – RPC Interactions

Reviewing Status of Deployed Contract Transaction

```
curl -i -H 'Content-Type: application/json' -X POST http://localhost:8685/v1/user/getTransactionReceipt -d
'{"hash":" 8b50ea6c9a422……9720f184fb03c16bf3 "}'
```

JSON Request

```
{
   "hash":"8b50ea6c9a42221bfe8d20c4d87f79077
           c8a3154a4d49720f18470fb03c16bf3"
}
```

Result of Request


```
{
   "result":{
      "hash":"8b50ea6c9a42221…..d49720f18470fb03c16bf3",
      "chainId":100,
      "from":"n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p",
      "to":"n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p",
      "value":"0",
      "nonce":"25",
      "timestamp":"1529427936",
      "type":"deploy",
      "data":"eyJTb3VyY2jcmlwdCYWiI…XJncyI6IltcIkRXCJdIn0
      "gas_price":"1000000",
      "gas_limit":"200000",
      "contract_address":"n1ohRhzEJ11……MdStb7r3baPkaZ",
      "status":1,
      "gas_used":"21653",
```


Status Codes:
1 = Successful
2 = Pending
3 = Failed

Hello World – RPC Interactions

Submit Data to Contract (step 1)

```
curl -i -H 'Accept: application/json' -X POST http://localhost:8685/v1/admin/sign -H 'Content-Type:
application/json' -d '{"transaction":{"from":"n1FBAEC1hrZh5k3LMg1jUz7A9won9i3SUuM“0
,"to":"n1ohRhzEJ11AVEqHysgMuMdStb7r3baPkaZ", "value":"0","nonce":27,"gasPrice":"1000000",“
gasLimit":"2000000","contract":{"function":"greetings","args":"[\“San Francisco\"]"}}, "passphrase": "
passphrase "}'
```

JSON Request


```
{
  "transaction":{
    "from":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE",
    "to":"n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p",
    "value":"0",
    "nonce":9,
    "gasPrice":"1000000",
    "gasLimit":"2000000",
    "contract":{
      "function":"greetings",
      "args":"[\“San Francisco\"]"
    }
  },
  "passphrase":"passphrase"
}
```


Result of Request


```
{
  "result":{
    "txhash":"a8a44ed34a5fa…….91d696a1d4160aa0eb21",
    "contract_address":""
  }
}
```

Hello World – RPC Interactions

Submit Data to Contract (step 2)

curl -i -H 'Accept: application/json' -X POST http://localhost:8685/v1/user/rawtransaction -H 'Content-Type: application/json' -d '{" data" : "iCrHtxyyIJks2/RErvBBA862…….D6iwAaGQ9OK1NisSGAuTBIYGiY1"}'

JSON Request


```
{
  "transaction":{
    "from":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE",
    "to":"n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p",
    "value":"0",
    "nonce":9,
    "gasPrice":"1000000",
    "gasLimit":"2000000",
    "contract":{
      "function":"greetings",
      "args":"[\“San Francisco\"]"
    }
  },
  "passphrase":"passphrase"
}
```


Result of Request


```
{
  "result":{
    "txhash":"a8a44ed34a5fa…….91d696a1d4160aa0eb21",
    "contract_address":""
  }
}
```

Hello World – RPC Interactions

Return the “Hello World” String

```
curl -i -H 'Content-Type: application/json' -X POST http://localhost:8685/v1/user/getEventsByHash -d
'{"hash":" c32bd5f6ffc1ddd3b3c7ef86b04159c1831c7628717e0804b7d8d2b5feb4c3b6 "}'
```

JSON Request

```
{
  "hash":"c32bd5f6ffc1d…ef86b04159c1831
          c7628717e0804b7d8d2b5feb4c3b6"
}
```

Result of Request


```
{
  "result":{
    "events":[
      {
        "topic":"chain.contract.greetings",
        "data":"Here is San Francisco. Hello World! By Nebulas Team."
      },
      {
        "topic":"chain.transactionResult",
        "data":{
          "hash":" c32bd5f6ffc1ddd3b3c7ef8….7e0804b7d8d2b5feb4c3b6",
          "status":1,
          "gas_used":"20123",
          "error":"",
          "execute_result":"Nebulas Team"
        }
      }
```

Locating Your Smart Contract TX Hash

Returns the deployment tx hash of a contract.

```
curl -i -H 'Content-Type: application/json' -X POST http://localhost:8685/v1/user/getTransactionByContract -d
'{"address":"n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p"}'
```

JSON Request

```
{
  "address":"n1vc4kuPJ8DFnkUL8seyRN5zEjb
             dJbGj76p"
}
```

Result of Request


```
{
  "result":{
    "hash":"e73bb5b94da91896d3f5787...e96054ec365b7ea07f3a0",
    "chainId":100,
    "from":"n1FBAEC1hrZh5k3LMg1jUz7A9won9i3SUuM",
    "to":"n1FBAEC1hrZh5k3LMg1jUz7A9won9i3SUuM",
    "value":"0",
    "nonce":"9",
    "timestamp":"1529427936",
    "type":"deploy",
    "data":"eyJTb3VyY2VU.....iwiQXJncyI6IltcIkRLXCJdIn0=",
    "gas_price":"1000000",
    "gas_limit":"200000",
    "contract_address":"n1ohRhzEJ11AVEqHysgM…Stb7r3baPkaZ",
    "status":1,
    "gas_used":"21653",
```

Create Wallet via CLI and RPC

Wallets can be created directly via CLI and RPC request

```
curl -i -H 'Content-Type: application/json' -X POST
http://localhost:8685/v1/admin/account/new -d '{"passphrase":"passphrase"}'
```

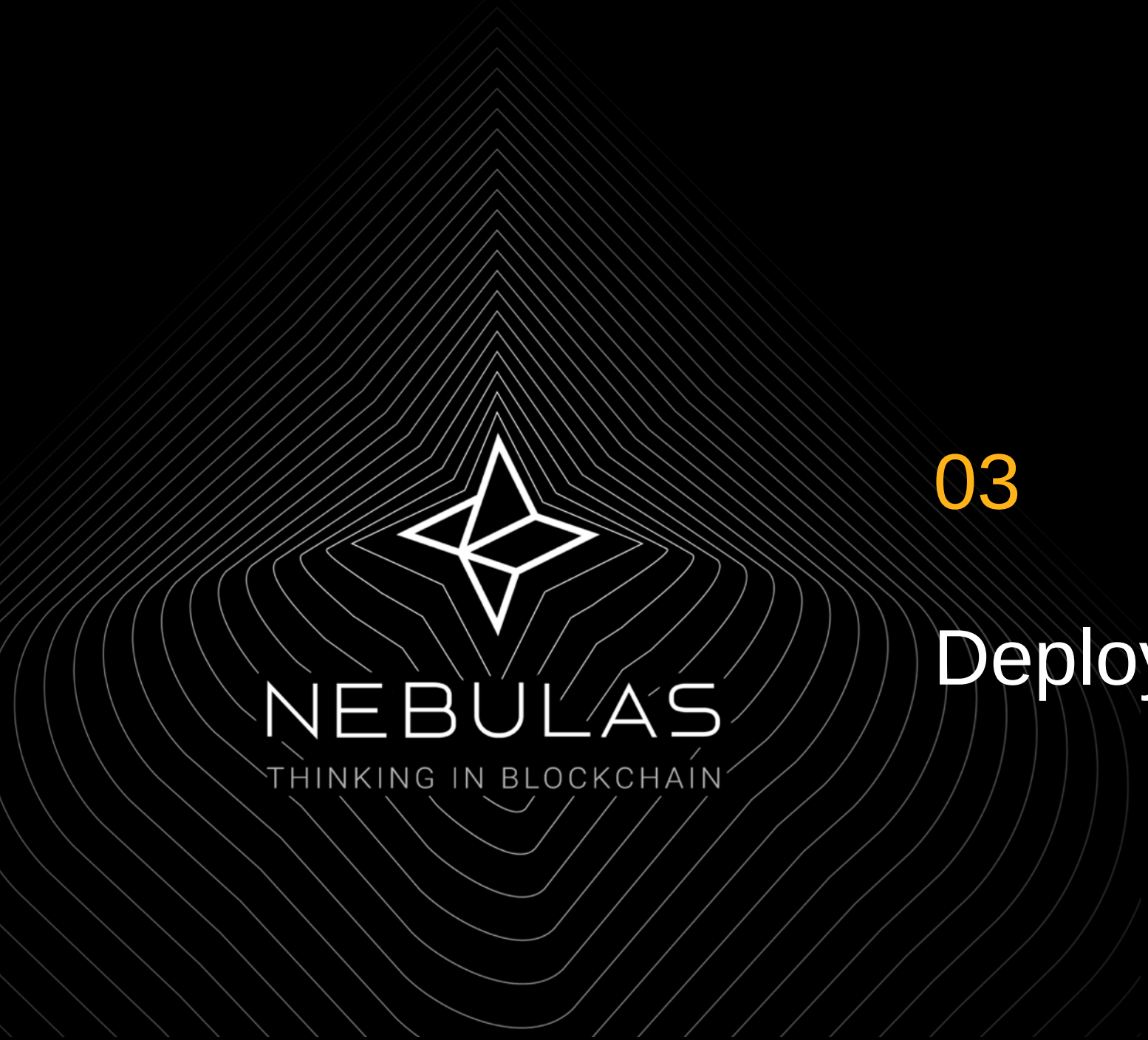
JSON Request

```
{
  " passphrase":"passphrase"
}
```

Result of Request

```
{
   "result":{
      "address":"n1U4iy97kmG3yr6hJtzED4LyujtXxjwQDSy"
   }
}
```

To create a wallet via command line/terminal, from your compiled directory, enter “./neb account new” and follow the instructions.

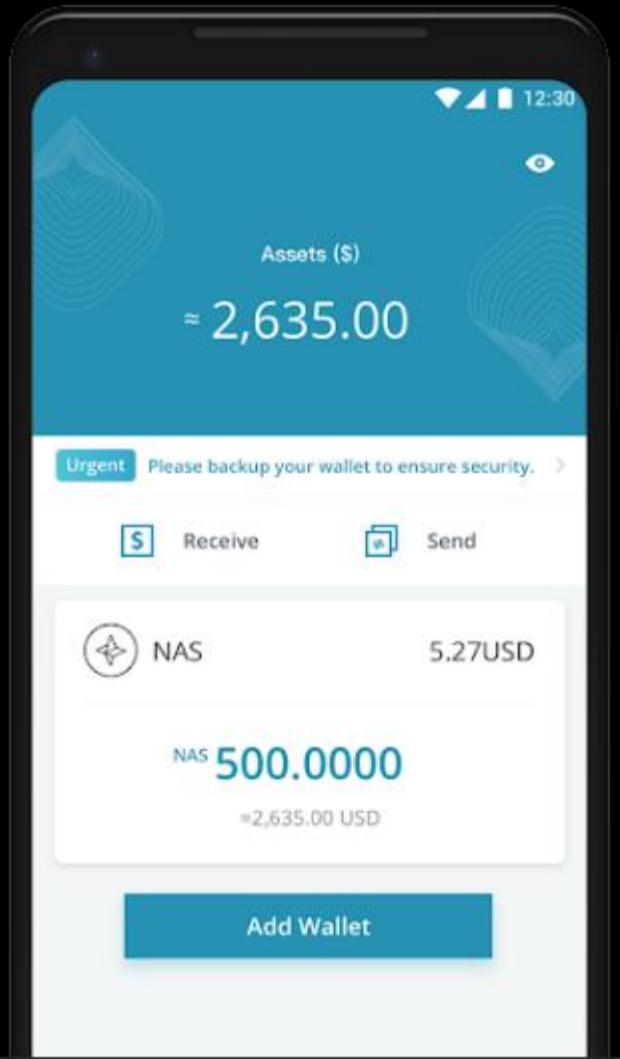


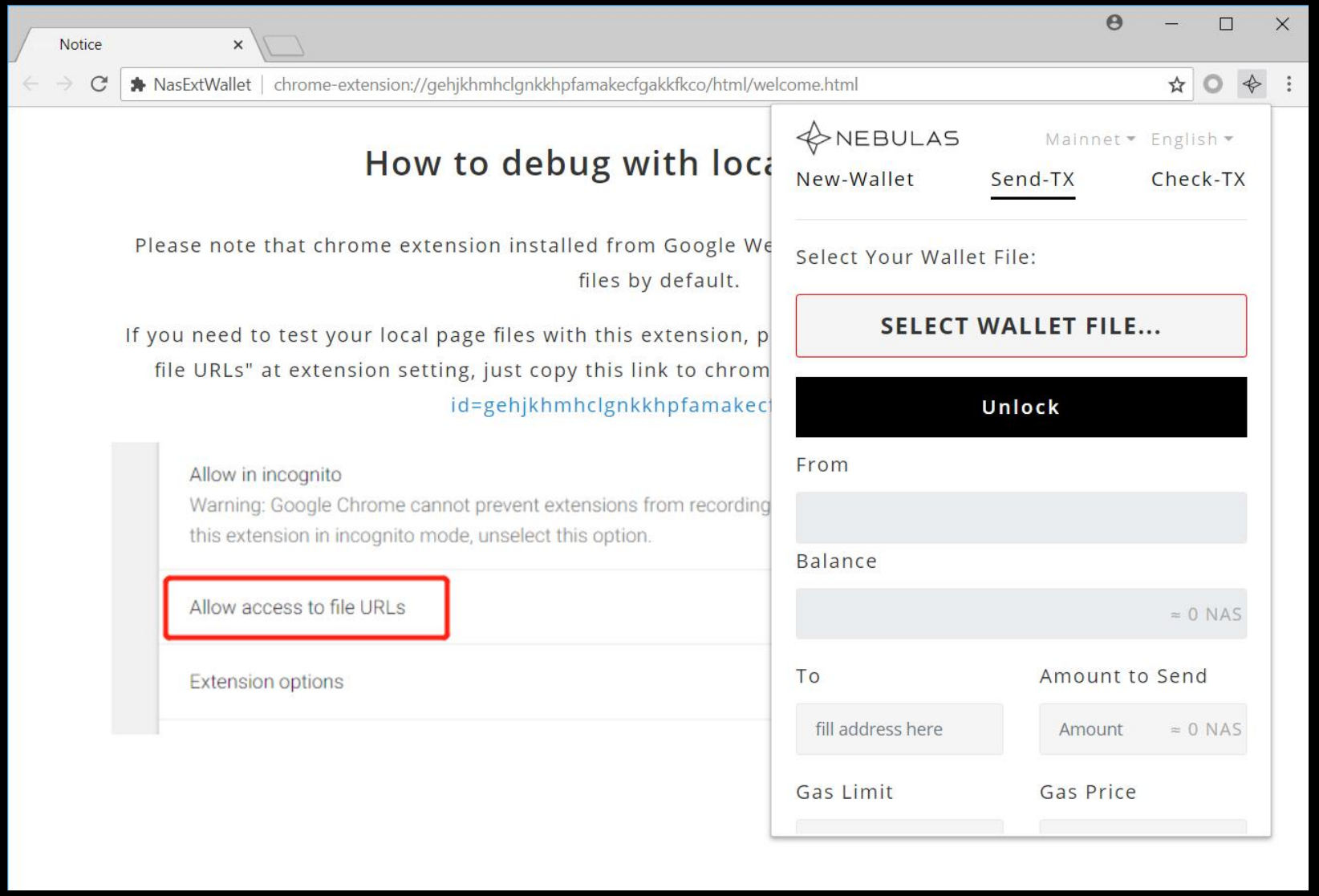
03

Deploy DApps

nebPay

nebPay allows for wallet integration simply with the Nebulas web-wallet and NASnano mobile wallet.
Learn more at https://github.com/nebulasio/nebPay





Available SDK's

Nebulas offers SDK packages and wallets for the most popular programming languages and mobile operating systems.

- NasNano Wallet available on the Apple Store
- SDK Package available on Github

- NasNano Wallet available on the Google Store
- SDK Package available on Github

- Web Wallet available for All Browsers across all Operating Systems
- Web Extension Wallet available on the Chrome Web Store

SDK Packages available on Github

All packages available at github.com/nebulasio

NEBULAS

DApps

Develop

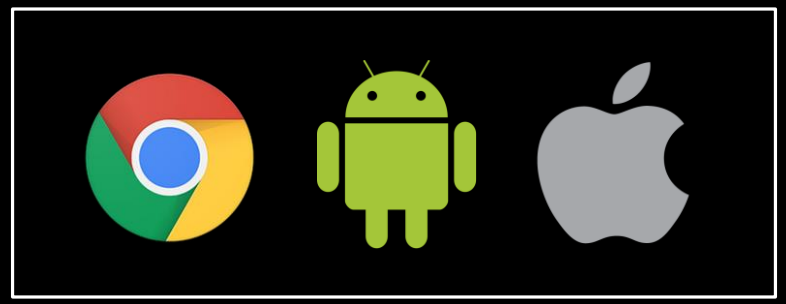
Deploy

NEBULAS

THINKING IN BLOCKCHAIN

Develop

Invoke API



Interact with user

Video Tutorials, commented smart contracts and this presentation available at

http://github.com/nebulas-learning

Nebulas Wiki and Tutorials available at

http://github.com/nebulasio/wiki

nebulas.io

t.me/nebulasio

nebulasio.herokuapp.com

twitter.com/nebulasio

/r/nebulas

Channel: Nebulas

medium.com/nebulasio

contact@nebulas.io