Nodix White Paper

1 General concept

Nodix is a modular blockchain engine, including distributed application server, which can be used to design customized blockchains, and build easily HTML5/js applications , based on a powerfull and portable C framework, to manipulate hierarchy of dynamic object with lockless multi thread access for maximum parallelism.

The design is very simple, programmed in C, it uses a system of portable binary module which can be created out of .dll or .so files, and used as base building block for distributed application using either C, HTML5/js, or the integrated script engine.

Bitcore and most blockchain client use a monolithic architecture, and alternative blockchains are created by forking the original bitcoin source code tree, changing some hard - coded values and recompiling the core to make the new coin which often contain the masternode server, the blockchain client, and the wallet in a single application executable, with the hard-coded values specific for the coin compiled in, which make it hard to use in a wider application framework.

Instead of monolithic design, each functionalities are encapsulated in different modules, which use enhanced json objects as configuration parameters, which allow flexible node definitions. Modules can be used as application modules used by scripts and other modules, or bound to http service using CGI or JSON/RPC protocol to be used by external application such as javascript.

The problem with monolithic design is that all applications based on the blockchain end up being run on centralized server, who run closed code on a web server on top of the blockchain, with full control of private key, which make them vulnerable to hacks, and not being decentralized or trustless. Most blockchain service are run in such kind of configuration.

With Nodix distributed framework, all the code and data for applications can be hosted and executed on nodes, and the interface provide the API to do full transaction signing and cryptography inside of the browser, which allow to open the RPC interface to the world without compromising private key.

The private keys are encrypted and decrypted inside of the browser, and the node never have to manipulate private key directly.

All together, the portable binary module, script engine and in browser cryptography allow for secure decentralization of applications with a dynamic HTML5 interface, and performant lockless parallel access to application objects from javascript or any language supporting asynchronous JSON/RPC request.

2 Running and configuration

As all the functionalities are designed with modularity in mind, it's very easy to adapt the node to specific blockchains, or to add new functionalities into custom blockchain by recompiling only the module containing the specific algorithm, or adding new plugin to the node via module RPC interface binding or script. Binary modules can used on any operating system with intel cpu as long as the libcon library is compiled for this host.

The design rely on a framework of dynamic data tree, which use an internal memory allocator using lockless reference counting mechanism rather than standard C allocation, which allow for easy sharing of data pointer and object hierarchy between different modules without pointer ownership, and to transmit complex data to javascript application or other nodes over the network via the RPC API exposed by the modules.

It allow flexible sharing of object between modules and threads, with the memory being managed automatically by the framework, with acquiring and releasing of objects references done manually in C.

The powerful system of dynamic data tree allow to represent a hierarchy of dynamically typed object/key nodes, used in native script engine to generate HTML5 web pages or json data for HTML5/js application.

The scripting engine is used to define the blockchain node protocol and coin specifics, as well as defining the modules API exposed through node services , and configuring the web server.

Blockchain configuration.

```
let NODE_GFX_OBJECT configuration = `
{
              "name":"nodix",
              "seed_node" :
              {
                     "host":"nodix.com",
                     "port" : 16714
              },
              "magic":0xD9BEFECA,
              "version":60018,
              (NODE_MODULE_DEF) "sign_mod":{"file":"modz/ecdsa.tpo"},
              "pubKeyVersion":0x19,
              "staking":
               {
                     "targetspacing":64,
                     "maxtargetspacing" : 640,
                     "targettimespan":960,
                     "limit":0x1B00FFFF,
                     "minStakeDepth" : 2,
                     "reward" : 150000000,
                     (NODE_MODULE_DEF) "pos_kernel" : {"file":"modz/stake_pos3.tpo"}
               },
               "mining":
               {
                     "targetspacing":64,
                     "maxtargetspacing" : 640,
                     "targettimespan":960,
                     "limit":0x1E0FFFFF,
                     "reward":10000000000000,
                     "last_pow_block":200,
                     "paytxfee":10000
              },
              "genesis":
              {
                     "version"           :1,
                     "time"              :1466419085,
                     "bits"              :0x1e0fffff,
                     "nonce"                    :579883,
                     "InitialStakeModifier":0,
                     "InitialStakeModifier2":0
              }
}`
```

2.1 Nodix coin script code

Node definition.

```
let NODE_BITCORE_NODE SelfNode = `
{
       "user_agent" : "Nodix",
       "paytxfee" : 0.01,
       "block_height" : 0,
       (NODE_GFX_INT)"version"  : 0,
       (NODE_GFX_INT)"current_pos_diff" : 0,
       (NODE_GFX_INT)"current_pow_diff" : 0,
       (NODE_BITCORE_BLK_HDR) "last_block" : {},
       (NODE_BITCORE_BLK_HDR) "lastPOSBlk" : {},
       (NODE_BITCORE_BLK_HDR) "lastPOWBlk" : {},
       (NODE_BITCORE_ADDR)"p2p_addr" :
       {
              "services": 0,
              (NODE_NET_IPV)"addr" : "127.0.0.1",
              (NODE_GFX_SHORT)"port" : 16819
       },
       (NODE_SERVICE)"http_service":
       {
              "port"        : 16820,
              "docroot"   : "web",
              "name"        : "nodix webservice",
              "indexpage" : "/nodix.site",
              "mimes" : {   "js":"text/javascript",
                            "css" : "text/css",
                            "png" : "image/png",
                            "svg" : "image/svg+xml",
                            "html": "text/html" },
              "defaultmime" : "text/plain",
              "maxpost"     : 1024,
              "modules"     :
              [
                     {"base" : "/jsonrpc", "type" : "rpc", (NODE_MODULE_DEF) "rpc_wallet"     : {"file":"modz/rpc_wallet.tpo"}},
                     {"base" : "/api/"   , "type" : "cgi", (NODE_MODULE_DEF) "block_explorer" : {"file":"modz/block_explorer.tpo"}}
              ]
       },
       (NODE_BITCORE_NODE_LIST) "peer_nodes" : [],
       (NODE_BITCORE_WALLET_ADDR_LIST) "addr scan list" : null,
       (NODE_BITCORE_MSG_LIST) "send queue" : [],
       (NODE_BITCORE_MSG_LIST) "emitted_queue" : [],
       (NODE_BITCORE_TX_LIST) "tx mem pool" : [],
       (NODE_BITCORE_BLK_HDR_LIST) "submitted blocks" : []
}`
```

Import modules

```
let NODE_MODULE_DEF  vec = `{"order":0, "file" : "modz/vec3.tpo"}`
let NODE_MODULE_DEF  protocol_adx = `{"order":1, "file" : "modz/protocol_adx.tpo"}`
let NODE_MODULE_DEF  block_adx = `{"order":2, "file" : "modz/block_adx.tpo"}`
let NODE_MODULE_DEF  wallet= `{"order":3, "file" : "modz/wallet.tpo"}`
let NODE_MODULE_DEF  node_adx = `{"order":4, "file" : "modz/node_adx.tpo"}`
let NODE_MODULE_DEF  nodix = `{"order":5, "file" : "modz/nodix.tpo"}`
```

Global variables

```
let NODE_BITCORE_BLK_HDR   genesis_blk = `{}`
let NODE_GFX_INT           ping_nonce  = 1
let NODE_GFX_BINT          block_reward = 0
let NODE_GFX_BINT          lost_reward = 0
let NODE_GFX_BINT          cur_len = 0
```

Node initialization code

```
proc init_node = `

       protocol_adx.init_protocol (configuration)
       block_adx.init_blocks      (configuration)
       node_adx.node_init_self    (SelfNode)

       loadmod                                      (configuration.staking.pos_kernel)
       configuration.staking.pos_kernel.init_pos    (configuration.staking)
       node_adx.node_init_service                   (SelfNode.http_service, configuration.staking.pos_kernel)

       set SelfNode.version = configuration.version;
       set SelfNode.block_reward = configuration.mining.reward;

       node_adx.node_load_block_indexes             ()

       block_adx.make_genesis_block      (configuration.genesis, genesis_blk)

       if (SelfNode.block_height = 0)
              node_adx.node_set_last_block                            (genesis_blk)
              nodix.compute_pow_diff                                  (genesis_blk, SelfNode.current_pow_diff)
              configuration.staking.pos_kernel.store_blk_staking  (genesis_blk)
       endif

       if (SelfNode.block_height > 1)
              node_adx.node_load_last_blks()

              block_adx.get_pow_reward(SelfNode.lastPOWBlk.height, block_reward)
              set SelfNode.pow_reward = block_reward;

              configuration.staking.pos_kernel.load_last_pos_blk(SelfNode.lastPOSBlk)    :

                     set SelfNode.lastPOSBlk = SelfNode.last_block;

                     configuration.staking.pos_kernel.find_last_pos_block(SelfNode.lastPOSBlk)
                     node_adx.node_store_last_pos_hash                            (SelfNode.lastPOSBlk)

              endor

              configuration.staking.pos_kernel.compute_last_pos_diff(SelfNode.lastPOSBlk, SelfNode.current_pos_diff) :
                     set SelfNode.current_pos_diff = configuration.staking.limit;
              endor

              configuration.staking.pos_kernel.stake_get_reward (SelfNode.lastPOSBlk.height, block_reward) ?
                     set SelfNode.pos_reward = block_reward;
              endor
       endif

       sethandler SelfNode.emitted_queue{ "cmd=verack" } = on_verack;
       sethandler SelfNode.emitted_queue{ "cmd=version" } = on_version;
       sethandler SelfNode.emitted_queue{ "cmd=ping" } = on_ping;
       sethandler SelfNode.emitted_queue{ "cmd=pong" } = on_pong;
       sethandler SelfNode.emitted_queue{ "cmd=inv" } = on_inv;
       sethandler SelfNode.emitted_queue{ "cmd=addr" } = on_addr;
       sethandler SelfNode.emitted_queue{ "cmd=block" } = on_block;


       node_adx.new_peer_node(configuration.seed_node)
       node_adx.queue_version_message(SelfNode.peer_nodes[0])
       node_adx.queue_getaddr_message(SelfNode.peer_nodes[0])
`
```

2.2 Example of bootstrapping the node based on script file

```c
int main(int argc, const char **argv)
{
	app_func_ptr			app_init, app_start, app_loop, app_stop;
	struct string			node_name = { PTR_NULL };
	mem_zone_ref			 params = { PTR_NULL }, script_vars = { PTR_NULL }, init_node_proc = { PTR_NULL };
	tpo_mod_file			*nodix_mod;
	int done = 0,n;

	init_mem_system			();
	init_default_mem_area		(24 * 1024 * 1024);
	set_exe_path			();

	network_init			();

	load_module		("modz/libbase.tpo", "libbase", &libbase_mod);

	load_script			= get_tpo_mod_exp_addr_name(&libbase_mod, "load_script", 0);
	resolve_script_var		= get_tpo_mod_exp_addr_name(&libbase_mod, "resolve_script_var", 0);
	get_script_var_value_str	= get_tpo_mod_exp_addr_name(&libbase_mod, "get_script_var_value_str", 0);
	get_script_var_value_ptr	= get_tpo_mod_exp_addr_name(&libbase_mod, "get_script_var_value_ptr", 0);
	execute_script_proc		= get_tpo_mod_exp_addr_name(&libbase_mod, "execute_script_proc", 0);
	tree_manager_init		= get_tpo_mod_exp_addr_name(&libbase_mod, "tree_manager_init", 0);

	tree_manager_init	(16 * 1024 * 1024);

	load_script		("nodix.node", &script_vars,3);

	if (!get_script_var_value_str(&script_vars, "configuration.name", &node_name, 0))
		make_string(&node_name, "nodix");

	if (!set_home_path(node_name.str))
	{
		console_print("could not set home dir 'nodix' \n");
		return 0;
	}


	get_script_var_value_ptr	(&script_vars, "nodix.mod_ptr"	, &nodix_mod);


	app_init = (app_func_ptr)get_tpo_mod_exp_addr_name(nodix_mod, "app_init", 0);
	app_start = (app_func_ptr)get_tpo_mod_exp_addr_name(nodix_mod, "app_start", 0);
	app_loop = (app_func_ptr)get_tpo_mod_exp_addr_name(nodix_mod, "app_loop", 0);
	app_stop = (app_func_ptr)get_tpo_mod_exp_addr_name(nodix_mod, "app_stop", 0);

	if (!app_init(&script_vars))
	{
		console_print("could not initialize app ");
		console_print(nodix_mod->name);
		console_print("\n");
		return 0;
	}
	resolve_script_var		(&script_vars,PTR_NULL, "init_node"	, 0xFFFFFFFF	,&init_node_proc);
	execute_script_proc		(&script_vars, &init_node_proc);

	if (daemonize(node_name.str) <= 0)
	{
		console_print("daemonize failed \n");
		return 0;
	}

	if (!app_start(&params))
	{
		console_print("could not start app ");
		console_print(nodix_mod->name);
		console_print("\n");
		return 0;
	}

	while (isRunning())
	{
		app_loop(PTR_NULL);
	}

	app_stop(PTR_NULL);
}
```

2.3 Node services

Additionally to basic block chain functions, nodes can expose services via http protocol.

Those services can be defined in 3 manners :

- From binary modules using classic cgi interface with parameters passed via the HTTP URL
- From binary modules using JSON/RPC over HTTP request.
- Either full web pages generated by the nodix script engine.

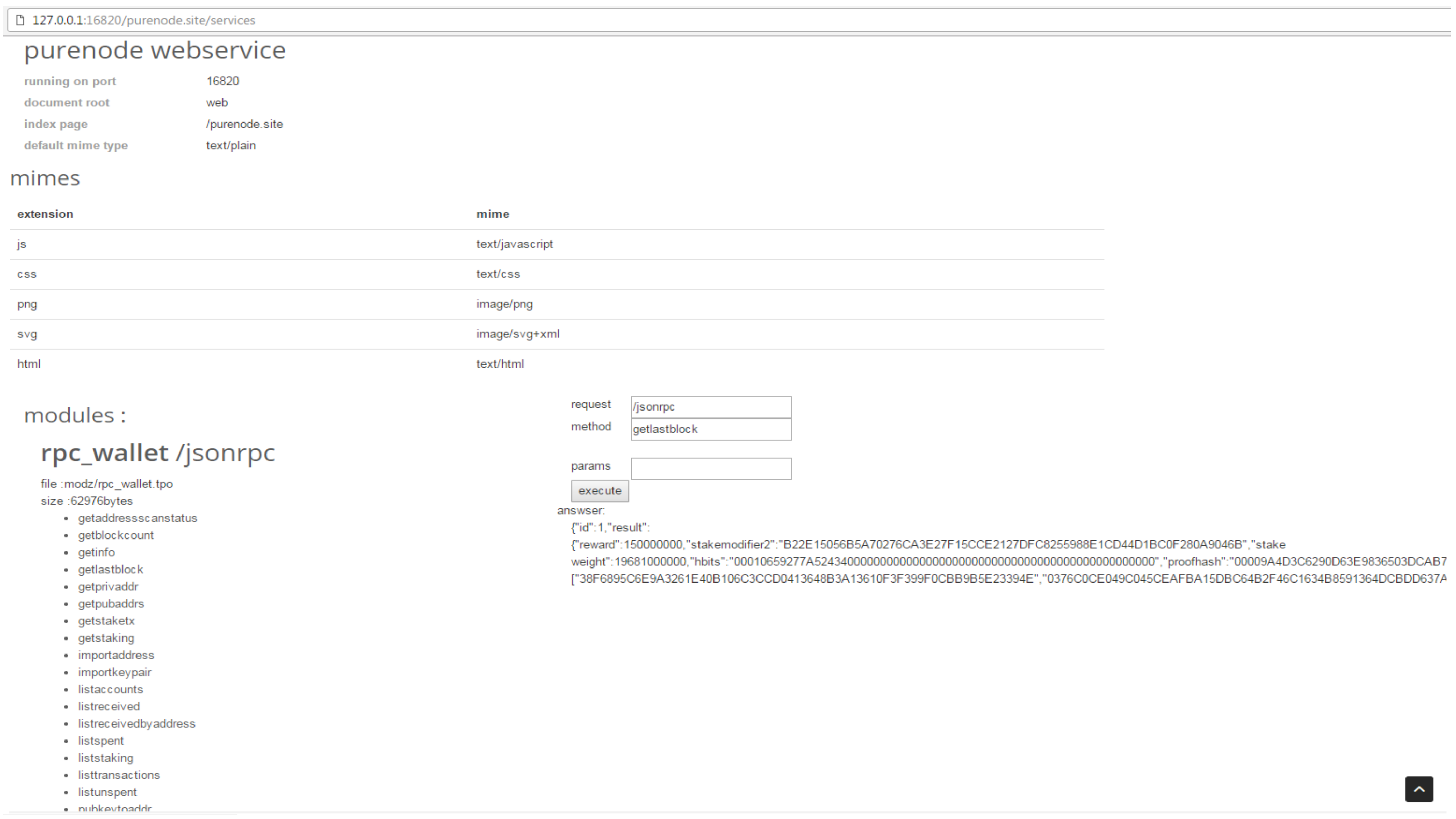
Sample page script from nodix.site

```
let NODE_JSON_ARRAY stylesheets = `[
       "//fonts.googleapis.com/css?family=Open+Sans:400,300,600&amp;subset=cyrillic,latin",
       "/assets/plugins/bootstrap/css/bootstrap.min.css",
       "/assets/css/style.css",
       "/assets/css/headers/header-default.css",
       "/assets/css/blocks.css",
       "/assets/css/footers/footer-v7.css",
       "/assets/plugins/animate.css",
       "/assets/plugins/line-icons/line-icons.css",
       "/assets/plugins/font-awesome/css/font-awesome.min.css",
       "/assets/plugins/brand-buttons/brand-buttons.css",
       "/assets/css/theme-skins/dark.css",
       "/assets/css/custom.css",
       "/assets/plugins/sky-forms-pro/skyforms/css/sky-forms.css",
       "/assets/plugins/sky-forms-pro/skyforms/custom/custom-sky-forms.css"
]`

let NODE_JSON_ARRAY scripts = `[
       "/assets/plugins/jquery/jquery.min.js",
       "/assets/plugins/jquery/jquery-migrate.min.js",
       "/assets/plugins/bootstrap/js/bootstrap.min.js",
       "/assets/plugins/back-to-top.js",
       "/assets/plugins/smoothScroll.js",
       "/assets/plugins/sky-forms-pro/skyforms/js/jquery-ui.min.js",
       "/assets/plugins/sky-forms-pro/skyforms/js/jquery.validate.min.js",
       "/assets/plugins/sky-forms-pro/skyforms/js/jquery.maskedinput.min.js",
       "/assets/plugins/scrollbar/js/jquery.mCustomScrollbar.concat.min.js",
       "/assets/js/custom.js",
       "/assets/js/app.js"
]`

let NODE_JSON_ARRAY metas = `[
       {"viewport":"width=device-width, initial-scale=1.0"},
       {"description":""},
       {"author":""}
]`

let NODE_MODULE_DEF  node_adx = `{"file" : "modz/node_adx.tpo"}`
let NODE_JSON_ARRAY node_modules = `[]`

page services = `
       html_head "NODIX SERVICE INFO ( SERVICES )"
       html_block "templates/menu.html"
       html_block "templates/services.html"
       html_scripts
       html_var SelfNode.http_service;

       html_js
              $(document).ready(function ()
              {
                     App.init();
                     App.initScrollBar();
                     site_base_url = '/nodix.site';
                     api_base_url ='';
                     lang         = 'en';

                     $('#serv_port').html(http_service.port);
                     $('#serv_name').html(http_service.name);
                     $('#serv_root').html(http_service.docroot);
                     $('#index').html(http_service.indexpage);
                     $('#defaultmime').html(http_service.defaultmime);

                     make_mime_table('mimes',http_service.mimes);

                     if(http_service.http_status==1)
                            $('#status').html   ('running');
                     else
                            $('#status').html   ('stopped');

                     mods = http_service.modules;
                     make_modules_html   ('service_modz_div',http_service.modules);
                     make_scripts_html   ('service_scriptz_div',http_service.nodescripts);
              });
       end_js

html_block "templates/footer.html"
```

`

Pages from site scripts can be accessed via http://node-ip:service-port/script-file.site/page-name

For example, the page implemented as the '*services*' method in the script '*nodix.site*' can be accessed at the address http://node-ip:service-port/nodix.site/services


127.0.0.1:16820/purenode.site/services

purenode webservice

running on port	16820
document root	web
index page	/purenode.site
default mime type	text/plain

mimes

extension	mime
js	text/javascript
css	text/css
png	image/png
svg	image/svg+xml
html	text/html

modules :

rpc_wallet /jsonrpc

file :modz/rpc_wallet.tpo
size :62976bytes

- getaddressscanstatus
- getblockcount
- getinfo
- getlastblock
- getprivaddr
- getpubaddrs
- getstaketx
- getstaking
- importaddress
- importkeypair
- listaccounts
- listreceived
- listreceivedbyaddress
- listspent
- liststaking
- listtransactions
- listunspent
- pubkeytoaddr

request	/jsonrpc
method	getlastblock
params	

answser:

```
{"id":1,"result":
{"reward":150000000,"stakemodifier2":"B22E15056B5A70276CA3E27F15CCE2127DFC8255988E1CD44D1BC0F280A9046B","stake
weight":19681000000,"hbits":"00010659277A52434000000000000000000000000000000000000000000000000","proofhash":"00009A4D3C6290D63E9836503DCAB7
["38F6895C6E9A3261E40B106C3CCD0413648B3A13610F3F399F0CBB9B5E23394E","0376C0CE049C045CEAFBA15DBC64B2F46C1634B8591364DCBDD637A
```

Functions exported from binary modules can be accessed as an RPC method on the service or bound to HTTP URL path with CGI API using the node configuration file.

```
(NODE_SERVICE)"http_service":
{
       "port"        : 16820,                            #port used to connect the service
       "docroot"    : "web",                             #root folder for web files
       "name"        : "nodix webservice",               #name of the service
       "indexpage" : "/nodix.site",                      #default index page
       "mimes" : {   "js":"text/javascript",             #list of mime-type / file extensions
                     "css" : "text/css",
                     "png" : "image/png",
                     "svg" : "image/svg+xml",
                     "html": "text/html" },
       "defaultmime"  : "text/plain",                    #default mime type
       "maxpost"     : 1024,                             #max size of HTTP POST data
       "modules"     :                                   #list of modules exposed through the http interface
       [
              {"base" : "/jsonrpc", "type" : "rpc", (NODE_MODULE_DEF) "rpc_wallet"     : {"file":"modz/rpc_wallet.tpo"}},
              {"base" : "/api/"    , "type" : "cgi", (NODE_MODULE_DEF) "block_explorer" : {"file":"modz/block_explorer.tpo"}}
       ]
}
```

This line bind functions exported from the binary module '`modz/rpc_wallet.tpo`' as RPC/JSON method to the URL path '/jsonrpc'.

```
{"base" : "/jsonrpc", "type" : "rpc", (NODE_MODULE_DEF) "rpc_wallet"     : {"file":"modz/rpc_wallet.tpo"}},
```

which can be called via http://node-ip:port/jsonrpc

This line bind function exported from binary module '`modz/block_explorer.tpo`' to the base URL '/api/'

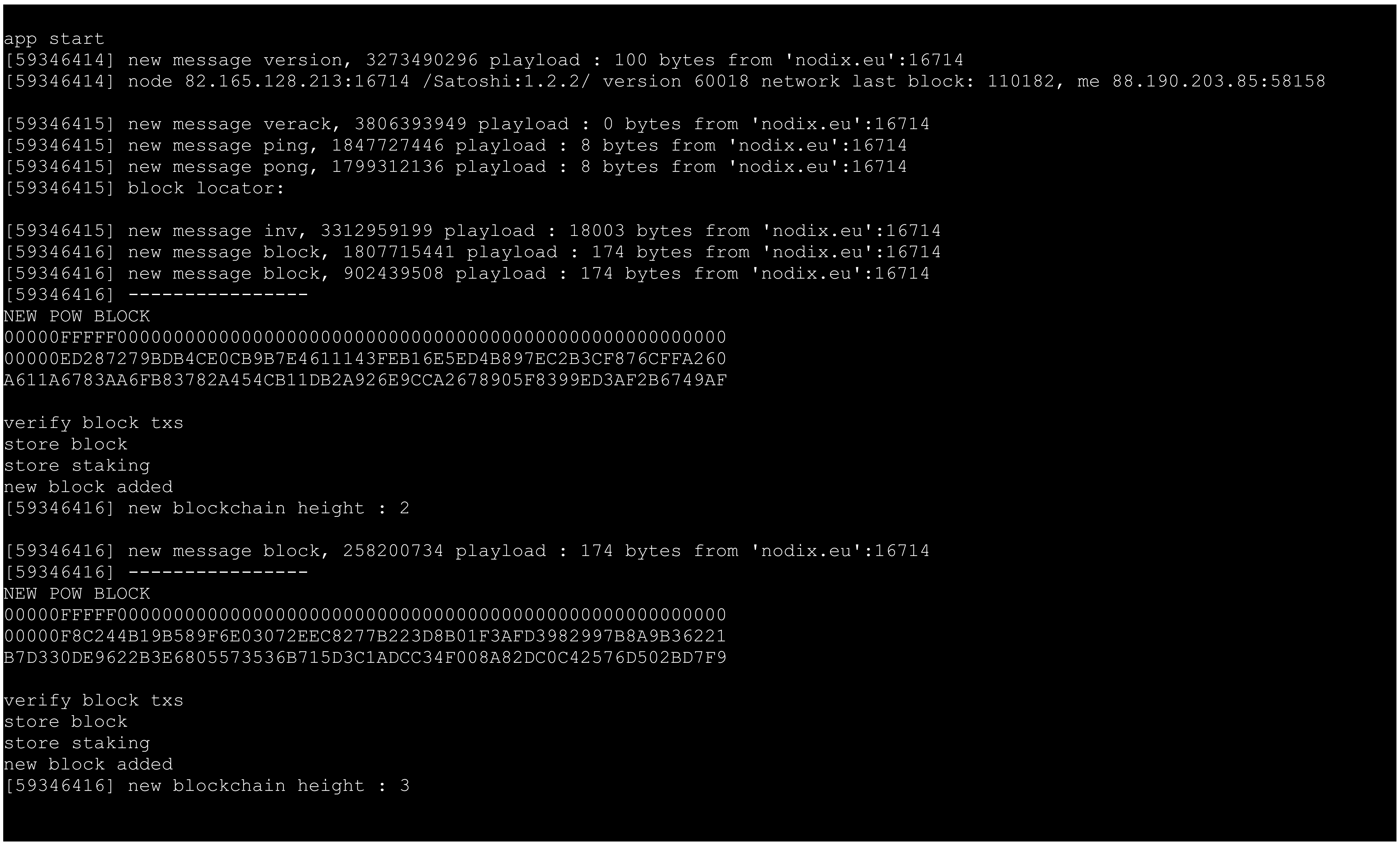
```
{"base" : "/api/"   , "type" : "cgi", (NODE_MODULE_DEF) "block_explorer" : {"file":"modz/block_explorer.tpo"}}
```

which can be called via http://node-ip:port/api/function_name

Function executed via JSON/RPC or CGI interface are always executed in a thread parrallelized from the main node thread and from each other.

The lockless architecture allow to manipulate object hierarchy in a thread safe manner transparently.

2.4 Logging and output



```
app start
[59346414] new message version, 3273490296 playload : 100 bytes from 'nodix.eu':16714
[59346414] node 82.165.128.213:16714 /Satoshi:1.2.2/ version 60018 network last block: 110182, me 88.190.203.85:58158

[59346415] new message verack, 3806393949 playload : 0 bytes from 'nodix.eu':16714
[59346415] new message ping, 1847727446 playload : 8 bytes from 'nodix.eu':16714
[59346415] new message pong, 1799312136 playload : 8 bytes from 'nodix.eu':16714
[59346415] block locator:

[59346415] new message inv, 3312959199 playload : 18003 bytes from 'nodix.eu':16714
[59346416] new message block, 1807715441 playload : 174 bytes from 'nodix.eu':16714
[59346416] new message block, 902439508 playload : 174 bytes from 'nodix.eu':16714
[59346416] ----------------
NEW POW BLOCK
00000FFFFF000000000000000000000000000000000000000000000000000000
00000ED287279BDB4CE0CB9B7E4611143FEB16E5ED4B897EC2B3CF876CFFA260
A611A6783AA6FB83782A454CB11DB2A926E9CCA2678905F8399ED3AF2B6749AF

verify block txs
store block
store staking
new block added
[59346416] new blockchain height : 2

[59346416] new message block, 258200734 playload : 174 bytes from 'nodix.eu':16714
[59346416] ----------------
NEW POW BLOCK
00000FFFFF000000000000000000000000000000000000000000000000000000
00000F8C244B19B589F6E03072EEC8277B223D8B01F3AFD3982997B8A9B36221
B7D330DE9622B3E6805573536B715D3C1ADCC34F008A82DC0C42576D502BD7F9

verify block txs
store block
store staking
new block added
[59346416] new blockchain height : 3
```

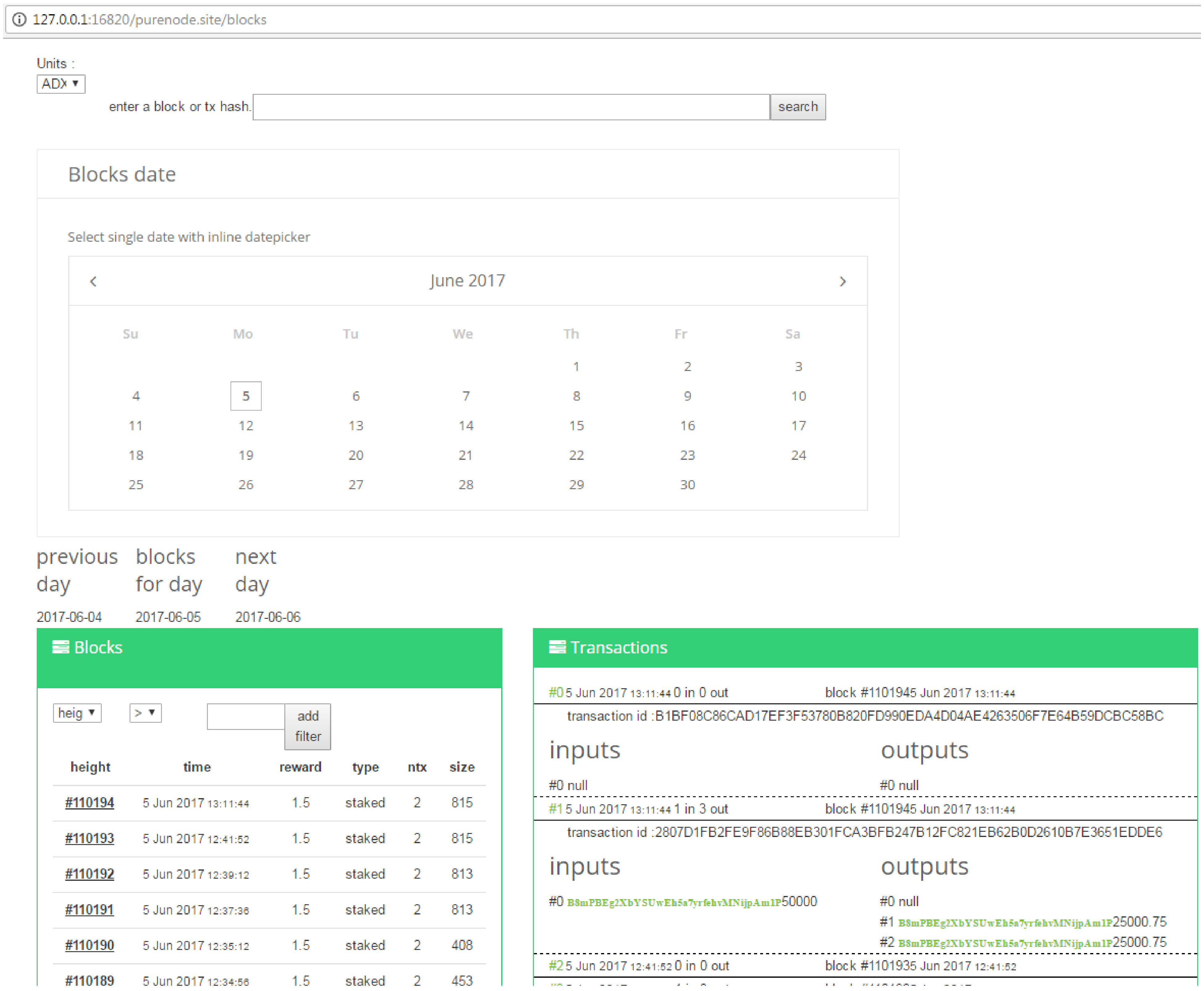
Nodix framework allow dumping of any dynamic object safely without using C/C++ stdio or streams, using it's own formatting of message from dynamic object definition, which allow easy debugging and logging of complex object hierarchy without running into unsafe text stream function of C runtime.

2.5 HTML5 block explorer

Blockchain nodes usually does not contain a block explorer, and it's generally another software bundle using its own database and node.js service, with nodix, the block explorer database and api is fully integrated in the node as an application service.

List of blocks by date and search engine / filtering

http://nodix.eu:16820/nodix.site/blocks


127.0.0.1:16820/purenode.site/blocks

Units : ADX

enter a block or tx hash. search

Blocks date

Select single date with inline datepicker

June 2017

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

previous day	blocks for day	next day
2017-06-04	2017-06-05	2017-06-06

Blocks

heig > add filter

height	time	reward	type	ntx	size
#110194	5 Jun 2017 13:11:44	1.5	staked	2	815
#110193	5 Jun 2017 12:41:52	1.5	staked	2	815
#110192	5 Jun 2017 12:39:12	1.5	staked	2	813
#110191	5 Jun 2017 12:37:36	1.5	staked	2	813
#110190	5 Jun 2017 12:35:12	1.5	staked	2	408
#110189	5 Jun 2017 12:34:56	1.5	staked	2	453

Transactions

#0 5 Jun 2017 13:11:44 0 in 0 out block #1101945 Jun 2017 13:11:44

transaction id :B1BF08C86CAD17EF3F53780B820FD990EDA4D04AE4263506F7E64B59DCBC58BC

inputs	outputs
#0 null	#0 null

#1 5 Jun 2017 13:11:44 1 in 3 out block #1101945 Jun 2017 13:11:44

transaction id :2807D1FB2FE9F86B88EB301FCA3BFB247B12FC821EB62B0D2610B7E3651EDDE6

inputs	outputs
#0 B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P50000	#0 null
	#1 B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P25000.75
	#2 B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P25000.75

#2 5 Jun 2017 12:41:52 0 in 0 out block #1101935 Jun 2017 12:41:52

http://nodix.eu:16820/api/blocks?BlockDate=2017-06-05

=>

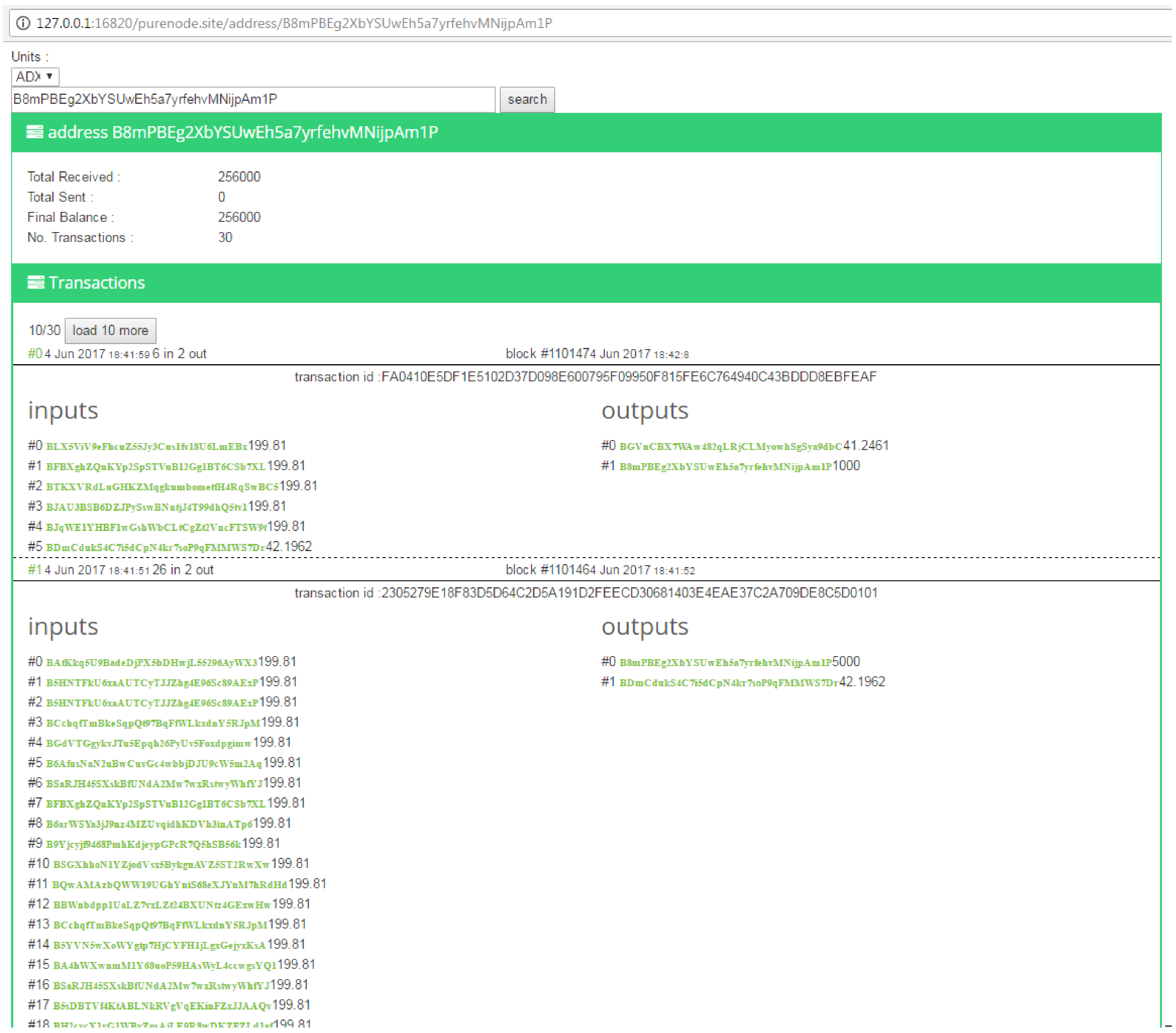
```
blocks:[,…]
   0:{reward: 150000000, stakemodifier2: "E44472EEAF3FD27715F048112D7716EA63308B96A81E2F3A6C82BF46BD141A76",…}
   bits:440209823
   confirmations:294
   difficulty:274722.125
   hash:"D86A837ACF95DA6BB9E6CBB0B69E094B30B6005ED72FD1CC6BEF600E810F1546"
   hbits:"0000000000003D119F00000000000000000000000000000000000000000000000"
   height:110221
   isCoinbase:false
   merkleroot:"7E8441A68E35753A7073103A8D6D679AAF42BACE179DF89D4EA531833347845D"
   nextblockhash:"D3D0BCB3B072F9D40AD85070DD31EC3BC9DEBAB21CCA4AB1879A66B7EFB4873F"
   nonce:0
   previousblockhash:"49FBB5C45C15B6C53AF5B2F8BAD595F71B379E1BF1FB3FA4BDE9CF97D371070B"
   proofhash:"000041F0292E0C63E1ABFA35FCAAFDA211FE7622E51F8F574151B5B3B98E256F"
   reward:150000000
   size:330
   stakemodifier2:"E44472EEAF3FD27715F048112D7716EA63308B96A81E2F3A6C82BF46BD141A76"
   time:1496670496
   tx:["4004A0841A2E4B8CD437766215A58EF3DCBEFAAEAD4968F4DB765D4B8D1250B5",…]
   0:"4004A0841A2E4B8CD437766215A58EF3DCBEFAAEAD4968F4DB765D4B8D1250B5"
   1:"4D499776222C39D2951B317AD1FECA1F2E3B0FB5959AD0AFD3842FE292C54D01"
   version:7
```

http://nodix.eu:16820/api/txs?BlockDate=2017-06-05&pageNum=0

=>

```
   txs:[{blockhash: "D86A837ACF95DA6BB9E6CBB0B69E094B30B6005ED72FD1CC6BEF600E810F1546",…},…]
   0:{blockhash: "D86A837ACF95DA6BB9E6CBB0B69E094B30B6005ED72FD1CC6BEF600E810F1546",…}
   1:{blockhash: "D86A837ACF95DA6BB9E6CBB0B69E094B30B6005ED72FD1CC6BEF600E810F1546",…}
   blockhash:"D86A837ACF95DA6BB9E6CBB0B69E094B30B6005ED72FD1CC6BEF600E810F1546"
   blockheight:110221
   blocktime:1508708742
   confirmations:294
   isCoinBase:false
   size:181
   time:1496670496
   txid:"4D499776222C39D2951B317AD1FECA1F2E3B0FB5959AD0AFD3842FE292C54D01"
        vin:[{value: 10066000000, addresses: ["BRcMT11Hfymgf5hViqiRg12zwSyLoSPibG"],…}]
        0:{value: 10066000000, addresses: ["BRcMT11Hfymgf5hViqiRg12zwSyLoSPibG"],…}
        addresses:["BRcMT11Hfymgf5hViqiRg12zwSyLoSPibG"]
        0:"BRcMT11Hfymgf5hViqiRg12zwSyLoSPibG"
        idx:2
        n:0
        prevhash:"D82713FE8E0F3C06D0A49F694658439EC933A849A948BEDDE755989258A62270"
        sequence:4294967295
        value:10066000000
   vout:
        0:{value: 0, n: 0, isNull: true}
        1:{value: 10216000000, n: 1, isNull: false,…}
```

Transaction by address



127.0.0.1:16820/purenode.site/address/B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P

Units :
ADX
B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P search

address B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P

Total Received :	256000
Total Sent :	0
Final Balance :	256000
No. Transactions :	30

Transactions

10/30 load 10 more

#0 4 Jun 2017 18:41:59 6 in 2 out block #1101474 Jun 2017 18:42:8

transaction id :FA0410E5DF1E5102D37D098E600795F09950F815FE6C764940C43BDDD8EBFEAF

inputs

#0 BLX5ViV9eFhcuZ55Jy3Cus1fv18U6LmEBx199.81
#1 BFBXghZQnKYp2SpSTVuB12Gg1BT6CSb7XL199.81
#2 BTKXVRdLuGHKZMqgkumbometfH4RqSwBC5199.81
#3 BJAU3BSB6DZJPySswBNutjJ4T99dhQ5tv1199.81
#4 BJqWE1YHBF1wGshWbCLtCgZt2VncFTSW9t199.81
#5 BDmCdukS4C7i5dCpN4kr7soP9qFMMWS7Dr42.1962

outputs

#0 BGVnCBX7WAw482qLRjCLMyowhSgSya9dbC41.2461
#1 B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P1000

#1 4 Jun 2017 18:41:51 26 in 2 out block #1101464 Jun 2017 18:41:52

transaction id :2305279E18F83D5D64C2D5A191D2FEECD30681403E4EAE37C2A709DE8C5D0101

inputs

#0 BAtKkq5U9BadeDjPX5bDHwjL55296AyWX3199.81
#1 B5HNTFkU6xaAUTCyTJJZhg4E96Sc89AExP199.81
#2 B5HNTFkU6xaAUTCyTJJZhg4E96Sc89AExP199.81
#3 BCchqfTmBkeSqpQt97BqFfWLkxdnY5RJpM199.81
#4 BGdVTGgykvJTu5Epqh26PyUv5Foxdpgimw199.81
#5 B6AfusNaN2uBwCuvGc4wbbjDJU9cW5m2Aq199.81
#6 BSaRJH45SXskBfUNdA2Mw7wxRstwyWhfYJ199.81
#7 BFBXghZQnKYp2SpSTVuB12Gg1BT6CSb7XL199.81
#8 B6arWSYa3jJ9nz4MZUvqidhKDVh3inATp6199.81
#9 B9Yjcyjf9468PmhKdjeypGPcR7Q5hSB56k199.81
#10 BSGXhhoN1YZjodVsx5BykguAVZ5ST2RwXw199.81
#11 BQwAMAzbQWW19UGhYniS68eXJYuM7hRdHd199.81
#12 BBWnbdpp1UaLZ7vxLZt24BXUNtz4GExwHw199.81
#13 BCchqfTmBkeSqpQt97BqFfWLkxdnY5RJpM199.81
#14 B5YVN5wXoWYgtp7HjCYFH1jLgxGejyxKsA199.81
#15 BA4hWXwnmM1Y68uoP59HAsWyL4ccwgsYQ1199.81
#16 BSaRJH45SXskBfUNdA2Mw7wxRstwyWhfYJ199.81
#17 B5sDBTVf4KtABLNkRVgVqEKinFZxJJAAQv199.81
#18 [illegible]199.81

outputs

#0 B8mPBEg2XbYSUwEh5a7yrfehvMNijpAm1P5000
#1 BDmCdukS4C7i5dCpN4kr7soP9qFMMWS7Dr42.1962

http://nodix.eu:16820/api/txs?address=BRcMT11Hfymgf5hViqiRg12zwSyLoSPibG&pageNum=0

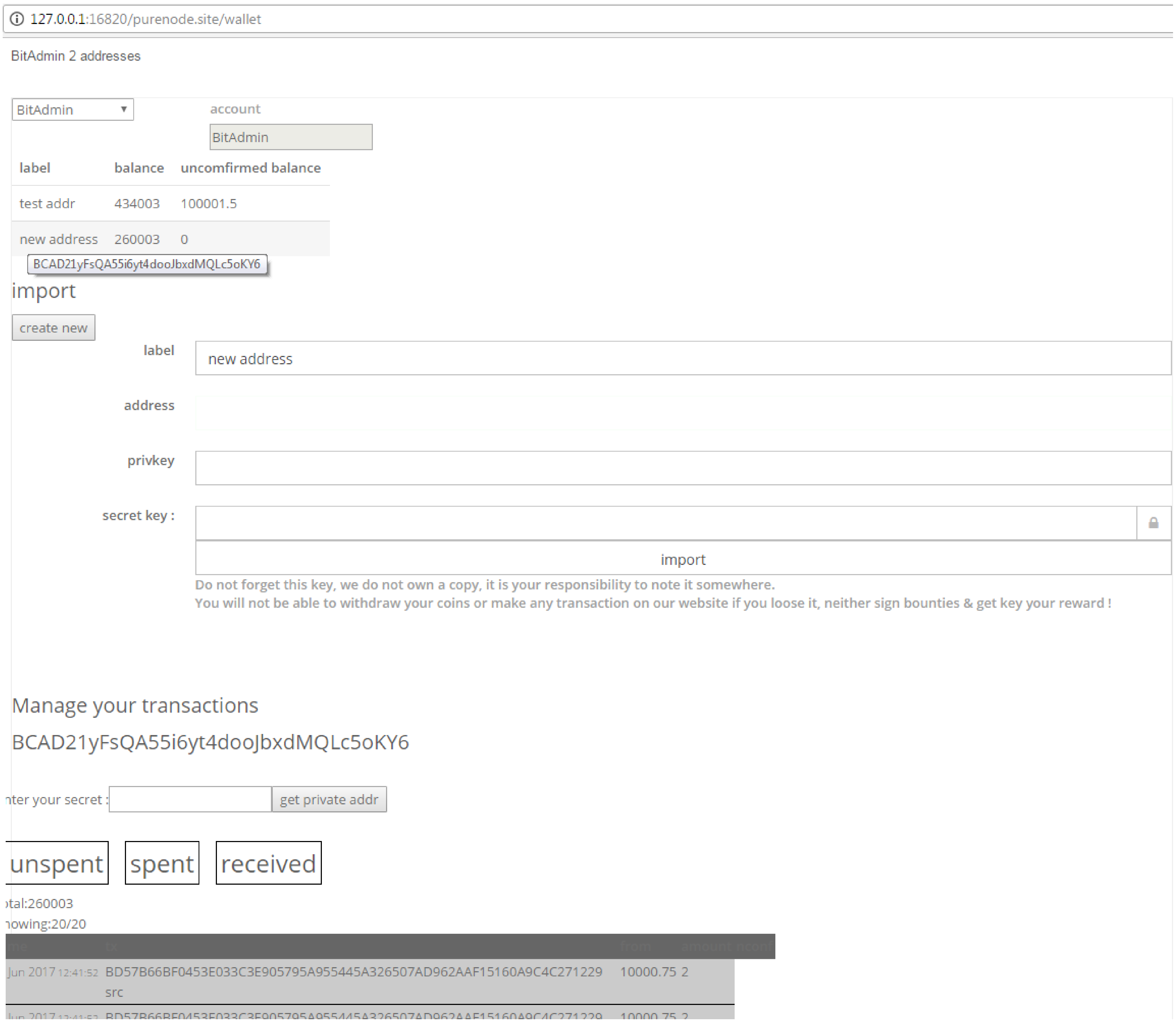
```
numtx:1402
txs:
0:
blockhash:"37531A82E21ADA1B310180851DAD72A09A36C501F9F2BD8C7EC7AC306895AC41"
blockheight:98
blocktime:1508671270
confirmations:110417
isCoinBase:true
size:93
time:1467221980
txid:"796ACDF1ACB361930E64D965FF7275044B87F9703DDE45475CF4CFD818D93DD8"
vin:[{coinbase: "01610103", n: 0, sequence: 4294967295}]
0:{coinbase: "01610103", n: 0, sequence: 4294967295}
coinbase:"01610103"
n:0
sequence:4294967295
vout:[{value: 10000000000000, n: 0, isNull: false,…}]
0:{value: 10000000000000, n: 0, isNull: false,…}
addresses:["BRcMT11Hfymgf5hViqiRg12zwSyLoSPibG"]
isNull:false
n:0
scriptPubKey:{hex: "76A914E81B99F7184A682459FABF6548C14B2989CFEC6B88AC", type: "paytoscript"}
value:10000000000000
```

2.6 HTML – javascript wallet

The node also include the RPCAPI for an HTML5/js wallet able to generate private key and sign transaction using javascript code, as well as staking in the browser, allowing maximum privacy, as the private key never leave the browser in clear form.

It's decrypted in the browser using user supplied secret, and then these private key are used to sign transaction right inside the browser.

Nodes never have to manipulate or access private key, but still keep track of transaction on public addresses, allowing read-only wallet, and manipulations of private key. Hash signing is made inside of the browser without the node knowing about the private key at all.



127.0.0.1:16820/purenode.site/wallet

BitAdmin 2 addresses

BitAdmin

account: BitAdmin

label	balance	uncomfirmed balance
test addr	434003	100001.5
new address	260003	0

BCAD21yFsQA55i6yt4dooJbxdMQLc5oKY6

import

create new

label	new address
address	
privkey	
secret key :	

import

Do not forget this key, we do not own a copy, it is your responsibility to note it somewhere.
You will not be able to withdraw your coins or make any transaction on our website if you loose it, neither sign bounties & get key your reward !

Manage your transactions

BCAD21yFsQA55i6yt4dooJbxdMQLc5oKY6

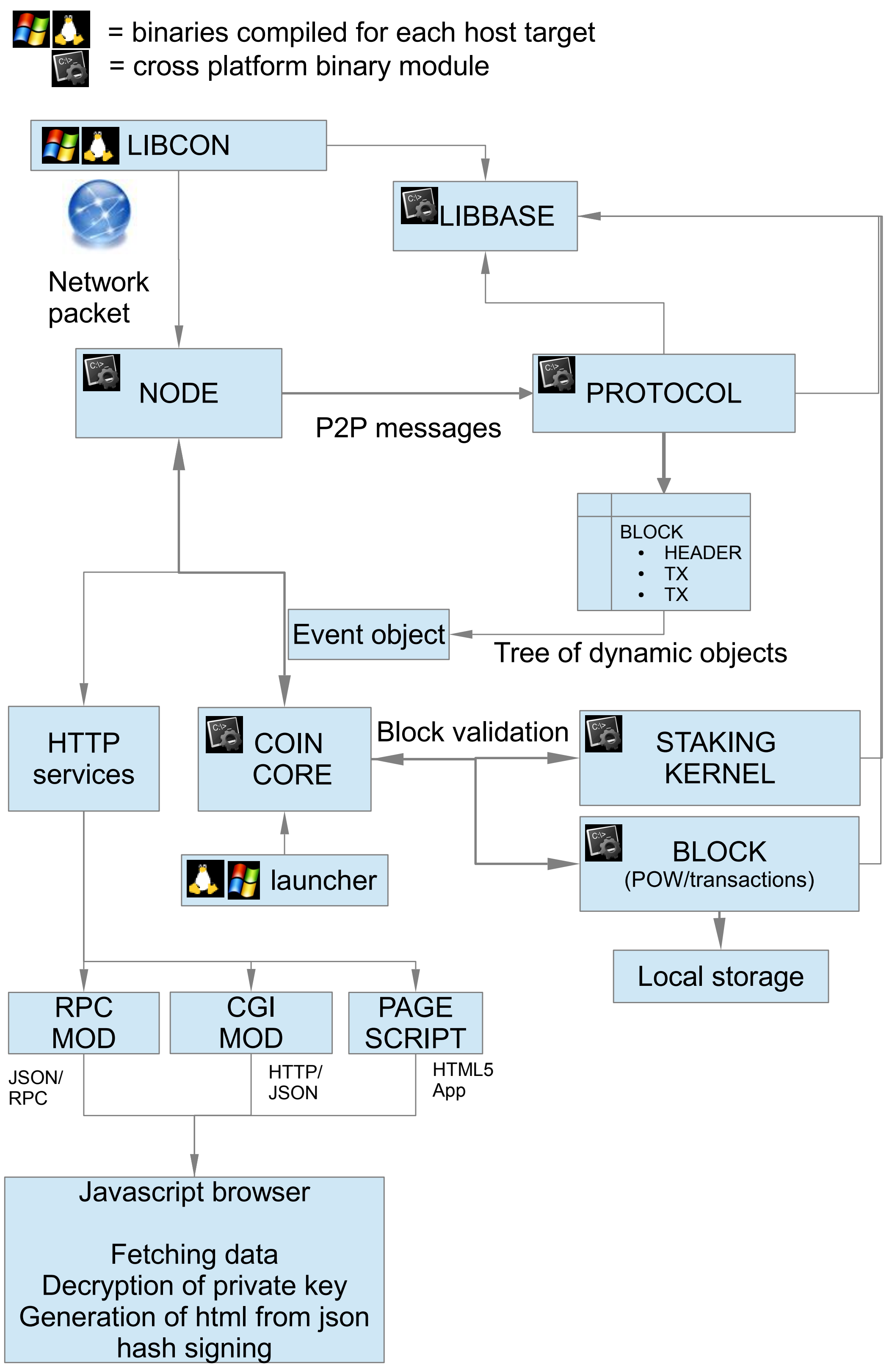
nter your secret : get private addr

unspent spent received

otal:260003
nowing:20/20

me	tx	from	amount	nconf
Jun 2017 12:41:52	BD57B66BF0453E033C3E905795A955445A326507AD962AAF15160A9C4C271229 src	10000.75	2	
Jun 2017 12:41:52	BD57B66BF0453E033C3E905795A955445A326507AD962AAF15160A9C4C271229	10000.75	2	

3 Node architecture



3.1 LIBCON (has to be compiled for the host)

- Memory allocation of reference counted pointer and basic memory manipulation.
- Strings and basic text manipulation, utf8.
- File system access and I/O.
- Socket and network.
- XML Parser (expat) used by uPnp.
- Zlib for compression/decompression.
- Binary module loader.
- Base C runtime.

3.2 LIBBASE (distributed as system agnostic binary module)

- Dynamic object tree based on references pointer.
- HTTP protocol.
- JSON parser.
- Nodix script parser.
- UPNP.
- hashes (SHA256,MD5,RIP160)

3.3 PROTOCOL (distributed as source)

- Parsing of bitcore protocol message and deserialization as dynamic object tree.
- Creation of new messages using the bitcore protocol based on dynamic objects.

3.4 BLOCK (distributed as source)

- Function to check block's proof of work.
- Functions to manipulate and verify transactions.
- Function to store block and transaction data
- Function to manipulate applications items.

3.5 SIGNATURE (distributed as source)

- Functions to generate key pairs
- Functions to extract public key from private key
- Function to verify data signature

3.6 NODE(distributed as system agnostic binary module)

- Reception and emission of P2P network packet.
- Management of blockchain state.
- Loading of RPC and CGI module.
- Handling of HTTP json/ajax/rpc request.
- Handling of web page generation requests.

3.7 WALLET(distributed as system agnostic binary module)

- Storing of public and crypted private key.
- Listing and management of transaction history.
- Generation of staking information.

3.8 RPC WALLET(distributed as source)

- Function to retrieve information on addresses compatible with bitcore RPC API, without the transaction signing related methods which are decentralized to client HTML5 WebApp.

3.9 BLOCK EXPLORER(distributed as source)

- Block explorer web api functions to output json formatted data (called from javascript in the block explorer page).

3.10 Staking kernel (distributed as source)

- Validation of proof of stake block and staking reward.
- Storing of pos specific data on the local system.
- Computing proof of stake difficulty re-targeting.
- Generation of new proof of stake block template for staking.

3.11 COIN (distributed as source)

- Loading of configuration file.
- Initialization of the proof of stake kernel module.
- Initialization of the node module.
- Global coin logic and network message handling

3.12 Example of application initialization

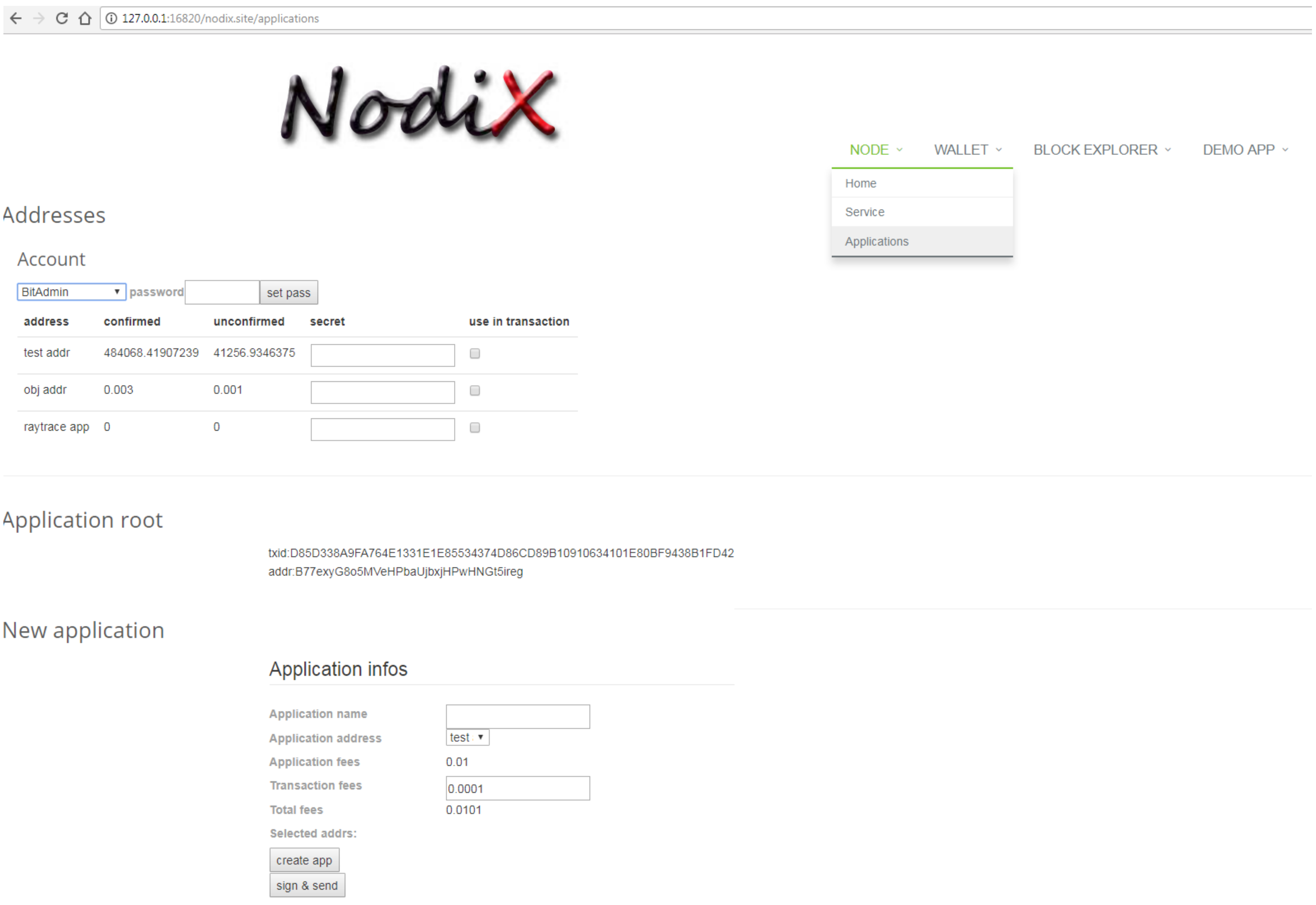
```
OS_API_C_FUNC(int) app_init(mem_zone_ref_ptr params)
{

       mem_zone_ref  log = { PTR_NULL };
       unsigned char *data;
       size_t        data_len;
       int           ret;

       pos_kernel = PTR_NULL;
       self_node.zone = PTR_NULL;
       node_config.zone = PTR_NULL;
       seed_node.zone = PTR_NULL;
       pos_kernel_def.zone = PTR_NULL;

       memset_c(null_hash, 0, 32);

       create_dir("txs");
       if (stat_file("txs") != 0)
       {
              log_message("unable to create tx dir \n", PTR_NULL);
              return 0;
       }

       create_dir("blks");
       if (stat_file("blks") != 0)
       {
              log_message("unable to create blks dir \n", PTR_NULL);
              return 0;
       }

       create_dir("adrs");
       if (stat_file("adrs") != 0)
       {
              log_message("unable to create adrs dir \n", PTR_NULL);
              return 0;
       }

       if (params != PTR_NULL)
       {
              mem_zone_ref  stake_mod_def = { PTR_NULL };
              ret=resolve_script_var            (params, PTR_NULL,"configuration"                        , NODE_GFX_OBJECT, &node_config);
              if (ret)ret = resolve_script_var  (params, PTR_NULL,"SelfNode"                             , NODE_BITCORE_NODE, &self_node);
              if (ret)ret = resolve_script_var  (params, PTR_NULL,"configuration.staking.pos_kernel"     , NODE_MODULE_DEF, &pos_kernel_def);
              if (ret)ret = resolve_script_var  (params, PTR_NULL,"configuration.seed_node"              , 0xFFFFFFFF, &seed_node);
              if(ret)node_set_script            (params);
       }

       if (!ret)
       {
              release_zone_ref(&seed_node);
              release_zone_ref(&node_config);
              release_zone_ref(&self_node);
       }
       return ret;
}
```

4 Distributed application framework

Nodix blockchain allow to represent full application on the blockchain, including object type definitions, objects data, files, layout and code, as a full MVC (Movel View Controller) stack.

First an application root has to be included on the blockchain, then application definitions can be included as child of this root application, and application elements added as child of the application definition.

The applications page in the html wallet can be used to create application root and new application entries.



Addresses

Account

BitAdmin password set pass

address	confirmed	unconfirmed	secret	use in transaction
test addr	484068.41907239	41256.9346375		☐
obj addr	0.003	0.001		☐
raytrace app	0	0		☐

Application root

txid:D85D338A9FA764E1331E1E85534374D86CD89B10910634101E80BF9438B1FD42
addr:B77exyG8o5MVeHPbaUjbxjHPwHNGt5ireg

New application

Application infos

Application name	
Application address	test
Application fees	0.01
Transaction fees	0.0001
Total fees	0.0101
Selected addrs:	

create app
sign & send

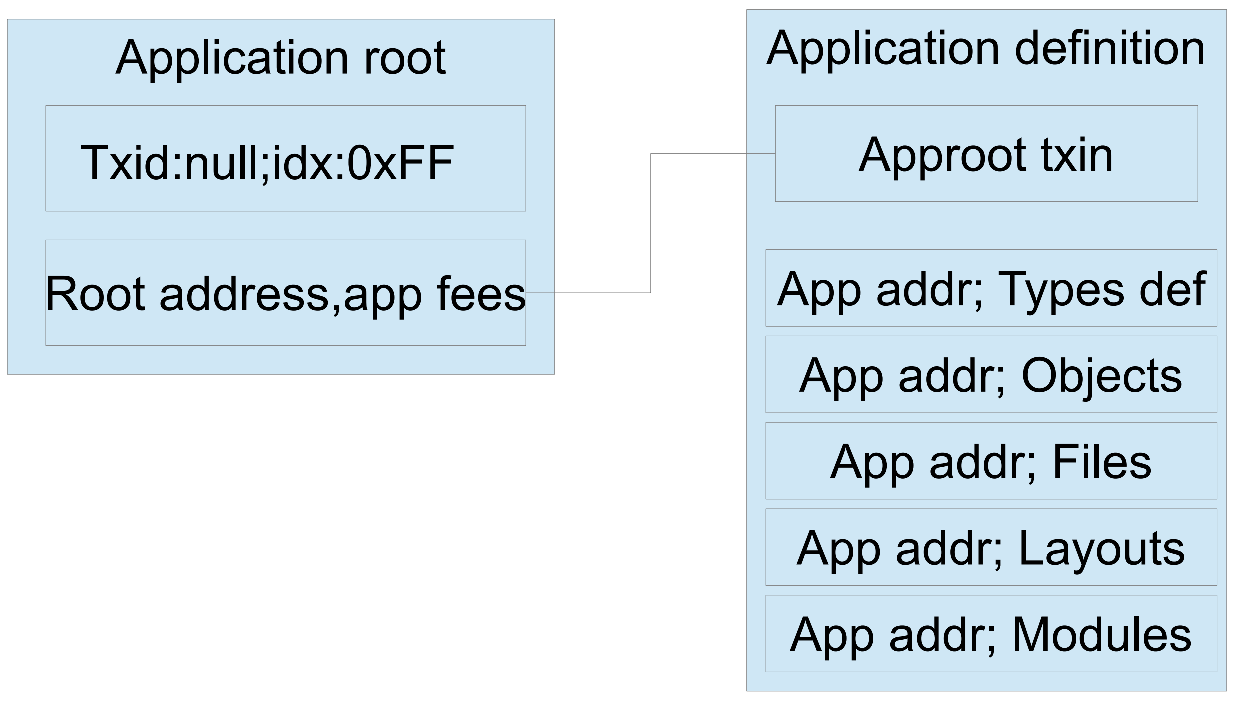
Transactions are first generated by the node from input parameters in the html UI, then send back to the UI with their signing hash. The javascript application then sign the transaction using the private key decrypted in the browser based on user supplied secret before being sent to the node's memory pool and propagated to the network.

4.1 Application root

The application root is the entry point for applications. Its first input contain null txid and idx, with the script containing a single text variable set to “approot”.

Its first output contain the application root address, and the amount of the utxo set the fee to pay to the app root address in order to add a new application.

4.2 Application definition



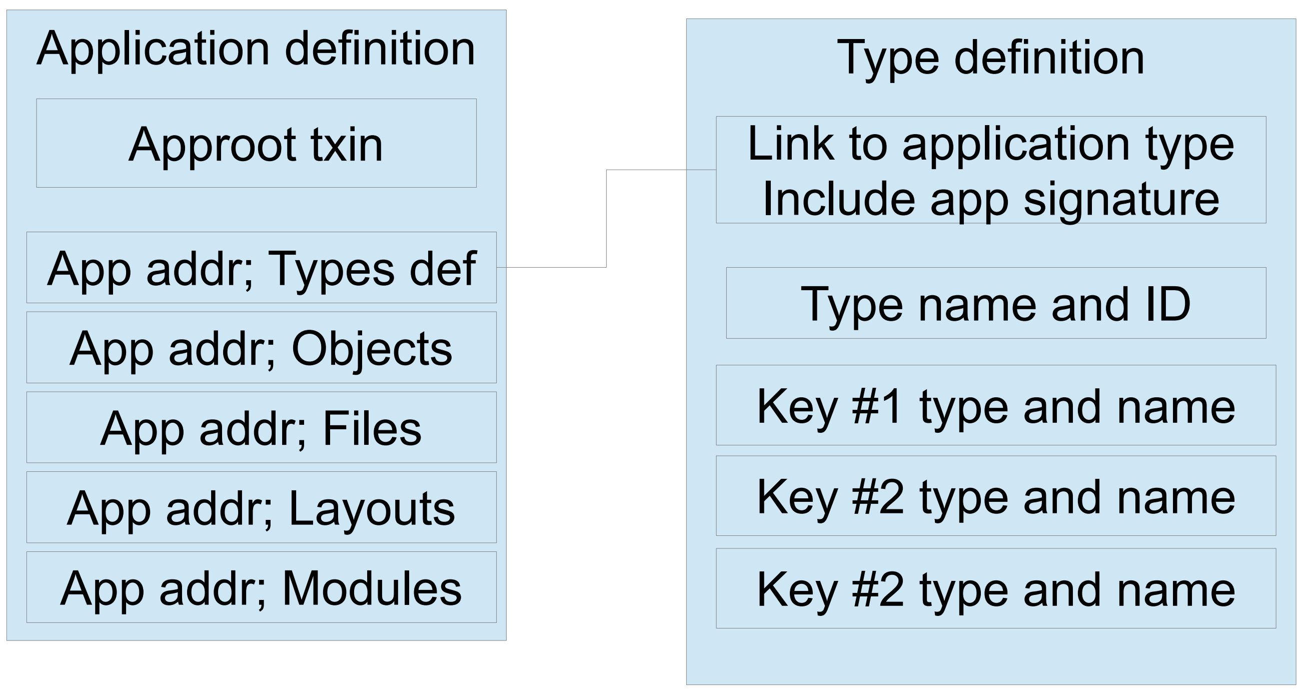
Application definitions are the entry point for applications. Their input link to the application root, and each output is used to add new entry in the application, containing the application address used to sign new entries to the application.

Additionally to the input linking to the application root and applications outputs, this transaction need to contain regular input and outputs that send the app fee to the address set in the application root.

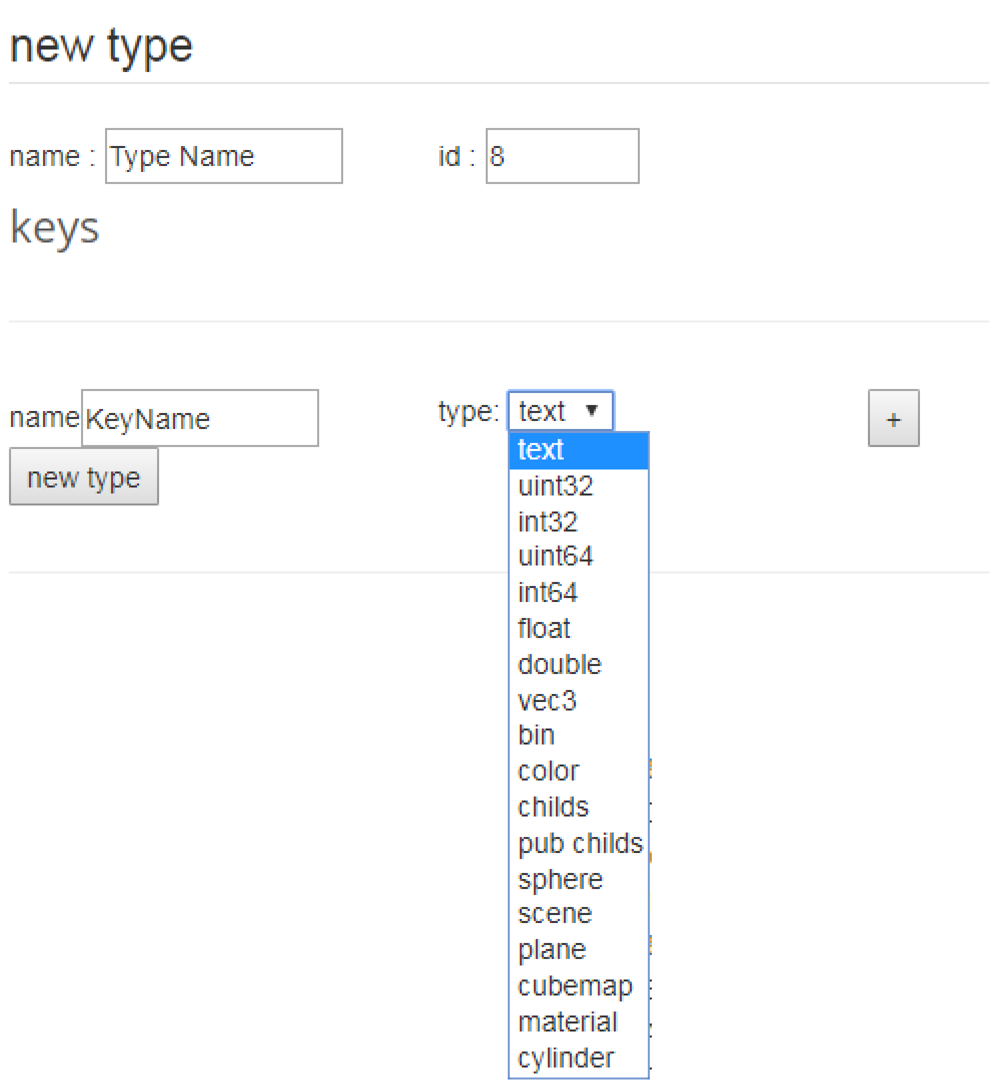
Once the application is created, it can be accessed via http://node-ip:port/application/**app_name**

4.3 Application types

Application types can be created on the application page. The address used to create the application need to be selected to add a new type.



Key's type can be built in base types, application type, references to other objects or files, or a list of private or public childs that can be added to the object.



new type

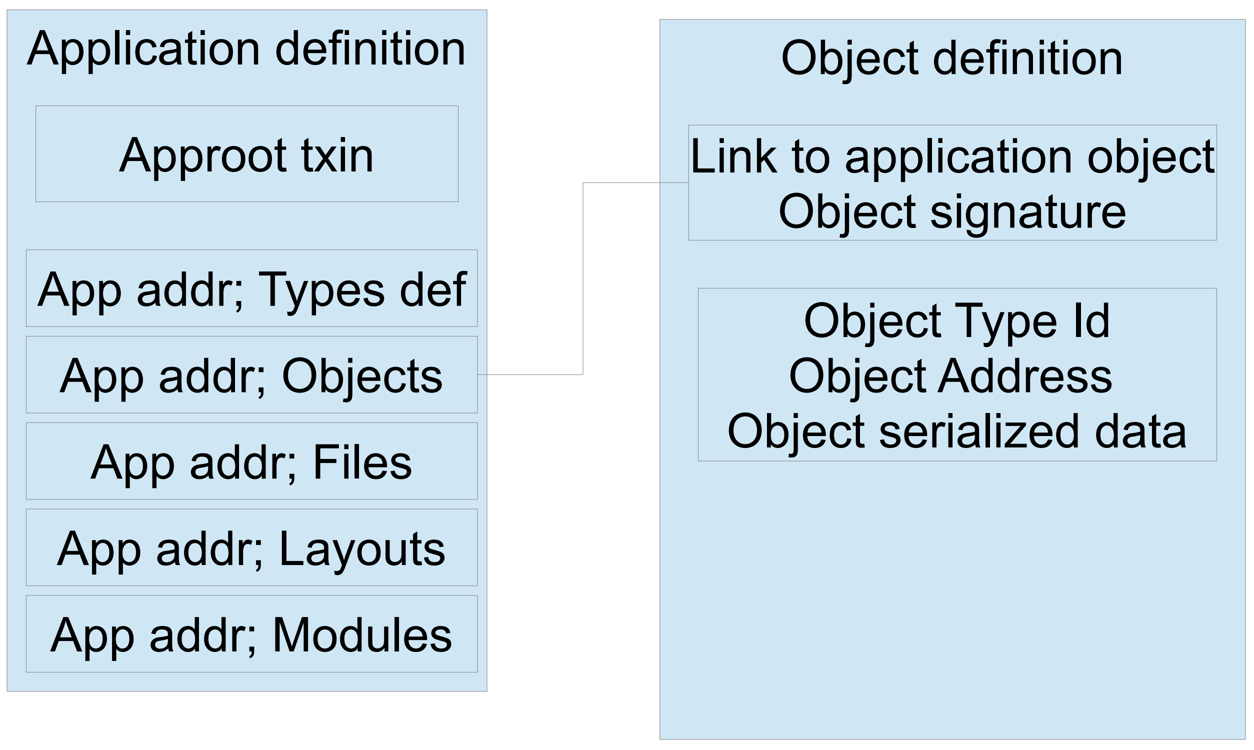
name : Type Name   id : 8

keys

name KeyName   type: text ▾   +

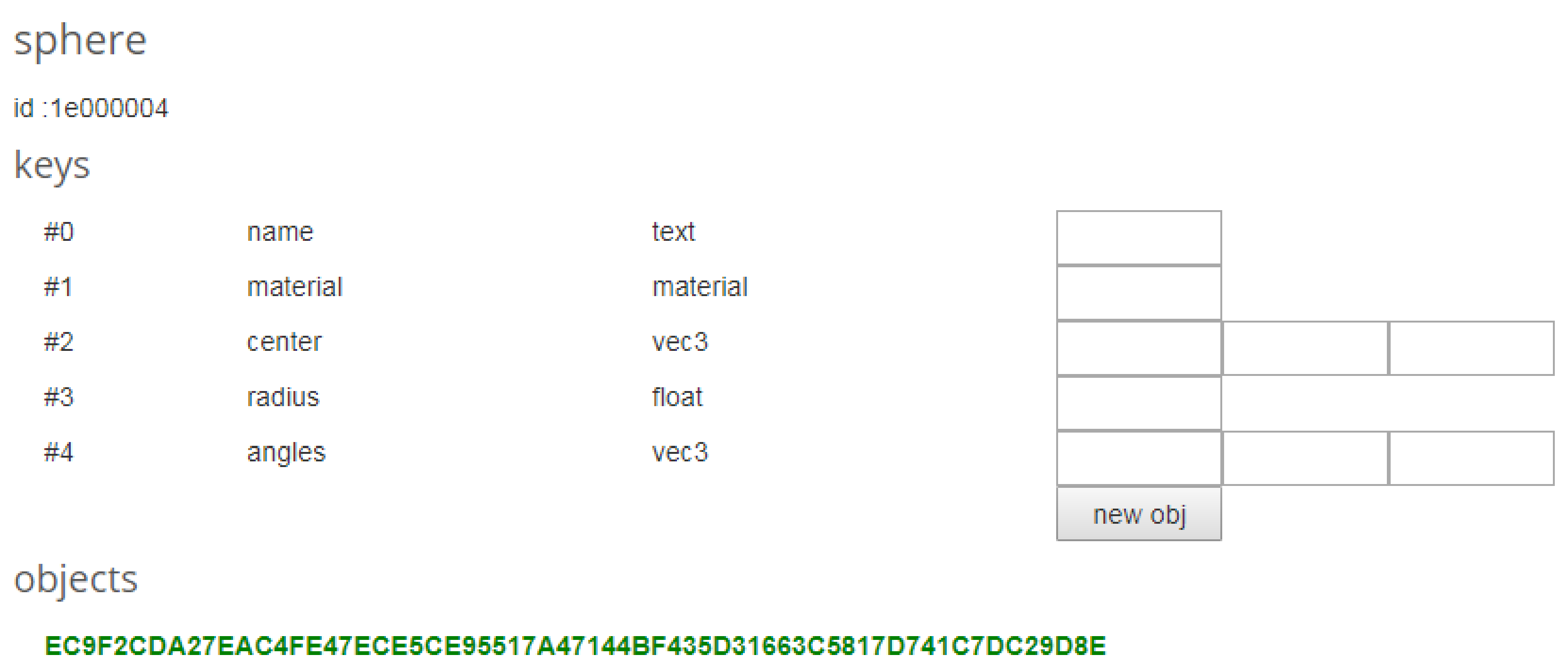
new type

4.4 Application objects



Once a type has been created, new objects can be created by filling the fields in the application page.

Objects are signed with user supplied address.



sphere

id :1e000004

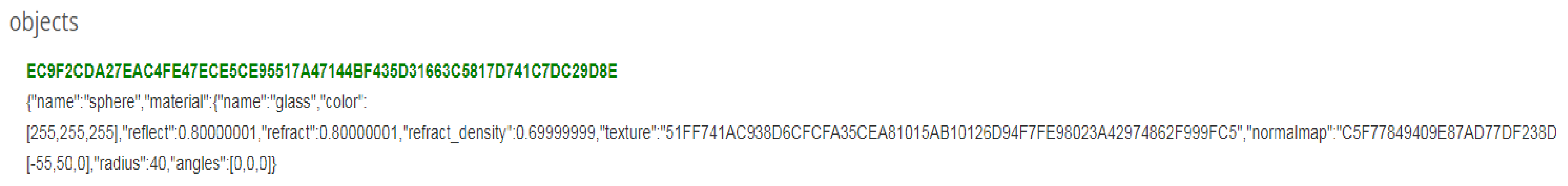
keys

#0	name	text	
#1	material	material	
#2	center	vec3	
#3	radius	float	
#4	angles	vec3	

objects

EC9F2CDA27EAC4FE47ECE5CE95517A47144BF435D31663C5817D741C7DC29D8E

Objects that have already been created for this application are listed in green below the type definition, and their data can be seen by clicking on the hash that correspond to the transaction that define the object.

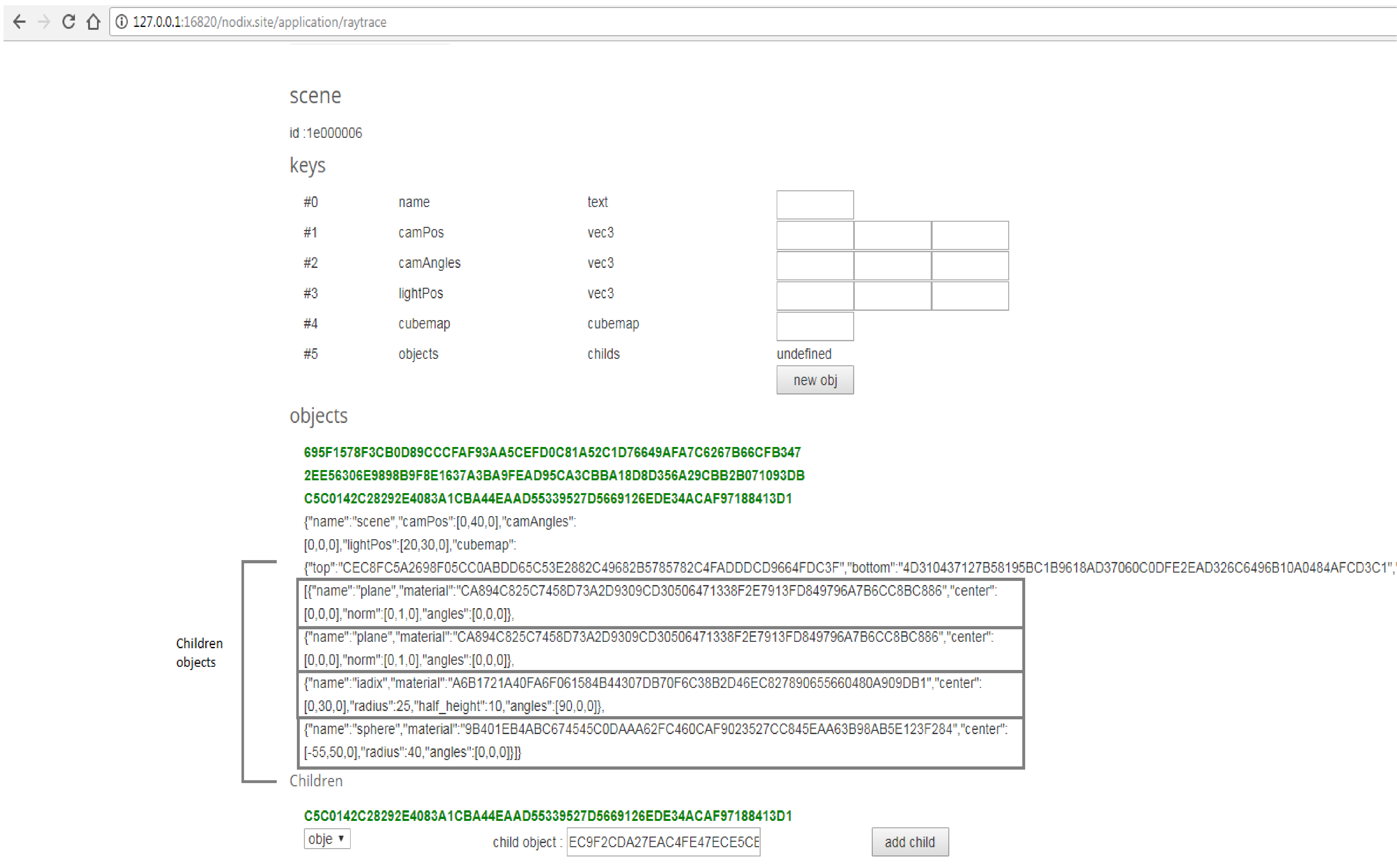


objects

EC9F2CDA27EAC4FE47ECE5CE95517A47144BF435D31663C5817D741C7DC29D8E

{"name":"sphere","material":{"name":"glass","color":
[255,255,255],"reflect":0.80000001,"refract":0.80000001,"refract_density":0.69999999,"texture":"51FF741AC938D6CFCFA35CEA81015AB10126D94F7FE98023A42974862F999FC5","normalmap":"C5F77849409E87AD77DF238D
[-55,50,0],"radius":40,"angles":[0,0,0]}

New children can be added to existing object that has a child key entry. If the child key is not public, the address used to create the object need to be selected in order to add a child to the object.



scene

id :1e000006

keys

#0	name	text	
#1	camPos	vec3	
#2	camAngles	vec3	
#3	lightPos	vec3	
#4	cubemap	cubemap	
#5	objects	childs	undefined

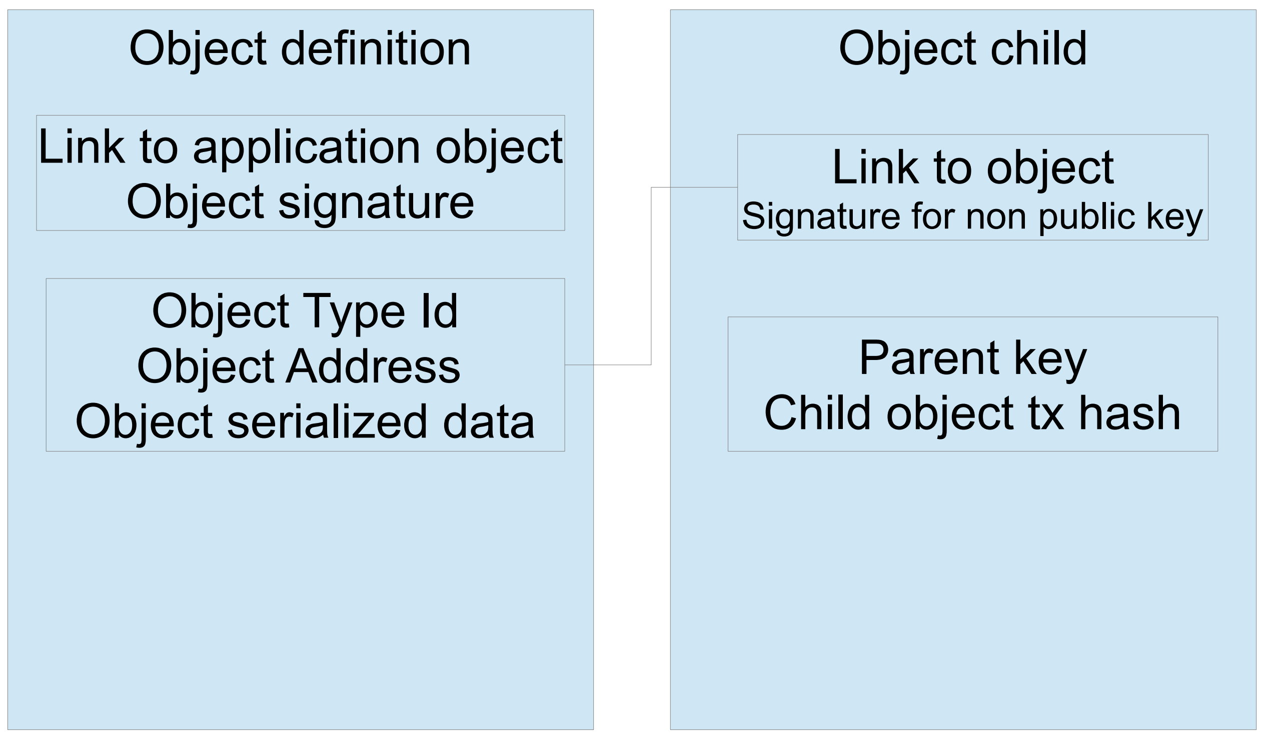
objects

695F1578F3CB0D89CCCFAF93AA5CEFD0C81A52C1D76649AFA7C6267B66CFB347
2EE56306E9898B9F8E1637A3BA9FEAD95CA3CBBA18D8D356A29CBB2B071093DB
C5C0142C28292E4083A1CBA44EAAD55339527D5669126EDE34ACAF97188413D1

{"name":"scene","camPos":[0,40,0],"camAngles":
[0,0,0],"lightPos":[20,30,0],"cubemap":
{"top":"CEC8FC5A2698F05CC0ABDD65C53E2882C49682B5785782C4FADDDCD9664FDC3F","bottom":"4D310437127B58195BC1B9618AD37060C0DFE2EAD326C6496B10A0484AFCD3C1",
[{"name":"plane","material":"CA894C825C7458D73A2D9309CD30506471338F2E7913FD849796A7B6CC8BC886","center":
[0,0,0],"norm":[0,1,0],"angles":[0,0,0]},
{"name":"plane","material":"CA894C825C7458D73A2D9309CD30506471338F2E7913FD849796A7B6CC8BC886","center":
[0,0,0],"norm":[0,1,0],"angles":[0,0,0]},
{"name":"iadix","material":"A6B1721A40FA6F061584B44307DB70F6C38B2D46EC827890655660480A909DB1","center":
[0,30,0],"radius":25,"half_height":10,"angles":[90,0,0]},
{"name":"sphere","material":"9B401EB4ABC674545C0DAAA62FC460CAF9023527CC845EAA63B98AB5E123F284","center":
[-55,50,0],"radius":40,"angles":[0,0,0]}]}

Children

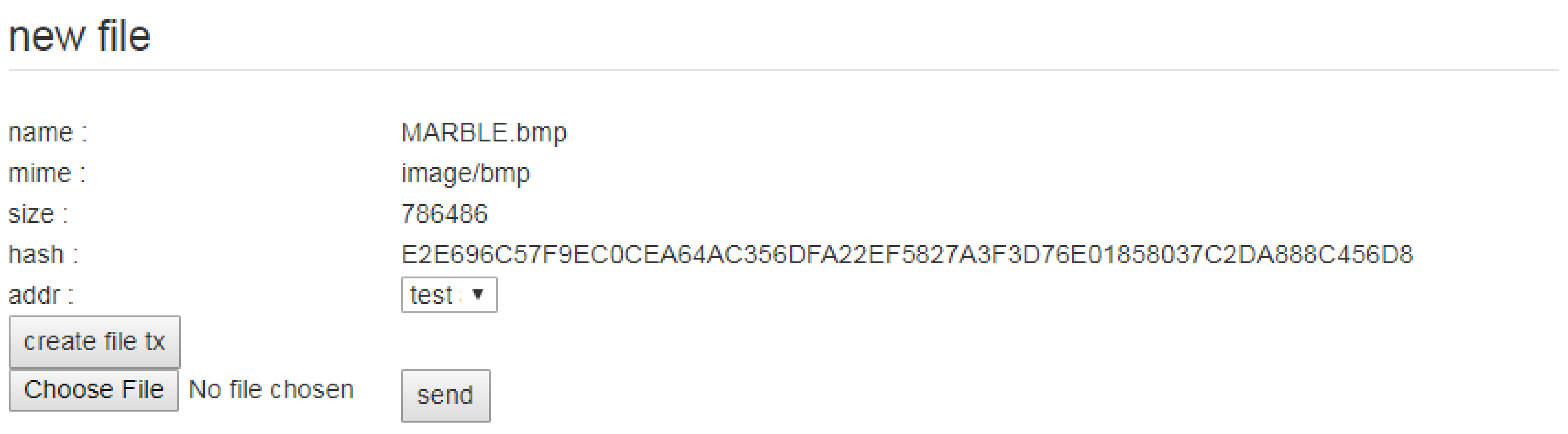
C5C0142C28292E4083A1CBA44EAAD55339527D5669126EDE34ACAF97188413D1



Unlike SQL database, the object hierarchy allow to access easily all children added to an object key without having to parse all objects of that type, and load a whole object hierarchy with a single command.

4.5 Application files

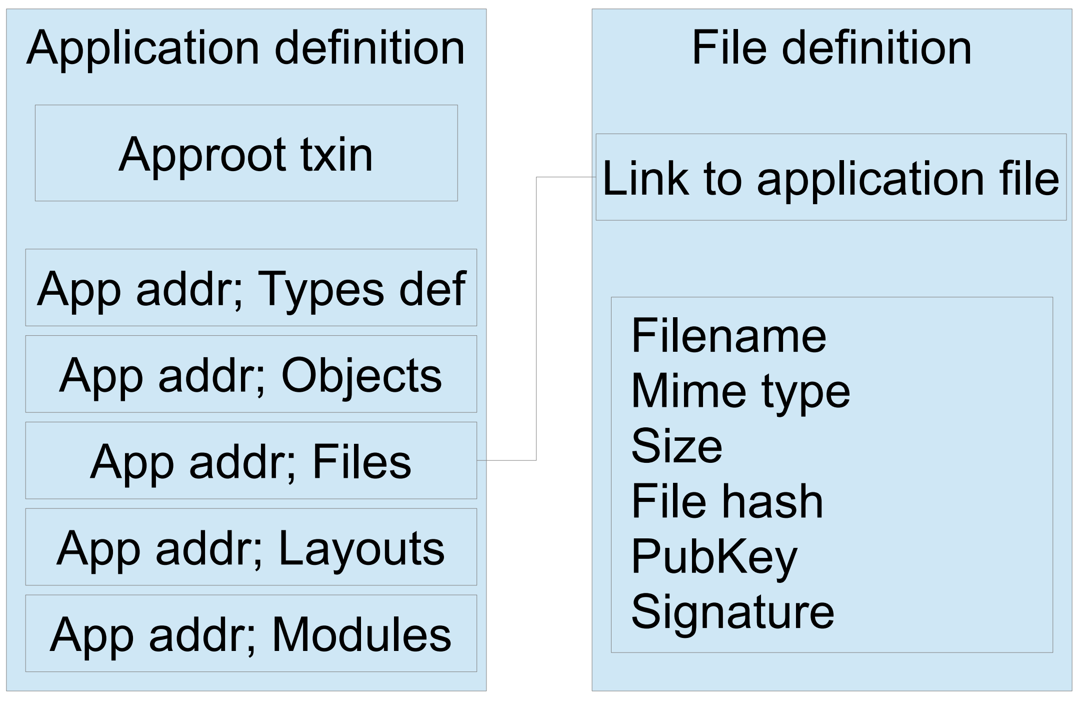
New files can be uploaded using the application page. First select the file using the file box, it will be then be uploaded as a temporary file, then select the address used to sign the file and create the transaction to store the file permanently on the node.



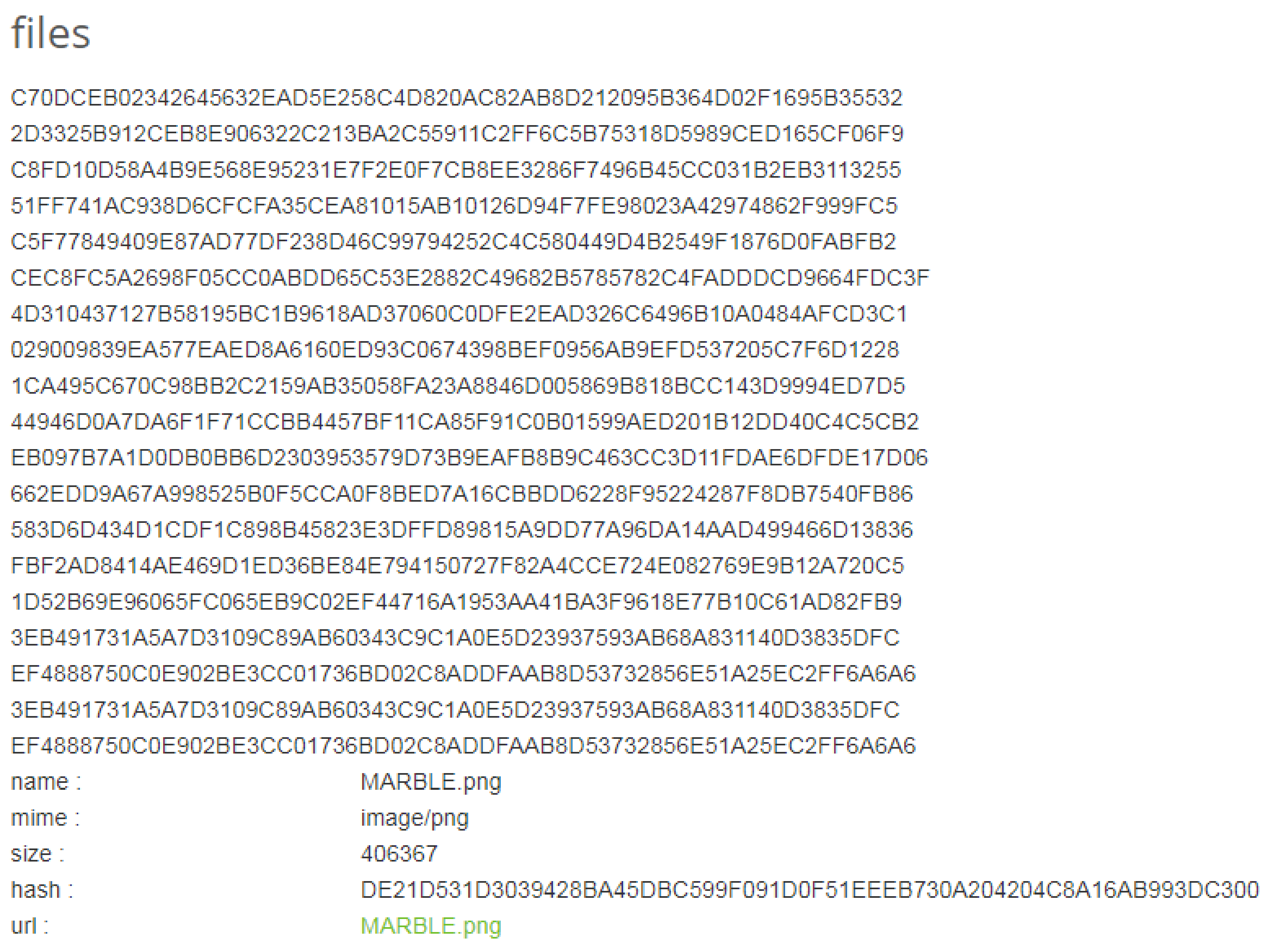
new file

name :	MARBLE.bmp
mime :	image/bmp
size :	786486
hash :	E2E696C57F9EC0CEA64AC356DFA22EF5827A3F3D76E01858037C2DA888C456D8
addr :	test

Only meta information about the file are stored on the blockchain, and a special protocol message combined with a getdata message are used to propagate the file based on this definition.



Files stored on the node can be viewed on this same application page.



files

```
C70DCEB02342645632EAD5E258C4D820AC82AB8D212095B364D02F1695B35532
2D3325B912CEB8E906322C213BA2C55911C2FF6C5B75318D5989CED165CF06F9
C8FD10D58A4B9E568E95231E7F2E0F7CB8EE3286F7496B45CC031B2EB3113255
51FF741AC938D6CFCFA35CEA81015AB10126D94F7FE98023A42974862F999FC5
C5F77849409E87AD77DF238D46C99794252C4C580449D4B2549F1876D0FABFB2
CEC8FC5A2698F05CC0ABDD65C53E2882C49682B5785782C4FADDDCD9664FDC3F
4D310437127B58195BC1B9618AD37060C0DFE2EAD326C6496B10A0484AFCD3C1
029009839EA577EAED8A6160ED93C0674398BEF0956AB9EFD537205C7F6D1228
1CA495C670C98BB2C2159AB35058FA23A8846D005869B818BCC143D9994ED7D5
44946D0A7DA6F1F71CCBB4457BF11CA85F91C0B01599AED201B12DD40C4C5CB2
EB097B7A1D0DB0BB6D2303953579D73B9EAFB8B9C463CC3D11FDAE6DFDE17D06
662EDD9A67A998525B0F5CCA0F8BED7A16CBBDD6228F95224287F8DB7540FB86
583D6D434D1CDF1C898B45823E3DFFD89815A9DD77A96DA14AAD499466D13836
FBF2AD8414AE469D1ED36BE84E794150727F82A4CCE724E082769E9B12A720C5
1D52B69E96065FC065EB9C02EF44716A1953AA41BA3F9618E77B10C61AD82FB9
3EB491731A5A7D3109C89AB60343C9C1A0E5D23937593AB68A831140D3835DFC
EF4888750C0E902BE3CC01736BD02C8ADDFAAB8D53732856E51A25EC2FF6A6A6
3EB491731A5A7D3109C89AB60343C9C1A0E5D23937593AB68A831140D3835DFC
EF4888750C0E902BE3CC01736BD02C8ADDFAAB8D53732856E51A25EC2FF6A6A6
```

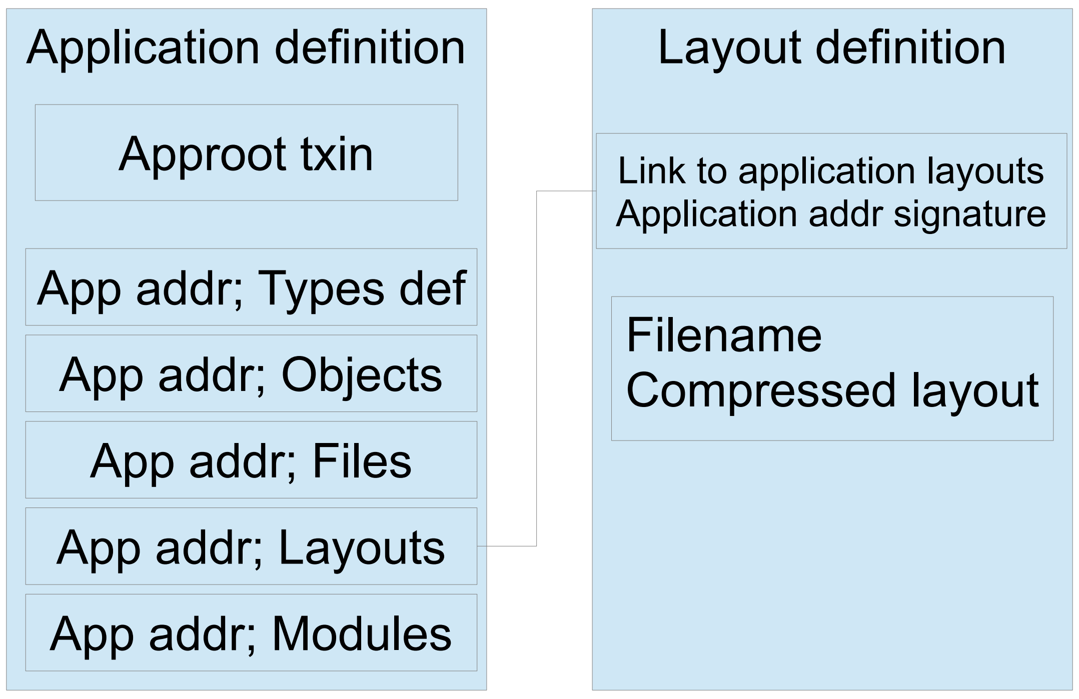
name :	MARBLE.png
mime :	image/png
size :	406367
hash :	DE21D531D3039428BA45DBC599F091D0F51EEEB730A204204C8A16AB993DC300
url :	MARBLE.png

They can be accessed via address like

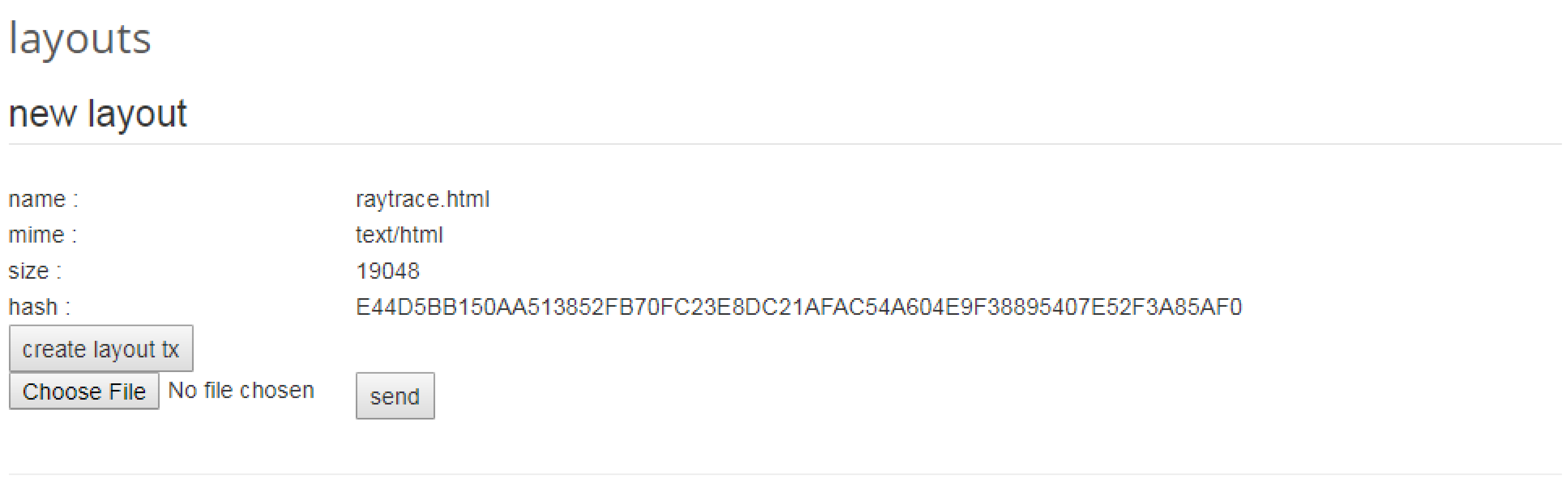
http://node-ip:port/app/**AppName**/file/**FileHash**

4.6 Application layouts

Layouts are blocks of html used to represent the application interface. They are stored in a compressed form on the blockchain and can be used in application scripts using the **html_layout** command in a page script.



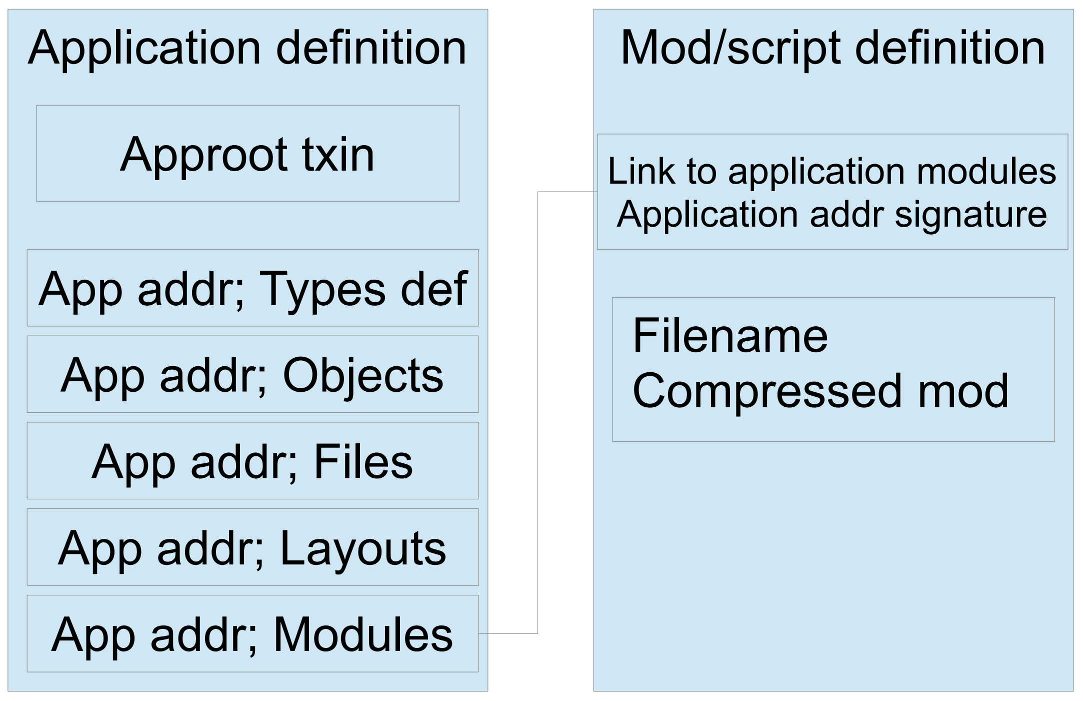
The address used to create the application need to be selected in order to create a new layout.


layouts

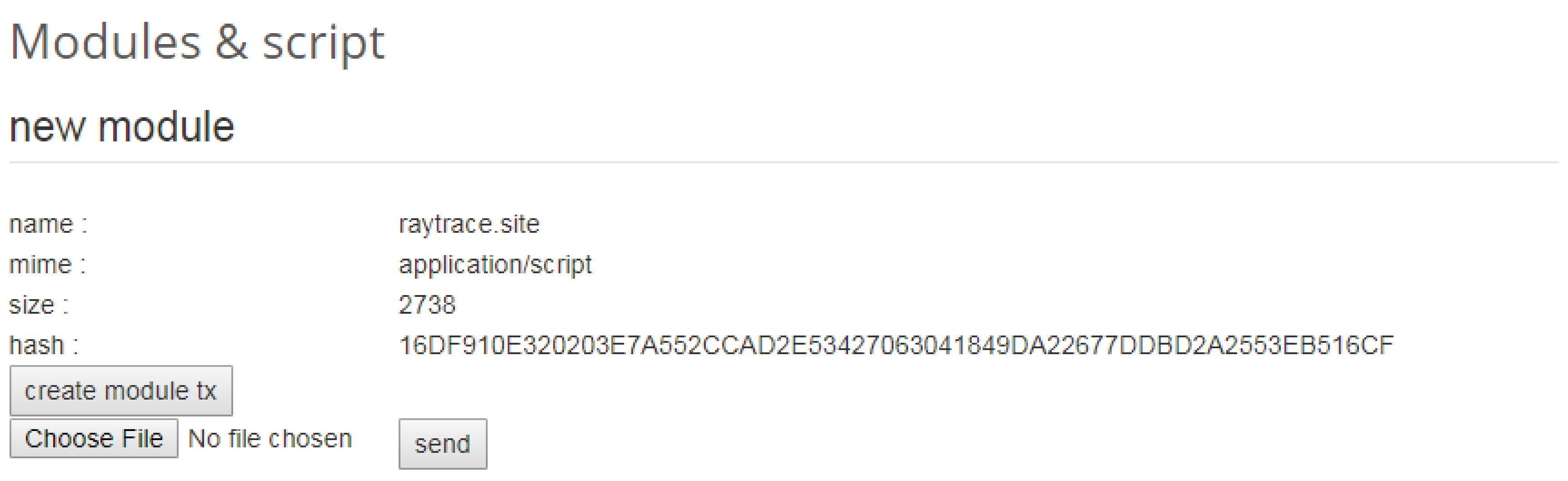
new layout

name :	raytrace.html
mime :	text/html
size :	19048
hash :	E44D5BB150AA513852FB70FC23E8DC21AFAC54A604E9F38895407E52F3A85AF0

4.7 Application scripts and modules



Applications scripts are used as a controller in MVC model, they can reference layouts and modules as well as applications files to generate the page. Each script can contain several pages entry, each with their own parameters and code.


Modules & script

new module

name :	raytrace.site
mime :	application/script
size :	2738
hash :	16DF910E320203E7A552CCAD2E53427063041849DA22677DDBD2A2553EB516CF

Scripts can contain module import of which function can be called from the script using script variable as parameters, as well as service module definition that are bound to the http interface for access from external application via CGI or JSON/RPC.

Module import

```
let NODE_MODULE_DEF tracer = ` {"file" : "apps/raytrace/modz/tracer.tpo"} `
```

Service module

```
let NODE_JSON_ARRAY service_modules = `[ {"base" : "/rt/", "type" : "cgi", (NODE_MODULE_DEF)
"tracer_rpc" : {"file" : "apps/raytrace/modz/tracer_rpc.tpo"}} ] `
```

The api in the service module can then be accessed via *http://node-ip:port/rt/function/param*

Service script

```
let NODE_MODULE_DEF tracer = ` {"file" : "apps/raytrace/modz/tracer.tpo"} `

let NODE_JSON_ARRAY service_modules = `[ {"base" : "/rt/", "type" : "cgi", (NODE_MODULE_DEF) "tracer_rpc" : {"file" :
"apps/raytrace/modz/tracer_rpc.tpo"}} ] `

page index(first) = `

     html_head  "NodiX Raytrace scenes"

     html_block "templates/menu.html"
     html_layout "scenes.html"

     tracer.get_nscene(nscenes)
     tracer.get_scene_list(scenes,first)

     html_scripts

     html_var scenes;
     html_var nscenes;


     html_js

            $(document).ready(function ()
            {
                  update_scenes(scenes,'scenes');
                  $('#nscenes').html(scenes.length);
                  $('#total').html(nscenes);
            });

     end_js
     html_block "templates/footer.html"
`
```

The pages in the service script can then be accessed via

http://node-ip:port/app/raytrace/page/raytrace.site/index

Javascript code in the layout can then access the service module API via service module binding.

5 Script code for processing of P2P protocol messages

```
handler on_verack(node,payload) = ` node_adx.queue_ping_message(node) `

handler on_ping(node,payload) = ` node_adx.queue_pong_message(node,payload.nonce) `

handler on_pong(node, payload) = ` set node.synching = 1; `

handler on_version(node,payload) = `

        set node.p2p_addr = payload.their_addr;
        set node.p2p_addr.services = payload.services;

        set node.my_addr = payload.my_addr;

        set node.user_agent = payload.user_agent;
        set node.version = payload.proto_ver;
        set node.block_height = payload.last_blk;

        node_adx.node_log_version_infos(node)

        node_adx.queue_verack_message(node)
`

handler on_inv(node, payload) = ` node_adx.queue_getdata_message(node, payload.hashes) `

handler on_addr(node, payload) = ` foreach payload.addrs node_adx.node_log_addr_infos `

handler on_block(node, payload) = `

        block_adx.set_block_hash(payload.header) : set payload.header.keep_block = 0; success endor

        if (node.testing_chain>0)

                if (payload.header.prev ! node.last_header_hash)
                        set payload.header.keep_block = 1;
                        success
                endif

            node_adx.node_add_block_header(node, payload.header)

                set cur_len = SelfNode.block_height;
                sub cur_len, node.testing_chain;

                if (node.block_headers* > cur_len)

                        set lost_reward  =  0;

                        accumulate add_reward(lost_reward) node.testing_chain,SelfNode.block_height;

                        node_adx.node_truncate_chain_to                       (node.testing_chain) :
                                success
                        endor

                        node_adx.sub_moneysupply                              (lost_reward)

                        configuration.staking.pos_kernel.compute_last_pos_diff(SelfNode.last_pos_block, SelfNode.current_pos_diff) :
                                set SelfNode.current_pos_diff = configuration.staking.limit;
                        endor

                        set node.testing_chain = 0 ;
                endif
        endif

        node_adx.node_is_next_block(payload.header) :
                node_adx.node_check_chain(node, payload.header) : success endor
        endor

        block_adx.remove_stored_block(payload.header)

        set payload.header.signature = payload.signature;

        nodix.accept_block(payload.header, payload.txs): log "rejected block" success endor

        log "store block"
        block_adx.store_block(payload.header, payload.txs) : log "error storing block" success endor
        log "store staking"
        configuration.staking.pos_kernel.store_blk_staking           (payload.header)
        configuration.staking.pos_kernel.store_blk_tx_staking  (payload.txs)
        log "new block added"

        block_adx.block_has_pow(payload.header.blkHash) ?
                nodix.compute_pow_diff                                (payload.header, SelfNode.current_pow_diff)
                block_adx.get_pow_reward                              (SelfNode.block_height, block_reward)
                set SelfNode.pow_reward = block_reward;
        endor

        block_adx.block_has_pow(payload.header.blkHash) :
                configuration.staking.pos_kernel.compute_last_pos_diff(payload.header, SelfNode.current_pos_diff)
                configuration.staking.pos_kernel.stake_get_reward            (SelfNode.block_height, block_reward)
                set SelfNode.pos_reward = block_reward;
        endor

        node_adx.add_money_supply          (block_reward)
        node_adx.node_set_last_block       (payload.header)
        node_adx.set_next_check                    (30)
success
```

5.1 Application main loop

```
OS_API_C_FUNC(int) app_loop(mem_zone_ref_ptr params)
{
	mem_zone_ref		blk_list = { PTR_NULL };
	mem_zone_ref		my_list = { PTR_NULL };
	mem_zone_ref		hash_list = { PTR_NULL };
	mem_zone_ref_ptr	blk = PTR_NULL;
	unsigned int		new_block = 0;

	update_peernodes	();
	node_check_services ();
	process_nodes		();

	if (tree_manager_find_child_node(&self_node, NODE_HASH("submitted blocks"), NODE_BITCORE_BLK_HDR_LIST, &blk_list))
	{
		tree_manager_create_node("hashes", NODE_BITCORE_HASH_LIST, &hash_list);

		for (	tree_manager_get_first_child(&blk_list, &my_list, &blk); ((blk != NULL) && (blk->zone != NULL));
			tree_manager_get_next_child (&my_list, &blk))
		{
			hash_t blk_hash = { 0 };
			struct string signature = { PTR_NULL };
			mem_zone_ref tx_list = { PTR_NULL };
			int			ret;
			if (!tree_manager_find_child_node(blk, NODE_HASH("txs"), NODE_BITCORE_TX_LIST, &tx_list))continue;
			if (!tree_manager_get_child_value_istr(blk, NODE_HASH("signature"), &signature, 0))continue;

			ret=accept_block				(blk, &tx_list);
			if (ret)
			{
				mem_zone_ref		new_hash = { PTR_NULL };
				ret= add_new_block(blk, &tx_list);
				if (ret)
				{
					tree_manager_get_child_value_hash(blk, NODE_HASH("blkHash"), blk_hash);
					if (tree_manager_add_child_node(&hash_list, "hash", NODE_BITCORE_BLOCK_HASH, &new_hash))
					{
						tree_manager_write_node_hash(&new_hash, 0, blk_hash);
						release_zone_ref(&new_hash);
					}
					new_block = 1;
				}
			}
			release_zone_ref				(&tx_list);
			free_string						(&signature);
			tree_manager_set_child_value_bool (blk, "done", 1);
		}
		tree_remove_child_by_member_value_dword(&blk_list, NODE_BITCORE_BLK_HDR, "done", 1);
		release_zone_ref(&blk_list);
	}
	if (new_block)
	{
		mem_zone_ref_ptr	node = PTR_NULL;
		mem_zone_ref		peer_nodes = { PTR_NULL };

		if (tree_manager_find_child_node(&self_node, NODE_HASH("peer_nodes"), NODE_BITCORE_NODE_LIST, &peer_nodes))
		{
			for (	tree_manager_get_first_child(&peer_nodes, &my_list, &node); ((node != NULL) && (node->zone != NULL));
				tree_manager_get_next_child (&my_list, &node))
			{
				queue_inv_message(node, &hash_list);
			}
			release_zone_ref(&peer_nodes);
		}
	}
	release_zone_ref(&hash_list);

	return 1;
}
```

5.2 Processing of node messages

```
int handle_message(mem_zone_ref_ptr node,const char *cmd,mem_zone_ref_ptr payload)
{
	if (!strncmp_c(cmd, "inv", 3))return handle_inv(node, payload);
	if (!strncmp_c(cmd, "getdata", 7))return handle_getdata(node, payload);
	return 0;
}

int process_node_messages(mem_zone_ref_ptr node)
{
	mem_zone_ref		msg_list = { PTR_NULL };
	mem_zone_ref_ptr	msg = PTR_NULL;
	mem_zone_ref		my_list = { PTR_NULL };


	if (!tree_manager_find_child_node(node, NODE_HASH("emitted_queue"), NODE_BITCORE_MSG_LIST, &msg_list))return 0;

	for (  tree_manager_get_first_child(&msg_list, &my_list, &msg); ((msg != NULL) && (msg->zone != NULL));
	       tree_manager_get_next_child (&my_list, &msg))
	{
		char			cmd[16];
		mem_zone_ref	payload_node = { PTR_NULL };
		int				ret;
		if (tree_mamanger_get_node_type(msg) != NODE_BITCORE_MSG) continue;
		if (!tree_manager_get_child_value_str	(msg, NODE_HASH("cmd"), cmd, 12, 16))continue;


		ret=node_process_event_handler		("emitted_queue", node, msg);
		if (!ret)
		{
			tree_manager_find_child_node(msg, NODE_HASH("payload"), NODE_BITCORE_PAYLOAD, &payload_node);
			ret = handle_message	(node, cmd, &payload_node);
			release_zone_ref		(&payload_node);
		}
		tree_manager_set_child_value_i32(msg, "handled", ret);
	}
	tree_remove_child_by_member_value_dword		(&msg_list, NODE_BITCORE_MSG, "handled", 1);
	tree_remove_child_by_member_value_lt_dword	(&msg_list, NODE_BITCORE_MSG, "recvtime", get_time_c()-100);

	release_zone_ref(&msg_list);
	return 1;
}
```

6 Code sample of RPC server

```
OS_API_C_FUNC(int) liststaking(mem_zone_ref_const_ptr params, unsigned int rpc_mode, mem_zone_ref_ptr result)
{
	mem_zone_ref_ptr addr;
	mem_zone_ref minconf = { PTR_NULL }, maxconf = { PTR_NULL }, unspents = { PTR_NULL }, addrs = { PTR_NULL };
	mem_zone_ref my_list = { PTR_NULL };
	mem_zone_ref last_blk = { PTR_NULL };
	struct string pos_hash_data = { PTR_NULL };
	int max = 2000;
	int ret = 0;
	unsigned int block_time;
	unsigned int target,iminconf=0;
	if (!tree_manager_find_child_node(&my_node, NODE_HASH("last block"), NODE_BITCORE_BLK_HDR, &last_blk))return 0;

	if (!tree_manager_add_child_node(result, "unspents", NODE_JSON_ARRAY, &unspents))
		return 0;

	if (tree_manager_get_child_at(params, 0, &minconf))
	{
		tree_mamanger_get_node_dword(&minconf, 0, &iminconf);
		release_zone_ref (&minconf);
	}

	if (iminconf < min_staking_depth)
		iminconf = min_staking_depth;

	tree_manager_get_child_at (params, 1, &maxconf);
	tree_manager_get_child_at (params, 2, &addrs);

	get_target_spacing (&target);
	tree_manager_get_child_value_i32(&last_blk, NODE_HASH("time"), &block_time);

	tree_manager_set_child_value_i32(result, "block_target", target);
	tree_manager_set_child_value_i32(result, "now", get_time_c());
	tree_manager_set_child_value_i32(result, "last_block_time", block_time);

	for (tree_manager_get_first_child(&addrs, &my_list, &addr); ((addr != NULL) && (addr->zone != NULL));
	tree_manager_get_next_child(&my_list, &addr))
	{
		if (max > 0)
		{
			btc_addr_t my_addr;
			tree_manager_get_node_btcaddr(addr, 0, my_addr);
			list_staking_unspent(&last_blk, my_addr, &unspents, iminconf, &max);
		}
	}
	release_zone_ref(&last_blk);
	release_zone_ref(&unspents);
	release_zone_ref(&addrs);
	release_zone_ref(&maxconf);
	return 1;
}


OS_API_C_FUNC(int) getstaketx(mem_zone_ref_const_ptr params, unsigned int rpc_mode, mem_zone_ref_ptr result)
{
	unsigned char chash[65];
	hash_t txHash, blkhash;
	btc_addr_t pubaddr;
	char toto = 0;
	mem_zone_ref vout = { PTR_NULL }, prevtx = { PTR_NULL }, newtx = { PTR_NULL }, pn = { PTR_NULL };
	struct string sPubk = { PTR_NULL }, script = { PTR_NULL }, null_str = { PTR_NULL };
	uint64_t amount;
	unsigned int OutIdx, newTxTime,n;
	int ret;

	null_str.str = &toto;
	null_str.len = 0;
	null_str.size = 1;


	tree_manager_get_child_at (params, 0, &pn);
	tree_manager_get_node_str (&pn, 0, chash, 65, 0);
	release_zone_ref (&pn);

	n = 0;
	while (n < 32)
	{
		char hex[3];
		hex[0] = chash[n * 2 + 0];
		hex[1] = chash[n * 2 + 1];
		hex[2] = 0;
		txHash[31 - n] = strtoul_c(hex, PTR_NULL, 16);
		n++;
	}


	tree_manager_get_child_at (params, 1, &pn);
	tree_mamanger_get_node_dword (&pn, 0, &OutIdx);
	release_zone_ref (&pn);
```

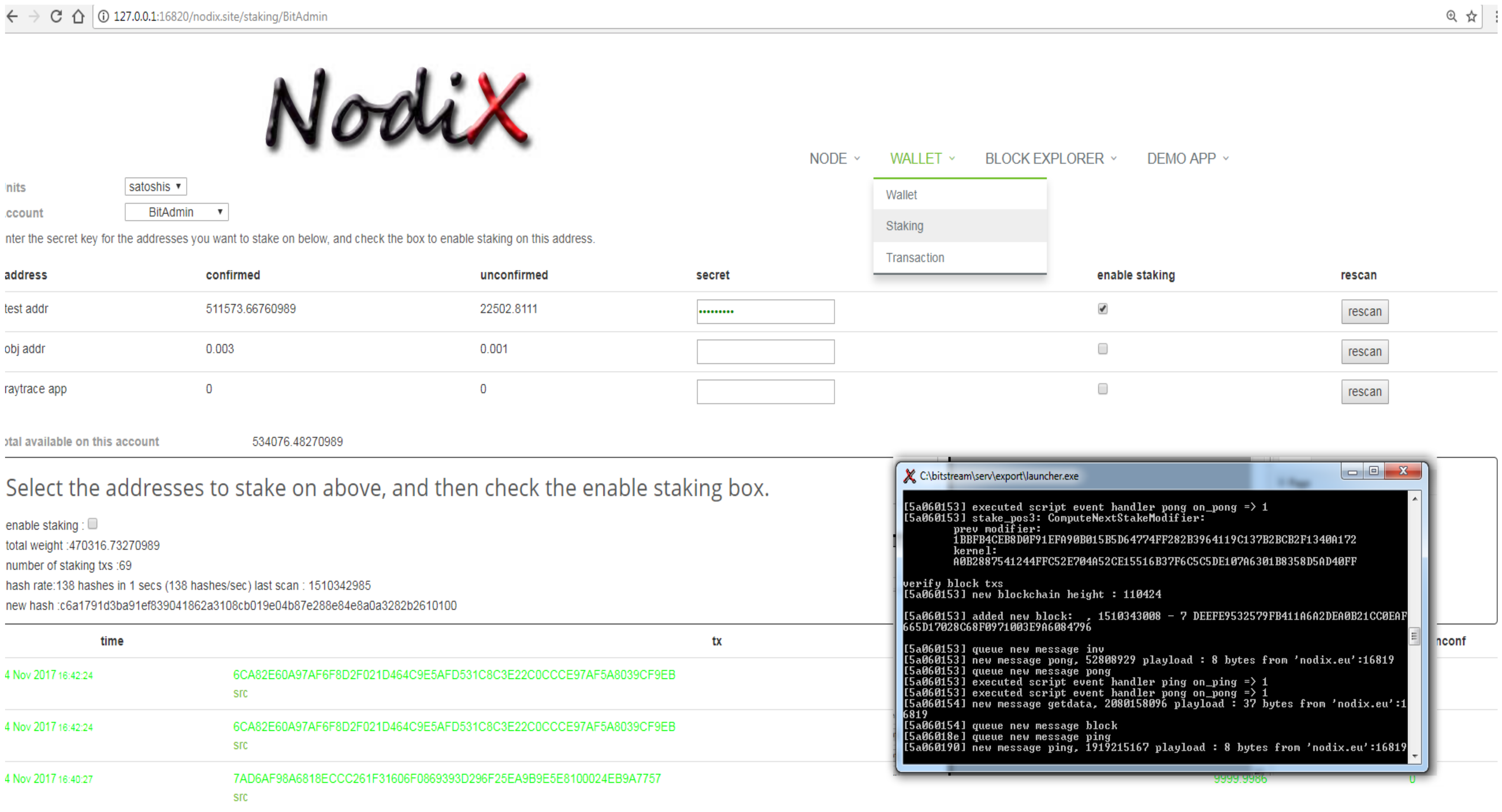
```c
	tree_manager_get_child_at (params, 2, &pn);
	tree_mamanger_get_node_dword (&pn, 0, &newTxTime);
	release_zone_ref (&pn);

	ret = load_tx(&prevtx, blkhash, txHash);

	if (ret)
		ret = get_tx_output(&prevtx, OutIdx, &vout);
	if (ret)
		ret = tree_manager_get_child_value_istr(&vout, NODE_HASH("script"), &script, 0);
	if (ret)
		ret = tree_manager_get_child_value_i64(&vout, NODE_HASH("value"), &amount);

	get_out_script_address(&script, &sPubk, pubaddr);

	if (ret)
	{
		uint64_t half_am,rew;
		ret = 0;
		get_stake_reward (&rew);
		half_am = muldiv64 (amount+rew, 1, 2);
		if (tree_manager_add_child_node(result, "transaction", NODE_BITCORE_TX, &newtx))
		{
			hash_t txh;
			unsigned int hash_type = 1;

			if (new_transaction(&newtx, newTxTime))
			{
				mem_zone_ref last_blk = { PTR_NULL };
				hash_t StakeMod, pos_hash, out_diff;
				hash_t prevOutHash;
				uint64_t weight;
				unsigned int prevOutIdx,last_diff;
				unsigned int ModTime;

				tx_add_input (&newtx, txHash, OutIdx, &script);
				tx_add_output(&newtx, 0, &null_str);
				tx_add_output(&newtx, half_am, &script);
				tx_add_output(&newtx, half_am, &script);
				tree_manager_find_child_node(&my_node, NODE_HASH("last block"), NODE_BITCORE_BLK_HDR, &last_blk))
				get_last_stake_modifier(&last_blk, StakeMod, &ModTime);
				compute_tx_pos(&newtx,StakeMod,newTxTime,pos_hash, prevOutHash, &prevOutIdx );
				memset_c (out_diff, 0, sizeof(hash_t));
				get_tx_output_amount (prevOutHash, prevOutIdx, &weight);
				last_diff = get_current_pos_difficulty();
				if (last_diff == 0xFFFFFFFF)
				{
					unsigned int nBits;
					tree_manager_get_child_value_i32(&last_blk, NODE_HASH("bits"), &nBits);
					mul_compact(nBits, weight, out_diff);
				}
				else
					mul_compact(last_diff, weight, out_diff);

				//check proof of stake
				if (cmp_hashle(pos_hash, out_diff) >= 0)
				{
					hash_t rtxhash;
					mem_zone_ref node_txs = { PTR_NULL };

					compute_tx_sign_hash (&newtx, 0, &script, hash_type, txh);

					n = 32;
					while (n--)rtxhash[n] = txh[31 - n];
					tree_manager_set_child_value_hash (result, "txhash" , rtxhash);
					tree_manager_set_child_value_btcaddr(result, "addr" , pubaddr);
					if (tree_manager_find_child_node(&my_node, NODE_HASH("tx mem pool"), NODE_BITCORE_TX_LIST, &node_txs))
					{
						tree_manager_set_child_value_bhash(&newtx, "tx hash", txh);
						tree_manager_node_add_child (&node_txs, &newtx);
						release_zone_ref (&node_txs);
					}
					ret = 1;
				}
			}
			release_zone_ref(&newtx);
		}
	}
	release_zone_ref(&vout);
	release_zone_ref(&prevtx);
	free_string(&script);
	return ret;
}
```

```c
OS_API_C_FUNC(int) getpubaddrs(mem_zone_ref_const_ptr params, unsigned int rpc_mode, mem_zone_ref_ptr result)
{
	mem_zone_ref username_n = { PTR_NULL }, addr_list = { PTR_NULL };
	struct string username = { PTR_NULL };
	struct string user_key_file = { PTR_NULL };
	size_t keys_data_len = 0;
	uint64_t conf_amount, unconf_amount, minconf;
	unsigned char *keys_data = PTR_NULL;

	if (!tree_manager_add_child_node(result, "addrs", NODE_JSON_ARRAY, &addr_list))
		return 0;

	tree_manager_get_child_at(params, 0, &username_n);
	tree_manager_get_node_istr(&username_n, 0, &username, 0);
	release_zone_ref(&username_n);

	make_string(&user_key_file, "keypairs");
	cat_cstring_p(&user_key_file, username.str);

	minconf = 1;

	if (get_file(user_key_file.str, &keys_data, &keys_data_len))
	{
		mem_ptr keys_ptr=keys_data;
		while (keys_data_len > 0)
		{
			mem_zone_ref new_addr = { PTR_NULL };
			conf_amount = 0;
			unconf_amount = 0;

			get_balance (keys_ptr, &conf_amount, &unconf_amount, minconf);
			if(tree_manager_add_child_node(&addr_list, "addr" , NODE_GFX_OBJECT, &new_addr))
			{
				tree_manager_set_child_value_btcaddr(&new_addr , "address", keys_ptr);
				tree_manager_set_child_value_i64(&new_addr, "amount" , conf_amount);
				tree_manager_set_child_value_i64(&new_addr, "unconf_amount", unconf_amount);
				release_zone_ref(&new_addr);
			}
			keys_ptr = mem_add(keys_ptr ,(sizeof(btc_addr_t) + sizeof(dh_key_t)));
			keys_data_len -= (sizeof(btc_addr_t) + sizeof(dh_key_t));
		}
		free_c(keys_data);
	}

	release_zone_ref (&addr_list);
	free_string (&user_key_file);
	free_string (&username);
	return 1;
}
```

7 Code sample of javascript staking in the browser

```
<script src="/ecdsa_bundle.js"></script>
<script src="/jsSHA-2.2.0/src/sha_dev.js"></script>
<script src="/jquery-3.1.1.min.js"></script>
<script language="javascript">

    var ec;
    var addrs = null;
    var unspents = null;
    var stake_unspents = null;
    var totalweight = 0;
    var pubkey;
    var privkey;
    var unit = 1;
    var staketimer = null;
    var block_target, now, last_block_time;
    var nHashes = 0;


    function rpc_call(in_method, in_params, in_success) {

       $.ajax({
              url: /jsonrpc ,
              data: JSON.stringify({ jsonrpc: 2.0 , method: in_method, params: in_params, id: 1 }), // id is needed !!
              type: "POST",
              dataType: "json",
              success: in_success,
              error: function (err) { alert("Error"); }
       });
    }


    function staking_loop(hash_data, time_start, time_end, diff) {
        var ct;
        for (ct = time_start; ct < time_end; ct += 16) {
            str = hex32(ct);
            total = hash_data + str;

            h = sha256(total);
            h2 = sha256(h);

            if (compare_hash(h2, diff)) {
                console.log('staking found ' + ct + ' ' + h2 + ' ' + diff);
                $('#newhash').html(h2);
                return ct;
            }
            nHashes++;
        }
        return 0;
    }

function list_staking(addresses) {
    rpc_call('liststaking', [0, 9999999, addresses], function (data) {
        var n;
        stake_unspents = data.result.unspents;
        block_target = data.result.block_target;
        now = data.result.now;
        last_block_time = data.result.last_block_time;

        update_staking();

        if (stake_unspents.length > 0)
            staketimer = setTimeout(check_all_staking, 10000);
        else
            clearTimeout(staketimer);
    });
}

function get_addrs(username)
{
       rpc_call( getpubaddrs , [username], function (data) {
              var n;
              var arAddr=[];
              addrs = data.result.addrs;
              update_addrs();

              for (n = 0; n < addrs.length; n++) {
                     arAddr[n] = addrs[n].address;
              }
              list_staking(arAddr);
       });
}

$(document).ready(function () {
       ec = new EC( secp256k1 );
       generateKeys();
       get_addrs(username);
});
```

```
function check_all_staking() {
        if ($('#do_staking').prop('checked')) {
            if (stake_unspents != null) {
                var n;
                var time_start, time_end;
                var timeStart = Math.floor(new Date().getTime() / 1000);
                var timeBegin = Math.floor((timeStart + 15) / 16) * 16;
                var num_stake_unspents = stake_unspents.length;
                if (last_block_time > (now - block_target)) {
                    time_start = Math.floor((last_block_time + 15) / 16) * 16;
                    time_end = time_start + block_target;
                }
                else {
                    time_start = timeBegin - 16;
                    time_end = timeBegin + 16;
                }
                nHashes = 0;

                for (n = 0; n < num_stake_unspents; n++) {
                    var txtime, staking;
                    staking = stake_unspents[n];
                    //console.log('staking : ' + staking.txid + '[' + staking.vout + '] ' + time_start + ' to ' + time_end);
                    txtime = staking_loop(staking.hash_data, time_start, time_end, staking.difficulty);
                    if (txtime > 0) {
                        var pubkey = $('#dostake_' + staking.dstaddr).attr('pubkey');
                        rpc_call('getstaketx', [staking.txid, staking.vout, txtime, pubkey], function (staketx) {
                            var txh, txa, secret;
                            if (staketx.error == 0) {
                                txh = staketx.result.txhash;
                                bh = staketx.result.newblockhash;
                                txa = staketx.result.addr;

                                rpc_call('getprivaddr', [accountName, txa], function (keyData) {

                                    if (keyData.error == 0) {
                                        secret = $('#secret_' + txa).val();
                                        var DecHexkey = strtoHexString(un_enc(secret, keyData.result.privkey.slice(0, 64)));
                                        var keys = ec.keyPair({ priv: DecHexkey, privEnc: 'hex' });
                                        // Sign message (must be an array, or it'll be treated as a hex sequence)
                                        var pubk = keys.getPublic().encodeCompressed('hex');
                                        var signature = keys.sign(txh, 'hex');
                                        // Export DER encoded signature in Array
                                        //var derSign = signature.toDER('hex');
                                        var derSign = signature.toLowS();
                                        rpc_call('signstaketx', [bh, derSign, pubk], function (txsign) {
                                            var hash = txsign.result.newblockhash;
                                            var blocksignature = keys.sign(hash, 'hex');
                                            //var derSign = blocksignature.toDER('hex');
                                            var derSign = blocksignature.toLowS();

                                            rpc_call('signstakeblock', [hash, derSign, pubk], function (blksign) {
                                            });
                                        });
                                    }
                                });
                            }
                            else
                                alert('stake tx rejected');
                        });
                        $('#do_staking').prop('checked', false);
                        return 0;
                    }
                }
                var timeEnd = Math.ceil(new Date().getTime() / 1000);
                var timespan = (timeEnd - timeStart);
                var hashrate = nHashes / timespan;

                $('#hashrate').html(nHashes + ' hashes in ' + timespan + ' secs (' + hashrate + ' hashes/sec) last scan : ' + timeStart);
            }
        }
        staketimer = setTimeout(check_all_staking, 10000);
    }



</script>
```



127.0.0.1:16820/nodix.site/staking/BitAdmin

nits	satoshis
ccount	BitAdmin

nter the secret key for the addresses you want to stake on below, and check the box to enable staking on this address.

address	confirmed	unconfirmed	secret	enable staking	rescan
test addr	511573.66760989	22502.8111	•••••••••	✓	rescan
obj addr	0.003	0.001		☐	rescan
raytrace app	0	0		☐	rescan

otal available on this account 534076.48270989

Select the addresses to stake on above, and then check the enable staking box.

enable staking : ☐
total weight :470316.73270989
number of staking txs :69
hash rate:138 hashes in 1 secs (138 hashes/sec) last scan : 1510342985
new hash :c6a1791d3ba91ef839041862a3108cb019e04b87e288e84e8a0a3282b2610100

time	tx		nconf
4 Nov 2017 16:42:24	6CA82E60A97AF6F8D2F021D464C9E5AFD531C8C3E22C0CCCE97AF5A8039CF9EB src		
4 Nov 2017 16:42:24	6CA82E60A97AF6F8D2F021D464C9E5AFD531C8C3E22C0CCCE97AF5A8039CF9EB src		
4 Nov 2017 16:40:27	7AD6AF98A6818ECCC261F31606F0869393D296F25EA9B9E5E8100024EB9A7757 src	9999.9986	0

C:\bitstream\serv\export\launcher.exe

```
[5a060153] executed script event handler pong on_pong => 1
[5a060153] stake_pos3: ComputeNextStakeModifier:
        prev modifier:
        1BBFB4CEB8D0F91EFA90B015B5D64774FF282B3964119C137B2BCB2F1340A172
        kernel:
        A0B2887541244FFC52E704A52CE15516B37F6C5C5DE107A6301B8358D5AD40FF

verify block txs
[5a060153] new blockchain height : 110424

[5a060153] added new block:  , 1510343008 - 7 DEEFE9532579FB411A6A2DEA0B21CC0EAF
665D17028C68F0971003E9A6084796

[5a060153] queue new message inv
[5a060153] new message pong, 52808929 playload : 8 bytes from 'nodix.eu':16819
[5a060153] queue new message pong
[5a060153] executed script event handler ping on_ping => 1
[5a060153] executed script event handler pong on_pong => 1
[5a060154] new message getdata, 2080158096 playload : 37 bytes from 'nodix.eu':1
6819
[5a060154] queue new message block
[5a06018e] queue new message ping
[5a060190] new message ping, 1919215167 playload : 8 bytes from 'nodix.eu':16819
```