How to create a Smart Contract in NEO Blockchain

Igor M. Coelho (igor.machado@ime.uerj.br)
Fabio Cardoso (slynrick@gmail.com)

BSL Hackathon 2018 Santiago/Chile, 3-4 Aug. 2018

Workshop Agenda

- What is NEO?
- What is NeoResearch?
- How to code on NEO
- How to test your code
- Examples and hands-on!

What is NEO?

- NEO is an open-source blockchain ecosystem, supporting asset transfers and Turing-complete smart contracts written in popular programming languages
- The distributed consensus mechanism allows dealing with byzantine failures, while keeping a higher number of transactions per second (TPS)
- Two main global assets: NEO and GAS
- As a governance token, NEO allows voting and GAS generation. GAS is used for paying transactions.

What is NEO?

- The purpose of NEO is to promote the Smart Economy, usually represented as the triplet: Digital Assets + Digital Identity + Smart Contract

More information:

- Organization website: neo.org
- GitHub project: github.com/neo-project
- Reddit: https://www.reddit.com/r/NEO
- Discord: https://discord.gg/umsfhqs

What is NeoResearch?

NeoResearch

Developing groundbreaking research for Neo Blockchain Ecosystem.

worldwide https://neoresearch.io

Repositories 16 | People 15 | Teams 0 | Projects 0 | Settings

Pinned repositories

Customize pinned repositories

neocompiler-eco
NeoCompiler Eco: compile, deploy and test NEO smart contracts (https://neocompiler-eco.neoresearch.io)
JavaScript 21 8

ODBFT
Optimized Delegated Byzantine Fault Tolerance
C++ 4

neo-tests
This project includes building and testing techniques for validating Neo nodes and Smart Contracts
JavaScript 6 2

neon-opt
Neo AVM optimizer: minimization of opcodes (NOP removal, unreachable code detection, ...)
JavaScript 2 2

neoresearch.github.io
NeoResearch website
HTML 2

learning-examples
Learning Material and Examples for NEO development
1 1

What is NeoResearch?

- NeoResearch is an open-source group for developing and doing research on NEO blockchain technologies
- City of Zion (CoZ) and NewEconoLabs (NEL) are the biggest global development communities for NEO
- CoZ focuses on western English speaking communities: cityofzion.io
- NEL focuses on Chinese communities: nel.group
- NeoResearch has most of it members from Brazil and South America

What is NeoResearch?

- An open-source Research & Development team formed by scientists and developers, focused on NEO blockchain technologies (formed in January 2018)
- Our homepage is **NeoResearch.io**
- GitHub group: github.com/neoresearch
- **Three main objectives:** (i) Development of platforms to provide integration of NEO ecosystem; (ii) Data collection/analysis from NEO projects; (iii) Report the results for NEO community on interesting R&D topics.

What is NeoResearch?

- We collaborate with different development/academic groups that may provide tools for the best interest of the NEO community
- The **SciChain.org** initiative is being conducted by NeoResearch group for over an year (since Dec. 2016) in order to use NEO technology to unify academic publishers and researchers towards a more transparent and reproducible science

How to code on NEO?

- Is it easy?
- Yes. Specially when you have good tools and materials in hand.

That’s our community mission :)

How to code on NEO

- NEO Smart Contracts can be coded in (virtually) any programming language! Pick your preferred one :)
- The original project developers proposed C# and Java compilers for NEO
- Community developers created Python and Go compilers (and many other proposals going on including Typescript, Haskell, …)
- Currently, the most popular programming languages for NEO are C# and Python

Hello World hackaton devs :) (C# / Python)

```
using Neo.SmartContract.Framework.Services.Neo;

namespace Neo.SmartContract
{
    public class HelloWorldNotification : Framework.SmartContract
    {
        public static void Main()
        {
            Runtime.Notify("Hello World");
        }
    }
}

```

```
def Main():
    print("Hello World")

```

How to test your code?

- NEO MainNet is supposed to be used for deploying final products (as GAS prices will need to be payed)
- NEO TestNet can be used for testing in a “real” environment
- TestNet GAS can be obtained from NEO groups, but it is usually better for fine-tuned projects
- NEO PrivateNet is the most common approach for starting new projects!

Testing locally with CoZ python private net

C# 13 10 MIT Updated 4 days ago

neo-local

Quickly setup a development environment for NEO dApps!

tooling blockchain neo development-environment coz

Shell 18 9 MIT Updated 6 days ago

neo-local-faucet

Node faucet for neo-local

JavaScript 1 1 MIT Updated 7 days ago

neon-wallet

Light wallet for the NEO blockchain

JavaScript 931 288 MIT Updated 7 days ago

NeoCompiler Eco

- The **NeoCompiler.io** website (and neocompiler-eco project) was started in December 2017 as an integration hub for NEO smart contract development
- It allows to online compile a smart contract (currently in C#, Java, Python and Go)
- It also allows easy testing of smart contracts code in a public network (“shared privatenet”) called Eco. This network resets periodically (usually 12-hours), now being expanded to larger periods (few days)

NeoCompiler Eco (https://neocompiler.io)

NeoCompiler Eco

Seguro | https://neocompiler.io/#/

NeoCompiler Eco

Online: 8 | Time Left: 22:22:11 | RPC_PY:https://rest.neocompiler.io -> /NEO-PYTHON:0.7 | Access: 68075 | https://neoscan.neocompiler.io -> H3:5663 | RPC_C#:https://node1.neocompiler.io/ -> /NEO:2.7.6.1/ -

EcoLab | NeoScan-PrivateNet | About

You can do it anywhere!
Using our page or our mobile app!

NeoCompiler Eco can help you edit, build and test your Smart Contract

GET STARTED!

Download a mobile version of NeoCompiler Eco

ANDROID APK (V1.5)

NeoCompiler Eco

Seguro | https://neocompiler.io/#/ecolab

NeoCompiler Eco

Online: 8 | Time Left: 22:20:49 | RPC_PY:https://rest.neocompiler.io -> /NEO-PYTHON:0.7.5/ -> H2:5673 | Access: 68077 | https://neoscan.neocompiler.io -> H3:5672 | RPC_C#:https://node1.neocompiler.io -> /NEO:2.7.6.1/ -> H1:5673 -> MP:0

EcoLab | NeoScan-PrivateNet | About

PythonNotifications | Compile

NodeInfo

C# | Python | Go | Java

SELECT EXAMPLE

```
1  using Neo.SmartContract.Framework;
2  using Neo.SmartContract.Framework.Services.Neo;
3  using System;
4  using System.Numerics;
5  namespace NeoContract1
6  {
7      public class Contract1 : SmartContract
8      {
9          public static void Main()
10         {
11             Storage.Put(Storage.CurrentContext, "Hello", "World");
12         }
13     }
14 }
15
```

Optimize opcodes with neon-opt (BETA! USE WITH CAUTION!)

COMPILE

C# HelloWorld (Notification) - Step 1 - Compile

```
using Neo.SmartContract.Framework.Services.Neo;

namespace Neo.SmartContract
{
    public class HelloWorldNotification : Framework.SmartContract
    {
        public static void Main()
        {
            Runtime.Notify("Hello World");
        }
    }
}

```

C# HelloWorld (Notification) - Step 1 - Compile

Compile Warning/Errors:

```
 convert succ
 gen abi succ
 write:NeoContract1.avm
 write:NeoContract1.abi.json
 SUCC
Done Building Project "/tmp/NeoContract1/NeoContract1.csproj" (default targets).
Done Building Project "/tmp/NeoContractTeste.sln" (default targets).

Build succeeded.
   0 Warning(s)
   0 Error(s)

Time Elapsed 00:00:04.60
```

C# HelloWorld (Notification) - Step 1 - Compile

AVM (in hex):

00c56b51c576000b48656c6c6f20576f726c64c46168124e656f2e52756e74696d652e4e6f74696679616c7566

ABI:

ScriptHash (reversed): 0xd71ee2627458f77b92d98751c3b6798a85ac86b6

Entry Point:Main

Functions:

    Void Main();

Events:

What is AVM, ABI and ScriptHash?

- The AVM represents the commands executed by your smart contract (each hex pair represents an opcode)
- Each operation has a price (in GAS), so NEO economic model incentives developers to generate efficient code
- http://docs.neo.org/en-us/sc/systemfees.html
- Currently, all operations below 10 GAS are free! But it is **recommended** to add a small fraction for priority

What is AVM, ABI and ScriptHash?

- The ABI represents the public functions you expose as the Application Interface of your smart contract
- Every contract execution starts in Main() function
- Function calls are performed by passing parameters to Main(parameter1, parameter2, …)
- It is recommended to adopt a canonical signature, which receives a string and a list of objects:

```
public static object Main(string operation, params object[] args)
{
```

What is AVM, ABI and ScriptHash?

- Basic ABI for NEP-5 tokens

```
Entry Point:Main
Functions:
    String Name();
    String Symbol();
    Integer Decimals();
    Any Main(String operation, Array args);
    Boolean Deploy();
    Boolean MintTokens();
    Integer TotalSupply();
    Boolean Transfer(ByteArray from, ByteArray to, Integer value);
    Integer BalanceOf(ByteArray address);
```

What is AVM, ABI and ScriptHash?

- The ScriptHash format represents a smart contract AVM, and can hold assets in a network
- Usually, a ScriptHash is represented in Address format, which is easier to read
- HelloWorld-ScriptHash: 0x7080009f67a84c7bfdcd7e81f82dda10513e529a
- HelloWorld-Address: AVqrEQwM6mCwbpPD2hdiFVcq9djehQRkzc

C# CheckWitness - Step 1 - Compile

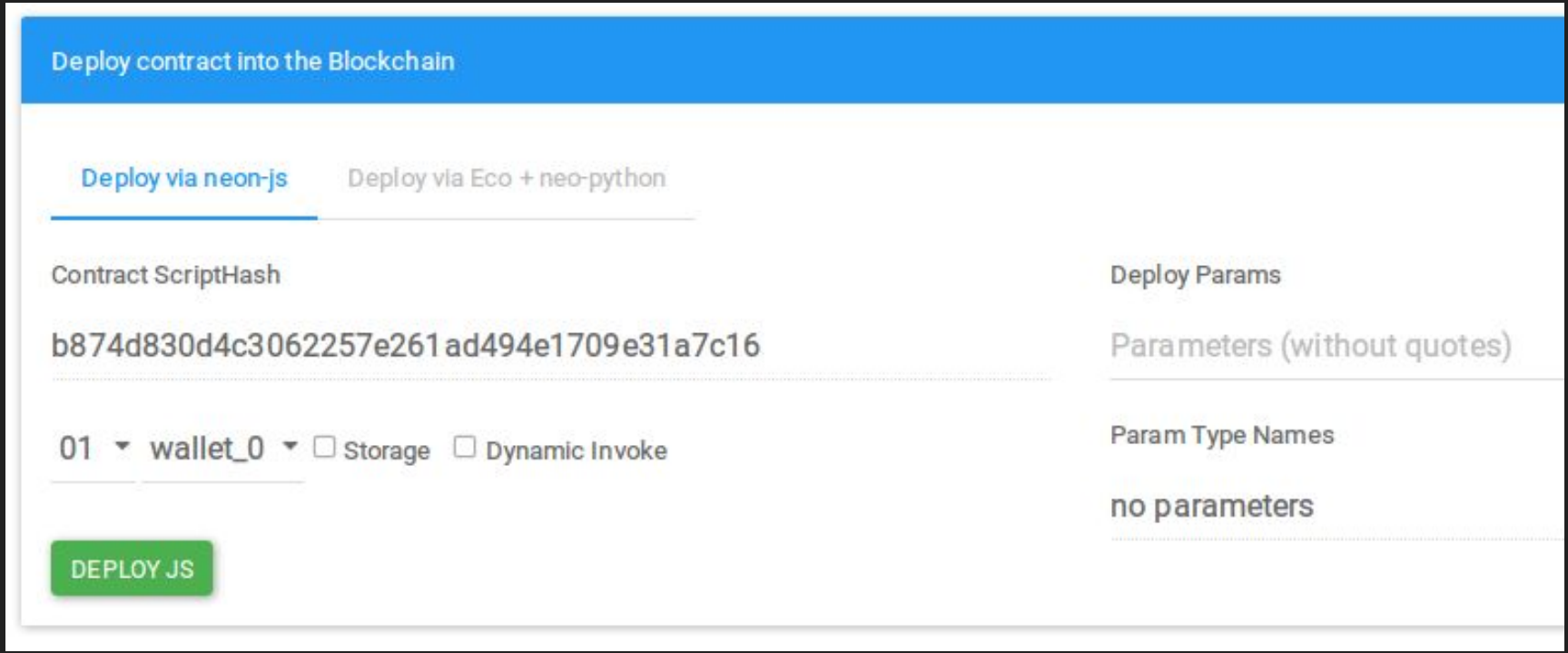
```
public static readonly byte[] Owner = "AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y".ToScriptHash();

public static bool Main() {

    if (Runtime.Trigger == TriggerType.Verification) {
        return Runtime.CheckWitness(Owner);
    }

    if (Runtime.Trigger == TriggerType.Application) {
        bool result = Runtime.CheckWitness(Owner);
        if (result) {
            Runtime.Notify("OWNER is caller");
            return true;
        }
        return false;
    }
    return false;
}
```

What are Verification and Application triggers?

- Currently, NEO supports two types of coding: Verification and Application
- Verification allows to determine who owns the assets of the contract (controlling transfers)
- Application defines the commands to be executed when invoked as a dapp (distributed application)

CheckWitness function is used to verify the identity of the invoker

C# CheckWitness - Step 2 - Deploy



C# CheckWitness - Step 2 - Deploy

ID	txType	txScriptHash	AppLog	txParams	TX on NeoScan	InvokeRestore
0	Deploy	7080009f67a84c7	HALT, BREAK	DeployParams	80ddf...8ecf0	:)

RELOAD TX'S STATUS

How much does a deploy cost?

- Currently, NEO deploy costs 100 GAS (without storage) and 500 GAS (with storage)
- The price motivates people to actually deploy only well-tested codes, and since operations are free after that (< 10 GAS), the economic model strongly compensates the use of NEO platform
- Remember that on the **private net**, you have access to an infinite amount of NEO and GAS :)

C# CheckWitness - Step 3 - Invoke

Invoke via neon-js | Invoke via Eco + neo-python

System fee (in GAS) 0 | Attach NEO 0 | GAS 0 | wallet_0

Contract ScriptHash

b874d830d4c3062257e261ad494e1709e31a7c16

Params

[]

Function to Invoke ☐ use param array	Type	Param 1: ☐ in array	Type	Param 2 ☐ in array	Type	Para
Main	None	Script Param 1	None	Script Param 2	None	Scr

INVOKE JS

C# CheckWitness - Step 3 - Invoke

ID	txType	txScriptHash	AppLog	txParams	TX on NeoScan	InvokeRestore
0	Deploy	7080009f67a84c7	HALT, BREAK	DeployParams	80ddf...8ecf0	:)
1	Invoke	88ab1d6c995730	HALT, BREAK	[]	8b862...187a4	:)

RELOAD TX'S STATUS

C# CheckWitness - Step 3 - Invoke

RPC Json Model

[{"jsonrpc":"2.0","id":5,"method":"getapplicationlog","params":["8b862b6d21958df983f1cd037e6605c2bdd7b046143621ec663e

Output RPC

[{"contract":"0x88ab1d6c995730ccabc6faa4072732d7440c12c1","state":{"type":"Array","value":
[{"type":"ByteArray","value":"4f574e45522069732063616c6c6572"}]}}]}}]

RPC CALL JSON | CONVERT NOTIFICATIONS

Notification 0 typeI: Array; typeII: ByteArray; value(hex): 4f574e45522069732063616c6c6572; value(string): OWNER is caller.

; value(hex): 4f574e45522069732063616c6c6572; value(string): OWNER is caller.

How to persist information of the blockchain

- Currently, NEO deploy costs 100 GAS (without storage) and 500 GAS (with storage)
- The price motivates people to actually deploy only well-tested codes, and since operations are free after that (< 10 GAS), the economic model strongly compensates the use of NEO platform
- Information can be read at any time (without any cost), using RPC command getstorage

Storage Examples (C# and Python)

```python
from boa.interop.Neo.Storage import Get, Put, GetContext
from boa.interop.Neo.Runtime import Notify

context = GetContext()

def Main():
    Put(context, "Hello", "World")
    value = Get(context, "Hello")
    Notify(value)
```

```csharp
using Neo.SmartContract.Framework;
using Neo.SmartContract.Framework.Services.Neo;

namespace Neo.SmartContract
{
    public class Storages : Framework.SmartContract
    {
        public static void Main()
        {
            Storage.Put(Storage.CurrentContext, "Hello", "World"); // wr
            byte[] data = Storage.Get(Storage.CurrentContext, "Hello");
            Runtime.Notify(data, data.AsString());
        }
    }
}
```

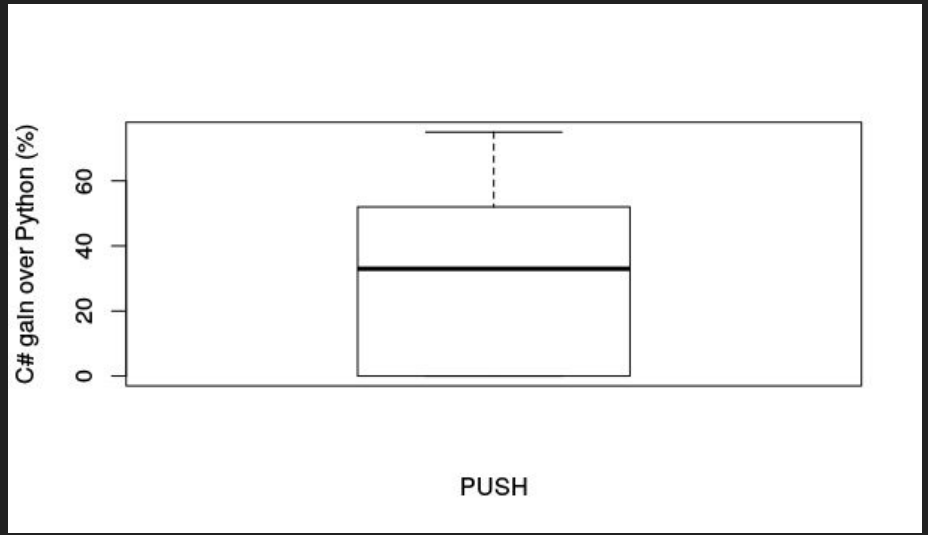
Notify and Event Examples (C# and Python)

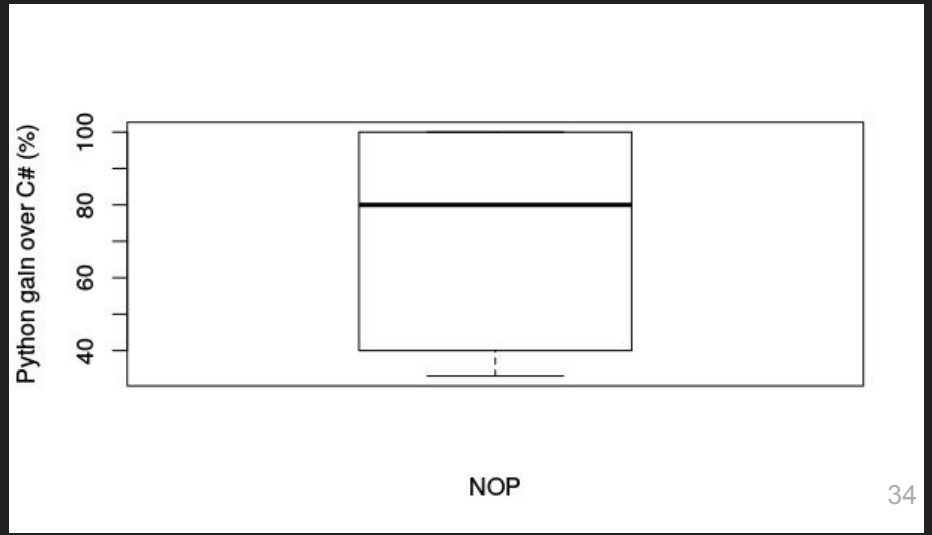
```csharp
using Neo.SmartContract.Framework;
using Neo.SmartContract.Framework.Services.Neo;
using Neo.SmartContract.Framework.Services.System;
using System;
using System.Numerics;
using System.ComponentModel;

namespace Neo.SmartContract
{
    public class NotifyEvent : Framework.SmartContract
    {
        [DisplayName("Event")]
        public static event Action<BigInteger, String> Event;

        public static void Main()
        {
            BigInteger i = 10;
            String j = "Hello";
            Runtime.Notify(i, j);
            Event(i,j);
        }
    }
}
```

```python
from boa.interop.Neo.Runtime import Notify
from boa.interop.Neo.Action import RegisterAction

Event = RegisterAction('Event', 'Number', 'String')

def Main():
    integer = 1
    string = "hello"
    Event(integer,string)
    Notify(integer,string)
```

Differences between compilers/languages

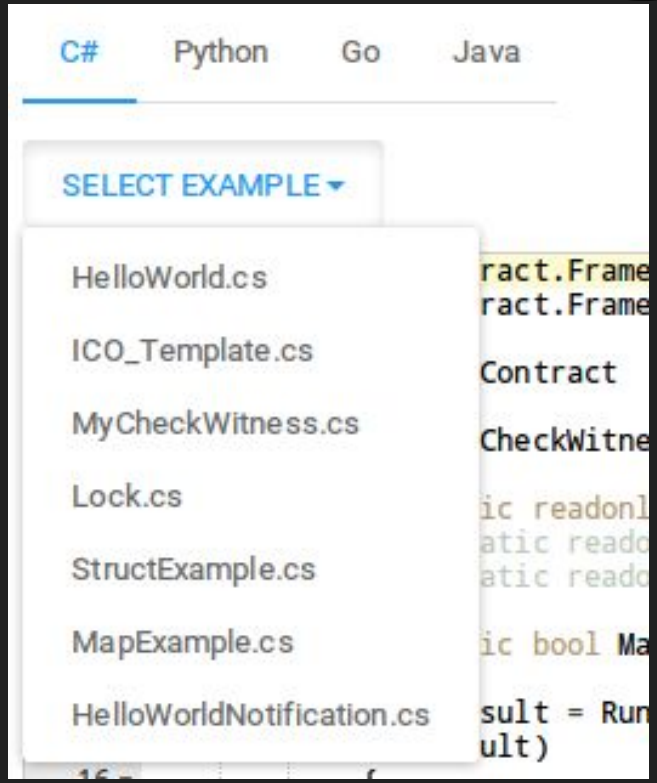
- Python compiler tends to produce 28% more PUSHBYTES operations and C# tends for 71% more NOPs. NeoResearch project neon-opt can fix that :)





Many more examples on neocompiler.io

- More examples can be found in many different places
- C# examples on neo-project github (examples-csharp)
- Python examples on neo-boa project (github.com/cityofzion)
- ICO and token examples on startups: NEX, Moonlight, ...



How to create a Smart Contract in NEO Blockchain

Igor M. Coelho (igor.machado@ime.uerj.br)
Fabio Cardoso (slynrick@gmail.com)

BSL Hackathon 2018 Santiago/Chile, 3-4 Aug. 2018