

# Download (select one)

- ***Github***

```
$ git clone https://github.com/nanaones/icon\_score\_dev\_workshop
```

- ***Docker Cloud hub***

```
$ docker run -it -p 9000:9000 nanaones/icon:0.3
```

- ***tar 파일 다운로드 받기***

```
ftp://192.168.0.1:21  
With SSID : icon
```

# ICON Network에서 SCORE 개발하기

나라

# 오늘의 목표 6가지!

1. SCORE 배우기
2. ICON 지갑 만들어보기
3. SCORE 로컬에 배포하기
4. SCORE 테스트넷에 배포하기
5. SCORE 만들어보기 (Lucky Draw)
6. (Github 에 올리기)

## 1. SCORE 배우기

# SCORE

SCORE란?

# 1. SCORE (Smart Contract On Reliable Environment)

→ SCORE는 ICON 네트워크 상에서 실행되는 Smart Contract.

## ◆ 특징

- 파이썬으로 작성
- SCORE == State Machine
  - VarDB, DictDB, ArrayDB
- **T-Bears**, SDK\* 를 통해 Deploy(배포) 가능

(ICON은 2018년 12월 현재 Java, Python, JavaScript SDK를 지원합니다.)\*

## 1. SCORE 배우기

# T-Bears

T-Bears란?

## 2. T-Bears (ICON SCORE development suite)

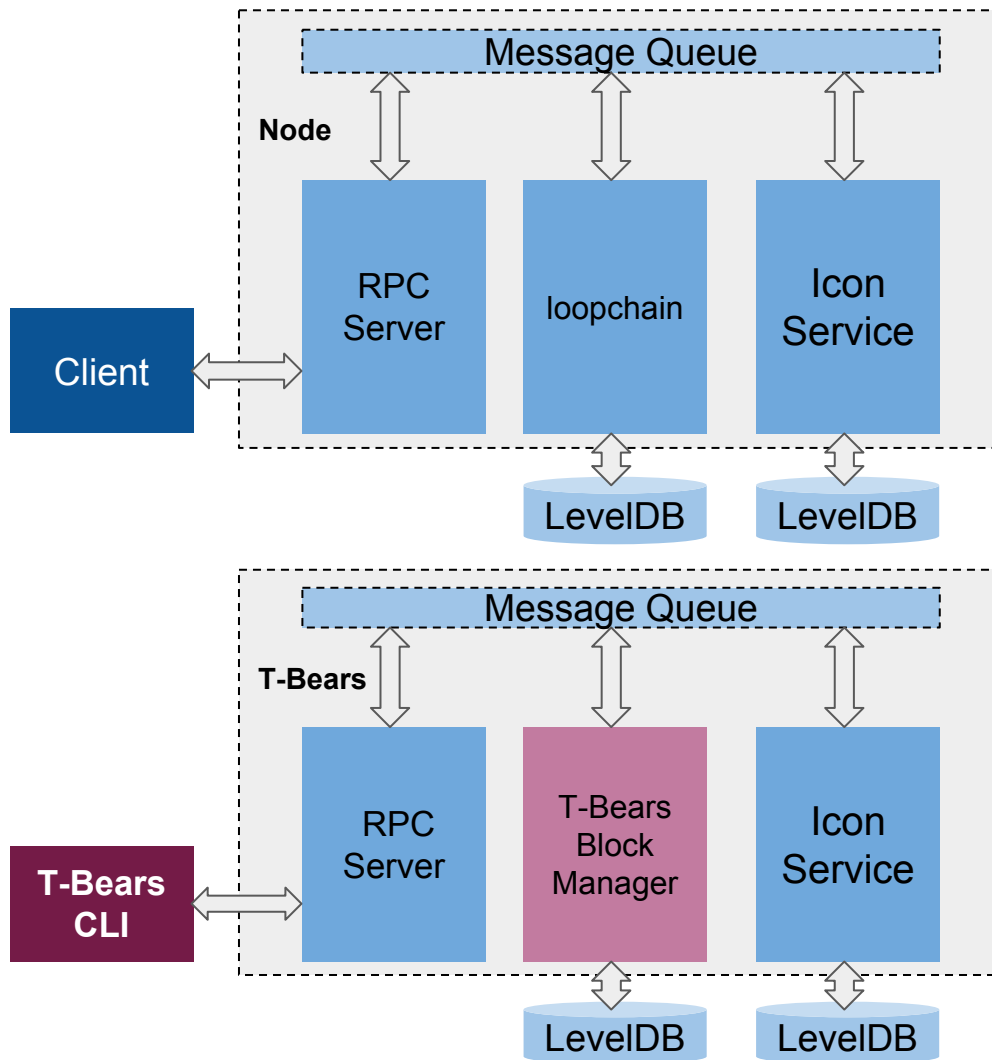
➤ SCORE 개발용 CLI 로컬 테스트 개발 툴

### ◆ 기능

- T-Bears는 바로 시작할 수 있도록 SCORE에 대한 프로젝트 템플릿을 제공한다.
- 에뮬레이트된 환경에서 로컬 테스트를 할 수 있으며, SCORE를 CLI 환경에서 로컬 네트워크와 ICON 네트워크에 배포 할 수 있다.

# T-Bears 구성요소

- **ICON RPC Server**
  - ICON JSON-RPC API 요청을 처리하고 응답을 클라이언트에 다시 보내는 모듈
- **ICON Service**
  - SCORE의 수명주기와 실행을 관리하는 모듈. SCORE의 상태 전이는 데이터베이스에 저장된다.
- **T-Bears Block Manager**
  - 트랜잭션을 처리하고 블록 생성을 에뮬레이트한다.
- **Message queue**
  - 메세지큐는 구성 요소 간 통신에 사용된다.





### 3. T-Bears 설치

Unix **Only**

Python 3.6 version **Only**

tbears python

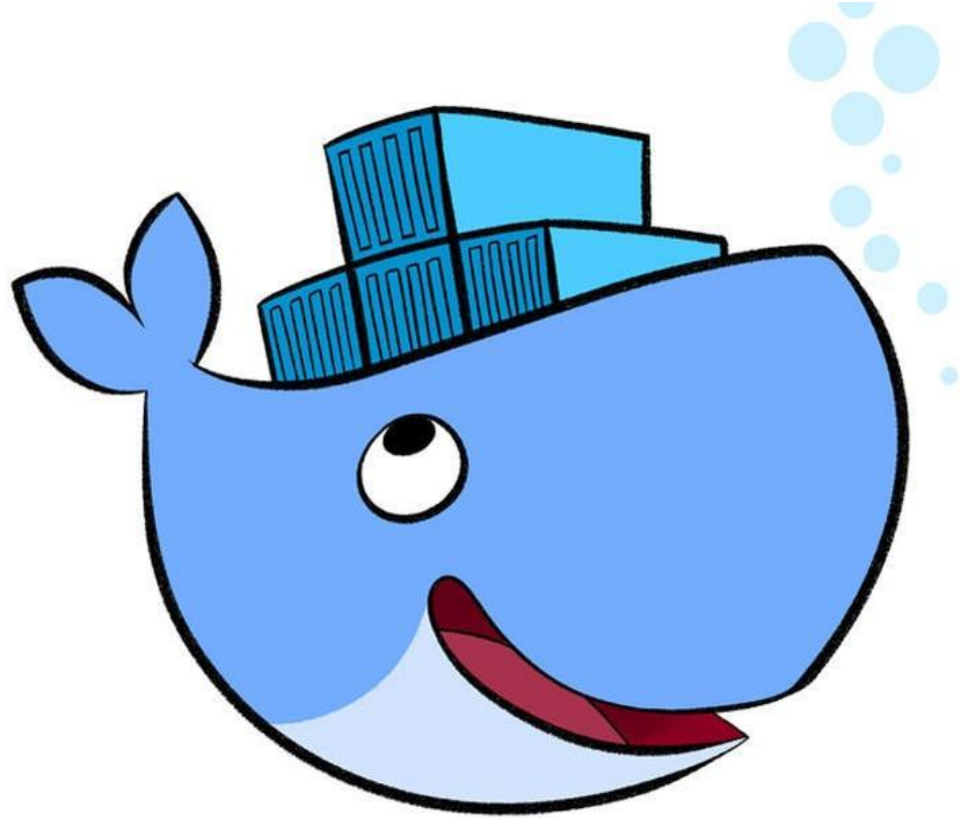
**requirements**

...

**t-bears/requirements.txt**

```
earlgray >= 0.0.4
iconcommons >= 1.0.5
iconrpcserver >= 1.1.2.1
iconservice >= 1.1.2
iconsdk >= 1.0.6
setproctitle == 1.1.10
requests == 2.20.0
plyvel == 1.0.5
secp256k1 == 0.13.2
eth-keyfile == 0.5.1
ipython == 6.4.0
.
.
.
.
```

**Use Docker !**



# Download(select one)

- ***Github***

```
$ git clone https://github.com/nanaones/icon\_score\_dev\_workshop
```

- **Docker Cloud hub**

```
$ docker run -it -p 9000:9000 nanaones/icon:0.3
```

- **tar 파일 다운로드 받기**

```
ftp://192.168.0.1:21  
With SSID : icon
```

# Load image

- **Docker Github에서 이미지 불러오기**

```
$ git clone https://github.com/nanaones/icon\_score\_dev\_workshop  
$ docker build --t icon .  
$ dpcker run -it -p 9000:9000 nara
```

- **Docker cloud hub에서 이미지 불러오기**

```
$ docker run -it -p 9000:9000 nanaones/icon:0.3
```

- **tar 파일에서 이미지 불러오기**

```
$ docker load --input icon.tar  
$ docker run -it -p 9000:9000 nanaones/icon:0.3
```

# **break time**

설치 문제가 생기면, 손을 들어주세요

## 2. 지갑 만들어보기

Appendix

# ICON 지갑 만들어보기

ICONex

## 2. 지갑 만들어보기

Appendix

ftp://192.168.0.1:21 / With SSID : icon

iconEX



ANDROID



## 2. 지갑 만들어보기

Appendix

**ftp://192.168.0.1:21 / With SSID : icon**

iconEX





## 2. 지갑 만들어보기

ftp://192.168.0.1:21 / With SSID : icon

### Appendix 지갑

ICONex 설치

Chrome market

Appendix\_link.md



ICONex 홈 / 내 지갑 / 코인

**Nara**

보유 코인

0 ICX

₩ 0 USD ▾

송금

입금 주소

hxe6d129275d46a9bd0bb787e1956c896120cdda6e

입금 주소 복사

QR코드 주소

· 위 주소로 코인을 입금할 수 있습니다.

· 모바일로 입금 시, 우측 QR코드를 스캔하면 주소가 입력됩니다.

## 2. 지갑 만들어보기

ftp://192.168.0.1:21 / With SSID : icon

### Appendix 지갑

## ICONex 에서 ICON 지갑 만들기

1. 지갑이름, 비밀번호 입력하기
  - a. 비밀번호는 적어도 8자 이상의 글자, 숫자, 특수문자를 포함하여야 합니다.
2. 백업용 Keystore file을 다운로드 받기
3. Private Key & Keystore file 안전하게 보관하기



## 2. 지갑 만들어보기

ftp://192.168.0.1:21 / With SSID : icon

### Appendix 지갑

## keyfile 불러오기

1. Add Wallets
2. 'Load Wallet' 체크
3. 불러들일 Wallet file 업로드
  - a. private key로도 지갑을 불러들일 수 있습니다.
4. 비밀번호 입력하기

wallets

COINS & TOKENS

Add wallets • Connect to Ledger



NaRa

1

0 USD

ICON

0 ICX

0 USD

No withdrawal history

Transfer



Loaded

1

0 USD

ICON

0 ICX

0 USD

No withdrawal history

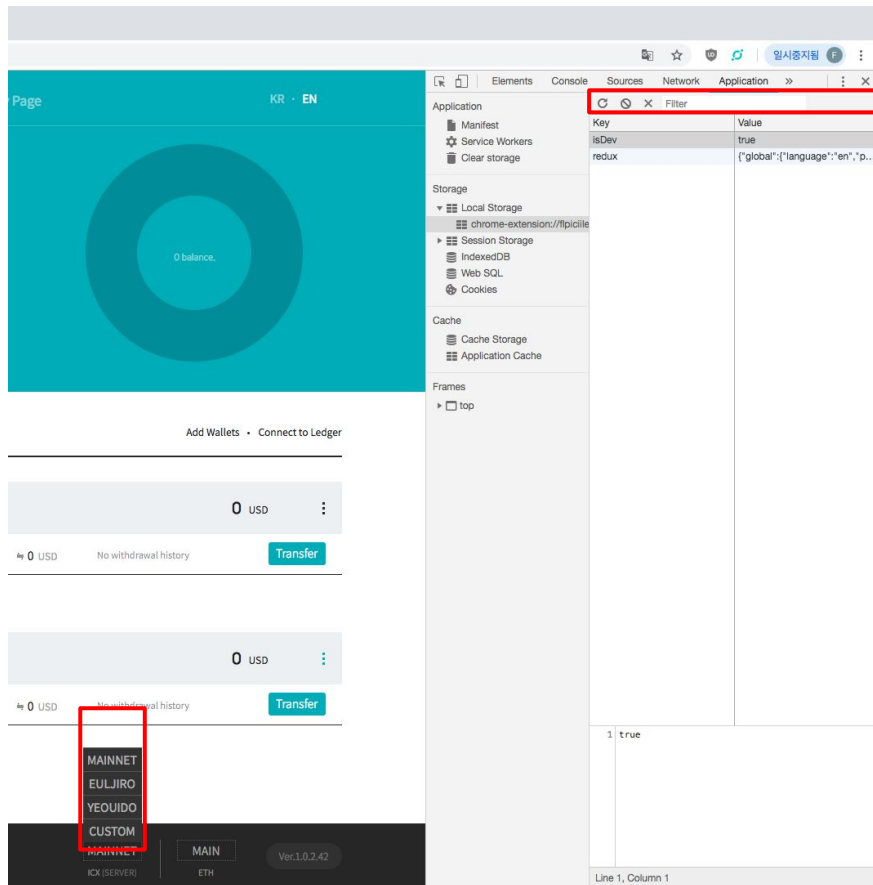
Transfer

# 지갑

ftp://192.168.0.1:21 / With SSID : icon

## testnet에 연결하기

1. 크롬 브라우저에서 개발자 메뉴켜기
2. Application 탭 클릭하기
3. 새로운 키 “isDev” 추가하기
4. 추가된 “isDev”키에 값“true” 넣기
5. 새로고침하기
6. 화면 오른쪽 하단에서 네트워크 선택하기(여의도)

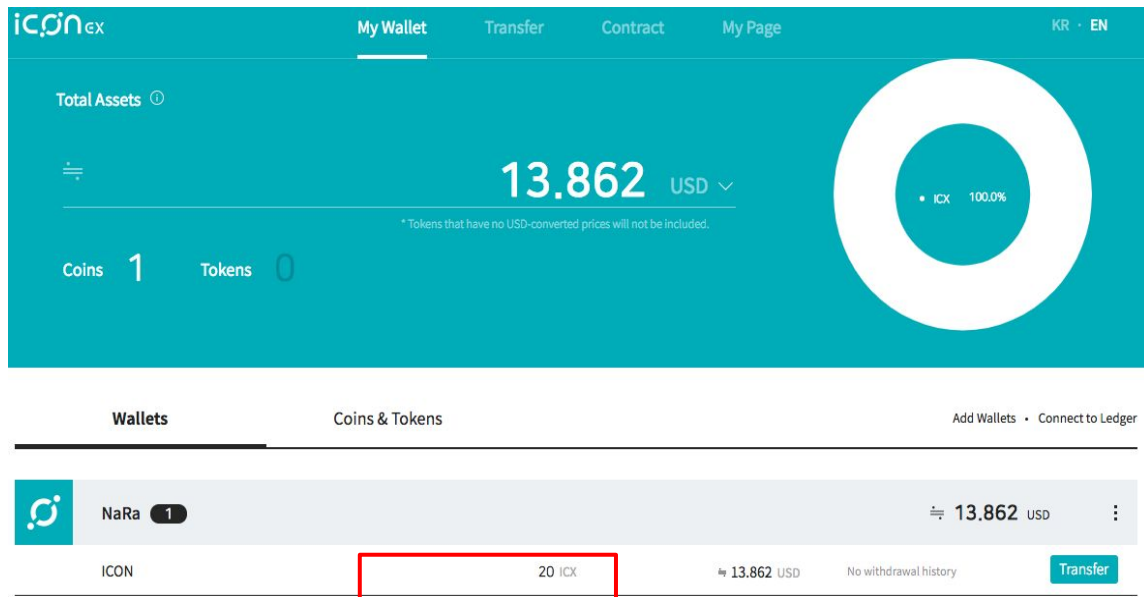


ftp://192.168.0.1:21 / With SSID : icon

# 지갑

## testnet ICX 요청

- Mail to **testicx@icon.foundation**

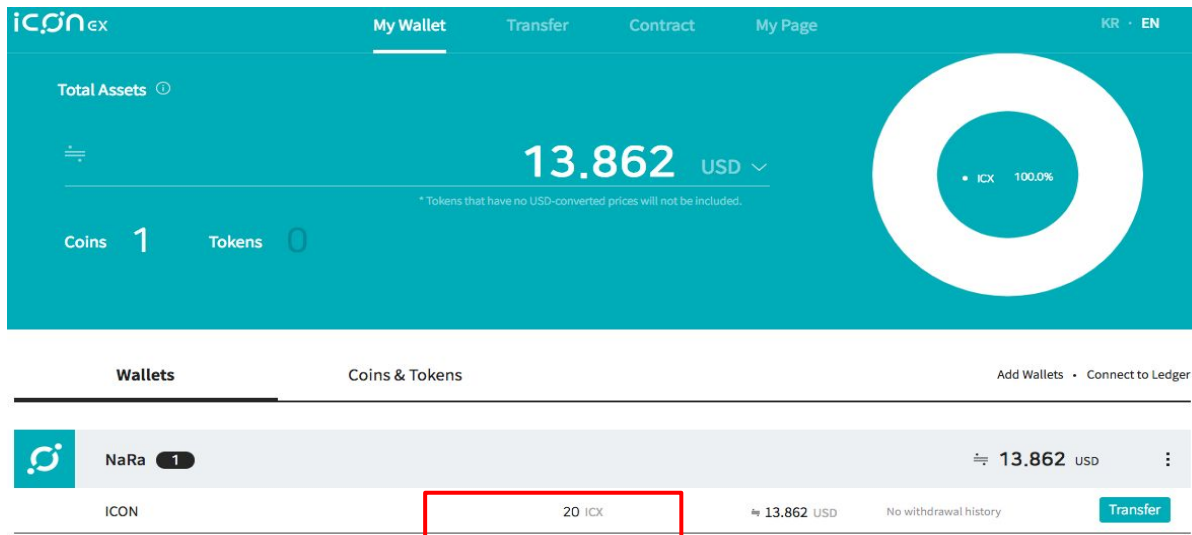


ftp://192.168.0.1:21 / With SSID : icon

# 지갑

## 테스트넷에서 사용할 ICX 받기

- <http://52.88.70.222> 접속하기
- Appendix\_link.md 참조



3. SCORE 로컬에 배포하기

ftp://192.168.0.1:21 / With SSID : icon

# T-Bears 사용해 보기

Docker

ftp://192.168.0.1:21 / With SSID : icon

# T-Bears 명령어 살펴보기 : \$ tbears -h

```
2.root@0b0743574bce: /home/tbears (docker)
root@0b0743574bce:/home/tbears# tbears -h
usage: tbears [-h] [-v] command ...

tbears v1.1.0 arguments

optional arguments:
  -h, --help      show this help message and exit
  -v, --verbose   Verbose mode

Available commands:
  If you want to see help message of commands, use "tbears com
command
start      Start tbears service
stop       Stop tbears service
deploy     Deploy the SCORE
clear      Clear all SCOREs deployed on tbears service
test       Run the unittest in the project
init       Initialize tbears project
samples    This command has been deprecated since v1.1.0
genconf    Generate tbears config files. (tbears_server_
tbears_cli_config.json and keystore_test1)
console    Get into tbears interactive mode by embedding
txresult   Get transaction result by transaction hash
transfer   Transfer ICX coin.
keystore   Create a keystore file in the specified path
balance    Get balance of given address in loop unit
totalsupply Query total supply of ICX in loop unit
scoreapi   Get score's api using given score address
txbyhash   Get transaction by transaction hash
lastblock  Get last block's info
blockbyhash Get block info using given block hash
blockbyheight Get block info using given block height
sendtx     Request icx_sendTransaction with the specifie
keystore file. If keystore file is not given,
request as it is in the json file.
call       Request icx_call with the specified json file.
```

|               |   |
|---------------|---|
| command       |   |
| start         | Start tbears service  |
| stop          | Stop tbears service   |
| deploy        | Deploy the SCORE  |
| clear         | Clear all SCOREs deployed on tbears service   |
| test          | Run the unittest in the project   |
| init          | Initialize tbears project   |
| samples       | This command has been deprecated since v1.1.0   |
| genconf       | Generate tbears config files. (tbears_server_config.json, tbears_cli_config.json and keystore_test1)  |
| console       | Get into tbears interactive mode by embedding IPython   |
| txresult      | Get transaction result by transaction hash  |
| transfer      | Transfer ICX coin.  |
| keystore      | Create a keystore file in the specified path  |
| balance       | Get balance of given address in loop unit   |
| totalsupply   | Query total supply of ICX in loop unit  |
| scoreapi      | Get score's api using given score address   |
| txbyhash      | Get transaction by transaction hash   |
| lastblock     | Get last block's info   |
| blockbyhash   | Get block info using given block hash   |
| blockbyheight | Get block info using given block height   |
| sendtx        | Request icx_sendTransaction with the specified json file and keystore file. If keystore file is not given, request as it is in the json file. |
| call          | Request icx_call with the specified json file.  |



# T-Bears 활용 : keystore

## 1. T-Bears 에서 Keystore 만들기

```
$ tbears keystore <Your Name>
```

```
root@9f9a7a309c60:/home/init_test/wallet# tbears keystore nara
Input your keystore password:
Retype your keystore password:
Made keystore file successfully
root@9f9a7a309c60:/home/init_test/wallet# ls -all
total 12
drwxr-xr-x 2 root root 4096 Nov 13 03:10 .
drwxr-xr-x 4 root root 4096 Nov 13 03:10 ..
-rw-r--r-- 1 root root 507 Nov 13 03:10 nara
root@9f9a7a309c60:/home/init_test/wallet#
```

8글자 이상, 영문, 특수문자, 숫자 포함

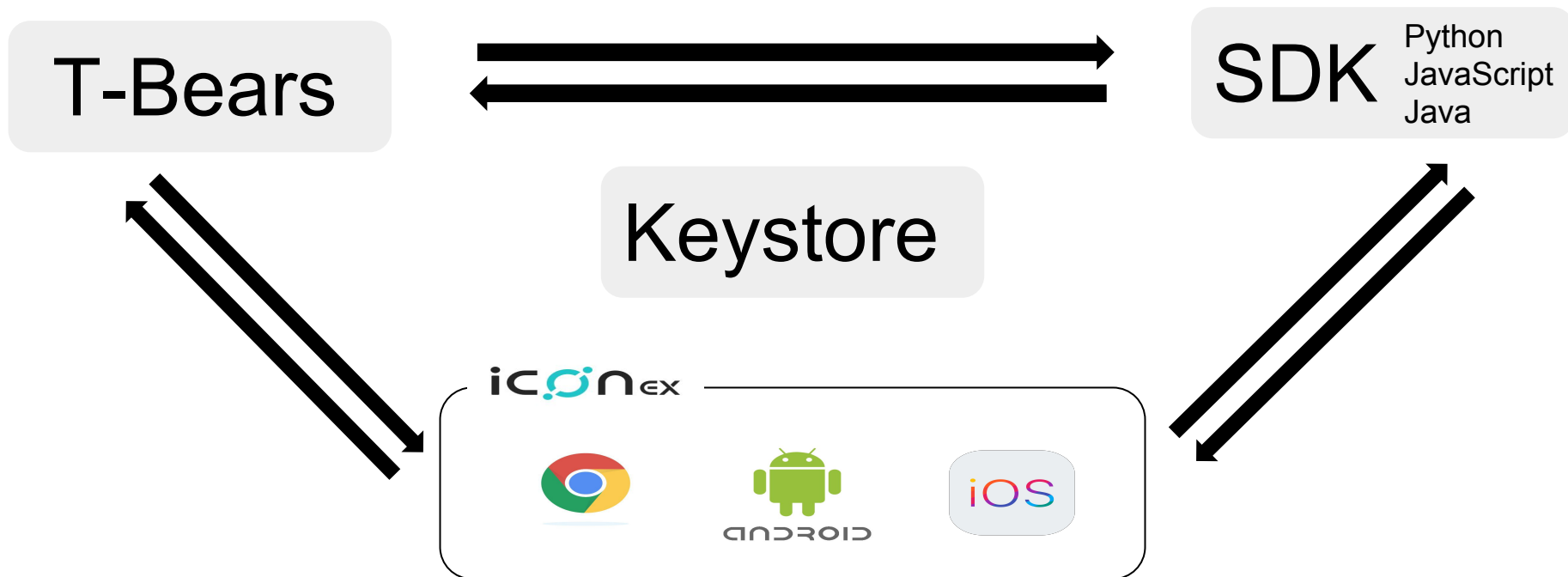
nara

```
{
  "address": "hx97207a7079d94c6cbd5fcd41e2a1365336c313f7",
  "crypto": {
    "cipher": "aes-128-ctr",
    "cipherparams": {
      "iv": "3cea78acc466443cb3f0e6bfa5788437"
    },
    "ciphertext": "9b166d680d6fb8b27f0897192784083da1d2c3158d2da54f0316e4e0d0705179",
    "kdf": "scrypt",
    "kdfparams": {
      "dklen": 32,
      "n": 16384,
      "r": 1,
      "p": 8,
      "salt": "612b2605b071c7a7e35eddf693a0cf7b"
    },
    "mac": "474fe97854640736d1506eac8295c58916d5bfb2cfc682c9f1b38cd38cc883a3",
    "id": "8bef1d4e-3446-4457-9a53-51bd4a20bff0",
    "version": 3,
    "coinType": "icx"
  }
}
```

ftp://192.168.0.1:21 / With SSID : icon

# T-Bears 활용 : keystore

Keystore file



# T-Bears 사용해 보기

- SCORE 프로젝트를 생성하는 `tbears init <projectName> <scoreClass>`
- SCORE를 Deploy 하는 `tbears deploy <projectName>`
- 트랜잭션 결과를 확인하는 `tbears txresult <txhash>`
- SCORE의 메서드 들을 알려주는 `tbears scoreapi <scoreAddress>`
- 메서드를 Call 해 보는 `tbears call <call.json>`

# T-Bears 활용

## 2. T-Bears 에서 제공하는 SCORE 살펴보기

### a. init 명령어를 통해 SCORE 프로젝트를 생성한다.

```
$ tbears init project SCORE_CLASS
```

```
root@9f9a7a309c60:/home/init_test# tree .
```

```
.
├── keystore_test1
├── project
│   ├── __init__.py
│   ├── package.json
│   └── project.py
├── tbears_cli_config.json
└── tbears_server_config.json
```

## b. 배포할 SCORE 살펴보기

```
$ tbears init project SCORE_CLASS
```

project.py

```
from iconservice import *

TAG = 'SCORE_CLASS'

class SCORE_CLASS(IconScoreBase):

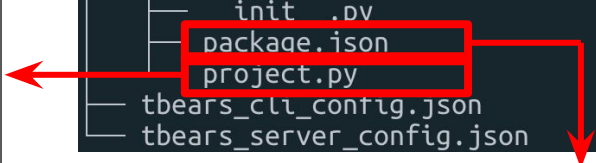
    def __init__(self, db: IconScoreDatabase) -> None:
        super().__init__(db)

    def on_install(self) -> None:
        super().on_install()

    def on_update(self) -> None:
        super().on_update()

    @external(readonly=True)
    def hello(self) -> str:
        Logger.debug(f'Hello, world!', TAG)
        return "Hello"
```

```
root@9f9a7a309c60:/home/init_test# tree .
.
├── keystore_test1
├── project
│   ├── init .py
│   ├── package.json
│   └── project.py
├── tbears_cli_config.json
└── tbears_server_config.json
```



package.json

```
{
  "version": "0.0.1",
  "main_file": "project",
  "main_score": "SCORE_CLASS"
}
```

## b. 배포할 SCORE 살펴보기

```
$ tbears init project SCORE_CLASS
```

project.py

```
from iconservice import *

TAG = 'SCORE_CLASS'

class SCORE_CLASS(IconScoreBase):

    def __init__(self, db: IconScoreDatabase) -> None:
        super().__init__(db)

    def on_install(self) -> None:
        super().on_install()

    def on_update(self) -> None:
        super().on_update()

    @external(readonly=True)
    def hello(self) -> str:
        Logger.debug(f'Hello, world!', TAG)
        return "Hello"
```

```
root@9f9a7a309c60:/home/init_test# tree .
.
├── keystore_test1
├── project
│   ├── init .py
│   ├── package.json
│   └── project.py
├── tbears_cli_config.json
└── tbears_server_config.json
```

package.json

```
{
  "version": "0.0.1",
  "main_file": "project",
  "main_score": "SCORE_CLASS"
}
```

ftp://192.168.0.1:21 / With SSID : icon

## c. 생성된 프로젝트를 T-Bears 에 배포하기

```
# tbears deploy project/
```

```
root@9f9a7a309c60:/home/init_test# tbears deploy project/
Send deploy request successfully.
If you want to check SCORE deployed successfully, execute txresult command
transaction hash: 0x939910bb11490c99fc42a1f2d04fafed134a07e658330ab12e293e8bf681ba65
root@9f9a7a309c60:/home/init_test#
```

```
root@9f9a7a309c60:/home/init_test# tree .
.
├── keystore_test1
├── project
│   ├── __init__.py
│   ├── package.json
│   └── project.py
├── tbears_cli_config.json
└── tbears_server_config.json
```

**ftp://192.168.0.1:21 / With SSID : icon**

#### d. 트랜잭션 결과 확인하기

```
$ tbears txresult <transaction hash>
```

[illegible]



## e. SCORE 가 제공하는 함수 목록 확인하기

배포한 SCORE의 주소로 해당 SCORE에서 호출할 수 있는 메서드의 확인이 가능하다.

```
# tbears scoreapi <scoreAddress>
```

```
root@9f9a7a309c60:/home/nara_test/nara_send_tx_# tbears scoreapi cx91e69b0135700166b14b84ce7e213174717e2fdd
```

```
SCORE API: [
  {
    "type": "fallback",
    "name": "fallback",
    "inputs": []
  },
  {
    "type": "function",
    "name": "hello",
    "inputs": [],
    "outputs": [
      {
        "type": "str"
      }
    ],
    "readonly": "0x1"
  },
  {
    "type": "function",
    "name": "name_Call",
    "inputs": [
      {
        "name": "_name",
        "type": "str"
      }
    ],
    "outputs": []
  },
  {
    "type": "function"
```

ftp://192.168.0.1:21 / With SSID : icon

# SCORE 자세히 알아보기

IconScoreBase

iconservice

Decorator

Audit

## 배포한 SCORE 파헤치기

```
project.py

from iconservice import *

TAG = 'SCORE_CLASS'

class SCORE_CLASS(IconScoreBase):

    def __init__(self, db: IconScoreDatabase) -> None:
        super().__init__(db)

    def on_install(self) -> None:
        super().on_install()

    def on_update(self) -> None:
        super().on_update()

    @external(readonly=True)
    def hello(self) -> str:
        Logger.debug(f'Hello, world!', TAG)
        return "Hello"
```

iconservice만 import 가능

IconScoreBase를 상속

IconScoreDatabase 를 DB 사용.  
인스턴스가 호출될 때 마다  
실행되어서 변수를 초기화 해 주는  
메서드

Decorator를 활용한 메서드 속성  
지정

## 배포한 SCORE 파헤치기 - IconScoreBase property

### project.py

```
from iconservice import *

TAG = 'SCORE_CLASS'

class SCORE_CLASS(IconScoreBase):

    def __init__(self, db: IconScoreDatabase) -> None:
        super().__init__(db)

    def on_install(self) -> None:
        super().on_install()

    def on_update(self) -> None:
        super().on_update()

    @external(readonly=True)
    def hello(self) -> str:
        Logger.debug(f'Hello, world!', TAG)
        return "Hello"
```

### IconScoreBase property

- **icx**
  - icx.transfer [Return only T]
  - icx.send [Return T/F]
- **db**
- **address**
- **owner**
- **block\_height**

## 배포한 SCORE 파헤치기 - IconScoreBase property

### project.py

```
from iconservice import *

TAG = 'SCORE_CLASS'

class SCORE_CLASS(IconScoreBase):

    def __init__(self, db: IconScoreDatabase) -> None:
        super().__init__(db)

    def on_install(self) -> None:
        super().on_install()

    def on_update(self) -> None:
        super().on_update()

    @external(readonly=True)
    def hello(self) -> str:
        Logger.debug(f'Hello, world!', TAG)
        return "Hello"
```

### IconScoreBase property

- **msg**
  - msg.sender
  - msg.value [ICX Amount]
- **tx**
  - tx.origin
  - tx.index
  - tx.hash
  - tx.timestamp
  - tx.nonce

## 배포한 SCORE 파헤치기 - iconservice utility function

### project.py

```
from iconservice import *  
  
TAG = 'SCORE_CLASS'  
  
class SCORE_CLASS(IconScoreBase):  
  
    def __init__(self, db: IconScoreDatabase) -> None:  
        super().__init__(db)  
  
    def on_install(self) -> None:  
        super().on_install()  
  
    def on_update(self) -> None:  
        super().on_update()  
  
    @external(readonly=True)  
    def hello(self) -> str:  
        Logger.debug(f'Hello, world!', TAG)  
        return "Hello"
```

### iconservice utility function

- **revert**  
DB상태변화 취소
- **sha3\_256**  
입력데이터 해싱
- **json\_dumps**
- **json\_loads**  
json 객체와 파이썬 객체 변환

## 배포한 SCORE 파헤치기

```
project.py

from iconservice import *

TAG = 'SCORE_CLASS'

class SCORE_CLASS(IconScoreBase):

    def __init__(self, db: IconScoreDatabase) -> None:
        super().__init__(db)

    def on_install(self) -> None:
        super().on_install()

    def on_update(self) -> None:
        super().on_update()

    @external(readonly=True)
    def hello(self) -> str:
        Logger.debug(f'Hello, world!', TAG)
        return 'Hello'
```

SCORE가 배포될 때 실행되는 메서드

SCORE를 업데이트 할 때 실행되는 메서드

'hello'라는 외부에서 호출 가능한 메서드  
readonly = True 라서, 읽기만 가능

Logger.[옵션] 로그 표출정도,  
T-Bears에서만 옵션 조절을 통한 확인 가능

## 배포한 SCORE 파헤치기 - Decorator

### project.py

```
from iconservice import *

TAG = 'SCORE_CLASS'

class SCORE_CLASS(IconScoreBase):

    def __init__(self, db: IconScoreDatabase) -> None:
        super().__init__(db)

    def on_install(self) -> None:
        super().on_install()

    def on_update(self) -> None:
        super().on_update()

    @external(readonly=True)
    def hello(self) -> str:
        Logger.debug(f'Hello, world!', TAG)
        return "Hello"
```

### Decorator

#### @external

외부에서 호출 가능

#### @payable

외부의 송금 수신 가능

#### @eventlog

트랜잭션의 결과를

txresult의 eventlog에 삽입



ftp://192.168.0.1:21 / With SSID : icon

## 배포한 SCORE 파헤치기 - fallback

```
root@9f9a7a309c60:/home/nara_test/nara_send_tx_# tbears scoreapi cx91e69b0135700166b14b84ce7e213174717e2fdd
```

```
{  
  "type": "fallback",  
  "name": "fallback",  
  "inputs": []  
},
```

fallback : 데이터가 없는 ICX전송이 일어났을 때  
불러지는 메서드, 직접 구현하여서 기능 추가가능

fallback이 구현되어 있지 않으면, ICX를 송금받을 수  
없습니다.

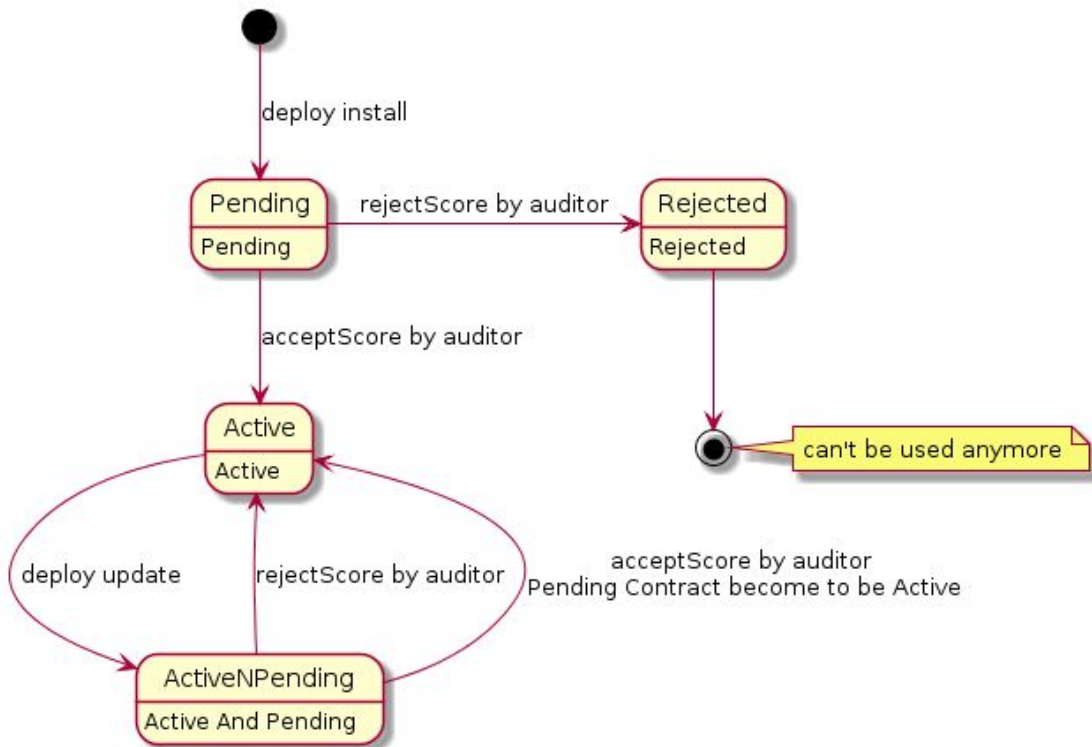
```
{  
  "type": "function",  
  "name": "hello",  
  "inputs": [],  
  "outputs": [  
    {  
      "type": "str"  
    }  
  ],  
  "readonly": "0x1"  
},  
{  
  "type": "function",  
  "name": "name_Call",  
  "inputs": [  
    {  
      "name": "_name",  
      "type": "str"  
    }  
  ],  
  "outputs": []  
},  
{  
  "type": "function"
```

ftp://192.168.0.1:21 / With SSID : icon

# Mainnet 에 배포한 SCORE는 바로 Deploy 되나요?

## Audit

SCORE State Diagram



ftp://192.168.0.1:21 / With SSID : icon

# 올바른 SCORE 를 위한 규칙

Timeout  
Unfinishing loop

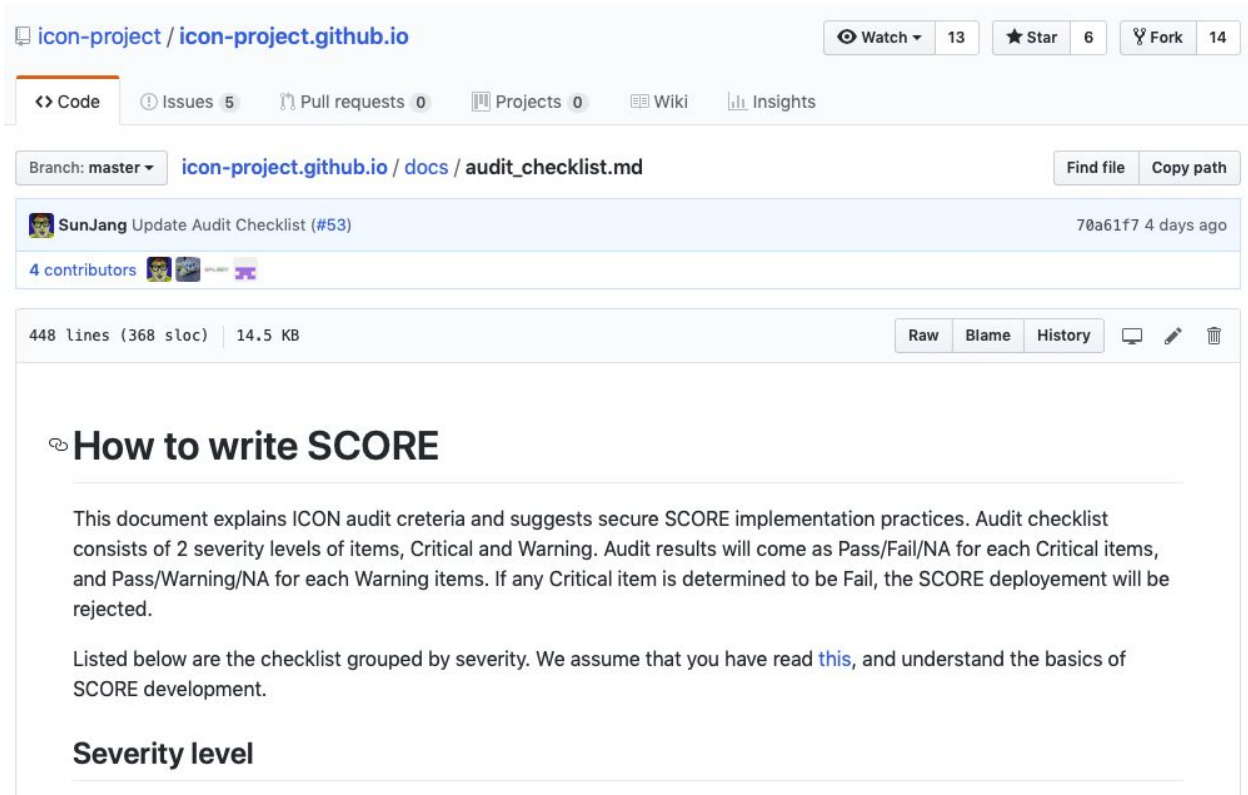
Package import  
System call

Randomness  
Outbound network call

IRC2 Token Standard compliance  
IRC2 Token parameter name

Eventlog on Token Transfer  
Eventlog without Token Transfer

ICXTransfer Eventlog  
Big Number Operation  
Instance Variable



The screenshot shows the GitHub interface for the 'icon-project' repository. The file 'audit\_checklist.md' is selected, showing its commit history and content. The commit history indicates it was updated by SunJang (#53) 4 days ago. The file content is titled 'How to write SCORE' and explains the audit criteria and implementation practices for SCORE. It mentions that the audit checklist consists of 2 severity levels of items, Critical and Warning, and that audit results will be Pass/Fail/NA for each Critical item, and Pass/Warning/NA for each Warning item. It also lists the checklist grouped by severity and mentions the assumption that the reader has read 'this' (a link) and understands the basics of SCORE development.

icon-project / icon-project.github.io

Watch 13 Star 6 Fork 14

Code Issues 5 Pull requests 0 Projects 0 Wiki Insights

Branch: master icon-project.github.io / docs / audit\_checklist.md Find file Copy path

SunJang Update Audit Checklist (#53) 70a61f7 4 days ago

4 contributors

448 lines (368 sloc) 14.5 KB Raw Blame History

## How to write SCORE

This document explains ICON audit criteria and suggests secure SCORE implementation practices. Audit checklist consists of 2 severity levels of items, Critical and Warning. Audit results will come as Pass/Fail/NA for each Critical items, and Pass/Warning/NA for each Warning items. If any Critical item is determined to be Fail, the SCORE deployment will be rejected.

Listed below are the checklist grouped by severity. We assume that you have read [this](#), and understand the basics of SCORE development.

### Severity level

자료 출처 : [https://github.com/icon-project/icon-project.github.io/blob/master/docs/audit\\_checklist.md](https://github.com/icon-project/icon-project.github.io/blob/master/docs/audit_checklist.md)

## 올바른 SCORE 를 위한 규칙 - Timeout

### Timeout

```
# Bad
```

```
@external
```

```
def airdrop_token(self, _value: int, _data: bytes = None):
```

```
    for target in self._very_large_targets:
```

```
        self._transfer(self.msg.sender, target, _value, _data)
```

```
# Good
```

```
@external
```

```
def airdrop_token(self, to: Address, value: int, data: bytes = None):
```

```
    if self._airdrop_sent_address[_to]:
```

```
        self.revert(f"Token was dropped already: {_to}")
```

```
    self._airdrop_sent_address[_to] = True
```

```
    self._transfer(self.msg.sender, _to, _value, _data)
```

ftp://192.168.0.1:21 / With SSID : icon

## 올바른 SCORE 를 위한 규칙 - Package import, System

..

Package import  
System call

```
# Bad  
import os
```

```
# Good  
from iconservice import *  
from .myclass import *
```

ftp://192.168.0.1:21 / With SSID : icon

## 올바른 SCORE 를 위한 규칙 - Randomness

Randomness  
Outbound network call

```
# Bad
# each node may have different outcome
won = datetime.datetime.now() % 2 == 0
```

```
# Bad
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(host, port)
```

ftp://192.168.0.1:21 / With SSID : icon

# 올바른 SCORE 를 위한 규칙 - Standard compliance

IRC2 Token Standard compliance  
IRC2 Token parameter name

```
# IRC2 functions
@external(readonly=True)
def name(self) -> str:

@external(readonly=True)
def symbol(self) -> str:

@external(readonly=True)
def decimals(self) -> int:

@external(readonly=True)
def totalSupply(self) -> int:

@external(readonly=True)
def balanceOf(self, _owner: Address) -> int:

@external
def transfer(self, _to: Address, _value: int, _data: bytes=None):
```

## 올바른 SCORE 를 위한 규칙 - Event log

```
# Good
```

```
@eventlog(indexed=3)
```

```
def Transfer(self, _from: Address, _to: Address, _value: int, _data: bytes):  
    pass
```

```
@external
```

```
def transfer(self, _to: Address, _value: int, _data: bytes = None):  
    self._balances[self.msg.sender] -= _value  
    self._balances[_to] += _value  
    self.Transfer(self.msg.sender, _to, _value, _data)
```

Eventlog on Token Transfer  
Eventlog without Token Transfer  
ICXTransfer Eventlog



ftp://192.168.0.1:21 / With SSID : icon

## 올바른 SCORE 를 위한 규칙 - instance Variable

```
# Bad
def __init__(self, db: IconScoreDatabase) -> None:
    super().__init__(db)

@external
def update_organizer(self, _organizer: Address) ->
None:
    self._organizer = _organizer

@external
def get_organizer(self) -> Address:
    return self._organizer
```

```
# Good
def __init__(self, db: IconScoreDatabase) -> None:
    super().__init__(db)
    self._organizer = VarDB(self._ORGANIZER, db,
value_type=Address)

@external
def update_organizer(self, _organizer: Address) -> None:
    self._organizer.set(_organizer)

@external
def get_organizer(self) -> Address:
    return self._organizer.get()
```

Instance Variable

# Review

## SCORE Deploy(배포)의 과정

### 1. SCORE 작성

- tbears init <projectName> <scoreClass>

### 2. SCORE 배포

- tbears deploy <projectName>

### 3. Deploy 결과확인

- tbears txresult <txhash>

### 4. SCORE 함수 목록 조회해 보기

- tbears scoreapi <scoreAddress>

### 5. SCORE call 해 보기

- tbears call <call.json>

### 6. “Goods\_event” SCORE 같이 완성해보기

# Next step

## 1. SCORE 작성

- tbears init <projectName> <scoreClass>

## 2. SCORE 배포

- tbears deploy <projectName>

## 3. Deploy 결과확인

- tbears txresult <txhash>

## 4. SCORE 함수 목록 조회해 보기

- tbears scoreapi <scoreAddress>

## 5. SCORE call 해 보기

- tbears call <call.json>

## 6. “Goods\_event” SCORE 같이 완성해보기



**Github repo에  
결과물 Pull Request**

ftp://192.168.0.1:21 / With SSID : icon

# Let's Do it !

문제가 생기면, 손을 들어주세요

## 5. SCORE 같이 만들어보기(Lucky Draw)

# SCORE 같이 완성하기

**Goods\_event.py** 를 같이 완성합니다.

진행이 어려운분은 손을 들어주세요

**ftp://192.168.0.1:21 / With SSID : icon**

**Q & A**