Dive into ICON - SCORE

ICON foundation

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. SCORE Implementation Guide

Step 3. SCORE Samples

Step 4. SCORE Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

1. What is SCORE
2. Characteristics of SCORE

Step 2. SCORE Implementation Guide

Step 3. SCORE Samples

Step 4. SCORE Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. SCORE Implementation Guide

1. Token & Crowdsale
2. IconScoreBase abstract methods
3. ...

Step 3. SCORE Samples

Step 4. SCORE Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. SCORE Implementation Guide

Step 3. SCORE Samples

1. Coin Flip
2. Simplified Blackjack
3. SCORE Style Guide

Step 4. SCORE Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. SCORE Implementation Guide

Step 3. SCORE Samples

Step 4. SCORE Hands-on Exercise

1. What do we do today ?
2. Development & QnA

1. Smart Contract - SCORE

Smart Contract - SCORE

- What is Smart Contract

1.1 What is SCORE

Definition of SCORE

- SCORE in Dictionary : the number of points, goals, etc. achieved in a game or competition (from Cambridge Dictionary)

- ICON SCORE : Abbreviation of Smart Contract on Reliable Environment
- Definition of SCORE : Smart contract running on ICON network

1.2 Characteristics of SCORE

Characteristics

- SCORE is written in python
- Uploaded as compressed binary data on the blockchain
- SCORE can be updated. SCORE address remains the same after update.

- SCORE code size is limited to about 64 KB (actually bounded by the maximum stepLimit value during its deploy transaction) after compression.
- SCORE must follow sandbox policy : file system access or network API calls are prohibited.

2. SCORE
Implementation Guide

SCORE Implementation Guide

- ICON Developers Portal
  - https://www.icondev.io/docs/overview

- iconservice API references
  - https://icon-project.github.io/score-guide/api-references.html

2.1 Token & Crowdsale

Token & Crowdsale

- SCORE by example

  https://www.icondev.io/docs/token-crowdsale

- Every SCORE classes must inherit **IconScoreBase** class.

2.2 IconScoreBase abstract methods

IconScoreBase abstract methods

- IconScoreBase (The highest parent class)

  https://www.icondev.io/docs/syntax#section-iconscorebase-the-highest-parent-class-

- Abstract methods : **`__init__, on_install, on_update`**

- Parent's function must be called as follows.

  `super().__init__(db) / on_install() / on_update()`

2.3 DB abstraction

DB abstraction

- VarDB, DictDB, ArrayDB

  https://www.icondev.io/docs/syntax#section-vardb-dictdb-arraydb

- Supported value types : **`int, str, Address, bytes`**

2.4 Decorator, fallback

Decorator

- Decorator (external, eventlog, payable)

  https://www.icondev.io/docs/syntax#section-external-decorator-external-

  external : **`@external`** can be called from outside the contract

  eventlog : **`@eventlog`** decorator will include logs in its TxResult as 'eventlogs'.

  payable : **`@payable`** decorator are permitted to transfer icx coins.

fallback

- fallback

  https://www.icondev.io/docs/syntax#section-fallback

  The function is executed whenever the contract receives plain icx coins without data.

2.5 Type hints, exception handling

Type hints, exception handling

- Type hints

  https://www.icondev.io/docs/syntax#section-type-hints

  Type hinting is highly recommended for the input parameters and return value.

- Exception handling

  https://www.icondev.io/docs/syntax#section-exception-handling

  Recommend to use revert function when handling exceptions in contract.

2.6 Global functions

Global functions

- Global functions

  https://www.icondev.io/docs/syntax#section-api-functions

- json_dumps, json_loads : Converts python object ⇔ JSON string.
- sha3_256 : Hash Algorithm for SCORE development.
- revert : Rollback all the changes in the state DB in current transaction.

2.7 InterfaceScore

InterfaceScore

- InterfaceScore

  https://www.icondev.io/docs/syntax#section-interfacescore

- Get InterfaceScore object by using IconScoreBase's built-in function :

  `create_interface_score('score address', 'interface class')`

2.8 Limitations

Limitations

- Limitations

  https://icon-project.github.io/score-guide/limitation.html

- The maximum limit of the total count of call, interface call and ICX transfer/send is 1024 in one transaction.

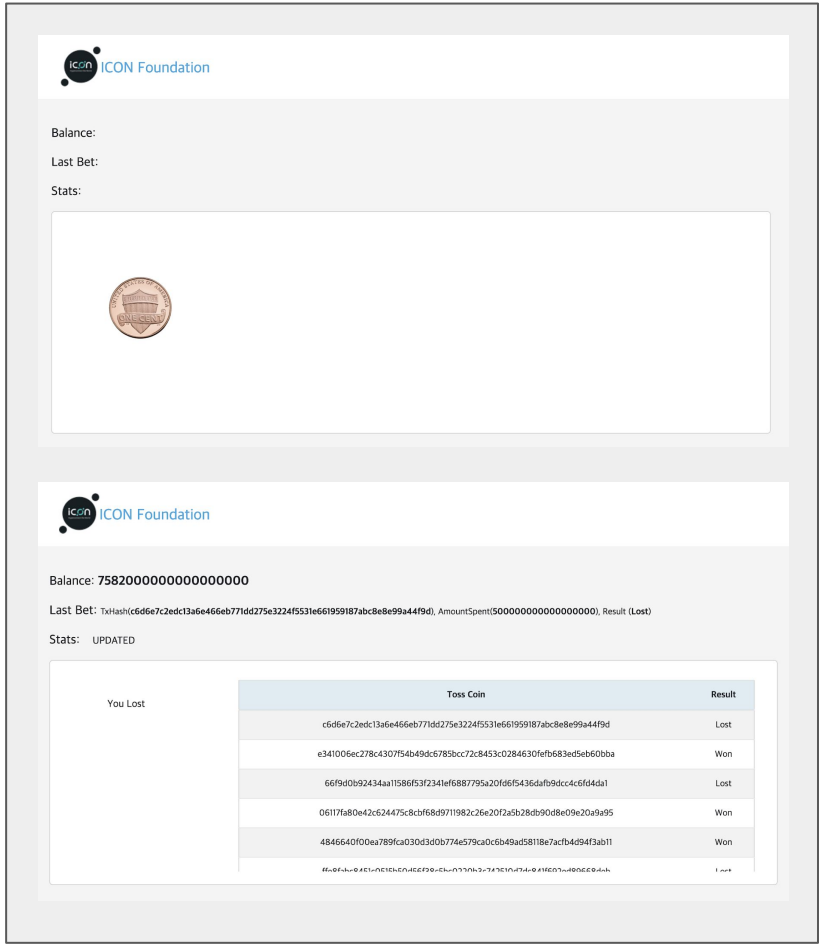
- The limitation of stack size increment by calling external SCORE is 64 in one transaction.

- Declaring member variables which not managed by states is prohibited.

3. SCORE Samples

SCORE Samples

- Coin Flip
  - SCORE + Front-End
- Simplified Blackjack
  - SCORE



3.1 Coin Flip

Coin Flip (ICON Dice Roll)

- Overview : Coin flip game using random generation. Supports single play.
- SCORE source : GitHubGist

  https://gist.github.com/hx57/cc8a027a596e1e3676d59a6193d62c58#file-diceroll-py

- DApp source : Medium Post

  https://medium.com/@2infiniti/icon-dapp-from-a-z-part-3-icon-dice-roll-dapp-7f0ca72057f5

- Demo : https://dapps.icon.support/icon-dice-roll/

3.2 Simplified Blackjack

Simplified Blackjack

- Overview : Sample SCORE implementing simplified blackjack game. Supports Player vs Player game.

- SCORE source : GitHub repo

  https://github.com/icon-workshops/Dive-into-ICON-2-SCORE/tree/master/samplegame

3.3 SCORE Style Guide

SCORE Style Guide

External Functions : ICON SCORE Sytle Guide

- **Function name** : camelCase
- **Parameters of function** : _camelCase
- **SCORE params (on_install)** : _camelCase
- **Function with eventlog decorator** : PascalCase

SCORE Style Guide

Internal Functions : PEP 8

- **Function name** : snake_case, _snake_case
- **Parameters of function** : snake_case
- **Variables** : snake_case, _snake_case

4. SCORE
Hands-on Exercise

4.1 What do we do today ?

What do we do today ?

- Declare & initialize variables
- Modify data into state DB
- Load data from the state DB
- Inter-SCORE function call (InterfaceScore)

- SCORE integration test (Optional)

Declare & initialize variables

- Declare variables : __init__
  1. Variable will be used as argument for create_interface_score. (VarDB)
  2. Variable for SCORE modified time as a iterable list. (ArrayDB)
  3. Variable for SCORE status`[SCORE_NAME, INTRODUCTION, APIS]`. (DictDB)

- Initialize variables : on_install
  1. Set the initial address value and save into VarDB.
  2. Set the initial status of SCORE and save into DictDB.
  3. Add the creation time to ArrayDB.

Modify data into state DB

- Change the introduction of SCORE in DictDB.
- Add the modified time to ArrayDB when the SCORE states change.

Load data from the state DB

- Retrieve status of SCORE from DictDB.
- Retrieve the list of state modified time.

Inter-SCORE function call (InterfaceScore)

- Deploy the HelloWorld which has only one external method. (Optional)
- Set the value of VarDB to designated SCORE address.
  (Can be skipped if you set the proper address value in on_install method)
- Create the InterfaceScore class of designated SCORE
- Call designated SCORE external function

SCORE integration test (Optional)

- SCORE integration test
  - Write test codes for SCORE external functions

4.2 Development & QnA

Dive into ICON - Appendix

Appendix A. Development Resources

Development Resources & Communities

- GitHub
- Developer Portal
- ICON Improvement Proposal

- Facebook 한국 개발자 그룹
- Medium 블로그
- Youtube 채널

GitHub https://github.com/icon-project

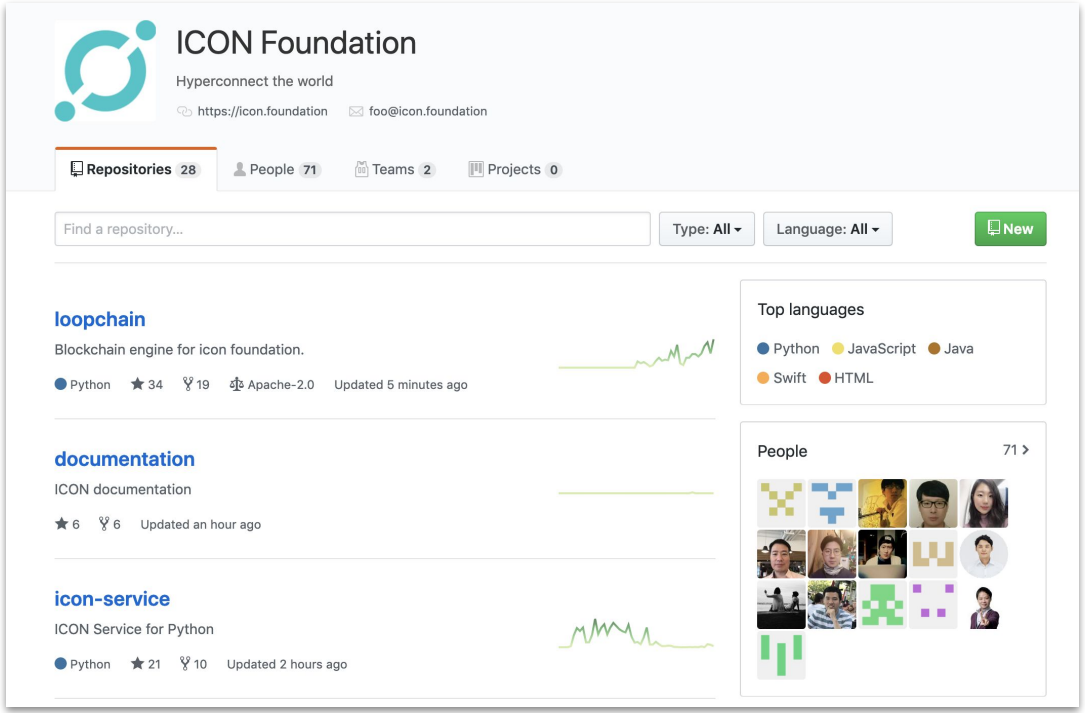
- loopchain
- icon-service
- icon-rpc-server

Node

- t-bears
- icon-sdk-python
- icon-sdk-java
- icon-sdk-js

Dev tools

- iconex_chrome_extension
- documentation
- ...



Developer Portal https://www.icondev.io

- Community portal for ICON DApp ecosystem

Announcement

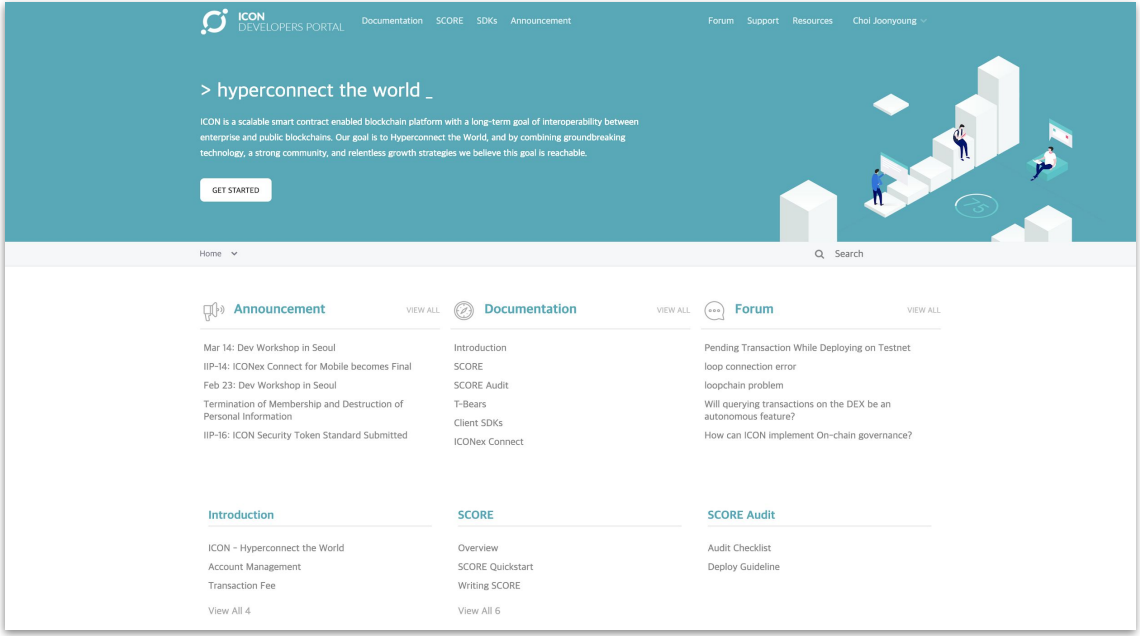
Documentation

Forum

Introduction

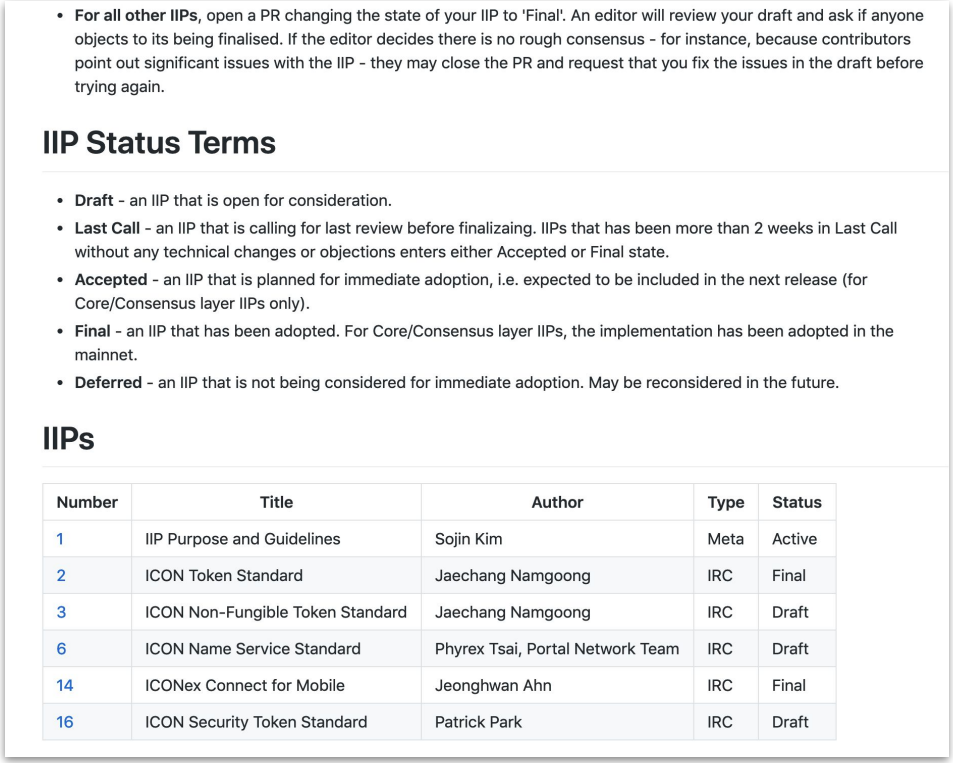
SCORE

SCORE Audit



ICON Improvement Proposal https://github.com/icon-project/IIPs

- IIP describes a standard for ICON platform.
- Anyone can prompt suggestions and discussions on new functions or improvement.
- Selected items will be implemented on ICON network.



- **For all other IIPs**, open a PR changing the state of your IIP to 'Final'. An editor will review your draft and ask if anyone objects to its being finalised. If the editor decides there is no rough consensus - for instance, because contributors point out significant issues with the IIP - they may close the PR and request that you fix the issues in the draft before trying again.

IIP Status Terms

- **Draft** - an IIP that is open for consideration.
- **Last Call** - an IIP that is calling for last review before finalizaing. IIPs that has been more than 2 weeks in Last Call without any technical changes or objections enters either Accepted or Final state.
- **Accepted** - an IIP that is planned for immediate adoption, i.e. expected to be included in the next release (for Core/Consensus layer IIPs only).
- **Final** - an IIP that has been adopted. For Core/Consensus layer IIPs, the implementation has been adopted in the mainnet.
- **Deferred** - an IIP that is not being considered for immediate adoption. May be reconsidered in the future.

IIPs

Number	Title	Author	Type	Status
1	IIP Purpose and Guidelines	Sojin Kim	Meta	Active
2	ICON Token Standard	Jaechang Namgoong	IRC	Final
3	ICON Non-Fungible Token Standard	Jaechang Namgoong	IRC	Draft
6	ICON Name Service Standard	Phyrex Tsai, Portal Network Team	IRC	Draft
14	ICONex Connect for Mobile	Jeonghwan Ahn	IRC	Final
16	ICON Security Token Standard	Patrick Park	IRC	Draft

Communities

- **Facebook 한국 개발자 그룹 : Dive into ICON**

  https://www.facebook.com/groups/DiveintoICON/

- **Medium 블로그 : B!ock.Chain**

  https://medium.com/b-ock-chain

- **Youtube 채널 : ICON developers**

  https://www.youtube.com/channel/UC8h4kVV7w94xmfCz6FbwHhg

Summary

- asdf
- GitHub
- Developer Portal
- ICON Improvement Proposal

- Facebook 한국 개발자 그룹
- Medium 블로그
- Youtube 채널