Dive into ICON - SCORE

ICON foundation

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. Implementation Guide

Step 3. SCORE Samples

Step 4. Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

1. What is SCORE
2. Characteristics of SCORE

Step 2. Implementation Guide

Step 3. SCORE Samples

Step 4. Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. Implementation Guide

1. IconScoreBase abstract methods
2. ...

Step 3. SCORE Samples

Step 4. Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. Implementation Guide

Step 3. SCORE Samples

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

Step 4. Hands-on Exercise

Dive into ICON - SCORE

Step 1. Smart Contract - SCORE

Step 2. Implementation Guide

Step 3. SCORE Samples

Step 4. Hands-on Exercise

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

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.

Implementation Guide

Implementation Guide

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

- iconservice API references
    - https://iconservice.readthedocs.io/en/latest/

- How to develop SCORE with PyCharm IDE (for Linux or macOS)
    - https://medium.com/b-ock-chain/score-with-pycharm-ce-4c295068c9aa

2.1 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()`

2.2 DB abstraction

DB abstraction

- VarDB, DictDB, ArrayDB

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

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

2.3 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.

2.4 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.

2.5 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.

2.6 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’)`

2.7 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.

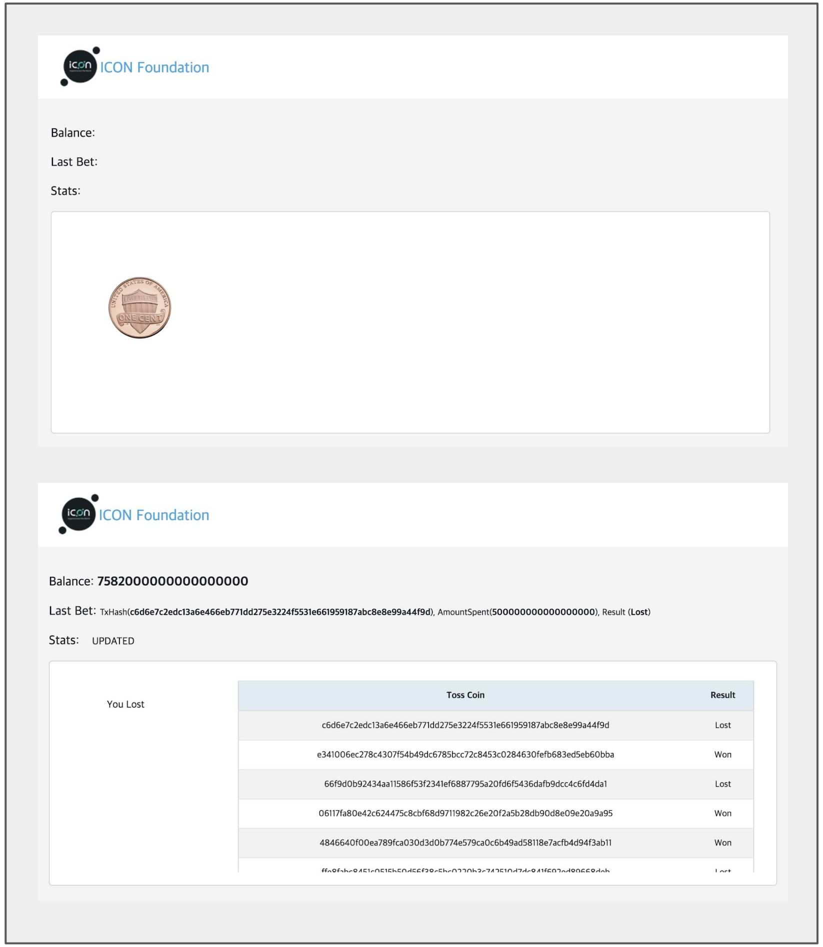
- Declaring member variables which not managed by states is prohibited.

SCORE Samples

SCORE Samples

- Coin Flip
  - SCORE + Front-End

- Simplified Blackjack
  - SCORE



ICON Foundation

Balance:

Last Bet:

Stats:

ICON Foundation

Balance: **7582000000000000000**

Last Bet: TxHash(**c6d6e7c2edc13a6e466eb771dd275e3224f5531e661959187abc8e8e99a44f9d**), AmountSpent(**5000000000000000000**), Result (**Lost**)

Stats: UPDATED

You Lost

Toss Coin	Result
c6d6e7c2edc13a6e466eb771dd275e3224f5531e661959187abc8e8e99a44f9d	Lost
e341006ec278c4307f54b49dc6785bcc72c8453c0284630fefb683ed5eb60bba	Won
66f9d0b92434aa11586f53f2341ef6887795a20fd6f5436dafb9dcc4c6fd4da1	Lost
06117fa80e42c624475c8cbf68d9711982c26e20f2a5b28db90d8e09e20a9a95	Won
4846640f00ea789fca030d3d0b774e579ca0c6b49ad58118e7acfb4d94f3ab11	Won
ffe8fabc8451c0515b50d56f38c5bc0220b3c742510d7dc841f692ed89668deb	Lost

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

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]`. (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.

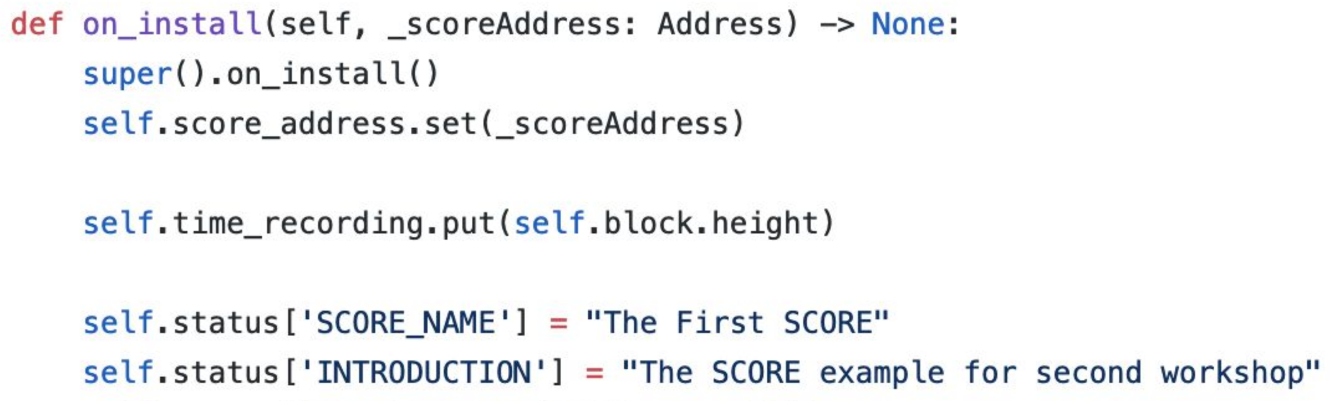
Declare & initialize variables

- Declare variables : __init__

```
def __init__(self, db: IconScoreDatabase) -> None:
    super().__init__(db)
    self._db = db
    self.score_address = VarDB("SCORE_ADDRESS", db, value_type=Address)
    self.status = DictDB("STATUS", db, value_type=str)
```

Declare & initialize variables

- Initialize variables : on_install



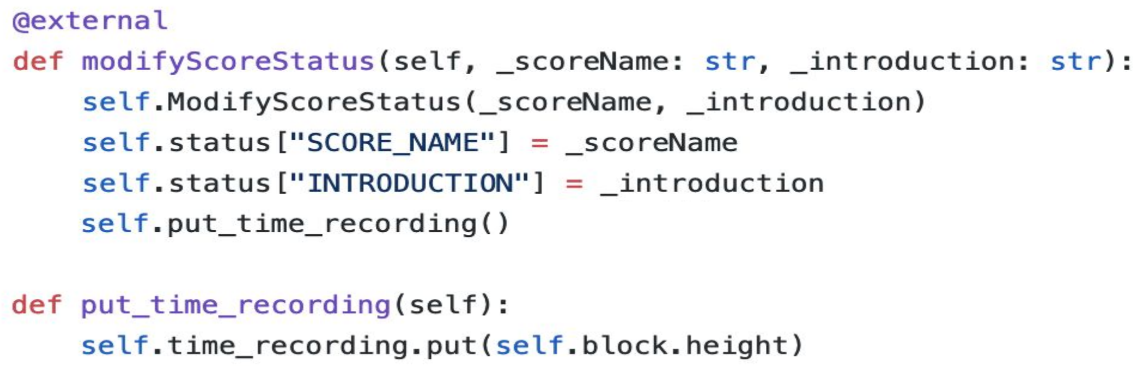
```
def on_install(self, _scoreAddress: Address) -> None:
    super().on_install()
    self.score_address.set(_scoreAddress)

    self.time_recording.put(self.block.height)

    self.status['SCORE_NAME'] = "The First SCORE"
    self.status['INTRODUCTION'] = "The SCORE example for second workshop"
```

Update SCORE state DB

- Modify the `SCORE_NAME, INTRODUCTION` of SCORE in DictDB.
- Add the modified time (block height) to ArrayDB when the SCORE states change.



```
@external
def modifyScoreStatus(self, _scoreName: str, _introduction: str):
    self.ModifyScoreStatus(_scoreName, _introduction)
    self.status["SCORE_NAME"] = _scoreName
    self.status["INTRODUCTION"] = _introduction
    self.put_time_recording()

def put_time_recording(self):
    self.time_recording.put(self.block.height)
```

Load data from the state DB

- Retrieve status of SCORE from DictDB.

```
@property
def score_status(self):
    score_name = self.status['SCORE_NAME']
    introduction = self.status['INTRODUCTION']
    return "SCORE_NAME : " + score_name + "\n" + "INTRODUCTION : " + introduction
```

- Retrieve the list of state modified time from ArrayDB.

```
@external(readonly=True)
def getTimeRecording(self) -> str:
    time_recorded = [time_recorded for time_recorded in self.time_recording]
    return str(time_recorded)

@property
def time_recording(self):
    return ArrayDB("TIME_RECORDING", self._db, value_type=int)
```

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)

```
@external
def setScoreAddress(self, _scoreAddress: Address):
    self.score_address.set(_scoreAddress)
```

Inter-SCORE function call (InterfaceScore)

- Create the InterfaceScore class of designated SCORE

```
class WorkshopInterface(InterfaceScore):
    @interface
    def hello(self):
        pass
```

- Call designated SCORE external function

```
@property
def interface_score_address(self):
    return self.score_address.get()

@external(readonly=True)
def interfaceScoreExercise(self) -> str:
    workshop_score = self.create_interface_score(self.interface_score_address, WorkshopInterface)
    result = workshop_score.hello()
    return result
```

SCORE integration test (Optional)

- SCORE integration test : Test sample for exercise
  - Write test codes for SCORE external functions

4.2 Development & QnA

Dive into ICON - Appendix

Development Resources

Development Resources & Communities

- GitHub
- Developer Portal
- ICON Improvement Proposal

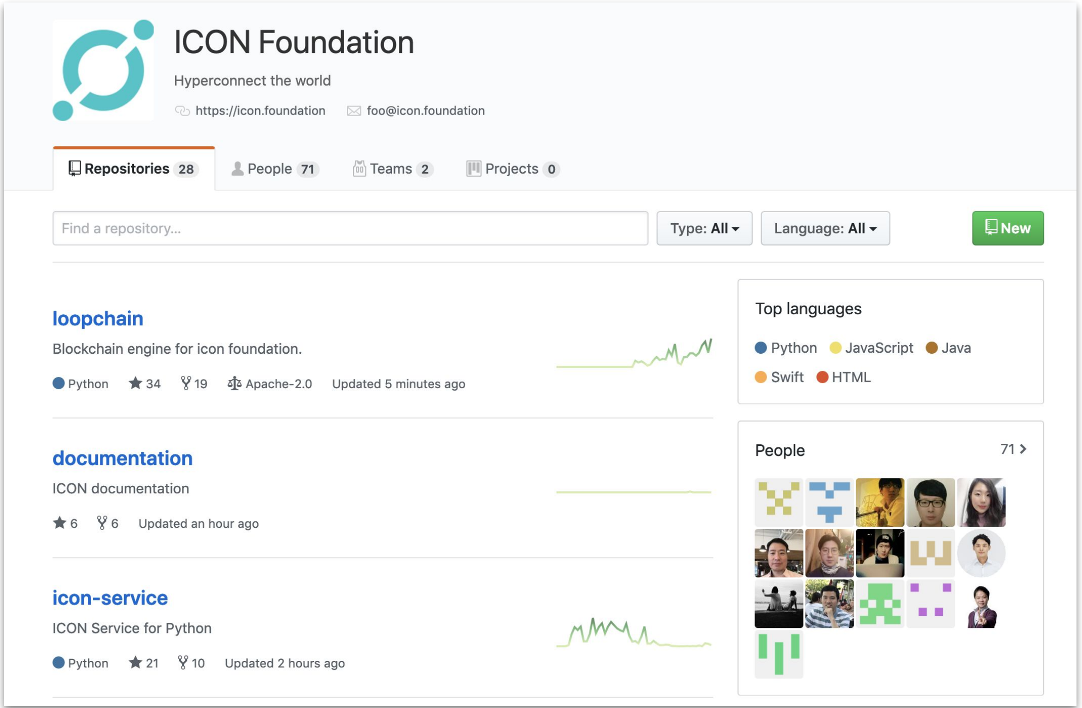
- Facebook 한국 개발자 그룹
- Medium 블로그
- Youtube 채널

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

- loopchain
- icon-service
- icon-rpc-server
- t-bears
- icon-sdk-python
- icon-sdk-java
- icon-sdk-js
- iconex_chrome_extension
- documentation
- ...

Node

Dev tools



ICON Foundation

Hyperconnect the world

https://icon.foundation foo@icon.foundation

Repositories 28 People 71 Teams 2 Projects 0

Find a repository... Type: All Language: All New

loopchain

Blockchain engine for icon foundation.

● Python ★ 34 19 Apache-2.0 Updated 5 minutes ago

documentation

ICON documentation

★ 6 6 Updated an hour ago

icon-service

ICON Service for Python

● Python ★ 21 10 Updated 2 hours ago

Top languages

● Python ● JavaScript ● Java ● Swift ● HTML

People 71

Developer Portal https://www.icondev.io

- Community portal for ICON DApp ecosystem

Announcement

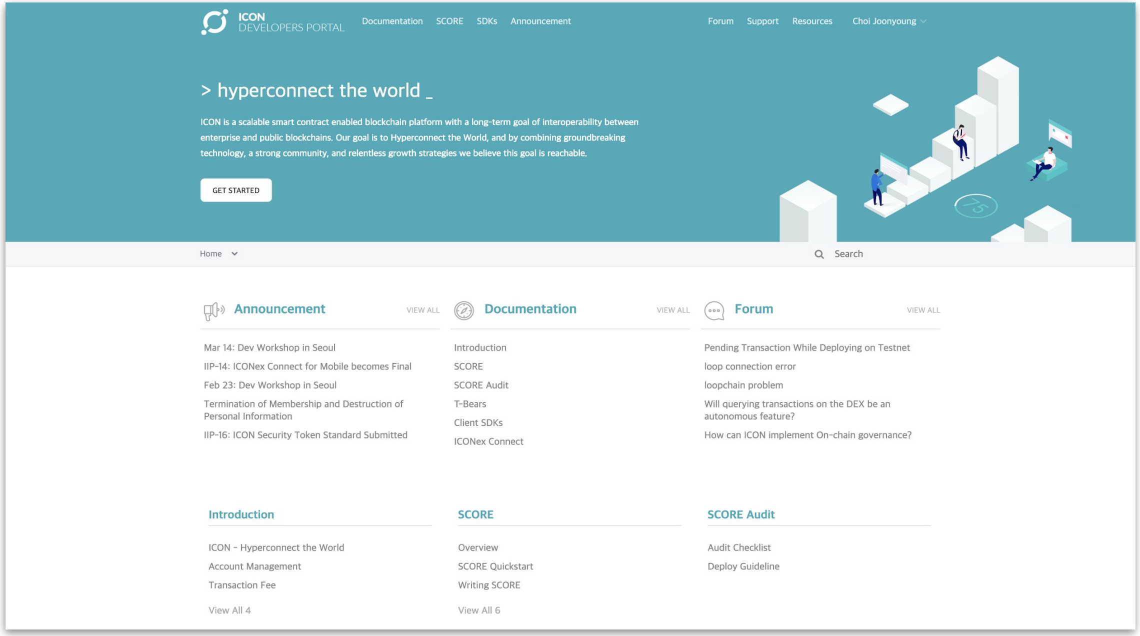
Documentation

Forum

Introduction

SCORE

SCORE Audit



> hyperconnect the world _

ICON is a scalable smart contract enabled blockchain platform with a long-term goal of interoperability between enterprise and public blockchains. Our goal is to Hyperconnect the World, and by combining groundbreaking technology, a strong community, and relentless growth strategies we believe this goal is reachable.

Announcement
- Mar 14: Dev Workshop in Seoul
- IIP-14: ICONex Connect for Mobile becomes Final
- Feb 23: Dev Workshop in Seoul
- Termination of Membership and Destruction of Personal Information
- IIP-16: ICON Security Token Standard Submitted

Documentation
- Introduction
- SCORE
- SCORE Audit
- T-Bears
- Client SDKs
- ICONex Connect

Forum
- Pending Transaction While Deploying on Testnet
- loop connection error
- loopchain problem
- Will querying transactions on the DEX be an autonomous feature?
- How can ICON implement On-chain governance?

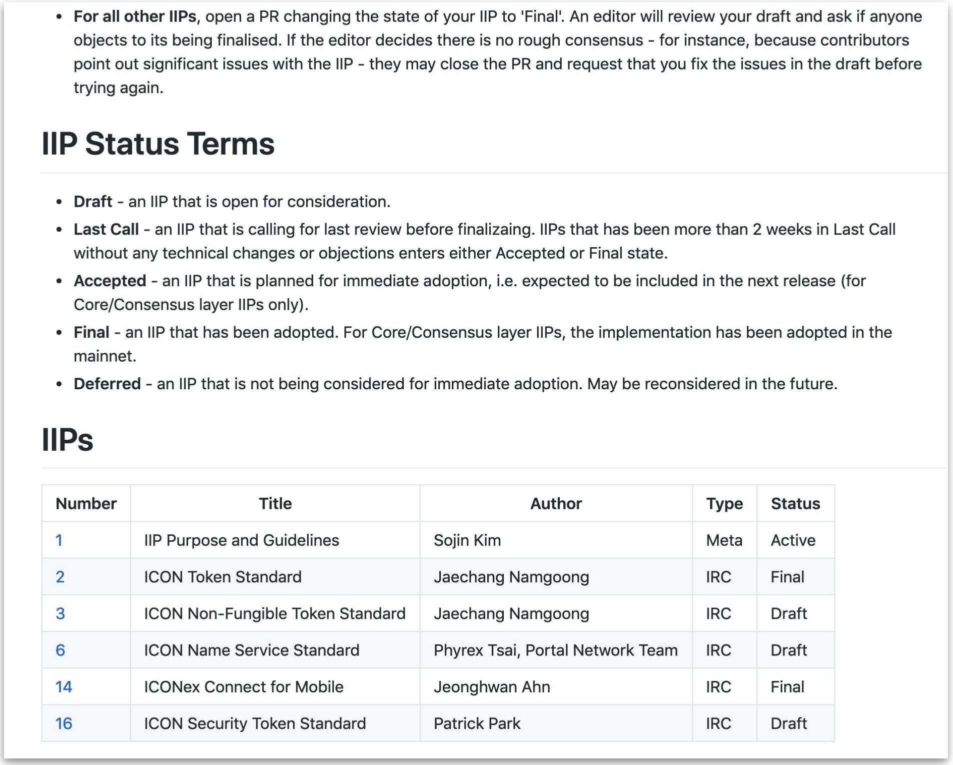
Introduction
- ICON - Hyperconnect the World
- Account Management
- Transaction Fee
- View All 4

SCORE
- Overview
- SCORE Quickstart
- Writing SCORE
- View All 6

SCORE Audit
- Audit Checklist
- Deploy Guideline

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

Thank you