ICONService API References

Feb 11, 2020

Contents

This document describes the API specification of `ICONService` for SCORE Developers. This document does not describe all APIs of `ICONService`, it contains the specifications of classes, decorators, global functions just for writing SCOREs.

Note: If you have just started writing a SCORE, please visit SCORE Guide first.

CHAPTER 1

Contents

1.1 Classes

These are the API classes for writing SCOREs. The key class is `IconScoreBase`. You can start writing a SCORE from it.

1.1.1 Address

class Address(*address_prefix: iconservice.base.address.AddressPrefix*, *address_body: bytes*, *ignore_length_validate: bool = False*)

Address class

body

Returns 20-byte address body part

Returns 20 byte data standing for address

static from_bytes(*buf: bytes*) → Optional[iconservice.base.address.Address]

Create Address object from bytes data

Parameters **buf** – `bytes` bytes data including Address information

Returns *Address*

static from_bytes_including_prefix(*buf: bytes*) → Optional[iconservice.base.address.Address]

static from_data(*prefix: iconservice.base.address.AddressPrefix*, *data: bytes*) → Optional[iconservice.base.address.Address]

creates an address object using given body bytes

Parameters

- **prefix** – address prefix
- **data** – 20-bytes address body

Returns *Address*

static from_string(*address: str*)
creates an address object from given 42-char string *address*

Returns *Address*

is_contract
Whether the address is SCORE

Returns True(contract) False(Not contract)

prefix
Returns address prefix part

Returns *AddressPrefix* AddressPrefix.EOA(0) or AddressPrefix.CONTRACT(1)

to_bytes() → bytes
Returns data as bytes from the address object

Returns `bytes` data including information of Address object

to_bytes_including_prefix() → bytes

1.1.2 AddressPrefix

class AddressPrefix
Enumeration of Address prefix

- CONTRACT: Contract Account
- EOA: Externally Owned Account

CONTRACT = 1

EOA = 0

1.1.3 ArrayDB

class ArrayDB(*var_key: K*, *db: IconScoreDatabase*, *value_type: type*)
Utility classes wrapping the state DB. ArrayDB supports length and iterator, maintains order.

K [int, str, Address, bytes]

V [int, str, Address, bytes, bool]

get(*index: int = 0*) → V
Gets the value at index

Parameters **index** – index

Returns value at the index

pop() → Optional[V]
Gets and removes last added value

Returns last added value

put(*value: V*) → None
Puts the value at the end of array

Parameters **value** – value to add

1.1.4 DictDB

class DictDB(*var_key: K, db: Union[IconScoreDatabase, IconScoreSubDatabase], value_type: type, depth: int = 1*)

Utility classes wrapping the state DB. DictDB behaves more like python dict. DictDB does not maintain order.

K [int, str, Address, bytes]

V [int, str, Address, bytes, bool]

remove(*key: K*) → None

Removes the value of given key

Parameters **key** –

1.1.5 IconScoreBase

class IconScoreBase(*db: iconservice.database.db.IconScoreDatabase*)

A base class of SCOREs. This class provides facilities and environments to SCORE to run.

__init__(*db: iconservice.database.db.IconScoreDatabase*) → None

A Python init function. Invoked when the contract is loaded at each node. Do not put state-changing works in here.

address

The current SCORE address

Returns *Address* current address

block

Deprecated property

Use block_height and now() instead.

block_height

Current block height

Returns current block height

call(*addr_to: iconservice.base.address.Address*, *func_name: str*, *kw_dict: dict*, *amount: int = 0*)

Calls an external function provided by another SCORE. *func_name* can be *None* if fallback calls

Parameters

- **addr_to** – *Address* the address of another SCORE
- **func_name** – function name of another SCORE
- **kw_dict** – Arguments of the external function
- **amount** – ICX value to enclose with. in loop.

Returns returning value of the external function

static create_interface_score(*addr_to: iconservice.base.address.Address, interface_cls: Callable[[Address], T]*) → T

Creates an object, through which you have an access to the designated SCORE's external functions.

Parameters

- **addr_to** – SCORE address
- **interface_cls** – interface class

Returns An instance of given class

db
An instance used to access state DB

Returns *IconScoreDatabase* db

fallback() → None
fallback function can not be decorated with *@external*. (i.e., fallback function is not allowed to be called by external contract or user.) This fallback function is executed whenever the contract receives plain icx coins without data. If the fallback function is not decorated with *@payable*, it is not listed on the SCORE APIs also cannot be called.

icx
An object used to transfer icx coin

- icx.transfer(addr_to(address), amount(integer)) -> bool Transfers designated amount of icx coin to `addr_to`. If exception occurs during execution, the exception will be escalated. Returns True if coin transfer succeeds.
- icx.send(addr_to(address), amount(integer)) -> bool Sends designated amount of icx coin to `addr_to`. Basic behavior is same as transfer, the difference is that exception is caught inside the function. Returns True when coin transfer succeeded, False when failed.

Returns *Icx* instance of icx

msg
Holds information of calling the SCORE

- msg.sender : Address of the account who called this function. If other contact called this function, msg.sender points to the caller contract's address.
- msg.value : Amount of icx that the sender attempts to transfer to the current SCORE.

now() → int
Timestamp of current block in microseconds

Returns timestamp in microseconds

on_install(***kwargs*) → None
Invoked when the contract is deployed for the first time, and will not be called again on contract update or deletion afterward. This is the place where you initialize the state DB.

on_update(***kwargs*) → None
Invoked when the contract is deployed for update. This is the place where you migrate old states.

owner
Address of the account who deployed the contract

Returns *Address* owner address

static revert(*message: Optional[str] = None*, *code: int = 0*)
Deprecated method

Use global function *revert()* instead.

tx
Holds information of the transaction

Returns *Transaction* transaction

1.1.6 IconScoreDatabase

class IconScoreDatabase(*address: Address*, *context_db: ContextDatabase*, *prefix: bytes = None*)
It is used in IconScore

IconScore can access its states only through IconScoreDatabase

delete(*key: bytes*)
Deletes the key/value pair for the specified key.

Parameters **key** – key to delete

get(*key: bytes*) → bytes
Gets the value for the specified key

Parameters **key** – key to retrieve

Returns value for the specified key, or None if not found

get_sub_db(*prefix: bytes*) → iconservice.database.db.IconScoreSubDatabase
Returns sub db with a prefix

Parameters **prefix** – The prefix used by this sub db.

Returns sub db

put(*key: bytes*, *value: bytes*)
Sets a value for the specified key.

Parameters

- **key** – key to set
- **value** – value to set

1.1.7 Icx

class Icx(*context: IconScoreContext*, *address: Address*)
Class for handling ICX coin transfer These functions are intended to be used for SCORE development.

get_balance(*address: Address*) → int
Returns the ICX balance of given address

Parameters **address** – address

Returns ICX balance of given address

send(*addr_to: Address*, *amount: int*) → bool
transfer the amount of icx to the given 'addr_to'

Parameters

- **addr_to** – receiver address
- **amount** – the amount of icx to transfer

Returns True(success) False(failed)

transfer(*addr_to: Address*, *amount: int*) → None
transfer the amount of icx to the given 'addr_to' If failed, an exception will be raised

Parameters

- **addr_to** – receiver address
- **amount** – the amount of icx to transfer

1.1.8 Transaction

class Transaction(*tx_hash: Optional[bytes] = None*, *index: int = 0*, *origin: Optional[Address] = None*, *to: Optional[Address] = None*, *timestamp: int = None*, *nonce: int = None*)

Holds information of the transaction

hash
Transaction hash

index
Transaction index in a block

nonce
(optional) nonce of a transaction request. random value

origin
The account who created the transaction.

timestamp
Timestamp of a transaction request in microseconds This is NOT a block timestamp

to
The account of tx to.

1.1.9 VarDB

class VarDB(*var_key: K*, *db: IconScoreDatabase*, *value_type: type*)

Utility classes wrapping the state DB. VarDB can be used to store simple key-value state.

K [int, str, Address, bytes]

V [int, str, Address, bytes, bool]

get() → Optional[V]
Gets the value

Returns value of the var db

remove() → None
Deletes the value

set(*value: V*) → None
Sets the value

Parameters **value** – a value to be set

1.2 Decorators

These decorators are pre-defined by `ICONService` for interacting between SCOREs-users or SCOREs-SCOREs.

eventlog(*func=None*, ***, *indexed=0*)

Functions with *@eventlog* decorator will include logs in its TxResult as 'eventlogs'. If indexed parameter is set in the decorator, designated number of parameters in the order of declaration will be indexed and included in the Bloom filter. Indexed parameters and non-indexed parameters are separately stored in TxResult. Possible data types for function parameters are primitive types (int, str, bytes, bool, Address).

It is recommended to declare a function without implementation body. Even if the function has a body, it does not be executed. When declaring a function, type hinting is a must. Without type hinting, transaction will fail.

The default value for the parameter can be set. At most 3 parameters can be indexed, And index can't exceed the number of parameters(will raise an error).

Parameters **indexed** – the number of indexed parameters count(maximum 3)

external (*func=None*, ***, *readonly=False*)
A decorator for the function whether the function exposes externally. If declared to the function, EOA or another SCORE can call it. These functions are registered on the exportable API list. Any attempt to call a non-external function from outside the contract will fail.

If a function is decorated with 'readonly' parameters, i.e., *@external(readonly=True)*, the function will have read-only access to the state DB. This is similar to view keyword in Solidity. If the read-only external function is also decorated with *@payable*, the function call will fail. Duplicate declaration of @external will raise an exception on import time.

Parameters **readonly** – True if the function have read-only access to the state DB.

interface (*func*)
A decorator for the functions of interface SCORE.

Declaring this decorator to the function can invoke the same form of the function of the external SCORE.

payable (*func*)
A decorator for the external function.

If the decorator is declared to the external function, it can receive the ICXs and process further works for it. If ICXs (msg.value) are passed to a non-payable function, that transaction will fail.

1.3 Global Functions

The Global functions can be used in anywhere of the SCORE project. Some functions can charge more STEPs due to the usage of CPU.

create_address_with_key (*public_key: bytes*) → Optional[iconservice.base.address.Address]
Create an address with a given public key

Parameters **public_key** – Public key based on secp256k1

Returns Address created from a given public key or None if failed

json_dumps (*obj: Any*) → str
Converts a python object *obj* to a JSON string

Parameters **obj** – a python object to be converted

Returns json string

json_loads (*src: str*) → Any
Parses a JSON string *src* and converts it to a python object

Parameters **src** – a JSON string to be converted

Returns a python object

recover_key (*msg_hash: bytes*, *signature: bytes*, *compressed: bool = True*) → Optional[bytes]
Returns the public key from message hash and recoverable signature

Parameters

- **msg_hash** – 32 bytes data
- **signature** – signature_data(64) + recovery_id(1)

- **compressed** – the type of public key to return

Returns public key recovered from msg_hash and signature (compressed: 33 bytes key, uncompressed: 65 bytes key)

revert (*message: Optional[str] = None*, *code: int = 0*) → None

Reverts the transaction and breaks. All the changes of state DB in current transaction will be rolled back.

Parameters

- **message** – revert message
- **code** – code

sha3_256 (*data: bytes*) → bytes

Computes hash using the input data

Parameters **data** – input data

Returns hashed data in bytes

CHAPTER 2

References

- SCORE Guide
- T-Bears

Index

Symbols

A

B

C

D

E

F

G

H

I

J

M

N

O

P

R

S

T

V