

EOS.IO

SMART CONTRACT DEV.

2018 EOS Developer Meetup, Seoul
mithrilcoin.io



Eric song
CTO at mithrilcoin.io

Table of contents

1. Wallet

2. VM on EOSIO

3. Features

4. EOSIO Account

5. Authorities And Permissions

6. eosd command

7. EOS Commander

8. EOSIO Smart Contract

8-1. Contract Skeleton

8-2. Types in Message

8-3. Contract 작성

8-3-1. header

8-3-2. abi

8-3-3. source

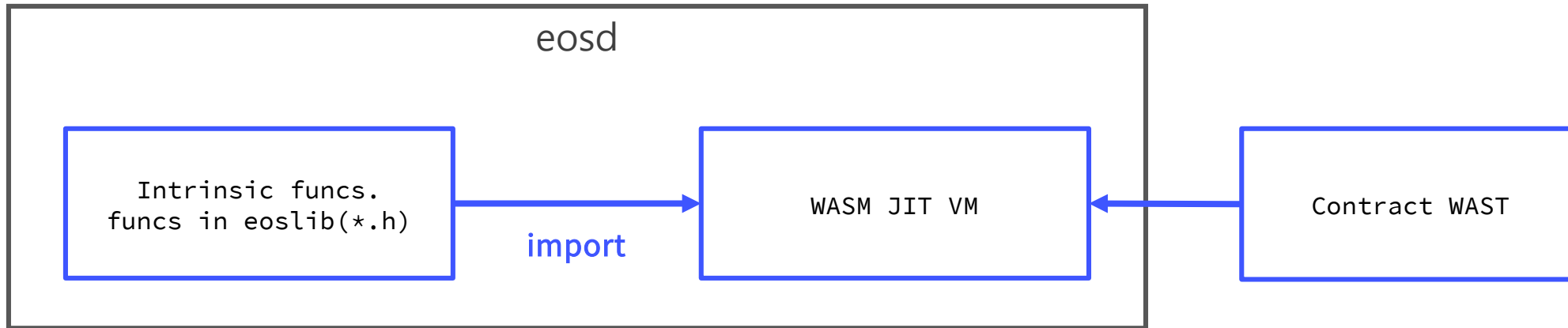
8-3-4. 작성 주의사항

8-4. eoscpp

9. Resources

- **Store keys**
 - Key(private,public) generation
 - Sign (ECDSA)
 - Broadcasts transactions to network
- **Curve params for signing**
 - secp256k1
 - EOS, Bitcoin, Ethereum, ...
 - secp256r1
 - EOS (since DAWN 3.0)
 - Supported by mobile device(iOS, Android) with special. hw.
 - This makes your mobile device hardware wallet!

- WebAssembly
- No need to learn web assembly, just use it.



- eoscpp (compile tool for EOS smart contract)
 - shell script
 - uses clang, llvm, llc, s2wasm

- C++
- **Message(Action), Transaction**
 - A message represents a single operation
 - A transaction is a collection of 1 or more messages
- **Limitation**
 - No floating point (float, double)
 - Transaction to be executed within 1 ms
 - Max tps: 30 tps per account (on current testnet)
- **Can update code at any time**
 - differ from ethereum.

- **No numerical address on EOS -> AccountName**
 - Human readable
 - Base32 encoding
 - ".12345abcdefghijklmnopqrstuvwxyz"
 - Max 13 자리, 마지막은 ".12345abcdefghi" 만
 - uint64 로 packing (fast !)
 - Sort 가 쉬운구조
- **ToString**
 - eosio::name(), N()

- 어떤 메시지의 권한이 올바른지 결정함
- 모든 account 는 2개의 native permission 있음
 - Owner (key)
 - Account 의 소유권, active permission 변경 권한
 - Active (key)
 - Owner 권한 이외의 대부분 권한
- Weight 와 threshold 지정 가능
 - Multi-sig 구현 가능
- User defined permission

- Tool for sending txn/querying state of eosd

- **eosioc** on DAWN3.x

- Create wallet, key

- ./create wallet
 - ./eosc create key

```
secp256k1 key( on DAWN2):  
public : EOS6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV  
private: 5KQwrPbwdL6PhXujxW37FSSQZ1JiwsST4cqQzDeyXtP79zkvFD3  
  
secp256r1 key( on DAWN3):  
Public:  EOSR16EPHFSKVYHBjQgxVGQPrwCxTg7BbZ69H9i4gztN9deKTEXTYne4  
private: EOSR1iyQmnyPEGvFd8uffnk152WC2WryBjgTrg22fXQryuGL9mU6qW
```

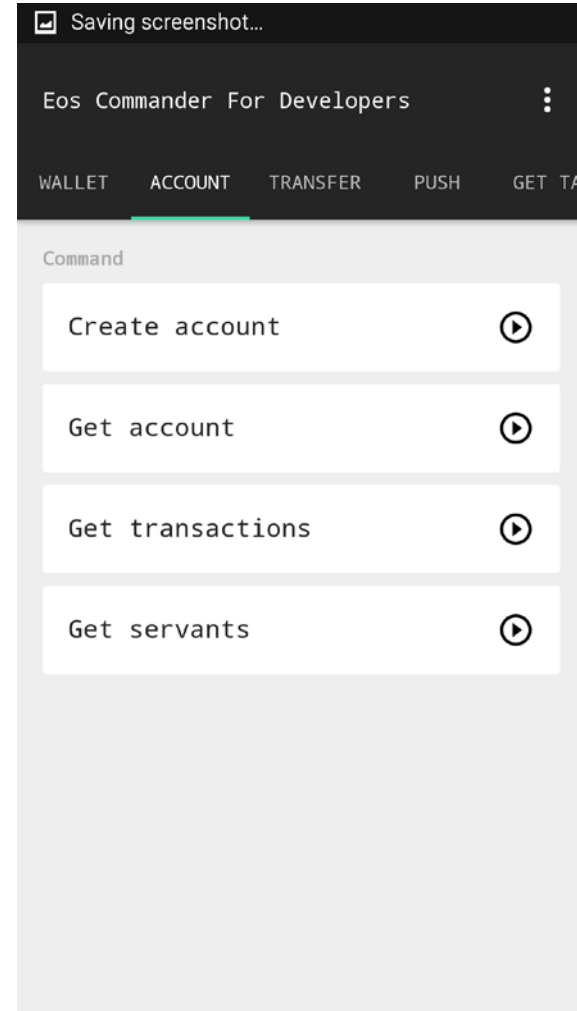
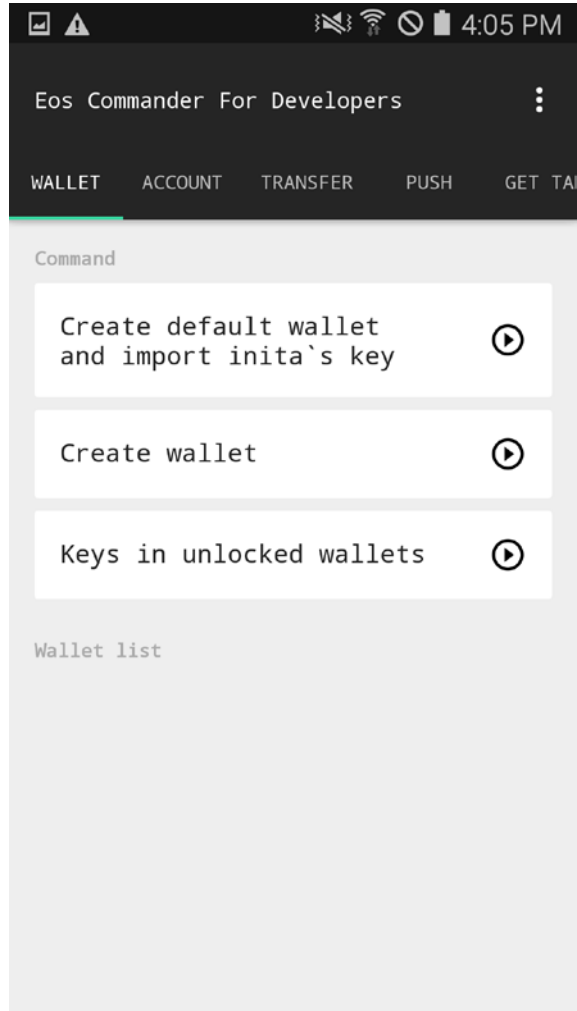
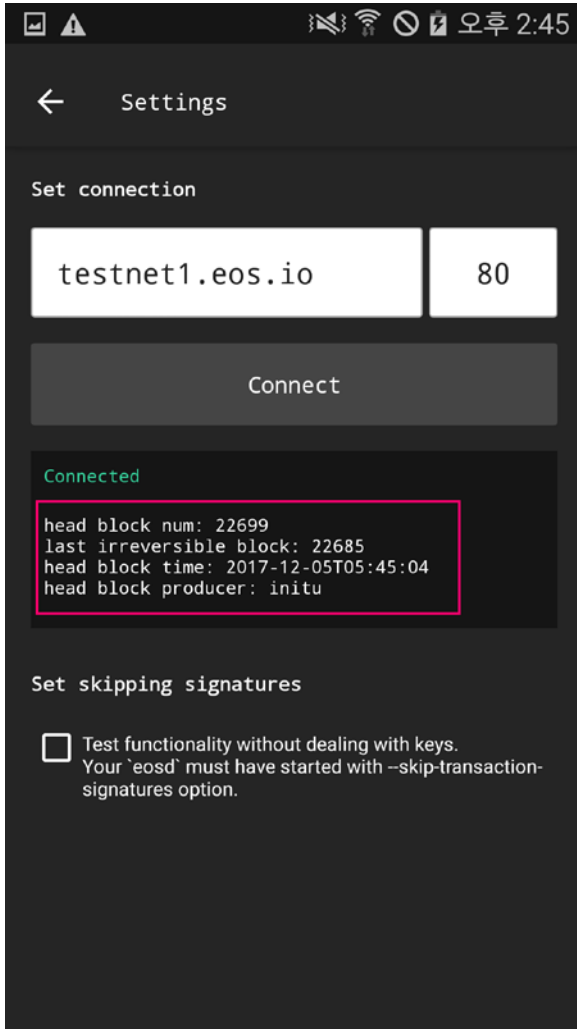
- create account

- ./eosc create account inita \$new_acc \$owner_pubK \$active_pubK

- **Transfer (eos)**
 - `./eosc transfer inita tester 1000`
- **set contract (update)**
 - `./eosc set contract currency ${wast_path} ${abi_path}`
- **send message (push transaction to contract)**
 - `./eosc push message(action) currency transfer '{"from":"currency","to":"tester","amount":50}'` ~~→~~
~~currency~~ ~~→ tester~~ -p currency@active
- **get table**
 - `./eosc get table tester currency account`

- **Support up to DAWN 2.x**
 - will support DAWN 3.x(+secp256r1 key) soon
- **Features**
 - built-in wallet function, offline signing
 - Create key/account, transfer, push, get table
 - Dynamic form ui for editing contract message
- **Resources**
 - <https://github.com/mithrilcoin-io/EosCommander>
 - <https://play.google.com/store/apps/details?id=io.mithrilcoin.eoscommander>

EOS Commander



EOS Commander

Eos Commander For Developers

LETACCOUNTTRANSFERPUSHGET TABLE

exchange1. input contract

2. get abi !
GET ABI

3. select action
Contract Action : buy

4. tap this!
FORM INPUTIMPORT JSON

Contract message
(Create message from ui or json file.)

Scope accounts (separate by space or

Action: buy

buyer (order_id)
name (account_name)
swapnibble
number (uint64)
12345

at_price (uint128)
123456

quantity (uint64)

expiration (time)PICKER

fill_or_kill (uint8)

OK

The image shows two overlapping screenshots from the EOS Commander app. The background screenshot shows the 'buy' action interface with fields for 'exchange', 'GET ABI', 'Contract Action' (set to 'buy'), and a JSON message editor. The JSON message is:

```
{  "buyer": {    "name": "swapnibble",    "number": 12345  },  "at_price": 123456,  "quantity": 100,  "expiration": "2018-01-17T06:39:37",  "fill_or_kill": 1}
```

. The foreground screenshot is a time picker dialog titled 'expiration' showing a date and time selection interface. The selected time is 'Today 06:39 AM'. A red arrow points from the 'OK' button in the time picker to the 'OK' button in the JSON editor.

Action: buy

buyer (order_id)

name (account_name)

swapnibble

expiration

01/15/2018 04 37

01/16/2018 05 38

Today 06 39 AM

01/18/2018 07 40 PM

01/19/2018 08 41

OK

Time picker supported.

message json is created.

OK

Eos Commander For Developers

LET ACCOUNT TRANSFER PUSH GET TABLE

exchange

GET ABI

Contract Action : buy

FORM INPUT IMPORT JSON

```
{  "buyer": {    "name": "swapnibble",    "number": 12345  },  "at_price": 123456,  "quantity": 100,  "expiration": "2018-01-17T06:39:37",  "fill_or_kill": 1}
```

Scope accounts (separate by space or

**SHOW ME THE
CODE**

AND STOP TALKING!

Contract Skeleton

```
eoscpp -n 컨트랙트이름
```

```
/**
 * The init() and apply() methods must have C calling convention so that the blockchain
 * can lookup and call these methods.
 */
extern "C" {

    /**
     * This method is called once when the contract is published or updated.
     */
    void init() {
        eosio::print( "Init World!\n" );
    }

    /// The apply method implements the dispatch of events to this contract
    void apply( uint64_t code, uint64_t action ) {
        eosio::print( "Hello World: ", eosio::name(code), "->", eosio::name(action), "\n" );
    }

} // extern "C"
```

- **type.h in eoslib**
 - 사용가능한 intrinsic type 정의가 있음.
- **Intrinsic type 주의사항**
 - uint32/int32 까지는 json 에 integer 로
 - uint64/int64 부터는 string 으로.
 - "amount": "12345678" or "coeff": "-789012891385621335"
 - Time 은 epoch time(int)이 아닌, string 으로.
 - http://en.wikipedia.org/wiki/ISO_8601
 - "2018-02-06T19:00:00"

- **hpp 파일엔 message/table구조 정의**
 - @abi action 으로 action name 지정
 - @abi table 로 table 구조 지정 (db)
- **cpp 파일에 코드 작성**
 - C++ name mangling 적용되지 않게 extern "C" 사용할것.
 - generate .wast by eoscpp
- **ABI (Application Binary Interface, json)**
 - Generated by eoscpp
 - Or write manually

Contract 작성 > header

```
#include <eoslib/eos.hpp>
#include <eoslib/db.hpp>

namespace proxy {

    //@abi action
    struct PACKED( set_owner ) {
        account_name owner;
        uint32_t      delay;
    };

    //@abi table
    struct config {
        config(){}
        const uint64_t      key = N(config); // key! Table 의 1번째 param 은 key 로 사용된다!
        account_name        owner = 0;
        uint32_t             delay = 0;
        uint32_t             next_id = 0;
    };

    using configs = eosio::table<N(proxy),N(proxy),N(configs),config,uint64_t>;

} /// namespace proxy
```

```
{
  "types": [{
    "new_type_name": "account_name",
    "type": "name"
  }
],
"structs": [{
  "name": "config",
  "base": "",
  "fields": [
    { "name": "key", "type": "name" },
    { "name": "owner", "type": "name" },
    { "name": "next_id", "type": "uint32" }
  ]
},{
  "name": "setowner",
  "base": "",
  "fields": [
    { "name": "owner", "type": "name" },
    { "name": "delay", "type": "uint32" }
  ]
}
],
"actions": [{
  "name": "setowner",
  "type": "setowner"
}
],
"tables": [{
  "name": "configs",
  "type": "config",
  "index_type": "i64",
  "key_names" : ["key"],
  "key_types" : ["name"]
}
]
}
```

```
extern "C" {

    void init() {
// 초기화 코드를 넣으세요. 예) 총 token 발행량, 초기 holder list 등
    }

    //
    void apply( uint64_t code, uint64_t action ) {

// 아래에서 N() 은 macro. String → eosio::account_name (uint64_t) 로 변환함.
// #define N(X) ::eosio::string_to_name(#X)

        if ( code == N(eosio) ) {
            if( action == N(transfer) ) {
                apply_transfer(code, unpack_action<native_currency::transfer>());
            } else if ( action == N(onerror)) {
                apply_onerror(deferred_transaction::from_current_action());
            }
        } else if (code == current_receiver() ) {
            if ( action == N(setowner)) {
                apply_setowner(current_action<set_owner>());
            }
        }
    }
}
```

- Refer to `/contracts/eoslib/*` files
- `external` (`stdlib` 포함) dependency 제거
 - 동일 기능을 별도 구현할 것.
- `mem alloc/dealloc` 주의(DAWN2.0 기준)
 - `eosio::malloc/free` 사용
 - `new/delete` overload (DAWN 3.0 에서 기본 제공됨)
- `std::string` -> `eosio::string` 으로 변경
 - `std::string::length()` -> `cstrlen(eosio::string::get_data())` 로
- `exception` 은 `assert()` 로
 - 무조건 throwing 은 `assert(0, "msg on throwing")` 형태로

- shell script
- wast generation

- eoscpp -o <source1> <source2>..

```
clang -emit-llvm -O3 --std=c++14 --target=wasm32 -ffreestanding -nostdlib -fno-threadsafe-statics -fno-rtti -fno-exceptions -I  
${EOSIO_INSTALL_DIR}/include -I $filePath -c $file -o $workdir/built/$name
```

- C++ -> llvm bit code -> llvm assembly-> wasm wast
 - stack size: 16384

- abi generation
 - eoscpp -g <hppfile>

Resources

- Github : <https://github.com/EOSIO/eos>
 - See wiki pages for tutorials and docs.
 - binary : <https://github.com/EOSIO/eos/releases>
- Request account on testnet
 - https://docs.google.com/forms/d/e/1FAIpQLSel3HVFb22zYaAJfUtu_IzFgIJ4OATb0jQ3H2FV-HbwnJ090g/viewform
- Community
 - <https://forums.eosgo.io/>
 - Telegram
 - <https://t.me/joinchat/EaEnSUPktgfol-XPfMYtcQ> (general dev.)
 - <https://t.me/EosGameDevelopers> (game dev.)