2 Creating a Simple Contract

When creating a contract you may note that most of the actions are typical. The differences lay in the implementation of the functions performed by the contract, which are implemented directly in the body of the contract. In this section instructions are given for creating a contract whose main function is to issue a greeting that comes in the form of «Hello, user».

2.1 Create a directory for contracts

Create a directory CONTRACTS_DIR, download the Contract Development Toolkit components necessary for compiling contracts in it.

```
cd CONTRACTS_DIR
git clone --recursive https://github.com/cyberway/cyberway.cdt --branch <bra
cd cyberway.cdt
./build.sh
sudo ./install.sh
```

2.2 Create the hello.cpp file

```
cd CONTRACTS_DIR
mkdir hello
cd hello
touch hello.cpp
```

Put «Hello, user» text message into hello.cpp file.

```
#include <eosiolib/eosio.hpp>
```

```
2
3  using namespace eosio;
4
5  class [[eosio::contract("hello")]] hello : public contract {
6    public:
7        using contract::contract;
8
9        [[eosio::action]]
10       void hi( name user ) {
11          print( "Hello, ", user);
12       }
13 };
14
15 EOSIO_DISPATCH( hello, (hi))
```

This action receives a parameter named «user» and displays a «Hello, user» message as a result. EOSIO_DISPATCH acts as a macro-operation to handle this action.

2.3 Compile hello.cpp

```
eosio-cpp -o hello.wasm hello.cpp --abigen
```

2.4 Set (unfold) contract

During the installation of the contract an account of this contract is created, as well as a public key of the account.

```
1  cleos wallet keys
2  cleos create account cyber hello <public key> -p cyber@active
```

A guide for creating a wallet as well as creating a development key can be found on the CyberWay website.

2.5 Set the absolute path to the created contract

Specify the absolute path `<contracts dir path>` to the contracts' directory in the following command:

```
cleos set contract hello CONTRACTS_DIR/hello -p hello@active
```

2.6 Check whether contract is running properly

To verify the operation of the contract you can call an action using user's name, for example, try sending a greeting to the user «Bob» using the following command:

```
cleos push action hello hi '["bob"]' -p hello@active
```

The operation is considered as successful if the following information is displayed on the monitor:

```
executed transaction: ...
#    hello.code <= hello.code::hi              {"user":"bob"}
>> Hello, bob
```

To expand the functions of the contract it is necessary to expand the logic of the hello.cpp file. It's the logic of the file file_name.cpp that determines the functionality of the contract being created.