# 编写 EOS 智能合约：发行代币

## EOS 编译、测试指南（二）

扫一扫关注欧链小秘书

EOS 团队于 2017 年 7 月 28 日推出了单机测试版，欧链团队也在第一时间对代码进行了编译和测试，发布了《EOS 编译、测试指南（一）》，详细介绍如何让 EOS 在自己的本地跑起来。两周以来，EOS 团队陆续在 Git 上更新了自己的代码，欧链团队也在持续跟进 EOS 代码，编译和测试，以下将以发行代币为例子，详细介绍如何在 EOS 上编写智能合约。OracleChain 团队使用 Mac 系统进行开发。

该指南主要依据 EOS 的官方文档，以及欧链团队的实践经验编写，开发者有任何技术问题可以关注欧链小秘书、在欧链科技社区里向 OracleChain 团队提问或者发送邮件到 contact@oraclechain.io。

EOS 文档：https://eosio.github.io/eos/

EOS 代码：https://github.com/EOSIO/eos

## 第一步：代码准备

开发者可以根据欧链团队编写的《EOS 编译、测试指南（一）》获取、编译 EOS 代码、并运行 EOS 单机节点 eosd 程序和使用 EOS 命令行工具 eosc。

因 EOS 在这两周的代码更新中，添加了账户签名和权限控制机制，因此原《EOS 编译、测试指南（一）》中的命令无法直接使用。需要对 EOS 代码进行一

些修改，避过权限交易机制，才能完成测试。

**注：EOS 官方文档中给出的命令行例子目前也还是只能在无权限控制版本下运行。**

首先获取 EOS 最新代码,包括两个分支的最新更新(不然会导致编译错误):

```
git pull
git submodule update --init --recursive
```

然后对 eos/libraries/chain/include/eos/chain/chain_controller.hpp 文件 339 行进行修改，将_skip_flags 赋值为 skip_authority_check。

```
339            uint64_t                              _skip_flags = skip_authority_check;
```

这样可以跳过权限校验过程。重新编译代码，至此已经可以发布和执行智能合约。

## 第二步：EOS 智能合约编写

按照 EOS 的官方文档，以及代码中给出的例子，EOS 的智能合约主要需要编写 abi 接口和用 C++编写逻辑，并使用 WASM 编译器编译成 wast 文件。

C++里面主要实现 init 和 apply 两个函数。

```
extern "C" {
   void init();
   void apply( uint64_t code, uint64_t action );
}
```

以 currency 为例:

```
extern "C" {
    void init()  {
       storeAccount( N(currency), Account( CurrencyTokens(1000ll*1000ll*1000ll) ) );
    }

    /// The apply method implements the dispatch of events to this contract
    void apply( uint64_t code, uint64_t action ) {
       if( code == N(currency) ) {
          if( action == N(transfer) )
             currency::apply_currency_transfer( currentMessage< TOKEN_NAME::Transfer >() );
       }
    }
}
```

实际上 currency 发行了一个代币 CurrencyTokens，并提供了代币之间的转账操作 apply_currency_transfer(currentMessage< TOKEN_NAME::Transfer >())。

EOS 代码在 eos/contracts/eoslib/token.hpp 给出了代币的模板，并给出了代币的一些简单操作，比如增减、比较等，这样开发者可以直接通过

```
14      typedef eos::token<uint64_t,N(currency)> CurrencyTokens;
```

定义发布自己的代币（参见 eos/contracts/currency/currency.hpp 第 14 行）。

## 第三步：发布和执行智能合约

通过编译可以得到 wast 文件，然后通过命令行可以发布 currency 合约（代币）。首先创建智能合约账号：

./eosc create account oraclechain currency
EOS4toFS3YXEQCkuuw1aqDLrtHim86Gz9u3hBdcBw5KNPZcursVHq
EOS6KdkmwhPyc2wxN9SAFwo2PU2h74nWs7urN1uRduAwkcns2uXsa

注意，需要保证 oraclechain 账户拥有 eos_balance，可通过账号 eos 进行转账。


发布智能合约：

./eosc setcode currency
../../contracts/currency/currency.wast
../../contracts/currency/currency.abi


在执行该智能合约前，需要对 eosc 代码进行一些小的修改（该处修改已经提交给 EOS 团队）。

首先修改 eos/programs/eosc/main.cpp 文件第 261 行，将“cmd_line”改为

“args”:

```
261     auto arg= fc::mutable_variant_object( "code", code )("action",action)("args", fc::json::from_string(json));
```

其次修改同一文件第 271 行，将“bincmd_line”该成“binargs”:

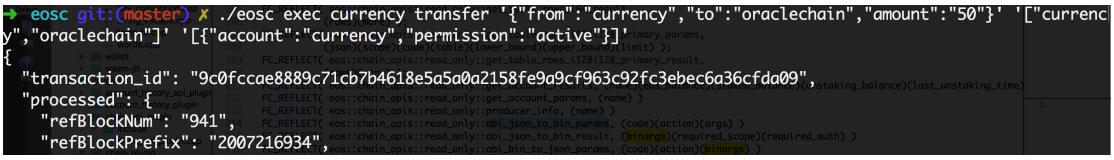
```
271         msg.data = result.get_object()["binargs"].as<Bytes>();
```

此处修改依据参考了

eos/plugins/chain_plugin/include/eos/chain_plugin/chain_plugin.hpp 文件中关于 abi_json_to_bin_params 函数和 abi_bin_to_json_params 函数对参数的定义。

然后可以通过命令行完成代币的转账:

./eosc exec currency transfer
'{"from":"currency","to":"oraclechain","amount":"50"}'
'["currency","oraclechain"]' '[{"account":"currency","permission":"active"}]'




至此，我们完成了代币发行和转账的智能合约编写。值得注意的是，目前该合约没有涉及到用户权限的控制，任何人都可以完成任意账户间的转账。欧链团队将持续跟进 EOS 此部分代码和文档，陆续提供更多编译、测试用例。

OracleChain 团队
2017 年 8 月 15 日