athena-ide-atom

Jul 14, 2020

Tutorial

Welcome to the athena ide atom wiki. This is an IDE (Integrated Development Environment) for aergo smart contract. It can compile, deploy and execute smart contract with an account. It also can create, import and export account.

CHAPTER 1

Getting Started

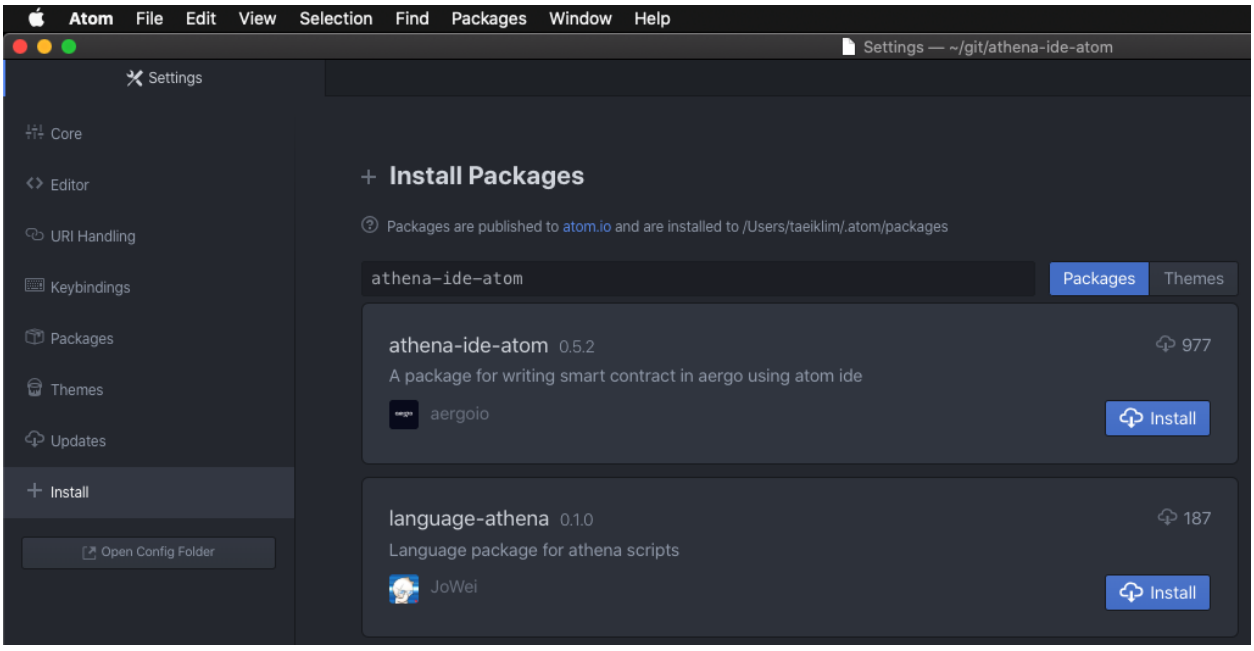
1.1 Install

Athena ide atom is based on atom. You have to install it first.

1.1.1 By apm

Athena ide atom is available in apm (atom package manager). You can install with it.

Atom -> Preferences -> Install



You can install using cli. For windows user, you can use git bash for cli environment.

```
> apm install athena-ide-atom
```

1.1.2 By installer

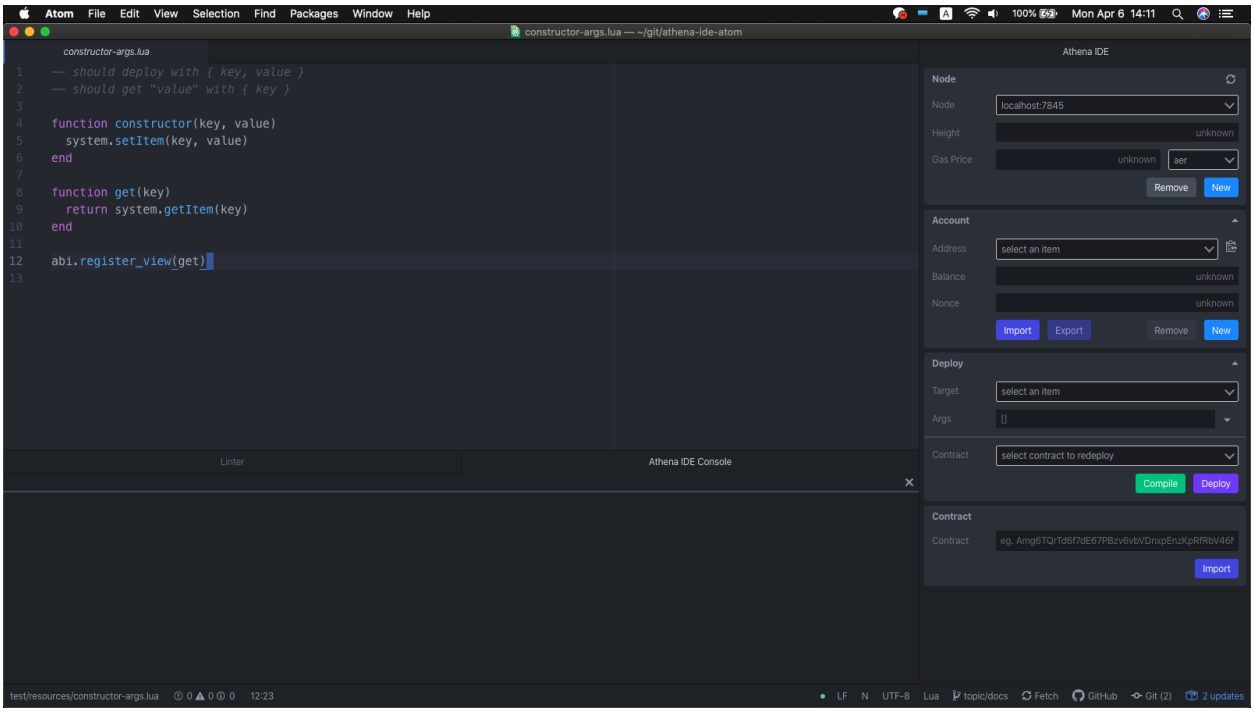
Installing with apm may be slow on slow network status. So we provide custom installer in releases. Download athena-ide-atom-x.x.x-installer.bin from releases and run in cli. For windows user, you can use git bash for cli environment.

```
> ./athena-ide-atom-x.x.x-installer.bin
```

1.2 Open

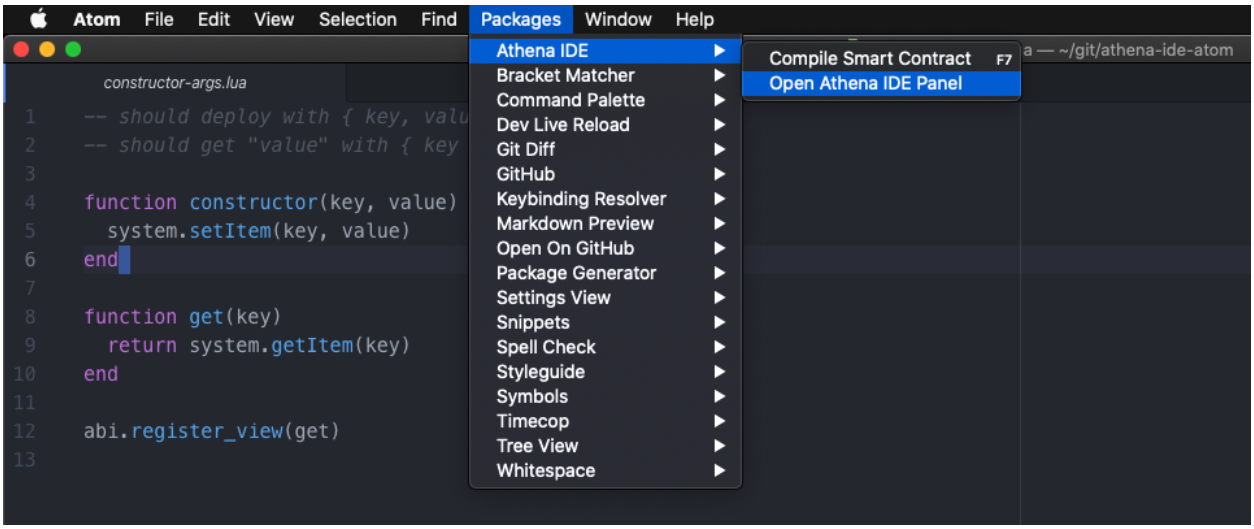
1.2.1 Using shortcut

- Mac : Option + Shirt + L
- Windows : Alt + Shirt + L



1.2.2 Using menu bar

Packages -> Athena IDE -> Open Athena IDE Panel

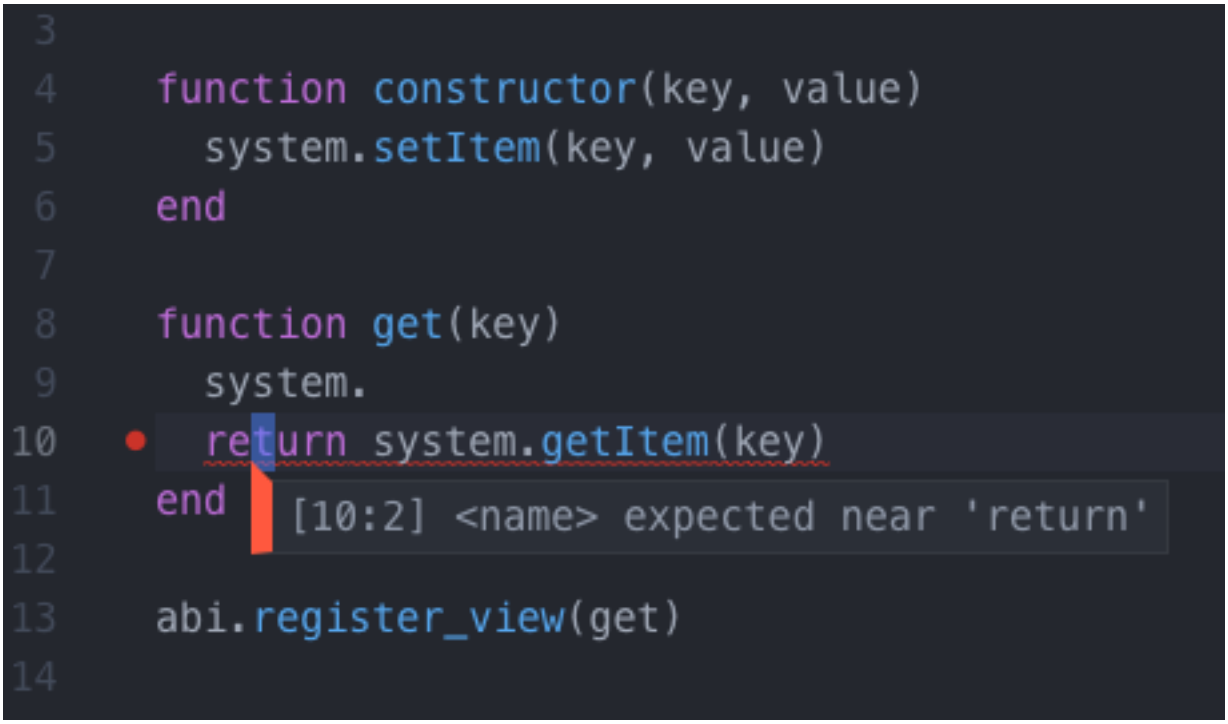


1.3 Auto Complete

Athena ide atom provides some basic autocomplete for lua.

1.4 Lint

Athena ide atom provides basic lint for lua.


function constructor(key, value)
system.setItem(key, value)
end
function get(key)
system.
return system.getItem(key)
end
[10:2] <name> expected near 'return'
abi.register_view(get)

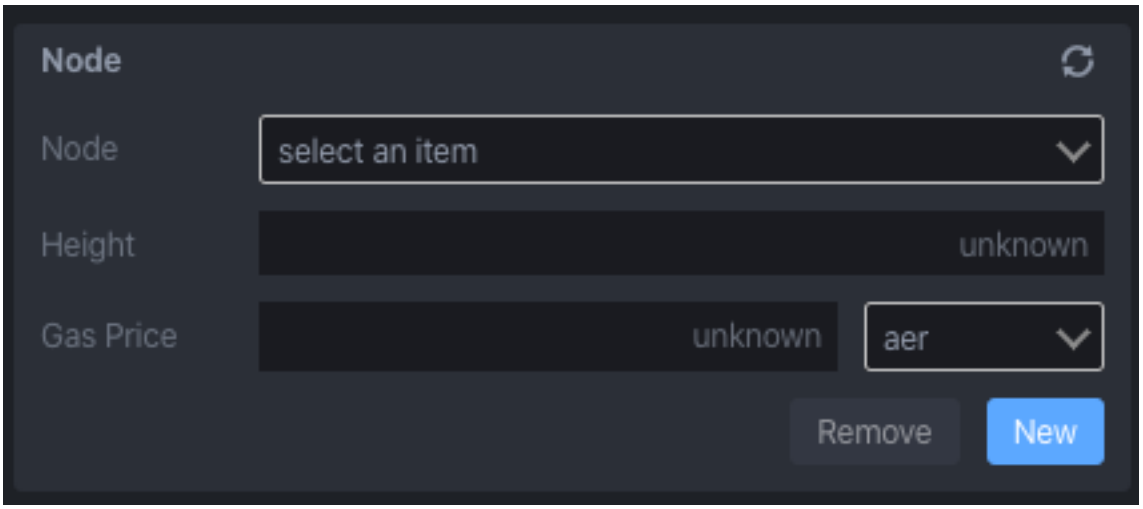
CHAPTER 2

Node

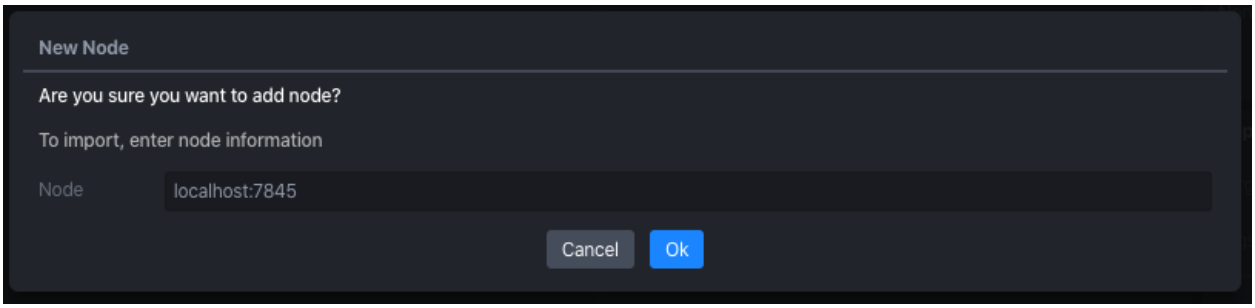
You can configure an aergo node whose smart contract interacting with. You can see height and gas price of it. Default configuration is localhost:7845.

2.1 New

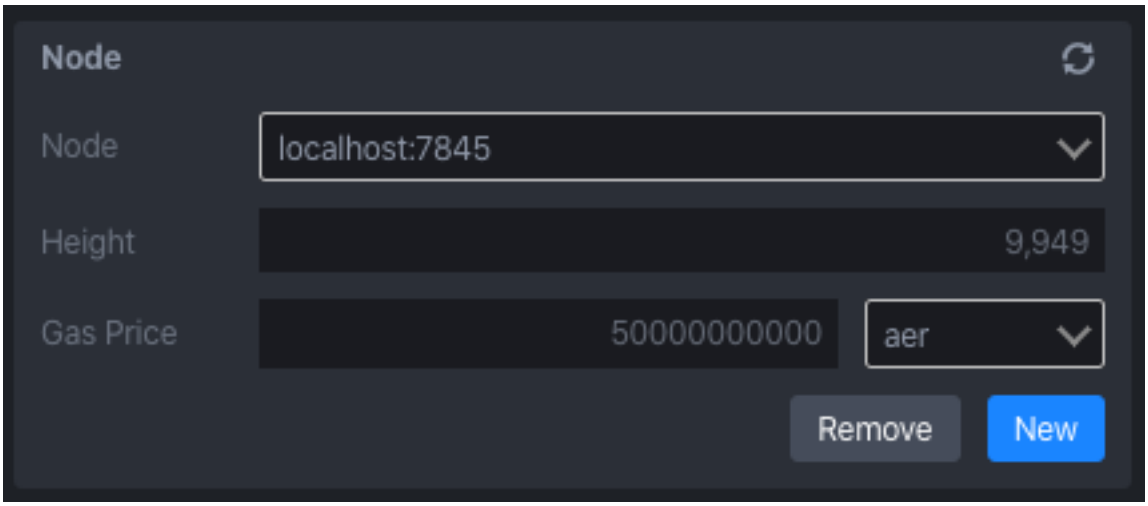
Click an new button



Enter node endpoint and click ok

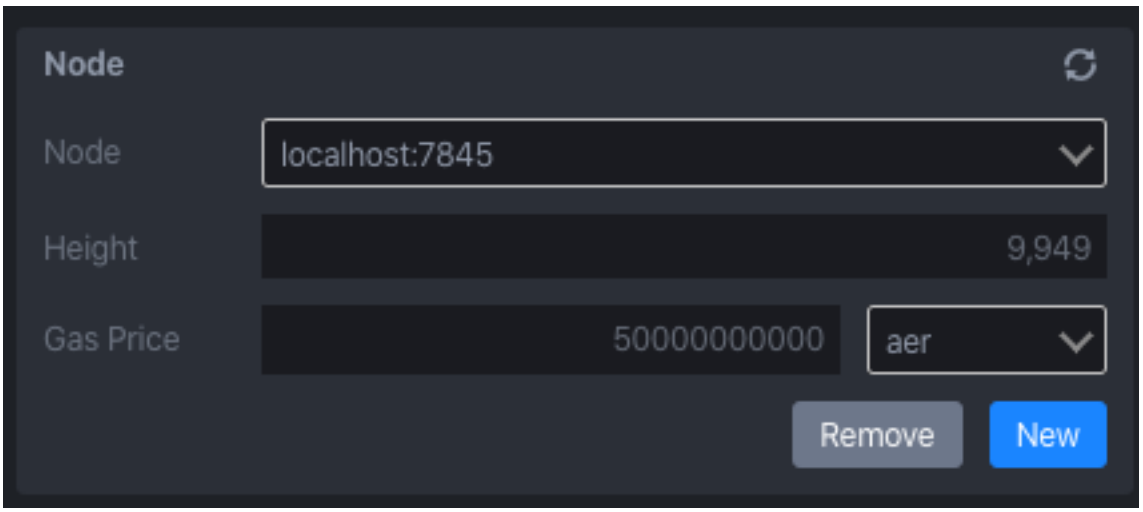


You can see height and gas price of it

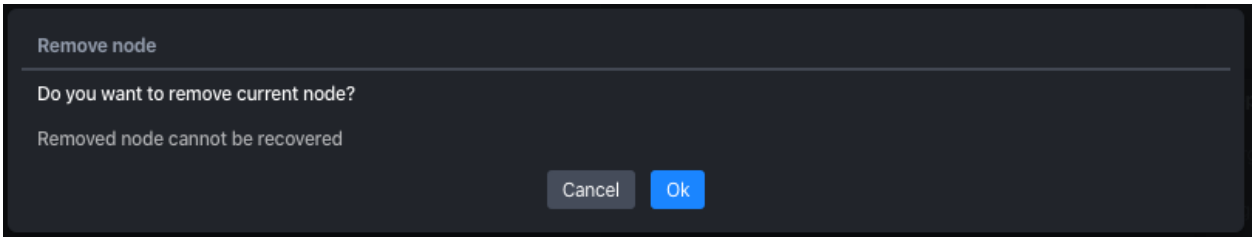


2.2 Remove

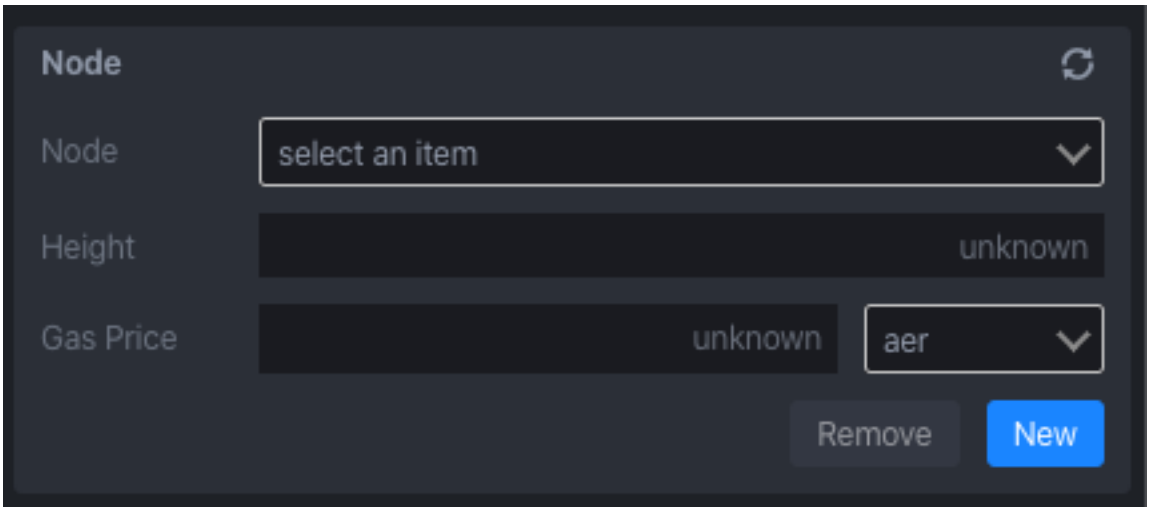
Click a remove button



Click ok button



Node removed from list



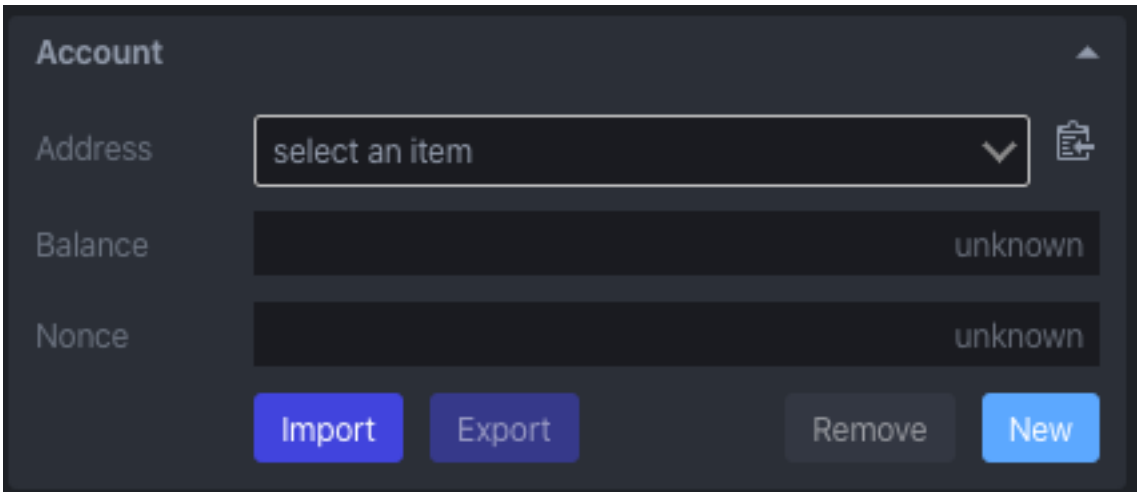
CHAPTER 3

Account

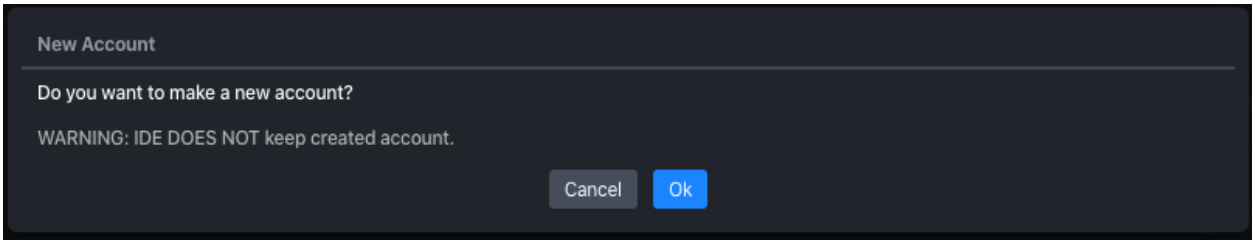
You can configure an account which is deploying, executing and querying smart contract. An account have to have some aergo token to make transaction.

3.1 New

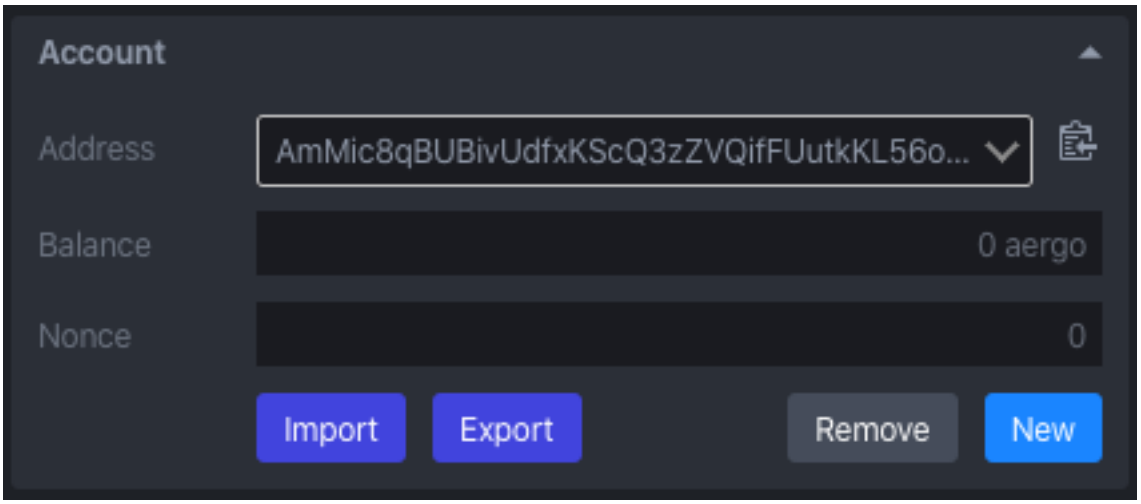
Click an new button



Click a ok button



Account created

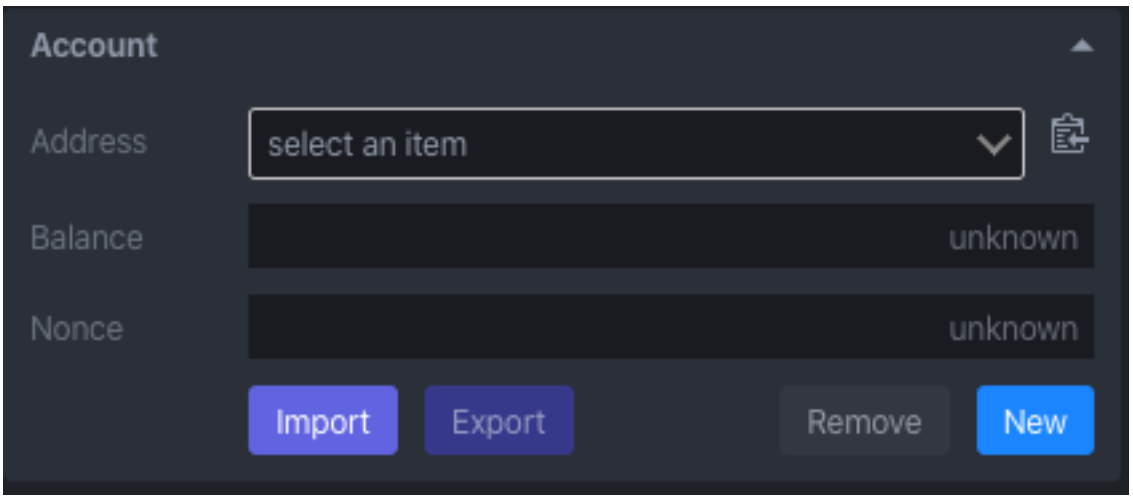


3.2 Import

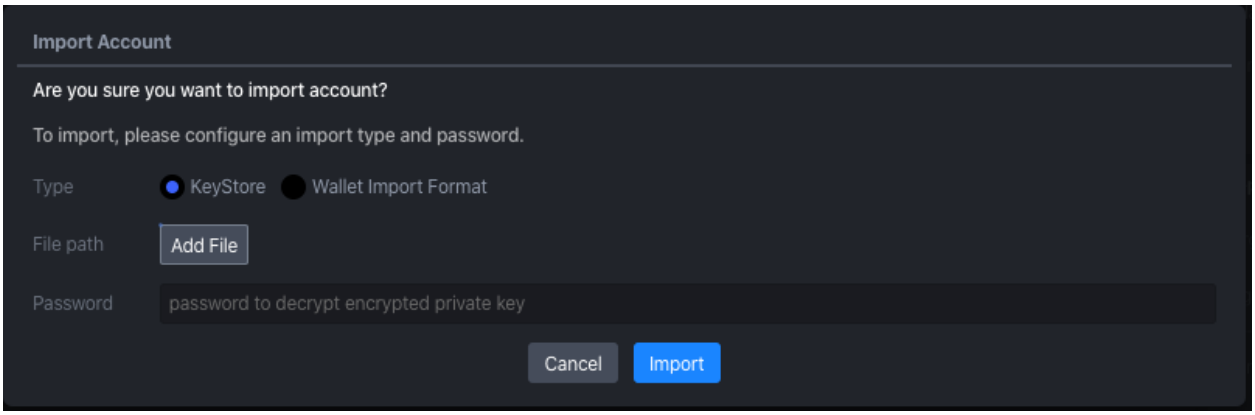
You can import account with an aergo specific keystore format or wallet import format. For details of keystore format, see aergo keystore proposal.

3.2.1 KeyStore

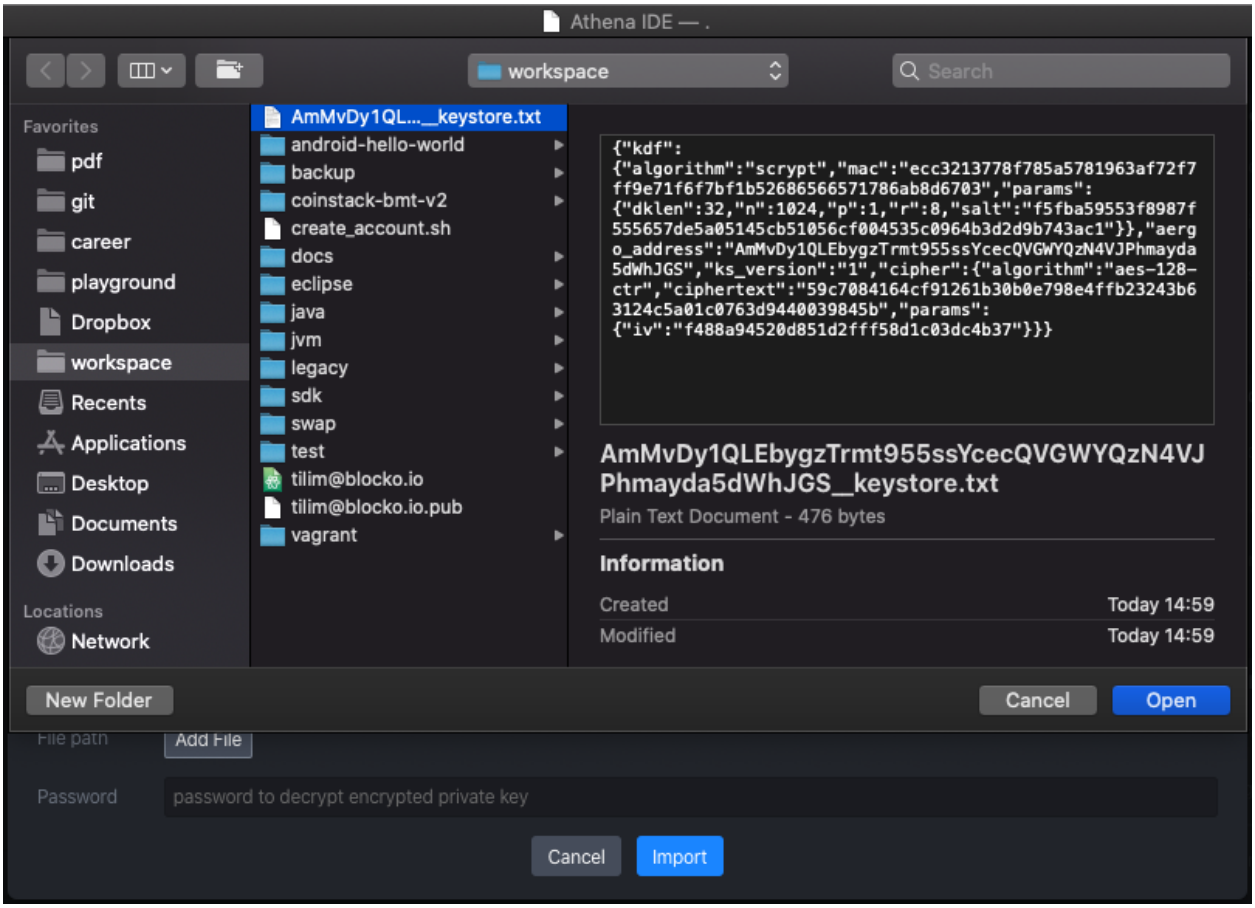
Click an import button



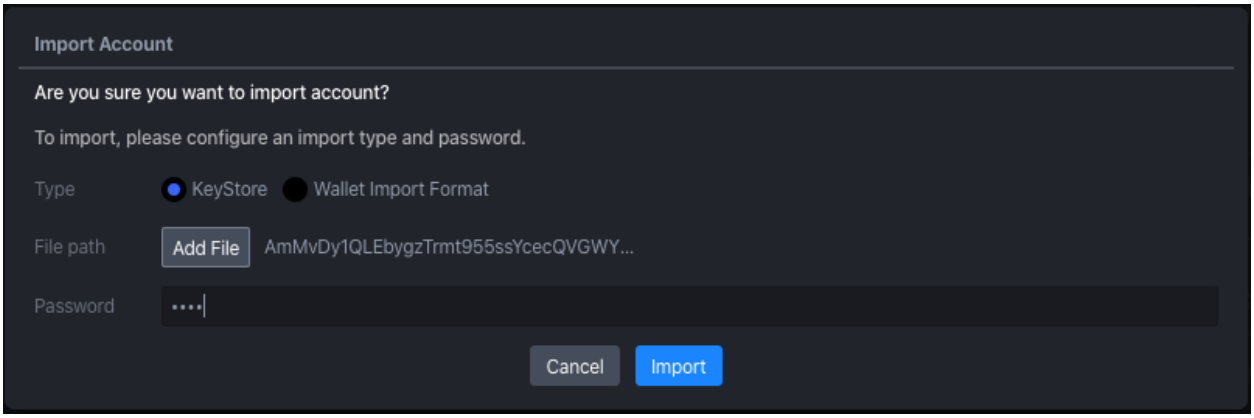
Click an add file button on popup



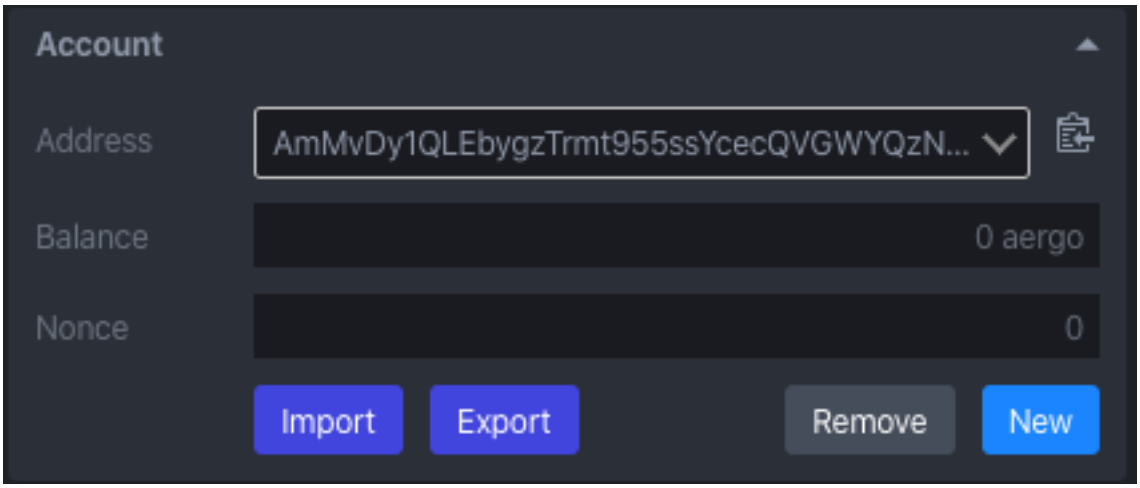
Select keystore file



Enter password to decrypt keystore

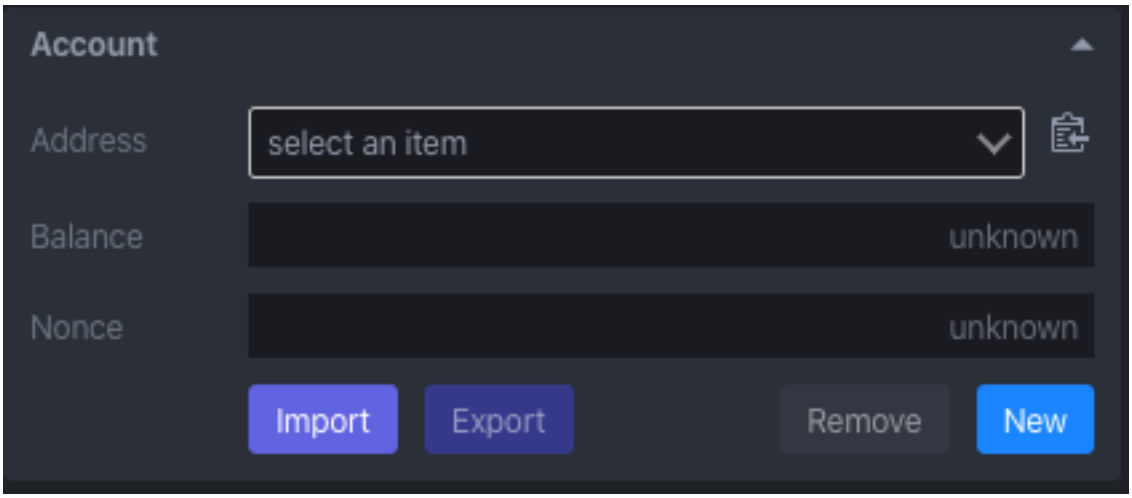


Account imported

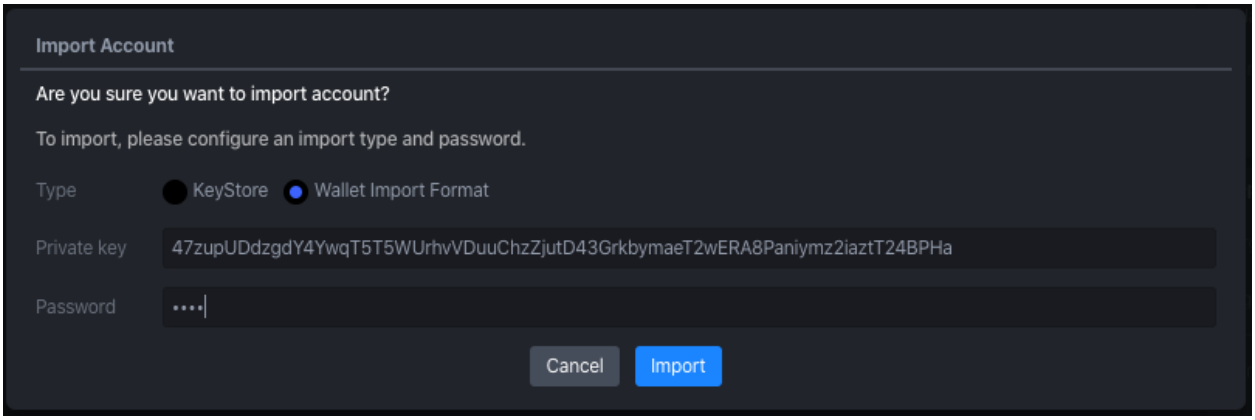


3.2.2 Wallet Import Format

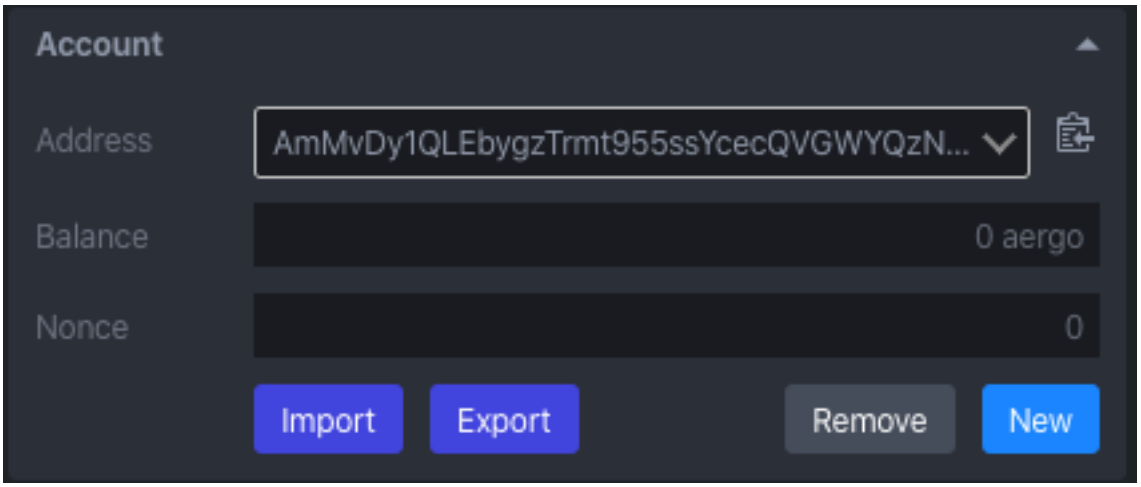
Click an import button



Enter wallet import format & password to decrypt it



Account imported

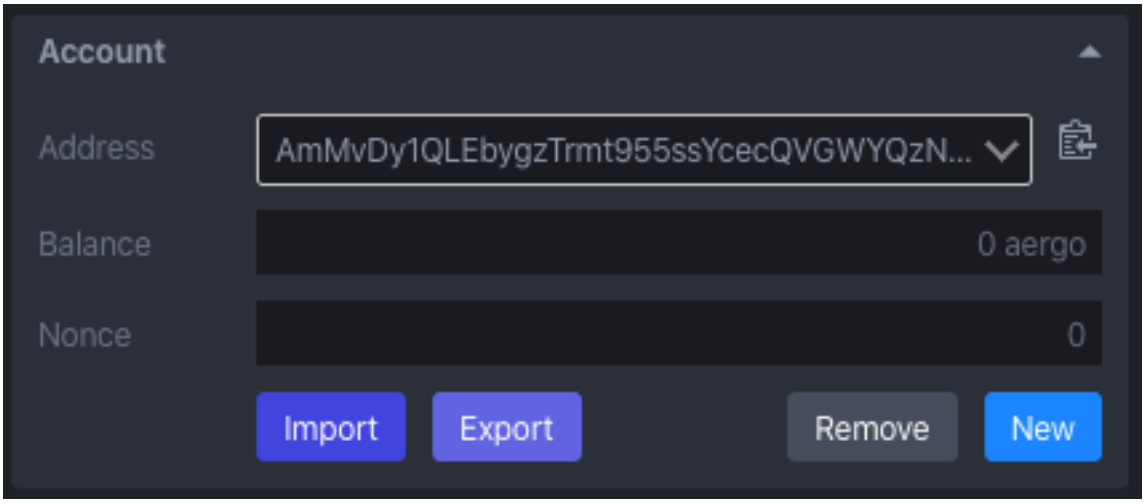


3.3 Export

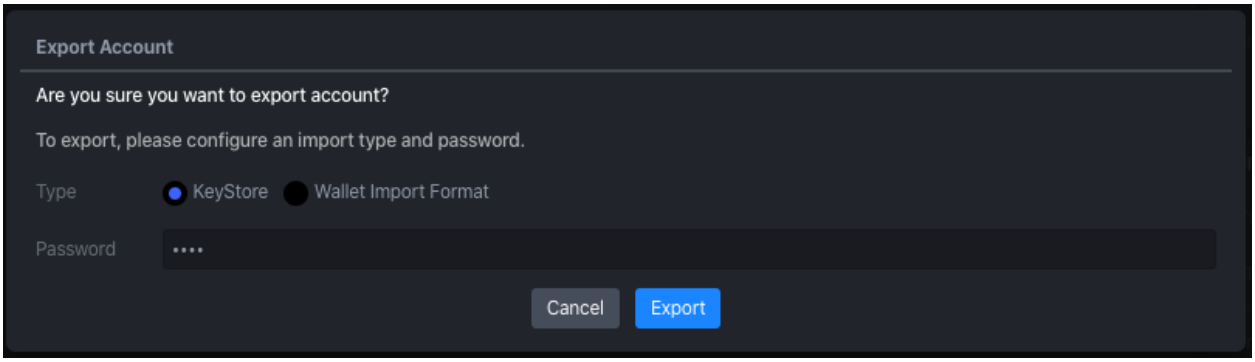
You can export account with an aergo specific keystore format or wallet import format. For details of keystore format, see aergo keystore proposal.

3.3.1 KeyStore

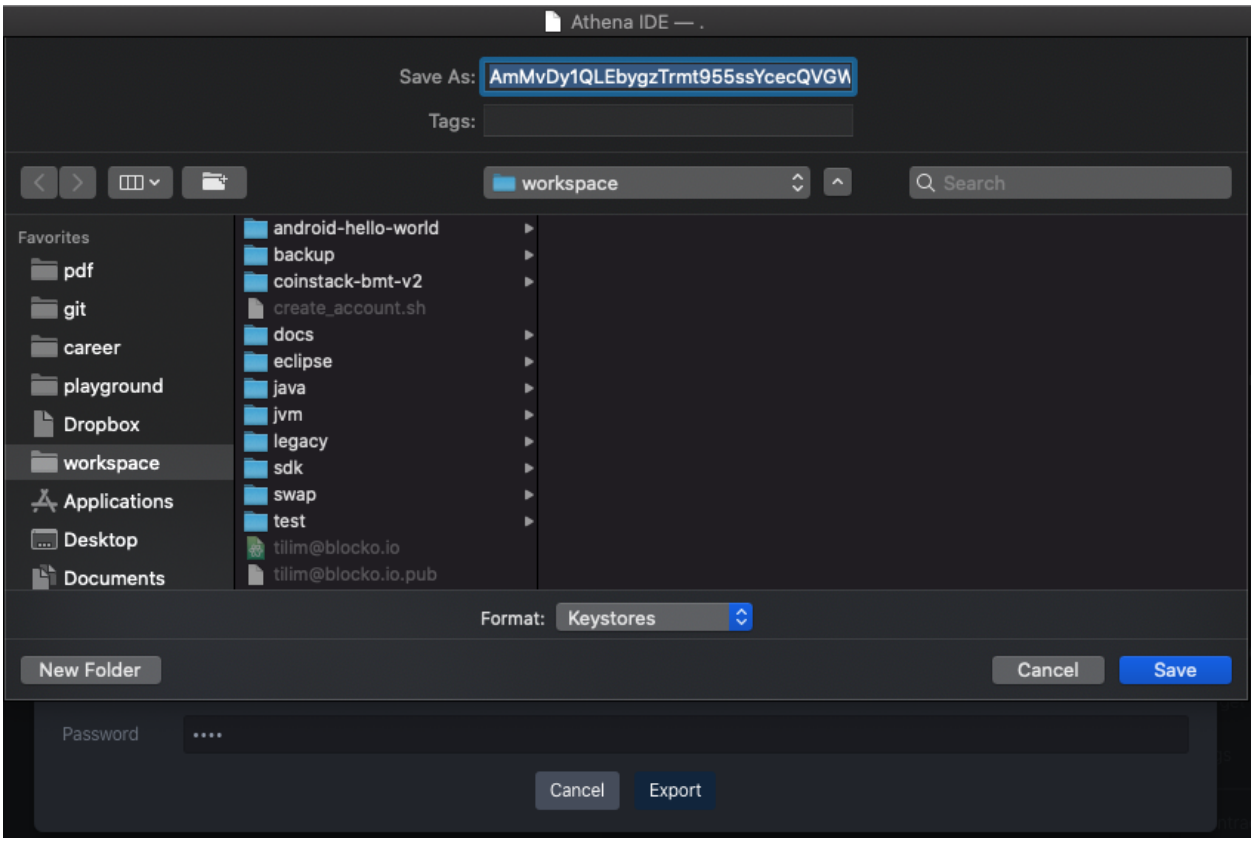
Click an export button



Enter a password to encrypt

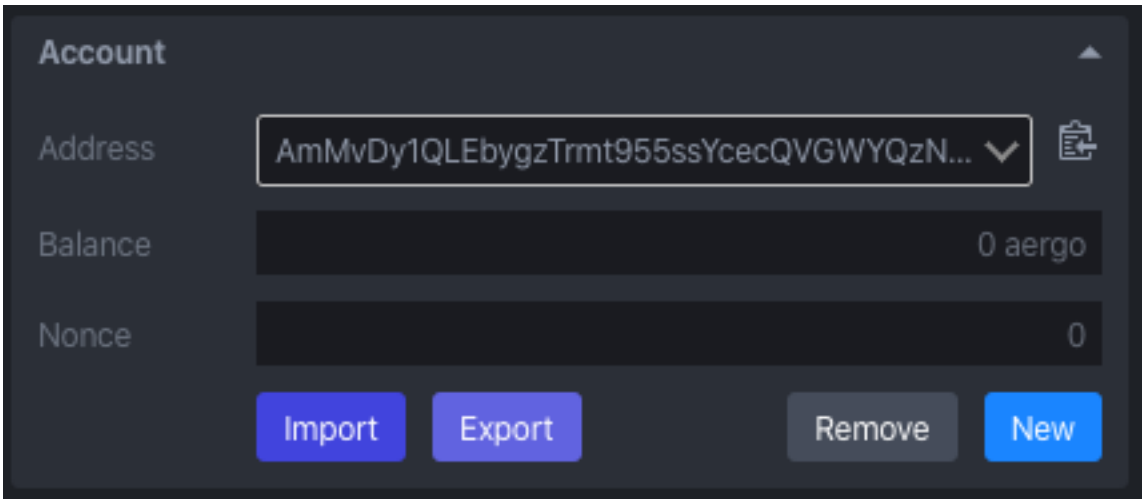


Choose save location

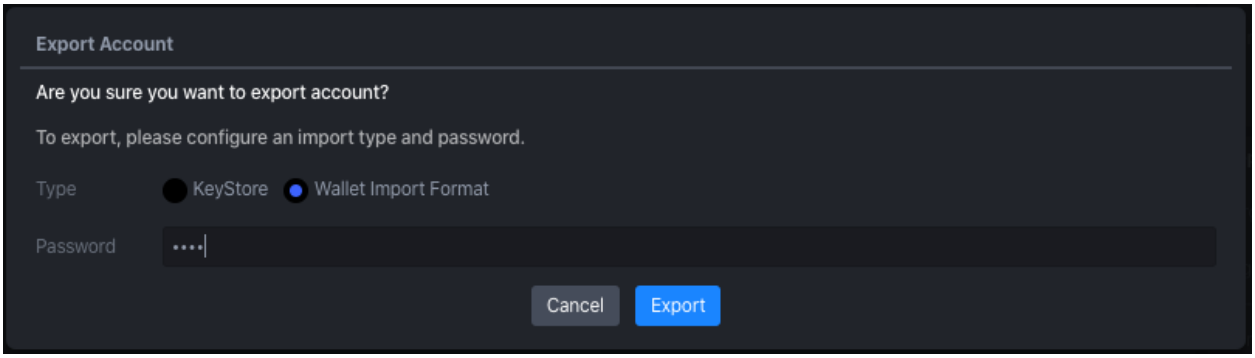


3.3.2 Wallet Import Format

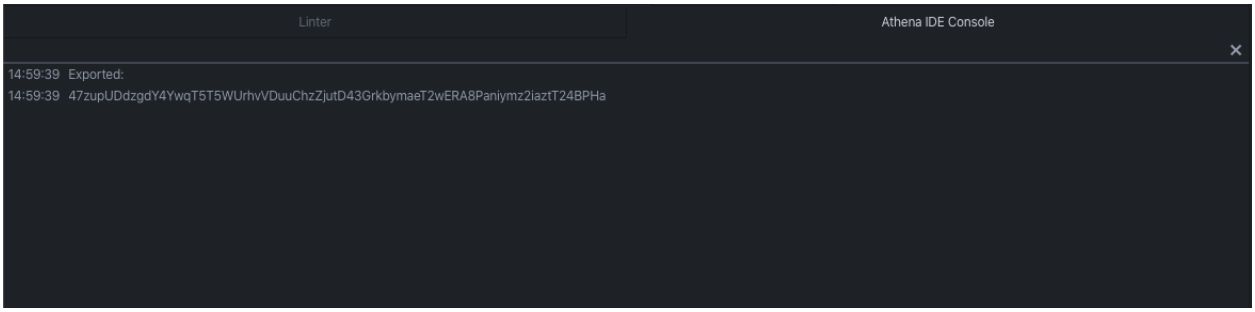
Click an export button



Enter a password to encrypt

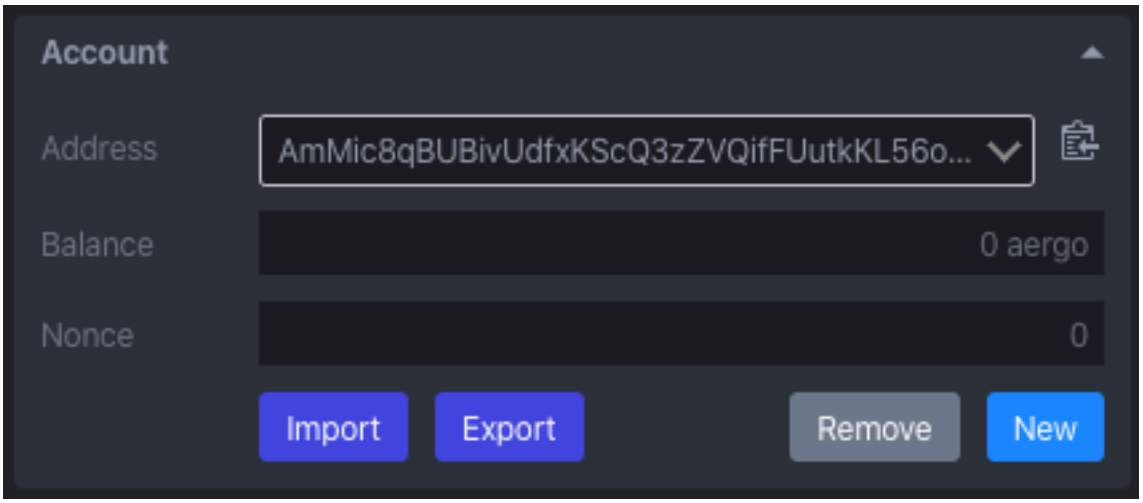


You can see wallet import format in a console

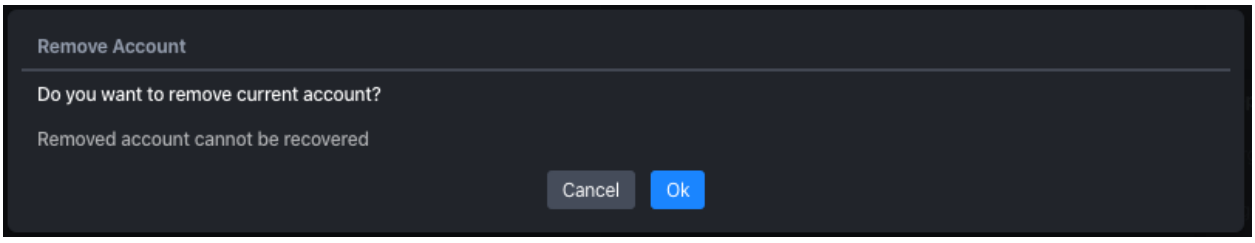


3.4 Remove

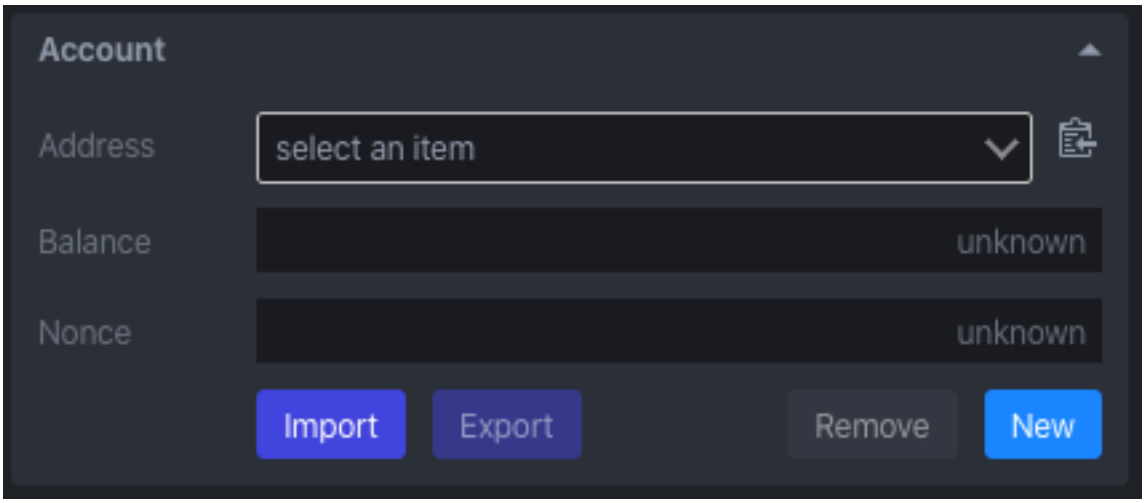
Click an remove button



Click a ok button



Account removed from list



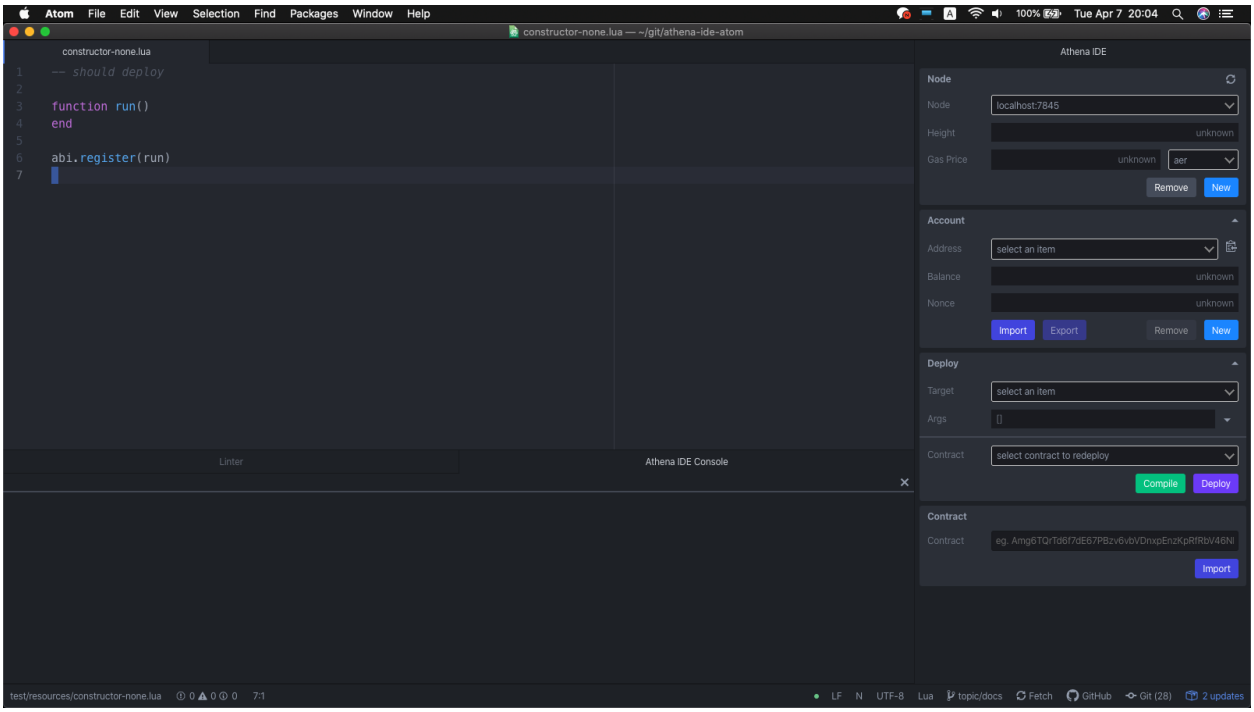
CHAPTER 4

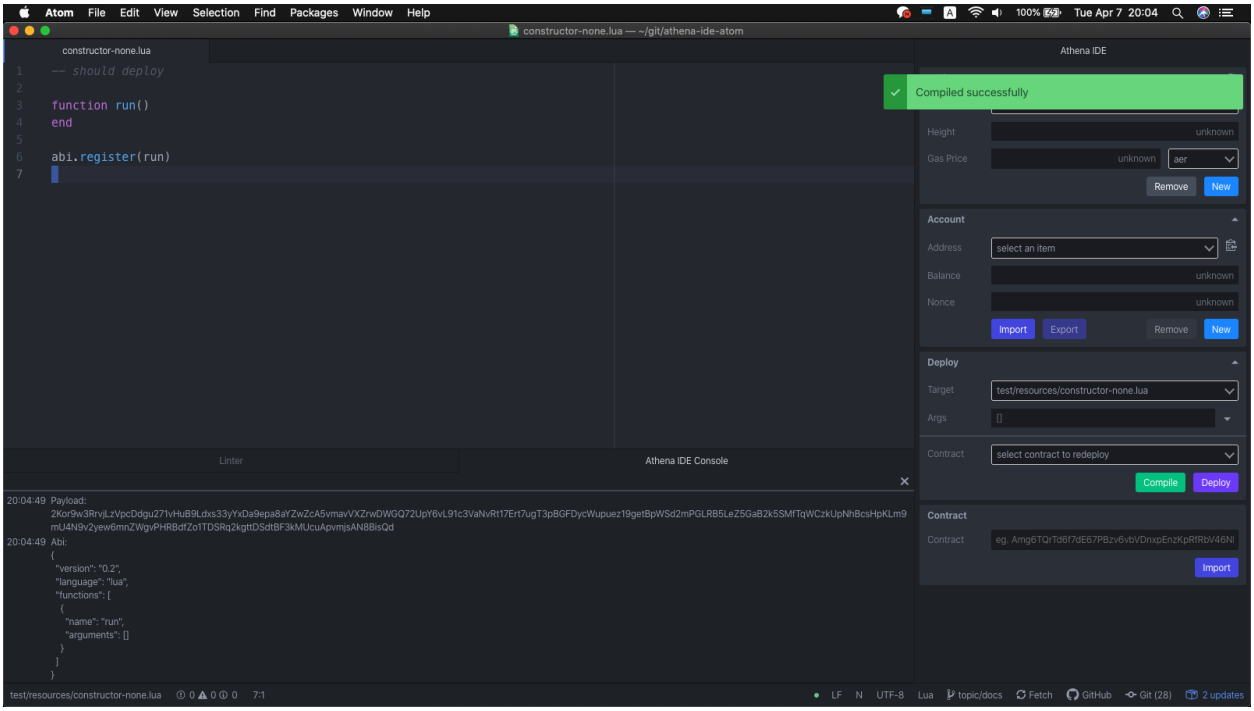
Contract

You can deploy/execute/query smart contract. Deploying and executing is done by making transaction. Make sure that you have enough aergo token to make transaction before deploying and executing contract.

4.1 Compile

Before deploying contract, you have to compile by **clicking compile button** or **pressing F7**.





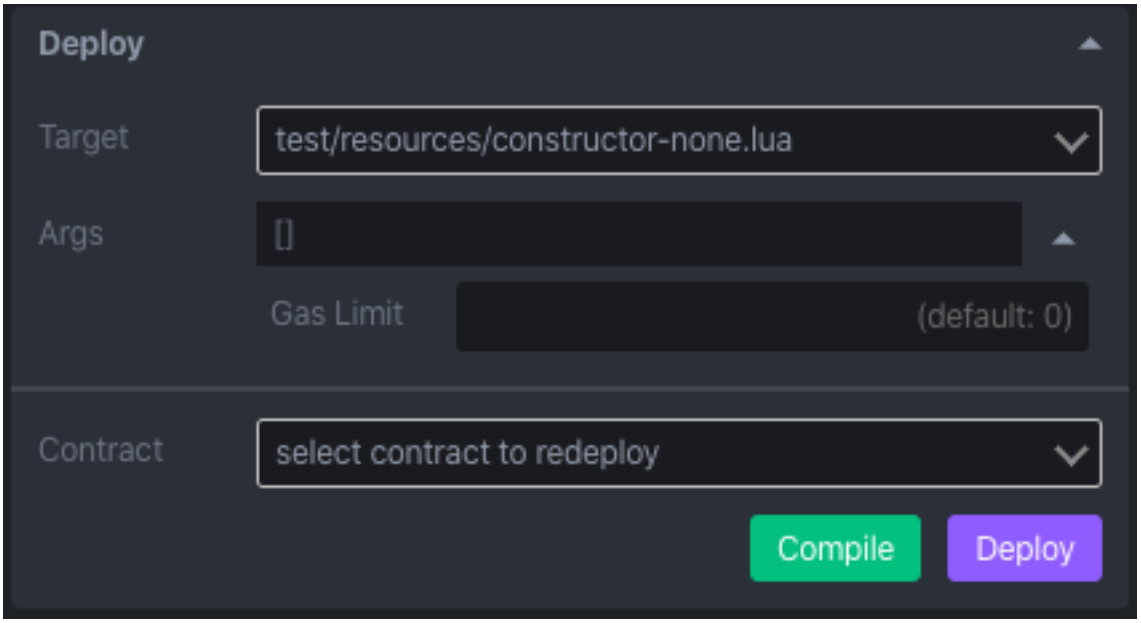
4.2 Deploy

After compiling contract, you can deploy contract.

4.2.1 Without args

You can deploy contract without constructor arguments.

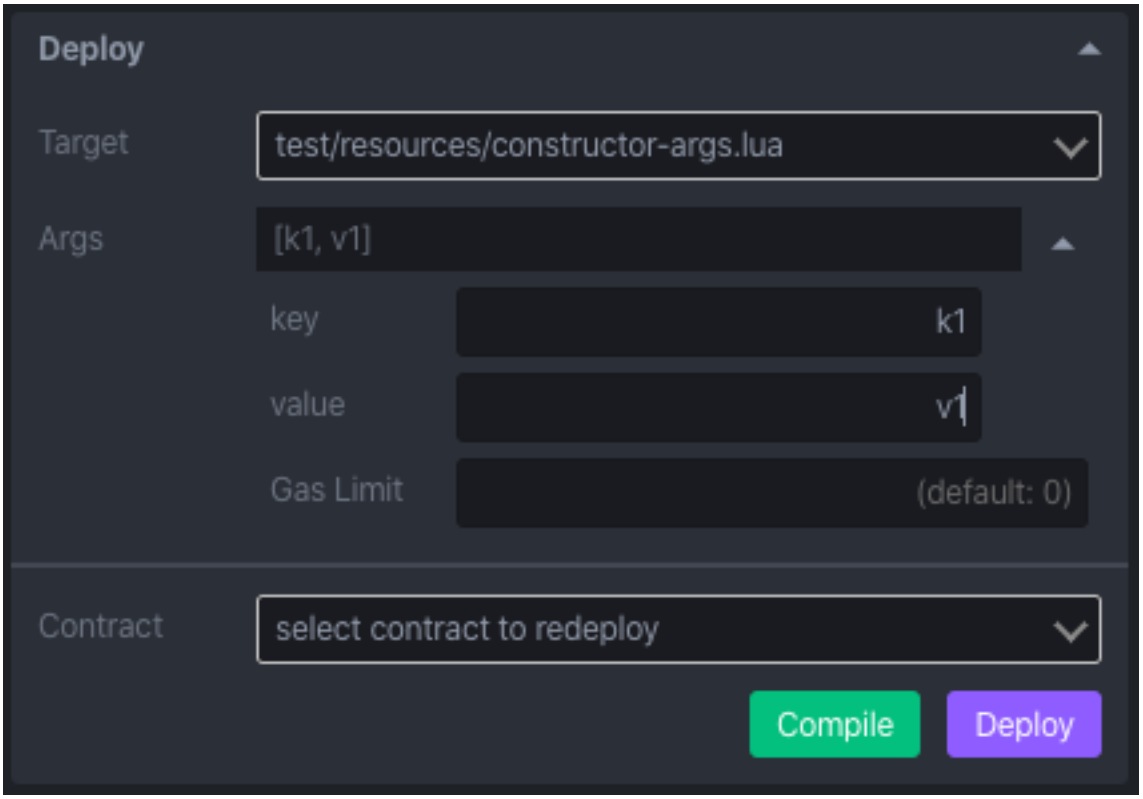
```
-- no arguments
function constructor()
  system.setItem("k1", "v1")
end

...
```



4.2.2 With args

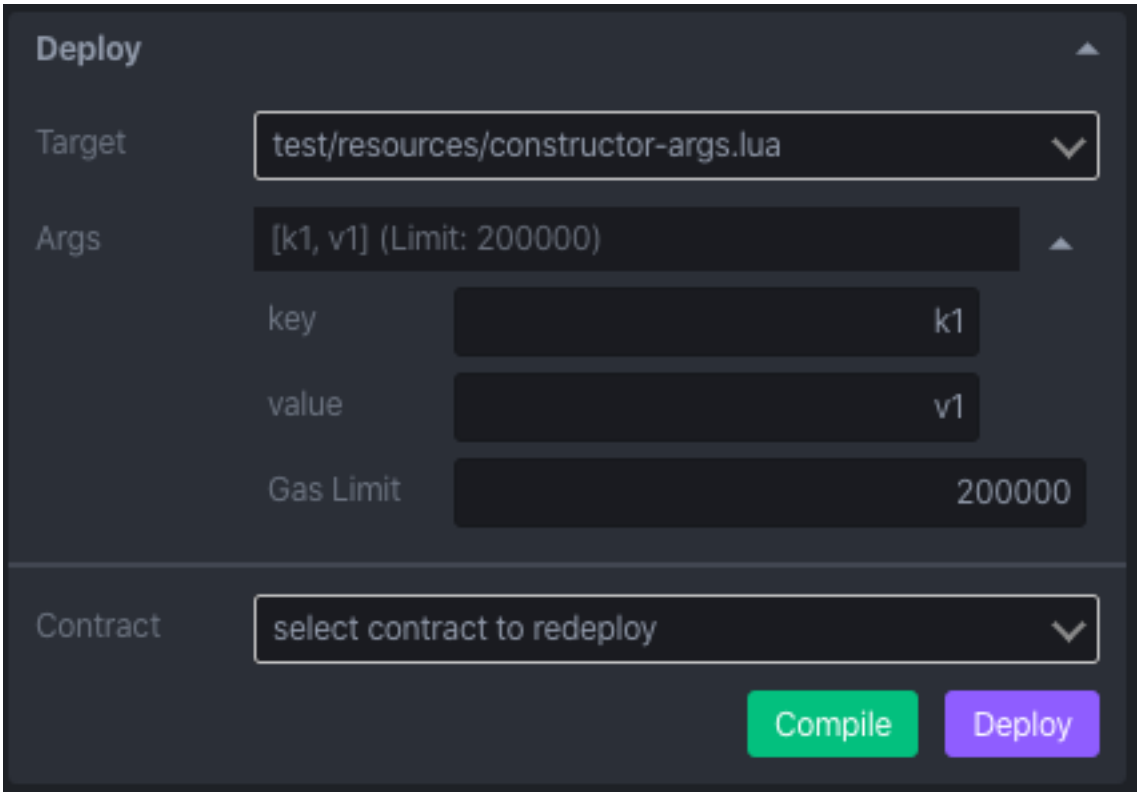
You can deploy contract with constructor arguments.

```
-- arguments (key, value)
function constructor(key, value)
  system.setItem(key, value)
end

...
```



4.2.3 Gas limit

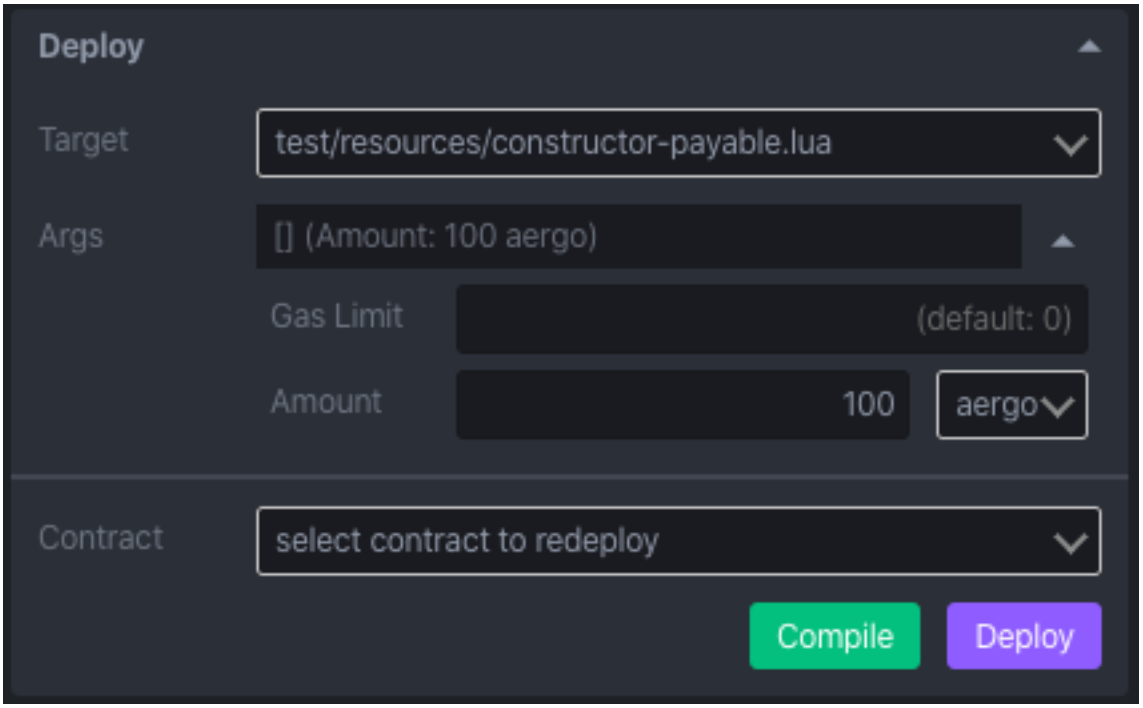
You can deploy contract with gas limit configuration. 0 limit means infinite (uses as much as possible).



4.2.4 Amount

You can deploy contract with aergo token. Make sure constructor is registered as payable.

```
function constructor()
  system.setItem("k1", "v1")
end

...

-- registered as payable
abi.payable(constructor)
```

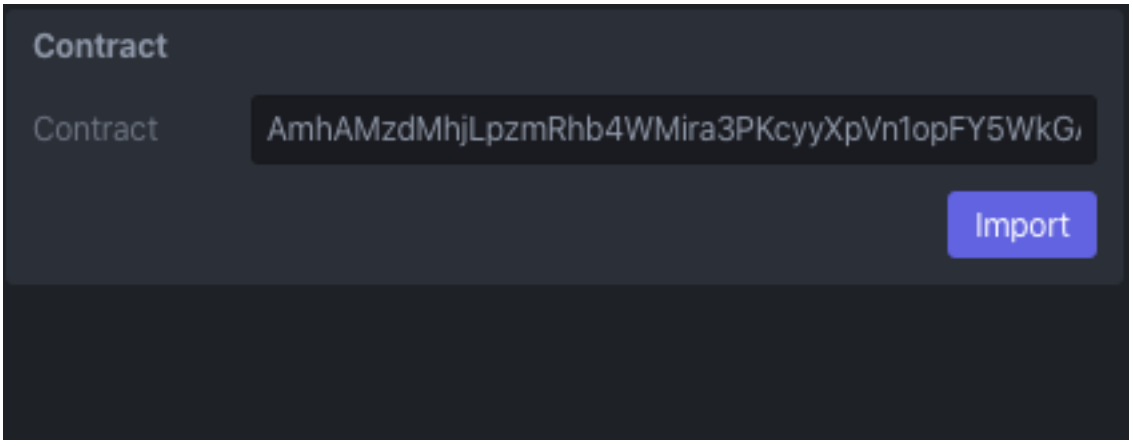


4.3 Import & Remove

4.3.1 Import

You can import already deployed contract.

Type contract address & click import button



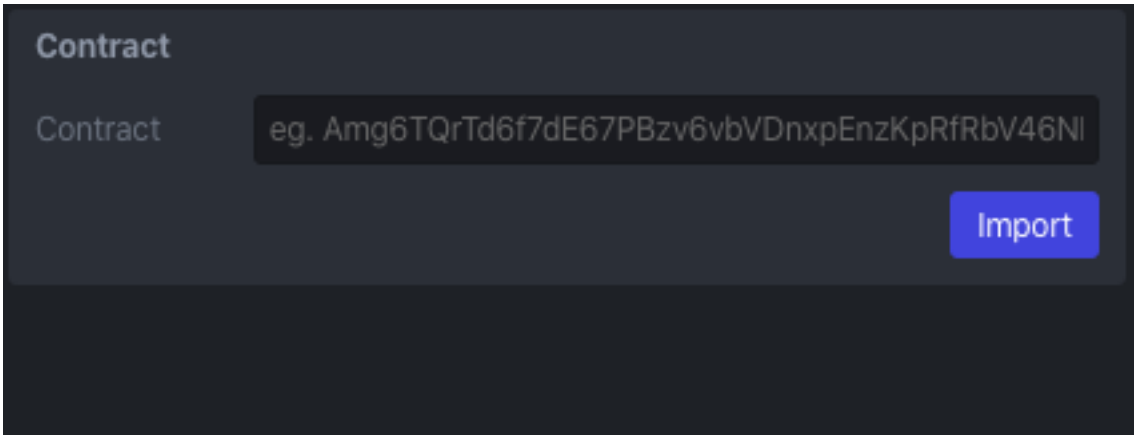
Contract imported

4.3.2 Remove

You can remove contract.

Click trash button

Contract removed



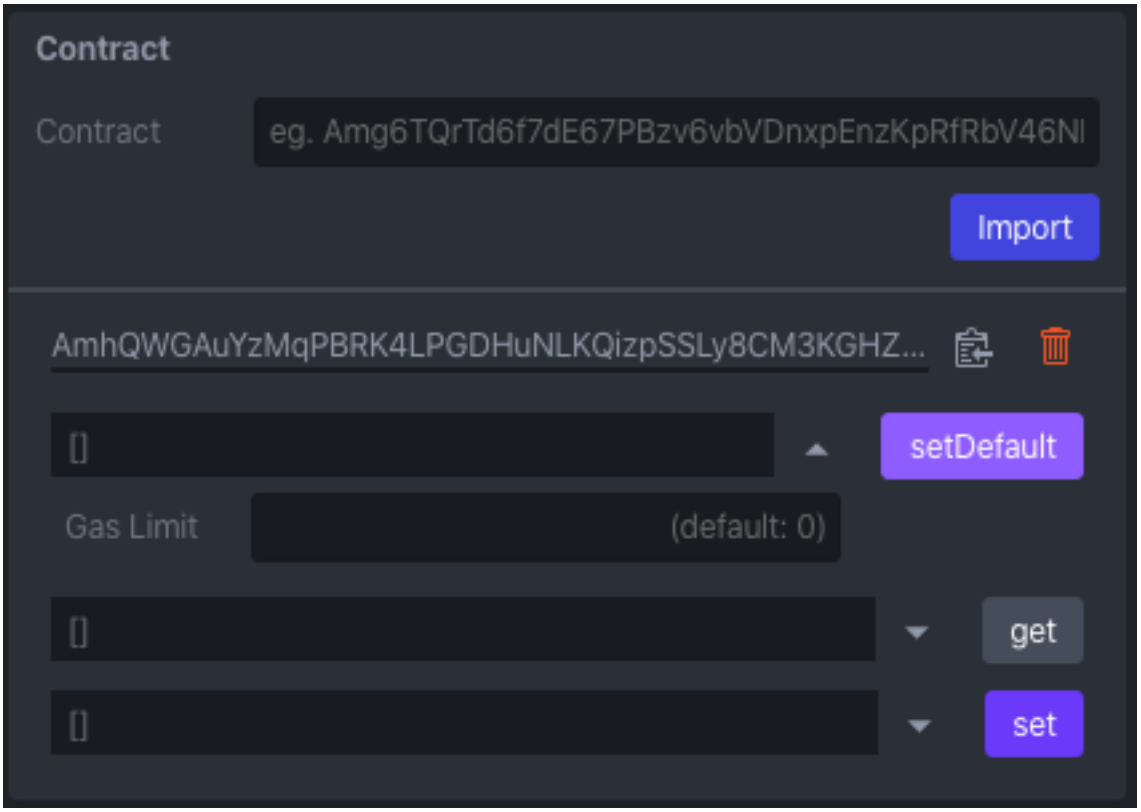
4.4 Execute

Contract execution can change status of contract state db. Any function registered as register can be executed.

4.4.1 Without args

You can execute contract without arguments.

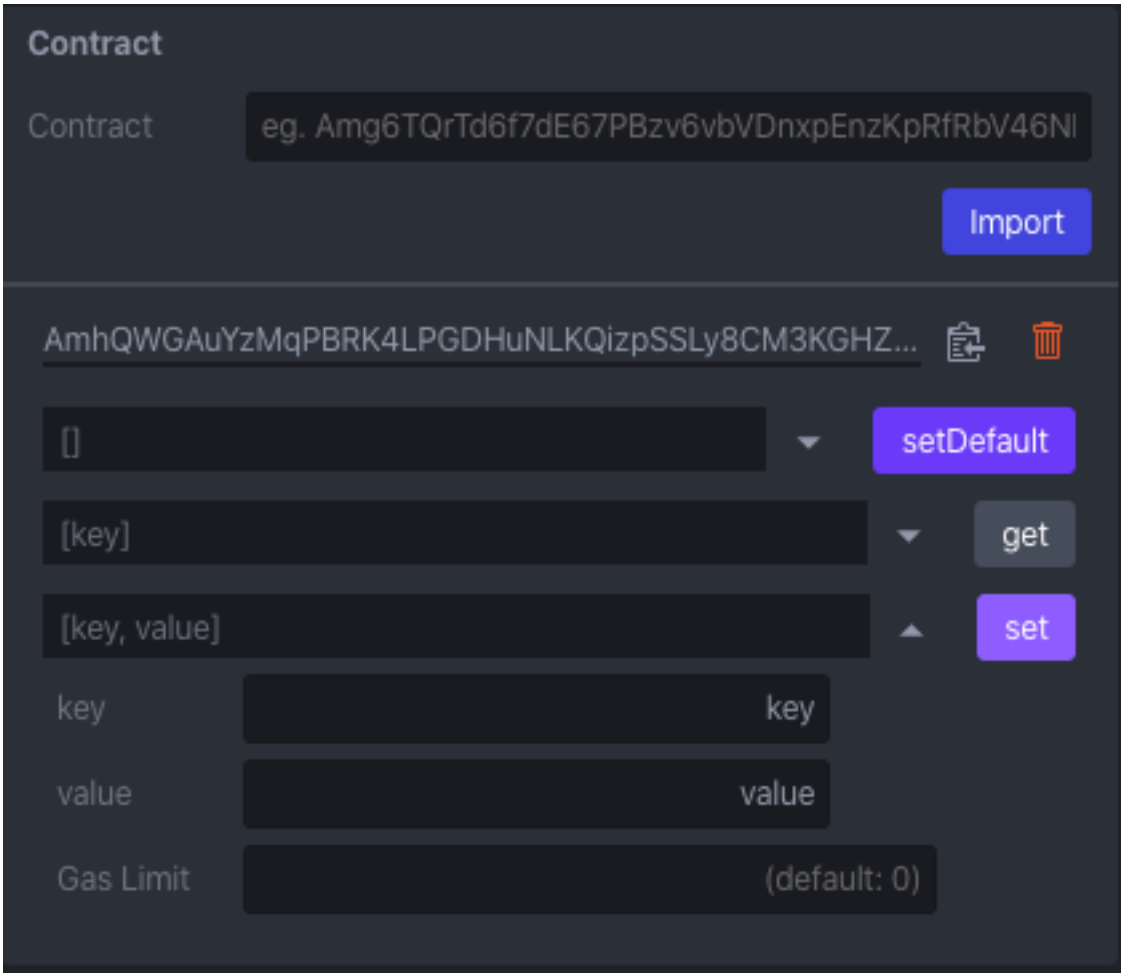
```
...

-- no arguments
function setDefault()
  system.setItem("k1", "v1")
end

...

-- register as execution
abi.register(setDefault)
```



4.4.2 With args

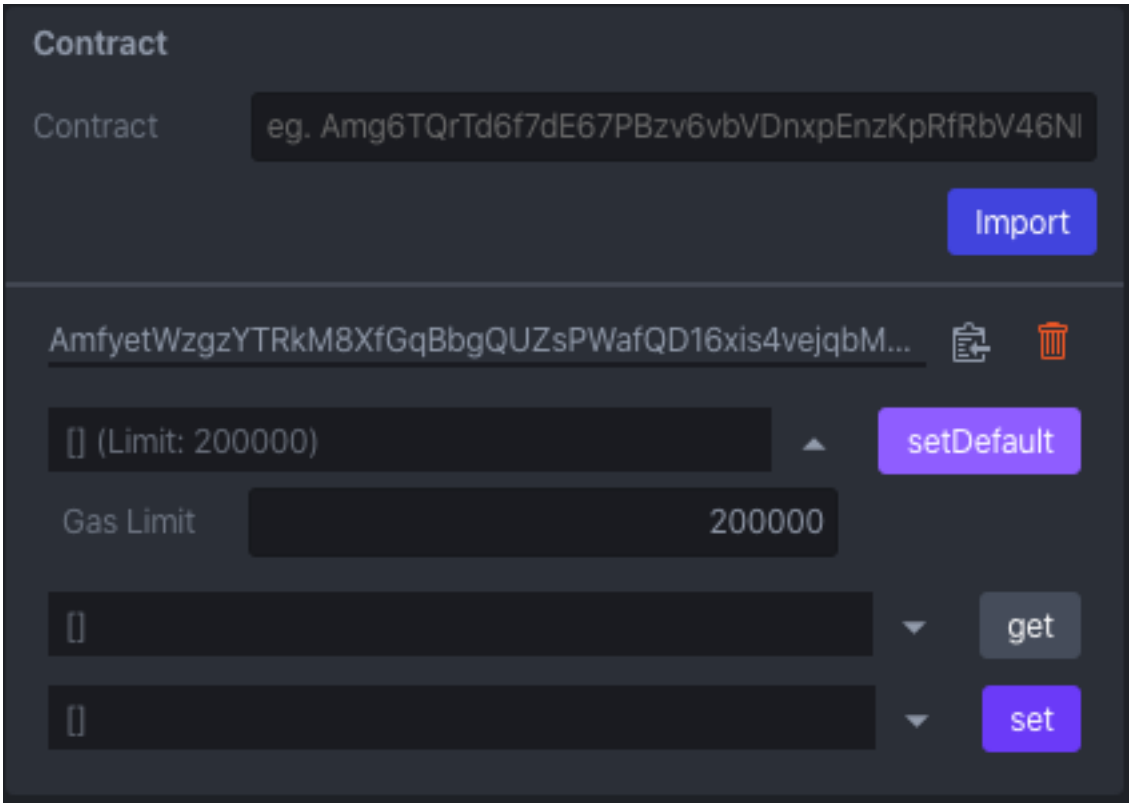
You can execute contract with arguments.

```
...

-- arguments (key, value)
function set(key, value)
  system.setItem(key, value)
end

...

-- register as execution
abi.register(set)
```



4.4.3 Gas limit

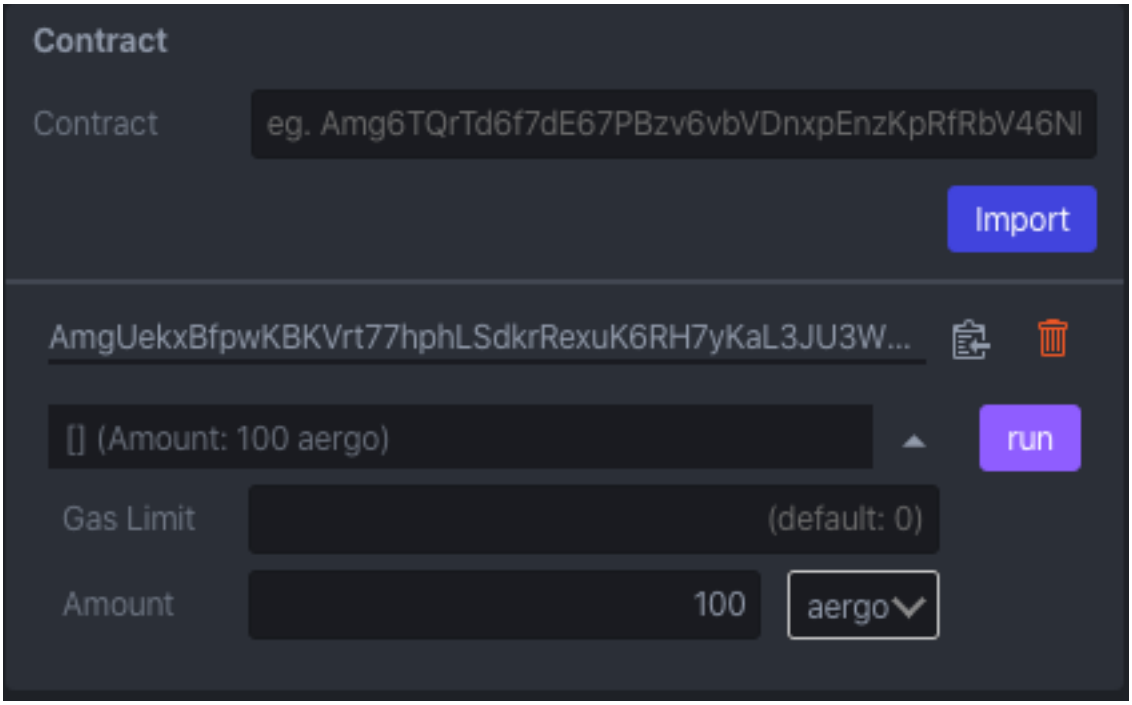
You can execute contract with configuring gas limit. 0 limit means infinite (uses as much as possible).



4.4.4 Amount

You can execute contract with aergo token. Make sure function is registered as payable.

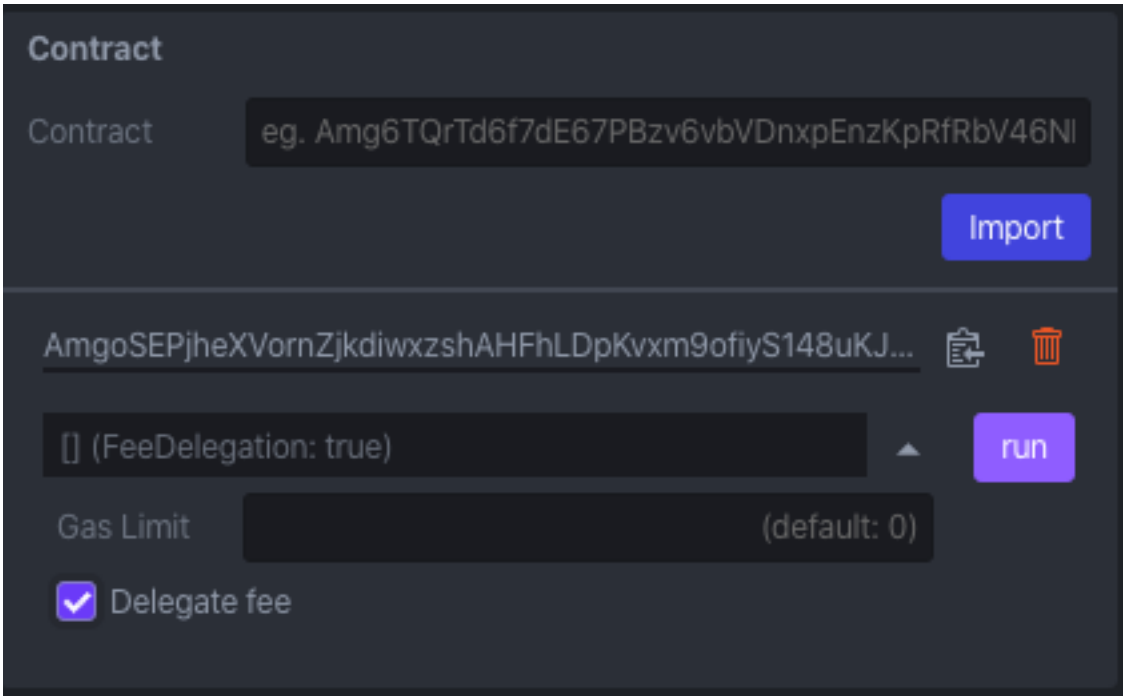
```
...

function run()
end

...

-- registered as payable
abi.payable(run)
```



4.4.5 Fee delegation

You can execute contract with fee delegation. When contract is executed with fee delegation, the contract pays fee on behalf of contract executor. Make sure function is registered as fee_delegation and a contract has enough aergo token.

```
...

function run()
end

-- registered as fee delegation
abi.fee_delegation(run)

-- register as execution
abi.register(run)
```



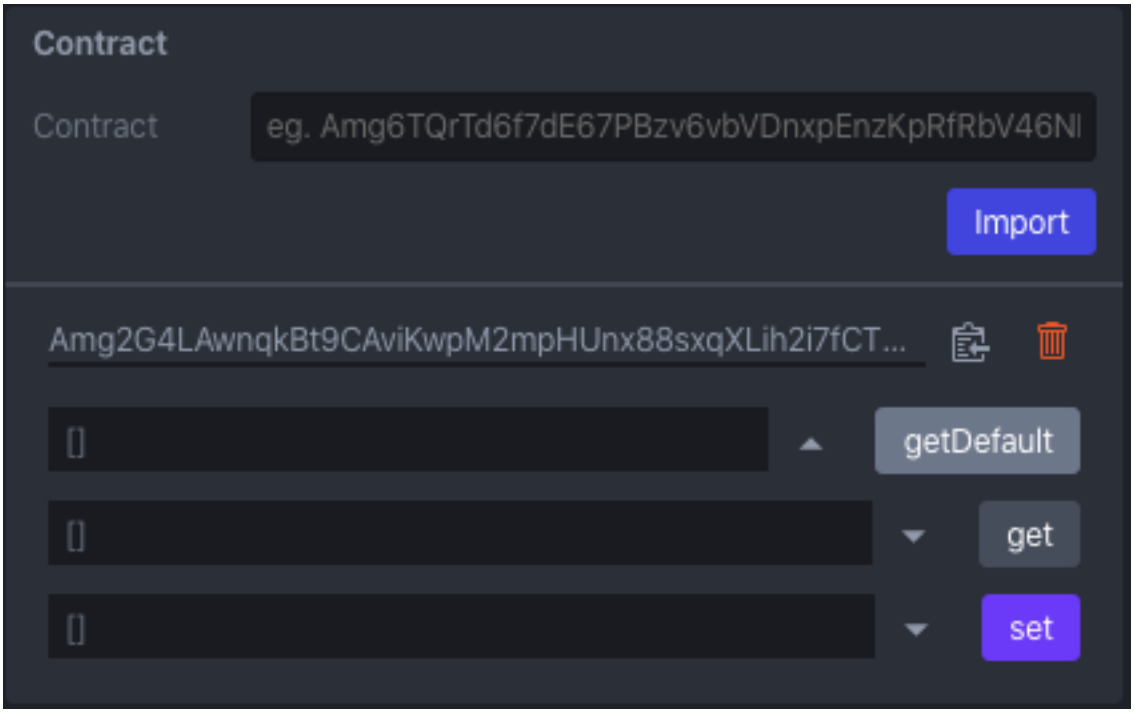
4.5 Query

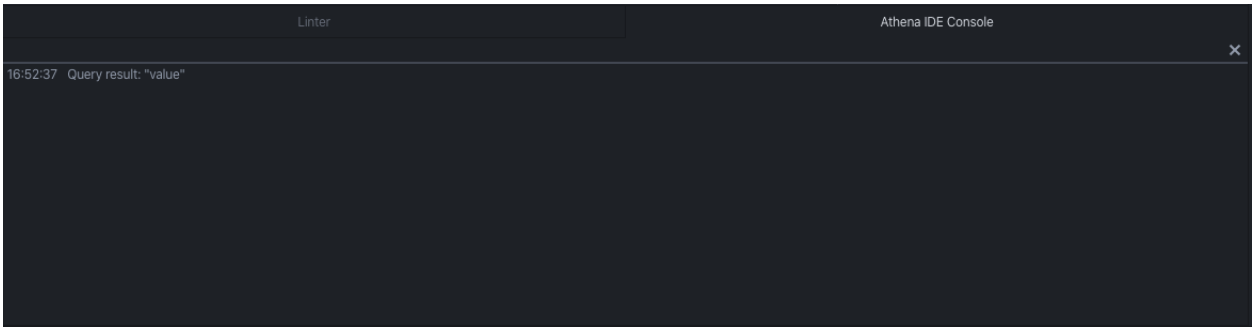
Contract query can check status of contract state db. Any function registered in register_view can be invoked as query.

4.5.1 Without args

You can query contract status without arguments.

```
...

-- no arguments
function getDefault()
  return system.getItem("k1")
end

-- registered as register_view
abi.register_view(getDefault)
```

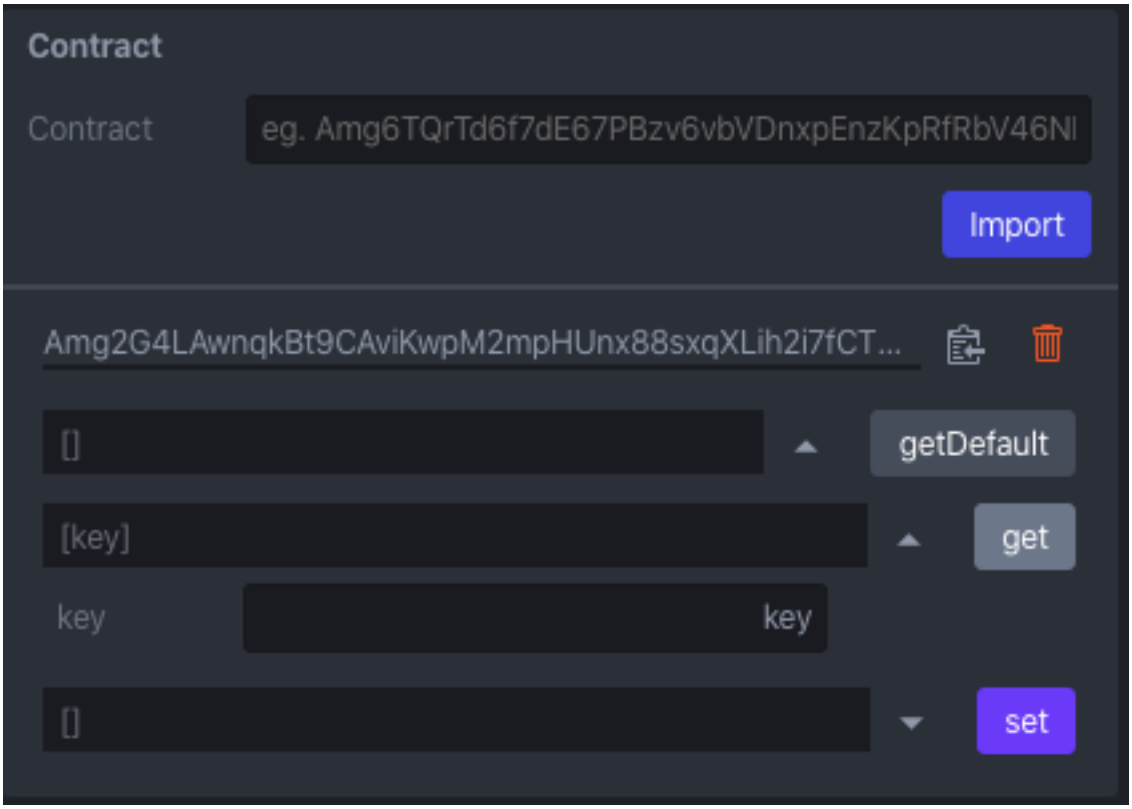


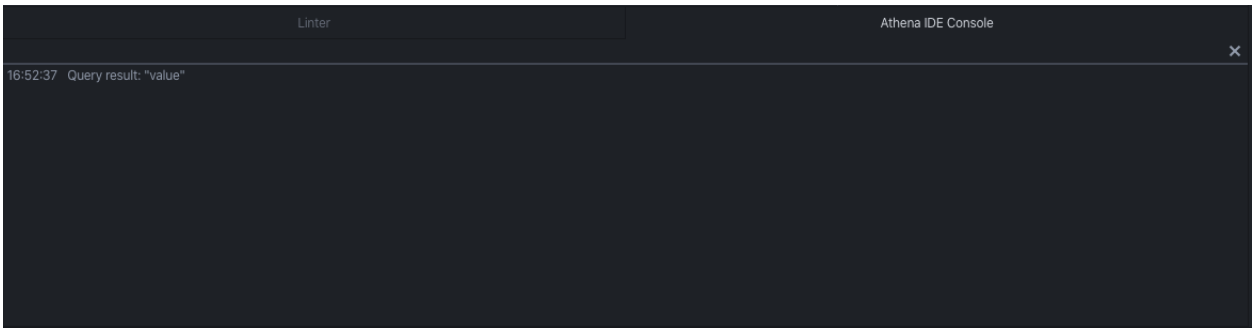


4.5.2 With args

You can query contract status with arguments.

```
...

-- arguments (key)
function get(key)
  return system.getItem(key)
end

-- registered as register_view
abi.register_view(get)
```





4.6 Varargs

Lua supports varargs. The varargs is denoted by ... in argument.

```
...

-- ... : varargs
function set(key, ...)
  local s = ""
  for i,v in ipairs{...} do
    s = s .. v
  end
  system.setItem(key, s)
end

...
```

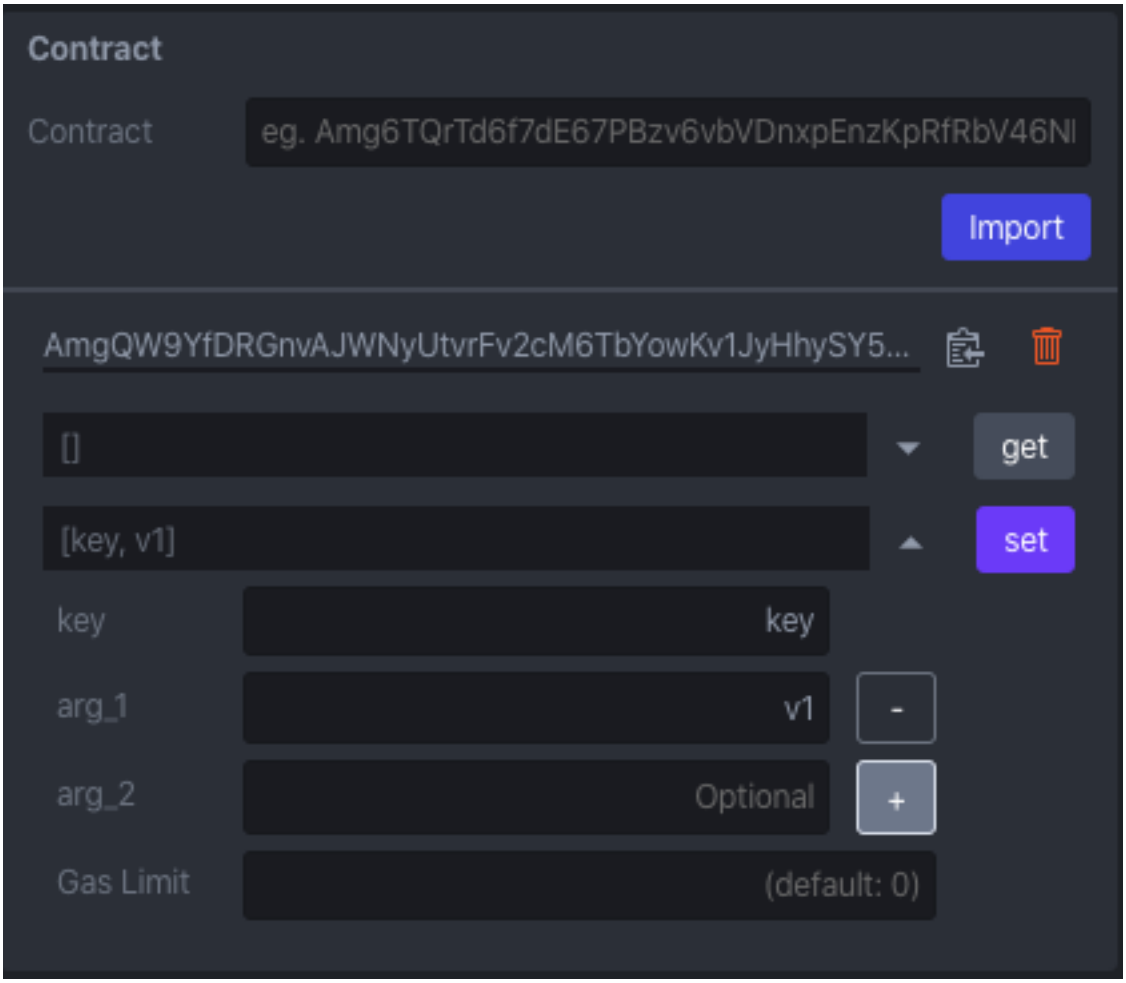
(continues on next page)

(continued from previous page)

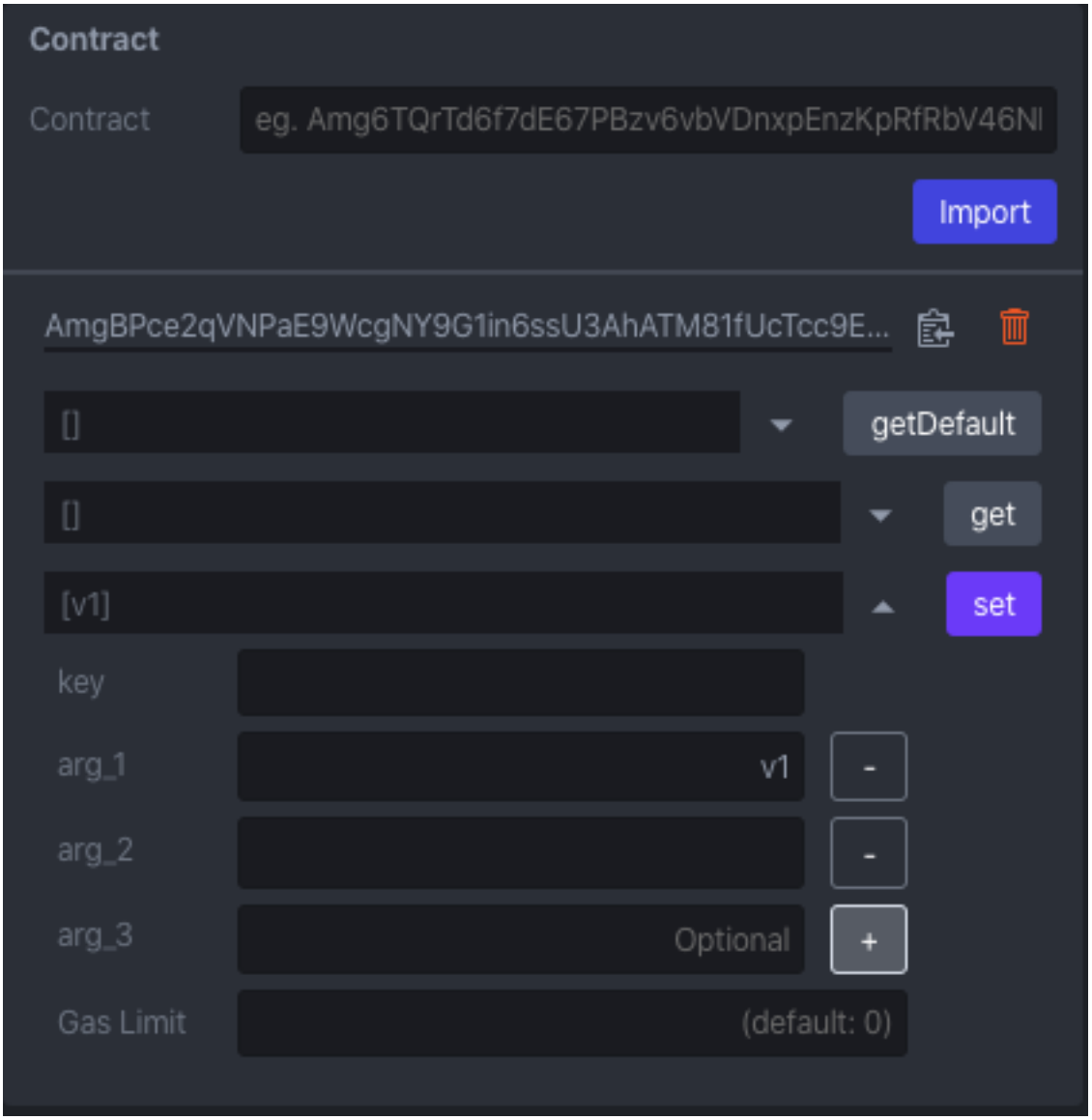
```
abi.register(set)
```

4.6.1 Add

Click **+** button

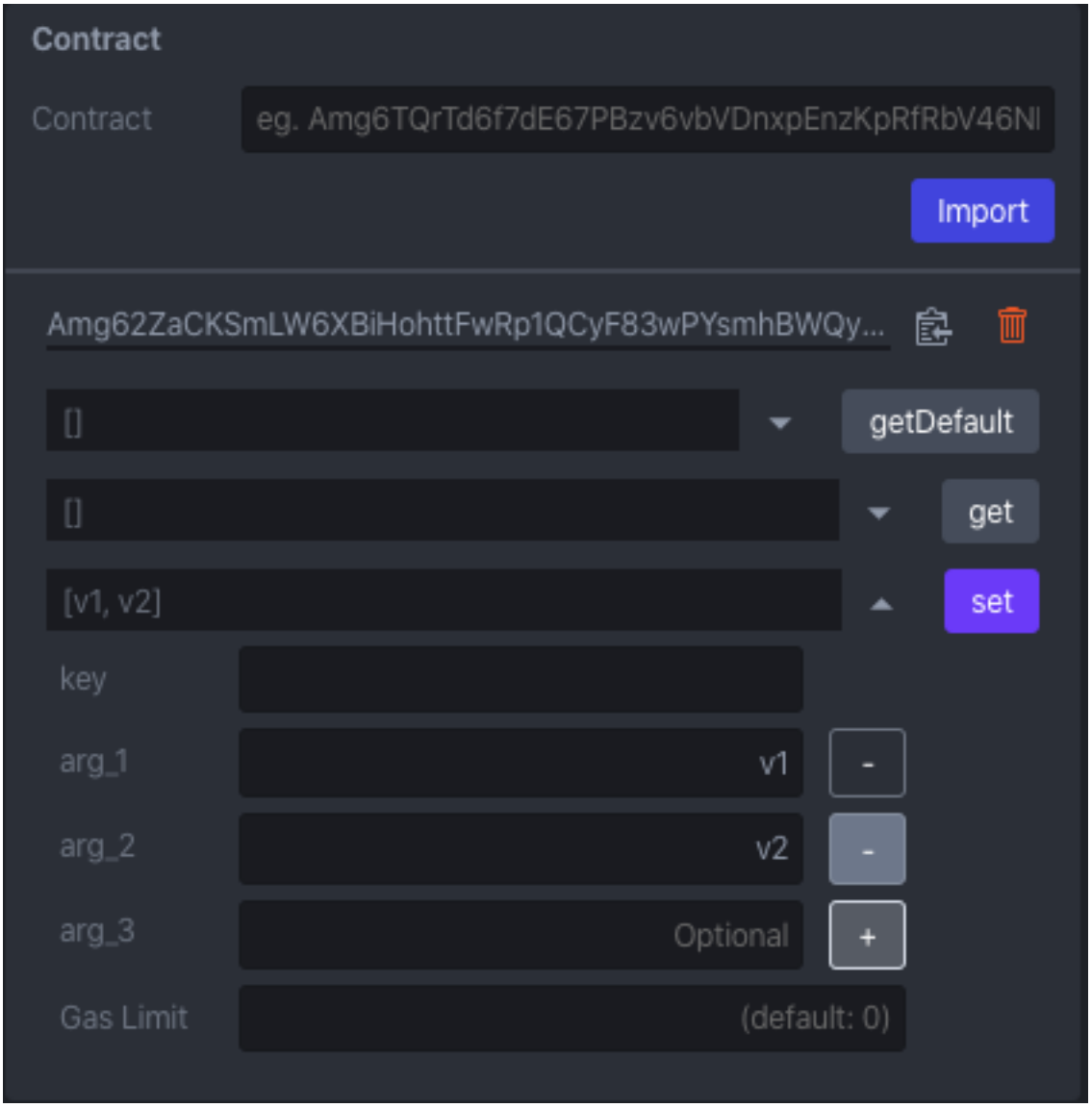


Argument added

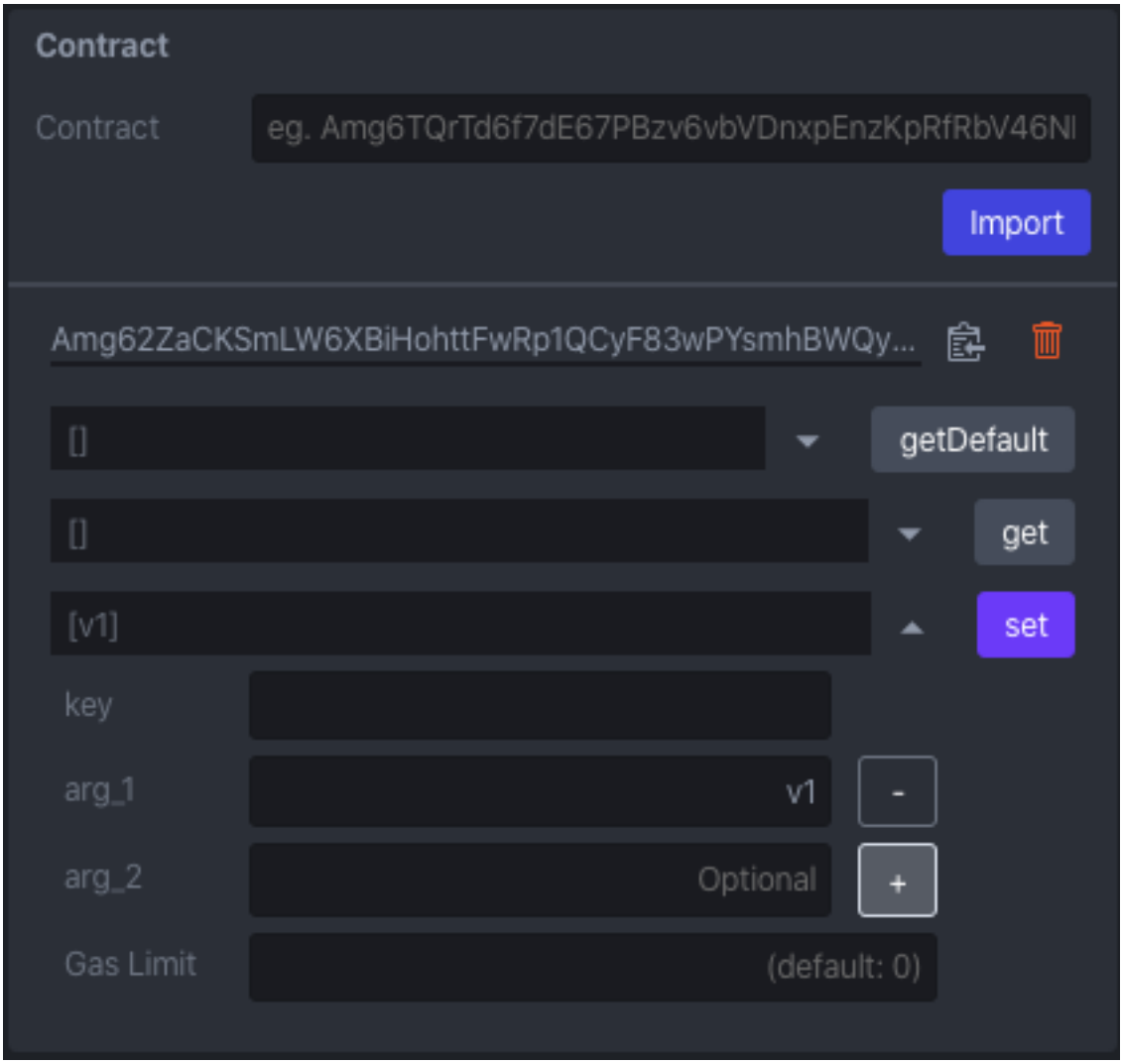


4.6.2 Remove

Click **-** button



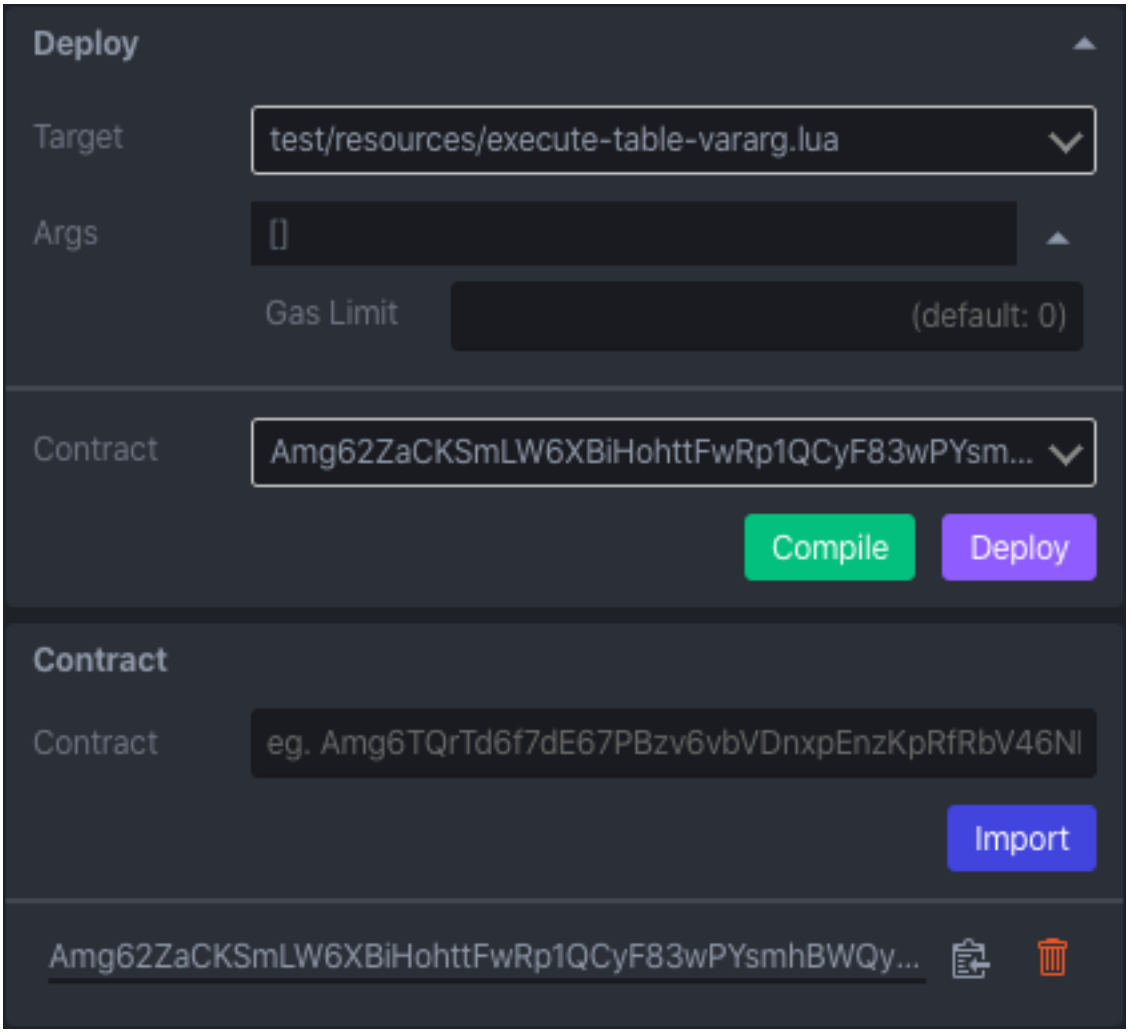
Argument removed



4.7 Redeploy (private mode only)

You can redeploy already deployed contract. This is supported in a private mode only. Make sure redeployer account is deployer of already deployed one.

To redeploy contract, select deployed contract and click deploy button.


Deploy
Target
test/resources/execute-table-vararg.lua
Args
[]
Gas Limit
(default: 0)
Contract
Amg62ZaCKSmLW6XBiHohttFwRp1QCyF83wPYsm...
Compile
Deploy
Contract
Contract
eg. Amg6TQrTd6f7dE67PBzv6vbVDnxpEnzKpRfRbV46Nl
Import
Amg62ZaCKSmLW6XBiHohttFwRp1QCyF83wPYsmhBWQy...