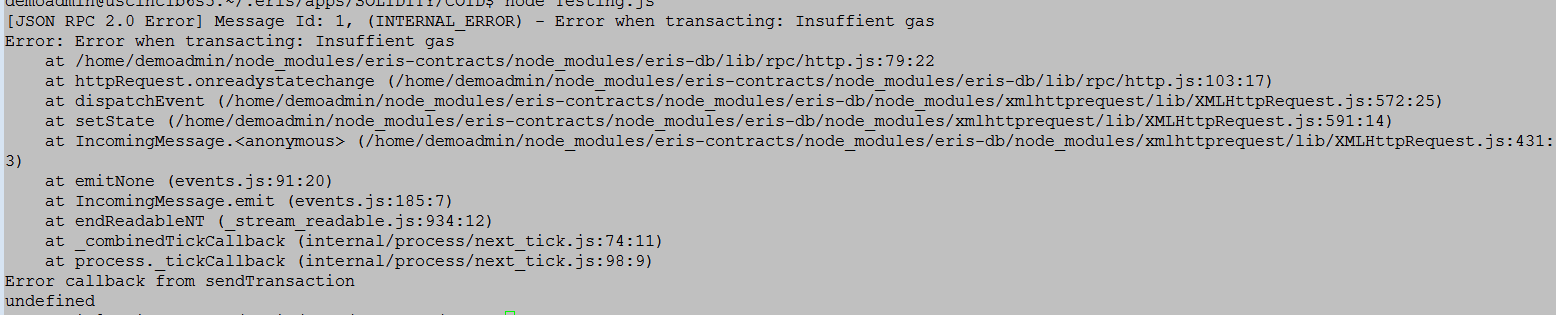
**ERIS ISSUE RECREATION** **Big Array Lookups**

Summary: The motive behind this function is to make light of, what perhaps may be a compiler issue. The contract makes use of looping through two arrays of size 100, and making lookups in the array. There are no out of bounds error in the contract.

Upon changing the size of the array to 10 (and changing nothing else), the issue went away.

Output: Here is the output when the arrays have size 100. (Out of gas)



Here is the output when the arrays have size 10. (No error)



Issue Recreation:

For the arrays with size 100, below we have an epm, a contract and a javascript file. Run the javascript file to get the issue. Following these, we have an epm, a contract and a javascript file when the arrays are size 10. In this case, there is no error.

**EPM (Case array size 100)**

jobs:

- name: Testing

job:

deploy:

contract: Testing.sol

wait: true

**CONTRACT (Case array size 100)**

contract Testing

{

function testing(bytes32[100] a, bytes32[100] b) returns (bytes32[100] aa, bytes32[100] bb)

{

for(uint i = 0; i < a.length; i++)

{

for(uint j = 0; j < b.length; j++)

{

aa[i] = b[j];

bb[i] = a[j];

}

}

}

}

**Javascript (Case array size 100)**

'use strict'

var contracts = require('eris-contracts')

var fs = require('fs')

var http = require('http')

//Set up addresses for deployed contracts

var address = require('./epm.json').Testing

var abi = JSON.parse(fs.readFileSync('./abi/' + address, 'utf8'))

var accounts = require('./accounts.json')

var chainUrl

var manager

var contract

var server

var hostname = 'localhost';

var port = 8004;

chainUrl = 'http://localhost:1337/rpc'

// Instantiate the contract object manager using the chain URL and the account data.

manager = contracts.newContractManagerDev(chainUrl, accounts.chainbreaker\_root\_000)

// Instantiate the contract object using the ABI and the address.

contract = manager.newContractFactory(abi).at(address)

var a = ["aaa","bbb","ccc"];

var b = ["aba", "bab", "cba"];

for(var i = a.length; i < 100; i++)

{

a[i] = "0";

b[i] = "0";

}

contract.testing(a,b,function(error,result)

{

console.log(result);

});

**EPM (Case array size 10)**

jobs:

- name: Testing

job:

deploy:

contract: Testing.sol

wait: true

**Contract (Case array size 10)**

contract Testing

{

function testing(bytes32[10] a, bytes32[10] b) returns (bytes32[10] aa, bytes32[10] bb)

{

for(uint i = 0; i < a.length; i++)

{

for(uint j = 0; j < b.length; j++)

{

aa[i] = b[j];

bb[i] = a[j];

}

}

}

}

**Javascript (Case array size 100)**

'use strict'

var contracts = require('eris-contracts')

var fs = require('fs')

var http = require('http')

//Set up addresses for deployed contracts

var address = require('./epm.json').Testing

var abi = JSON.parse(fs.readFileSync('./abi/' + address, 'utf8'))

var accounts = require('./accounts.json')

var chainUrl

var manager

var contract

var server

var hostname = 'localhost';

var port = 8004;

chainUrl = 'http://localhost:1337/rpc'

// Instantiate the contract object manager using the chain URL and the account data.

manager = contracts.newContractManagerDev(chainUrl, accounts.chainbreaker\_root\_000)

// Instantiate the contract object using the ABI and the address.

contract = manager.newContractFactory(abi).at(address)

var a = ["aaa","bbb","ccc"];

var b = ["aba", "bab", "cba"];

for(var i = a.length; i < 10; i++)

{

a[i] = "0";

b[i] = "0";

}

contract.testing(a,b,function(error,result)

{

console.log(result);

});