# **Creating Contract Instance**

**web3 0.20.7 :-**

var Echoupal2Instance = web3.eth.contract(Echoupal2ABI).at(Echoupal2Address);

**web3 1.0.0-beta.36 :-**

var Echoupal2Instance = new web3.eth.Contract(Echoupal2ABI, Echoupal2Address);

# **Deploying Contract**

**web3 0.20.7 :-**

const Echoupal1Object = web3.eth.contract(Echoupal1ABI);

Echoupal1Instance = Echoupal1Object.new(

{

data: '0x' + Echoupal1ByteCode,

from: web3.eth.coinbase,

gasPrice: web3.eth.gasPrice,

gas: 8000000

}, function (err, res) {

if (err) {

console.log(err.toString());

return;

}

if (res.address) {

//do something

}

**web3 1.0.0-beta.36 :-**

web3.eth.getCoinbase()

.then(function (coinbase) {

web3.eth.personal.unlockAccount(coinbase, "admin")

.then((res) => {

console.log("Coinbase Unlocked!");

web3.eth.getGasPrice()

.then(function (gasPrice) {

var transactionObject = {

from: coinbase,

gasPrice: gasPrice,

gas: 8000000

};

Echoupal1Instance.deploy({

data: '0x' + Echoupal1ByteCode,

})

.send(transactionObject, function (error, transactionHash1) {

console.log("Transaction Hash 1: " + transactionHash1)

})

.then(function (newEchoupal1Instance) {

//do something

}

# **Post Function with Single Transaction**

**web3 0.20.7 :-**

//Pre-seeding Participant's account

web3.eth.sendTransaction(

{ from: web3.eth.coinbase, to: ethereumAddress, value: web3.toWei(10, "ether") },

console.log("User Account Seeded with 10 Ethers"));

var UserRegistered = Echoupal1Instance.UserRegistered(); //event

var block = web3.eth.getBlock('latest').number;

console.log("Latest Block No.: " + block);

// Transaction object

var transactionObject = {

data: '0x' + Echoupal1ByteCode,

from: web3.eth.coinbase,

gasPrice: web3.eth.gasPrice,

gas: 8000000

};

Echoupal1Instance.registerUser.sendTransaction(

ethereumAddress, id, email, phoneNumber, fullName, firstName, lastName, localAddress,

participantType, transactionObject, function (err, result) {

if (err) {

console.log(err);

}

else {

UserRegistered.watch(function (error, result1) {

if (!error) {

if (result1.blockNumber > block) {

//do something

UserRegistered.stopWatching();

}

}

else {

console.log(error);

}

});

}

});

**web3 1.0.0-beta.36 :-**

web3.eth.getCoinbase()

.then(function (coinbase) {

web3.eth.personal.unlockAccount(coinbase, "admin")

.then((response) => {

console.log("Coinbase unlocked!")

web3.eth.personal.newAccount("admin")

.then(function (ethereumAddress) {

web3.eth.sendTransaction({

from: coinbase,

to: ethereumAddress,

value: web3.utils.toWei('10', 'ether')

})

.then(

console.log("User Account Seeded With 10 Ethers"),

web3.eth.getGasPrice()

.then(function (gasPrice) {

var transactionObject = {

from: coinbase,

gasPrice: gasPrice,

gas: 8000000

}; Echoupal1Instance.methods.registerUser(ethereumAddress, id, email,

phoneNumber, fullName, firstName, lastName, localAddress,

participantType)

.send(transactionObject)

.on('transactionHash', function (hash) { //console.log("Transaction Hash: " + hash);

})

.on('receipt', function (receipt) {

//do something

})

);

})

})

});

# **Post Function with Multiple Transactions**

**web3 0.20.7 :-**

Echoupal1Instance.createFarmer1.sendTransaction(

ethereumAddress, villageId, houseNo, educationId, dob, doa, fatherName, spouseName,

panNumber, gender, dateOfRegistration, farmerType, displayOrder,

statusId, transactionObject, function (err, result) {

if (err) {

console.log(err);

}

else {

FarmerCreated.watch(function (error, result1) {

if (!error) {

if (result1.blockNumber > block && result1.args.success == true) {

FarmerCreated.stopWatching();

Echoupal1Instance.createFarmer2.sendTransaction(

createdBy, createdDate, modifiedBy, modifiedDate, hashData,

transactionObject, function (err1, result2) {

if (err1) {

console.log(err1);

}

else {

var FarmerCreated = Echoupal1Instance.FarmerCreated({}, { fromBlock: result1.blockNumber + 1, toBlock: 'latest' });

FarmerCreated.watch(function (error1, result3) {

if (!error1) {

if (result3.blockNumber > block && result3.args.success == true) {

//do something

} FarmerCreated.stopWatching();

}

}

else {

console.log(error1);

}});}});}}

else {

console.log(error);

}});}});});

**web3 1.0.0-beta.36 :-**

Echoupal1Instance.methods.createFarmer1(ethereumAddress, villageId, houseNo,

educationId, dob, doa, fatherName, spouseName, panNumber, gender,

dateOfRegistration, farmerType, displayOrder, statusId)

.send(transactionObject)

.on('transactionHash', function (hash1) {

})

.on('receipt', function (receipt1) { Echoupal1Instance.methods.createFarmer2(createdBy, createdDate, modifiedBy,

modifiedDate, hashData)

.send(transactionObject)

.on('transactionHash', function (hash2) {

})

.on('receipt', function (receipt2) {

//do something

})

.on('error', console.error);

})

.on('error', console.error);

})})});

# **Get Function with Single Record Fetch**

**web3 0.20.7 :-**

var userProfile1 = Echoupal1Instance.viewUserProfile1(id);

var userProfile2 = Echoupal1Instance.viewUserProfile2(id);

//do something with data

**web3 1.0.0-beta.36 :-**

Echoupal1Instance.methods.viewUserProfile1(id).call({ from: ethereumAddress })

.then(function (userProfile1) {

Echoupal1Instance.methods.viewUserProfile2(id).call({ from: ethereumAddress })

.then(function (userProfile2) {

//do something with data

});

});

});

# **Get Function with Multiple Record Fetch**

**web3 0.20.7 :-**

var alertCount = Echoupal2Instance.getAlertCount();

for (var i = 0; i < alertCount; i++) {

var alertDetails = Echoupal2Instance.viewAlertByIndex(i);

var data = {

id: alertDetails[0],

alertTypeId: alertDetails[1],

raisedBy: alertDetails[2],

farmerId: alertDetails[3],

isClosed: alertDetails[4].toString()

}

alertList.push(data);

}

res.send(alertList);

});

**web3 1.0.0-beta.36 :-**

var ethereumAddress = req.query.EthereumAddress;

var alertCommentList = new Array;

Echoupal2Instance.methods.getAlertCommentCount().call({ from: ethereumAddress })

.then(function (alertCommentCount) {

getAlertCommentList(alertCommentCount);

});

async function getAlertCommentList(alertCommentCount) {

for (var i = 0; i < alertCommentCount; i++) {

var alertCommentDetails = await Echoupal2Instance.methods.viewAlertCommentByIndex(i).call({ from: ethereumAddress })

//do something with data

alertCommentList.push(data);

}

res.send(alertCommentList);

}});

# **Get all Events**

**web3 0.20.7 :-**

var echoupal1Events = Echoupal1Instance.allEvents({ fromBlock: 0, toBlock: 'latest' })

echoupal1Events.get(function (echoupal1Error, echoupal1Logs) {

if (!echoupal1Error) {

echoupal1Logs.forEach(function (result) {

var transactionHash = result.transactionHash;

var blockNumber = result.blockNumber;

. . . . . . . . . .

switch (eventType) {

case "SeedInputCreated":

success = result.args.success;

farmerId = result.args.farmerId;

hashValue = result.args.hashValue

break;

default:

additionalInfo = "Unknown event...";

}

var data = {

//create json data

};

EventList.push(data);

});}

**web3 1.0.0-beta.36 :-**

var EventList = new Array;

Echoupal1Instance.getPastEvents('allEvents', { fromBlock: 0, toBlock: 'latest' })

.then(function (echoupal1Events) {

echoupal1Events.forEach(function (result) {

var transactionHash = result.transactionHash;

var blockNumber = result.blockNumber;

. . . . . . . .

switch (eventType) {

case "UserRegistered":

success = result.returnValues.success;

ethereumAddress = result.returnValues.ethereumAddress

break;

default:

additionalInfo = "Unknown event...";

}

var data = {

//create json data };

EventList.push(data);

});});

# **ASCII to Hex Function**

**web3 0.20.7 :-**

web3.fromAscii(panNumber)

\****does work*** *with values that are already in Hex.*

**web3 1.0.0-beta.36 :-**

*web3.utils.asciiToHex(raisedBy)*

*\*****does not work*** *with values that are already in Hex.*

# **Listening To Specific Events**

**web3 0.20.7 :-**

UserUpdated.watch(function (error, result1) {

if (!error) {

if (result1.blockNumber > block) {

//do something

UserUpdated.stopWatching();

}}

else {

console.log(error);

}});

**web3 1.0.0-beta.36 :-**

Echoupal1Instance.events.UserUpdated({

fromBlock: previousBlockNumber + 1

})

.on('data', function (event) {

//do something

})

.on('changed', function (event) {

console.log(event)

})

.on('error', function (error) {

console.log(error)

});

# **Sending Signed Transaction (with Private Key)**

**web3 1.0.0-beta.36 :-**

var keyObject = keythereum.importFromFile(ethereumAddress, dataDir);

var privateKey = keythereum.recover("admin", keyObject);

web3.eth.getTransactionCount(ethereumAddress)

.then(function (nonce) {

web3.eth.getBlockNumber()

.then(function (previousBlockNumber) {

var data = Echoupal1Instance.methods.updateUser(id, email, phoneNumber, fullName,

firstName, lastName, localAddress, participantType).encodeABI();

var rawTransactionObject = {

nonce: nonce,

from: ethereumAddress,

to: Echoupal1Address,

value: '0x00',

gasPrice: '0x00',

gas: '0x7A1200',

data: data

};

var tx = new Tx(rawTransactionObject);

tx.sign(privateKey);

var serializedTx = tx.serialize();

web3.eth.sendSignedTransaction('0x' + serializedTx.toString('hex'))

.on('receipt', function () {

Echoupal1Instance.getPastEvents('UserUpdated', {

fromBlock: previousBlockNumber + 1,

toBlock: 'latest'

})

.then(function (events) {

var data = {

Success: events[0].returnValues.success

}

res.send(data)

});})})})

# **Sending Signed Transaction (with Passphrase)**

**web3 1.0.0-beta.36 :-**

web3.eth.getTransactionCount(ethereumAddress)

.then(function (nonce) {

web3.eth.getBlockNumber()

.then(function (previousBlockNumber) {

web3.eth.personal.signTransaction({

nonce: nonce,

from: ethereumAddress,

gasPrice: "0",

gas: "8000000",

to: Echoupal1Address,

value: "",

data: Echoupal1Instance.methods.updateUser(id, email, phoneNumber, fullName,

firstName, lastName, localAddress, participantType).encodeABI()

}, "admin")

.then(function (signedTransactionData) {

web3.eth.sendSignedTransaction(signedTransactionData.raw)

.on('receipt', function () {

Echoupal1Instance.events.UserUpdated({

fromBlock: previousBlockNumber + 1

})

.on('data', function (event) {

var data = {

Success: event.returnValues.success

}

res.send(data)

})

.on('changed', function (event) {

console.log(event)

})

.on('error', function (error) {

console.log(error)

});

});

});

});

});

* This method **DOES NOT** unlock your account.
* Faster than the previous method.

# **Creating external account (metamask) and sending raw transaction using private key from UI**

* import './leaders.html'
* import { Meteor } from 'meteor/meteor';
* import { Blaze } from 'meteor/blaze';
* var handle = require('ethereumjs-tx');
* Template['components\_leader\_dashboard'].onRendered(function(){
* var template = this;
* // Store the ABI inside a variable
* var CONTRACT\_INSTANCE = web3.eth.contract(ABI).at("0xC67CbC66E64f9b19C62b39E6d60DD3d0b3a851Fd");
* var nonce
* web3.eth.getTransactionCount("0x15845672c52Ff68f662D3063552d85994c4A99fF",function(error,result) {
* if (!error) {
* var data = CONTRACT\_INSTANCE.updateUser.getData("AGENT1", "[abc@itc.com](mailto:abc@itc.com)", "123", "sunilguna","sunil","guna", "bangalore", 0)
* var rawTransactionObject = {
* nonce: result,
* from: "0x15845672c52Ff68f662D3063552d85994c4A99fF",
* to:   "0xC67CbC66E64f9b19C62b39E6d60DD3d0b3a851Fd",
* value: '0x00',
* gasPrice: '0x00',
* gas: '0x7A1200',
* data: data
* };
* var tx = new handle(rawTransactionObject);
* var privateKey = new Buffer('76b50eb7a35aeb1bfcc7b5096495bf60cd347fe192048a74e860811ccd231aa9', 'hex')
* tx.sign(privateKey);
* var serializedTx = tx.serialize();
* console.log(serializedTx.toString('hex'));
* web3.eth.sendRawTransaction('0x' + serializedTx.toString('hex'), function(err, hash) {
* if (!err){
* console.log(hash);
* });}})

# **Launch Command to start Geth with Websocket enabled :-**

geth --identity "ITCPrivateNetwork" --networkid 2018 --datadir ./DataDir/ --gasprice "0" --rpc --rpcaddr "172.16.2.94" --rpcport "8595" --rpccorsdomain "\*" --rpcapi "db,http,eth,net,personal,web3" --ws --wsaddr "0.0.0.0" --wsport "8596" --wsorigins '\*' --wsapi "db,eth,net,web3,personal,admin" --ipcpath "/home/waqqar/Blockchain/DataDir/geth.ipc" --port "4003" --targetgaslimit "6000000000000" --preload ./script.js console

# **Changes to make in Node API for using Websocket :-**

Set provider to Websocket instead of Http:

const wsProvider = new Web3.providers.WebsocketProvider("ws://<IP Address>:<Websocket Port No.>")

web3 = new Web3(wsProvider);

# **Points to remember: -**

* Using Websocket protocol, we cannot deploy contracts whose Byte Code is **more than 28kB.** Hence, for now deployment will be done through Http only.
* The **“watch()”** previously watching events has been removed.
* Method signatures for s**endTransaction** & get **Calls** have changed for web3 1.0.0.
* It’s possible to **send signed transactions (without unlocking sender’s account)** in web3 0.2.x as well as 1.0.0.
* For **fetching multiple records** in web3 1.0.0 we need to loop the **async** function using **async/await** otherwise the **synchronous for loop** won’t wait for the async function to return. **Fibers** can also be used.
* It is also possible to send signed transaction via an External Account provided by Wallet Providers such as **Metamask**.