BitUnits Whitepaper

BitUnits – Giving a bit to get started

A method for two parties (peer-to-peer) to electronically exchange digital assets and value without having to deal with a middleman. BitUnits is powered by the Ethereum Classic (ETC) blockchain

Abstract
BitUnits is a digital currency that allows for peer-to-peer exchange of value in the same way you would use cash. These transactions are the result of a mathematical construct or blockchain that remains neutral without the oversight of a financial institution.

We created BitUnits on July 7, 2018 to help foster a more inclusive ecosystem.

Bitcoin is no longer a project that people can easily get involved in. Building a $2000 mining computer or investing in BTC isn't plausible for those in poverty. It would also be ridiculously expensive to airdrop BTC to people around the world who were interested in joining the cryptocurrency market. With BitUnits - we are capable of spreading tokens around so that we may all share in the revolution. We envision a financially liberated world and for this technology to succeed cryptocurrency must be shared with everyone. BitUnits will help spread the knowledge and use of

blockchain. Welcoming newcomers to embark on the technological revolution at no cost.

BitUnits was created to expand the internet of money by introducing a new audience to cryptocurrency. The ERC 223 standard was inspiration to began experimenting with innovative ideas. There are few projects introducing potential investors who are hesitant to join the volatile market. We are doing our part by offering a zero risk way to get more people involved in the growing ecosystem.

The code was designed to create a rare digital asset that could be manually distributed for several reasons. For one, we did not want to host an ICO or require people to buy our token to experience this technology. Making the integration process a learning experience for newcomers rather than solely an investment. The supply was limited so that token holders would be rewarded for supporting the ecosystem and if there was minimal growth. Our team is committed to see BitUnits become valued not for it's price point - but for being the project that familiarized people with cryptocurrency.

Imagine living in a country where the currency was hyper-inflated by 25,000% and wasn't even worth the paper it was printed on. Those living in poverty are unable to afford Bitcoin or Ethereum Classic. Investing in a virtual currency is nonsensical when you can barely cover the essentials. Even if it was for their benefit in the future. For the western world this would be a devastating situation to live in. Rendering people unable to experience this technology for themselves. This is where BitUnits comes in to play. We are here for those people. We are willing to spend the fees UNITS into the hands of those who are less privileged than us. And to ensure that BitUnits can be exchanged or sent to a third party we can even donate small amounts of ETC to cover fees. Our goal is helping as many people reach blockchain as we can.

BitUnits is helping the community transact in a manner that does not require a middleman. This is the community we are trying to build. Spreading the blockchain with everyone is crucial if we want to reshape the way we exchange goods, services, and value. The knowledge and use of Bitcoin was gradually spread by word of mouth since Satoshi released the protocol in 2009. And we want BitUnits' to mimic this growth over the next few years. First we will distribute 50% of the supply by airdrops and engagements with the community. While also encouraging our community to share what they've learned. Once the 50% distribution of BitUnits is complete and we will begin building our community from the ground up. Offering a development bounty for those interested in making a difference in the world.

Why use Ethereum Classic?

Ethereum Classic is a next generation blockchain platform that is capable of building a new internet infrastructure – one that can dramatically enhance the way that information and value are shared in the digital economy. Unlocking trillions of dollars in untapped economic surplus in the process. The platform lets anyone build and use decentralized applications, dApps for short, that run on blockchain technology. Like Bitcoin, no one controls or owns Ethereum Classic – it is an open-source project built by decentralized teams around the world. Unlike the Bitcoin protocol, Ethereum Classic was designed to be adaptable and flexible. BitUnits is also one of the few projects running on the Ethereum Classic network.

Openness

The rules of the game should be open for anyone to see and understand. No registration, identification, or other preconditions should inhibit participation in any layer of the system. Anyone is free to create their own client implementing an open protocol. The network and it's security is open to contribution. Enforcing restrictions on any of these risks centralization and prevents the network from scaling. The ETC platform is recommended to any new developers who have ideas of how we can use blockchain technology to shape a better future.

Neutrality

It is important for everyone participating in blockchain-enabled cooperation to be on an equal footing. The network is impartial to economic power, ethnicity, age, gender, and profession. Rules are exactly the same for everyone, period. Without neutrality, the system is inclined towards a specific group at the expense of another. A partial network is less likely to gain universal acceptance and maximize network value for everyone.

Immutability

Blockchain preserves a true and universally acceptable transaction history. One immutable sequence of events. What's true today will always be true, regardless of business or political interests, and no amount of lobbying can change that. Since it's not possible to change history, no resources are wasted on the effort. Any sufficiently motivated and determined groups seeking to exploit loopholes, will only diminish the network value for everyone. The rules governing the blockchain network are known in advance and are exactly the same for everyone. Only with 100% consensus can the rules be changed.

We are committed to an open, neutral, and immutable blockchain. This philosophy informs all our actions and positions towards any developments in the crypto world and beyond. Subversion of any these principals should be met with resistance. All changes that strengthen or introduce centralization on the blockchain should be

fought. Only developments that are conducive to decentralization and strengthening the 3 key blockchain characteristics should be supported and encouraged. The blockchain revolution will not be centralized. Let's make sure of it.

Web 3.0: A platform for decentralized apps

Ethereum Classic is the perfect foundation to build a secure internet. As intended by the developers the platform is a blank canvas and you have the freedom to build whatever you want. The protocol is meant to be generalized so the core features can be combined in arbitrary ways. Ideally, dApp projects will leverage the Ethereum Classic blockchain to build solutions that rely on decentralized consensus to provide new products and services that were not previously possible. Ethereum Classic is perhaps best described as a decentralized ecosystem: the core protocol is supported by various pieces of infrastructure, code, and a development community that is comprised of ETC DEV, ETC Cooperative, IOHK Grothendieck, ETC Labs, and ETC Commonwealth.

Moving Forward

The next financial revolution is inevitable and lays the foundation for emerging nations to develop economic security. In the coming years, we're likely to see the first countries adopt their own cryptocurrencies. Many forms of assets, stocks, and bonds will be tokenized and moved on the blockchain. A decentralized future is assured with the precondition that people are involved and educated with the truth.

The HODL Incentive

Once listed on an exchange the countdown will commence for the long HODL program which incentivizes the community to HODL BitUnits for one year. Holders will be rewarded with their patience and commitment by receiving a percentage of the lock up tokens. Continuing every year until the last token is distributed.

Our Decentralized Journey

BitUnits code was deployed on July, 2018. Since then we have commenced the airdrop and laid the foundation for future developments. In 2018, we want to cultivate a dynamic community of new cryptocurrency enthusiasts. Paving the way for the next wave of future cryptocurrency hodlers and investors. Half of the 10,000,000 UNITS is allocated to integrating beginners for the first wave of distribution. 2,500,000 UNITS are for recruiting developers and advisors who want to make the BitUnits vision a reality.

The Point of Sale / Wallet App

We are looking for developers to help us create a user friendly Point of Sale/wallet app. The app would be a beneficial tool for integrating small businesses eager to join the coin market. Making this app will provide small businesses with the ease of transaction and record keeping for the businesses they run. The easy to use interface will process transactions for the products they offer while managing inventory and services. The app will accept BitUnits and many of the mainstream coins and tokens, such as Bitcoin, Litecoin, and certainly Ethereum Classic. There will be in app exchanging powered by ShapeShift. Later we'll integrate lightning payments and side chain features. Ideally, the store owner will enter their inventory into the app, the prices in the local currency, barcodes, and descriptions of the inventory.

The wallet features will allow people to store many private keys from the variety of cryptocurrencies like many of the wallets we already have. When a merchant is making a sale to a consumer, the teller scans in all of the items, the app will provide the customer with the price in local currency they would like to use for payment, then the camera will activate and wait for the customer to present their cryptocurrency ID and email address which can be embedded in a QR code, the email is for the receipt, and this is only if the customer has the option setup in their device.

The teller's device will then present the total in the local currency and request the payment in the specified cryptocurrency. The customer will then scan the teller's device for the address and send the payment. The receipt will be sent after the purchase has been processed. The inventory of the merchant is also updated instantly and the merchant also gets the records for that business day emailed to their inbox.

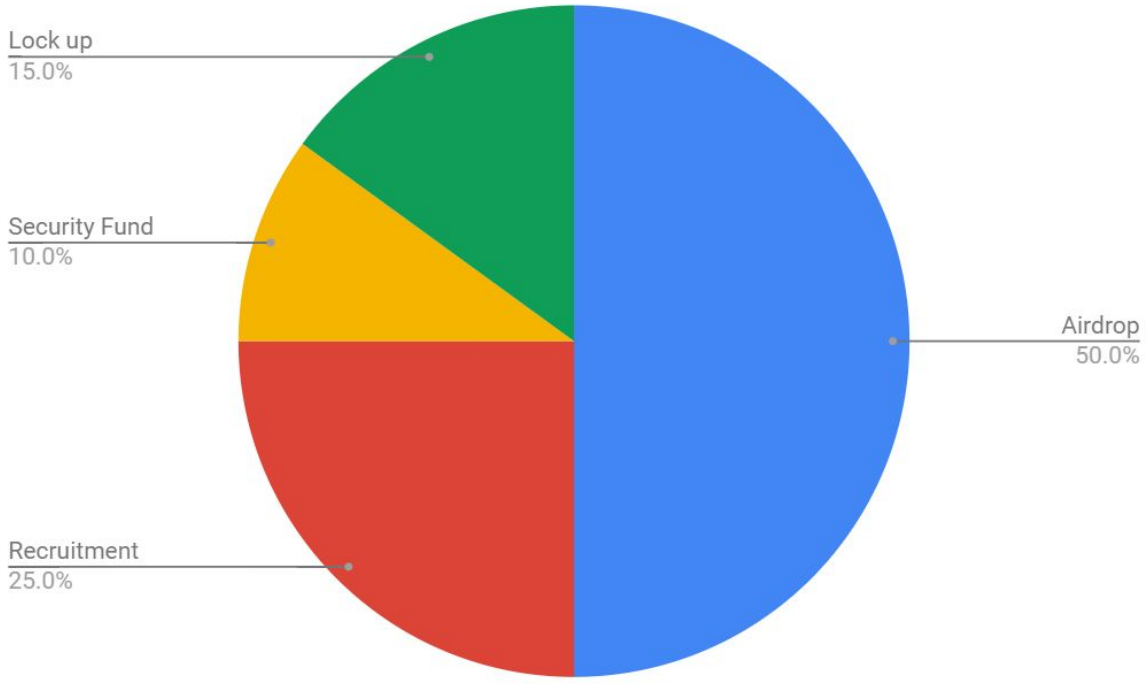
This app is achievable and will profoundly improve the developing world. The same would go for the community holding onto their BitUnits. Holders will earn a small percentage of the unfrozen tokens. We think of maybe having 25,000 to 50,000 added into circulation every year until all of the BitUnits have all been distributed into circulation.

We would like to thank you all in advance for the love and support.

Let's revolutionize money and build a tool that will benefit us all. Money is the language we use to communicate value and their is nothing more valuable than the human spirit. We all deserve to live a wealthy and healthy life. Let us make a change for all of humanity.

Graph Section and code and Team

BitUnits Emission



- **Total Supply: 10,000,000**

- **50% Community Airdrop**
- **25% Development Bounty**
- **10% Security Insurance Fund**
- **15% Locked-up for Future Emission**

BitUnits' smart contract source code

http://etherhub.io/addr/0xd1c10d433c888e6d1841ff924d0ce45157f0d5cd#tab_addr_3

```
contract ERC223 {
  uint public totalSupply;
  function balanceOf(address who) constant returns (uint);

  function name() constant returns (string _name);
  function symbol() constant returns (string _symbol);
  function decimals() constant returns (uint8 _decimals);
  function totalSupply() constant returns (uint256 _supply);

  function transfer(address to, uint value) returns (bool ok);
  function transfer(address to, uint value, bytes data) returns (bool ok);
  event Transfer(address indexed _from, address indexed _to, uint256 _value);
   event ERC223Transfer(address indexed _from, address indexed _to, uint256
_value, bytes _data);
}

contract ContractReceiver {
  function tokenFallback(address _from, uint _value, bytes _data);
}

contract ERC223Token is ERC223 {
  using SafeMath for uint;

  mapping(address => uint) balances;

  string public name;
  string public symbol;
  uint8 public decimals;
  uint256 public totalSupply;


  // Function to access name of token .
  function name() constant returns (string _name) {
```

```
      return name;
  }
  // Function to access symbol of token .
  function symbol() constant returns (string _symbol) {
      return symbol;
  }
  // Function to access decimals of token .
  function decimals() constant returns (uint8 _decimals) {
      return decimals;
  }
  // Function to access total supply of tokens .
  function totalSupply() constant returns (uint256 _totalSupply) {
      return totalSupply;
  }

   // Function that is called when a user or another contract wants to transfer funds .
   function transfer(address _to, uint _value, bytes _data) returns (bool success) {
    if(isContract(_to)) {
        return transferToContract(_to, _value, _data);
    }
    else {
        return transferToAddress(_to, _value, _data);
    }
}

  // Standard function transfer similar to ERC20 transfer with no _data .
  // Added due to backwards compatibility reasons .
  function transfer(address _to, uint _value) returns (bool success) {

    bytes memory empty;
    if(isContract(_to)) {
        return transferToContract(_to, _value, empty);
    }
    else {
        return transferToAddress(_to, _value, empty);
    }
}
```

```
  function isContract(address _addr) private returns (bool is_contract) {
      uint length;
      assembly {
                //retrieve the size of the code on target address, this needs
assembly
            length := extcodesize(_addr)
        }
        if(length>0) {
            return true;
        }
        else {
            return false;
        }
    }


   function transferToAddress(address _to, uint _value, bytes _data) private
returns (bool success) {
    if (balanceOf(msg.sender) < _value) revert();
    balances[msg.sender] = balanceOf(msg.sender).sub(_value);
    balances[_to] = balanceOf(_to).add(_value);
    Transfer(msg.sender, _to, _value);
    ERC223Transfer(msg.sender, _to, _value, _data);
    return true;
  }

  function transferToContract(address _to, uint _value, bytes _data) private
returns (bool success) {
    if (balanceOf(msg.sender) < _value) revert();
    balances[msg.sender] = balanceOf(msg.sender).sub(_value);
    balances[_to] = balanceOf(_to).add(_value);
    ContractReceiver reciever = ContractReceiver(_to);
    reciever.tokenFallback(msg.sender, _value, _data);
    Transfer(msg.sender, _to, _value);
    ERC223Transfer(msg.sender, _to, _value, _data);
    return true;
  }


  function balanceOf(address _owner) constant returns (uint balance) {
```

```
    return balances[_owner];
  }
}
/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {

  /**
  * @dev Multiplies two numbers, throws on overflow.
  */
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    if (a == 0) {
      return 0;
    }
    uint256 c = a * b;
    assert(c / a == b);
    return c;
  }

  /**
  * @dev Integer division of two numbers, truncating the quotient.
  */
  function div(uint256 a, uint256 b) internal pure returns (uint256) {
    // assert(b > 0); // Solidity automatically throws when dividing by 0
    uint256 c = a / b;
     // assert(a == b * c + a % b); // There is no case in which this doesn't
hold
    return c;
  }

  /**
   * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is
greater than minuend).
  */
  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    assert(b <= a);
    return a - b;
  }
```

```
  /**
  * @dev Adds two numbers, throws on overflow.
  */
  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}
pragma solidity ^0.4.24;



contract BitUnits is ERC223Token {
  using SafeMath for uint256;


  string public name = "BitUnits";

  string public symbol = "UNITS";

  uint public decimals = 4;

  uint public totalSupply = 10000000 * (10**decimals);

  address private treasury = 0xADACdb5BAF826FD607A00454C6ad1157E6bE4065;

  uint256 private priceDiv = 10000000;

  event Purchase(address indexed purchaser, uint256 amount);

  constructor() public {
    balances[msg.sender] = 10000000 * (10**decimals);
    balances[0x0] = 0 * (10**decimals);
  }

  function () public payable {
    bytes memory empty;
```

```
    if (msg.value == 0) { revert(); }
    uint256 purchasedAmount = msg.value.div(priceDiv);
    if (purchasedAmount == 0) { revert(); } // not enough ETC sent
    if (purchasedAmount > balances[0x0]) { revert(); } // too much ETC sent

    treasury.transfer(msg.value);
    balances[0x0] = balances[0x0].sub(purchasedAmount);
    balances[msg.sender] = balances[msg.sender].add(purchasedAmount);

    emit Transfer(0x0, msg.sender, purchasedAmount);
    emit ERC223Transfer(0x0, msg.sender, purchasedAmount, empty);
    emit Purchase(msg.sender, purchasedAmount);
  }
}
```

The team

Issatta C.
Founder, Lead dev.
iabcDigitals@BitUnits.Club
@BitUnits
Telegram

Rochelle C.
Co-Founder

Wari I
Community Manager,
Marketing

Abdulai M
Research and
Development

Francois P.
News and Public relations