|  |
| --- |
| *iRESLab, Inc.*  08/08/2018 |
|  |
| Disclaimer: All information and Intellectual Property contained in this document is the Proprietary and Confidential property of iRESlab, Inc. and intended client, and all rights herein are expressly reserved. By accepting this material the recipient agrees that the information contained herein be held in the strictest confidence and that it will not be used, copied, reproduced in whole or in part for any purpose other than it is intended. The contents shall not be revealed to any third party without iRESlab’s prior written consent. |
|  |
|  |

|  |
| --- |
| API specification document for  CoinClaim REST APIs |
| Version 0.8 |

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Author** | **Section(s)** | **Revision Notes** | **Date** |
| 0.1 | iRESlab |  | Initial draft | 05 Mar 2018 |
| 0.2 | iRESlab | 2 | Updated endpoint details | 06 April 2018 |
| 0.3 | iRESlab |  | Support for Ethereum (ETH) added | 17 May 2018 |
| 0.4 | iRESlab |  | Support for ‘CLM’ ERC-20 added | 29 June 2018 |
| 0.5 | iRESlab |  | Support for individual ERC-20 tokens added | 11 July 2018 |
| 0.6 | iRESlab |  | Support for User Account to User Address, Company Account to User Address added | 25 July 2018 |
| 0.7 | iRESlab |  | Additional response codes and save token details API updated | 08 August 2018 |
| 0.8 | iRESlab |  | Additional request parameters and response parameters in Transfer Tokens API and response codes. | 10 September 2018 |

**Table of Contents**

[1. Introduction 4](#_Toc520816245)

[2. REST APIs 4](#_Toc520816246)

[2.1. Access Token 4](#_Toc520816247)

[2.2. Generate Address 5](#_Toc520816248)

[2.3. Transfer Tokens 6](#_Toc520816249)

[2.4. Account Balance 9](#_Toc520816250)

[2.5. Save Token Details API 10](#_Toc520816251)

[3. Response Codes 11](#_Toc520816252)

# Introduction

This document contains the information about service endpoints, method types, request-response parameters for all the REST APIs to be consumed by CoinClaim.io backend system.

Services will be exposed as rest services over HTTPS (Http over SSL or HTTP Secure) protocol.

# REST APIs

## Access Token

API authentication and authorization implemented using OAUTH 2.0. Access token provides access to all other APIs.

**URL:** [{URL}/cc-blockchain-api/oauth/token?grant\_type=client**\_**credentials](http://localhost:8086/electra/oauth/token?grant_type=client_credentials)

**Request Type** : POST

**Headers:**

* **Content-Type :** application/json
* **Authorization :** Basic <Base64Hash(client\_id:client\_secret)>
  + **client\_id =** coinclaim
  + **client\_secret =** coincl@im

**Parameter List:** Nil

**Sample Request (JSON Payload):**

{ }

**Sample Response (JSON Payload):**

* ***Success:***

{

"access\_token": "395a94f6-7849-472d-806b-dc31eaf14640",

"token\_type": "bearer",

"expires\_in": 59,

"scope": "trust read write"

}

* ***Error:***

{

"timestamp": 1520255664497,

"status": 401,

"error": "Unauthorized",

"message": "Bad credentials",

"path": "/cc-blockchain-api/oauth/token"

}

## Generate Address

Generates unique address for supported blockchains i.e. Ethereum & Bitcoin and associates those addresses to a client/company or user based on *clientType* field & identified by *clientCorrelationId* in request.

**URL**: {URL}/cc-blockchain-api/address?access\_token=<access-token>

**Request Type** : POST

**Headers:**

* **Content-Type :** application/json

**Parameter List:**

* clientCorrelationId : uniquely identifies the company or a user
* clientType : identifies whether a client is a company or a user (Values - COMPANY or USER)

**Sample Request (JSON Payload):**

{

"clientCorrelationId": "4cc86e25641048788d87fad7df010604",

"clientType": "COMPANY"

}

{

"clientCorrelationId": "4cc86e25641048788d87fad7df010604",

"clientType": "USER"

}

**Sample Response (JSON Payload):**

* ***Success:***

{

"status": 200,

"code": 100,

"accountDetails": [{

"tokenType": "BTC",

"address": " mnNpnxsS7QVFHeR9XdB5RaNuorrPGPmPNA"

},

{

"tokenType": "ETH",

"address": "0x64aD6313dd83Fd4D074C78D5f291bD264c317510"

}

]

}

* ***Error*:**

{

"status": 400,

"code": 900,

"errors": [{

"code": 805,

"message": "Client or User already exists for Correlation Id"

}]

}

{

"error": "invalid\_token",

"error\_description": "Invalid access token: fdc31850-ac27-42ad-abe4-dc1e56685eca"}

**Notes:**

* List of all possible response ‘status’ & corresponding ‘description’ in JSON response will be predefined. Refer to Section x.x for response codes.

## Transfer Tokens

Transfer tokens (BTC/ETH/ERC20) from following:

* Company’s account to user’s account
* Company’s account to user’s address
* Coin Claim master’s account to user’s account. ERC-20 (‘CLM’) tokens will be transferred from deployed smart contract to beneficiary address.
* User’s account to user’s address

**URL**: {URL}/cc-blockchain-api/transfer?access\_token=<access-token>

**Request Type** : POST

**Headers:**

* **Content-Type :** application/json

**Parameter List**:

* clientCorrelationId
* userCorrelationId : identifies the user to whom coins/tokens will be allocated
* tokenType : identifies the type of token (Values - BTC or ETH or ERC20)
* tokenSymbol : represents the token which needs to be transacted (CLM for coin claim token else specific token symbol or code representing company’s token)
* noOfTokens : total number of tokens to be transferred
* beneficiaryAddress
* gasPrice : represents the gas price to be use in transaction
* gasLimit : represents the gas limit for the transaction

***Note –*** *Please make sure to send correct type (BTC for Bitcoin, ETH for Ethereum and ERC20 for ERC-20 tokens) and token code (CLM for Coin Claim token else specific token symbol representing company’s token)*

**Sample Request (JSON Payload):**

***Company To User Account:***

*BTC:*

{

"clientCorrelationId": "4cc86e25641048788d87fad7df010601",

"userCorrelationId": "67c86e25641048748687fa77df015601",

"tokenType": "BTC",

"noOfTokens": "0.15",

}

*ETH:*

{

"clientCorrelationId": "4cc86e25641048788d87fad7df010601",

"userCorrelationId": "67c86e25641048748687fa77df015601",

"tokenType": "ETH",

"noOfTokens": "0.15",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

*ERC20:*

{

"clientCorrelationId": "4cc86e25641048788d87fad7df010601",

"userCorrelationId": "67c86e25641048748687fa77df015601",

"tokenType": "ERC20",

"tokenSymbol": "XYZ",

"noOfTokens": "1.5",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

***Company To User Address:***

*BTC:*

{

"clientCorrelationId": "4cc86e25641048788d87fad7df010601",

"beneficiaryAddress": "mzmPq51AAbQVW3bbqmxBUncLDZnJpP5tKK",

"tokenType": "BTC",

"noOfTokens": "0.1"

}

*ETH:*

{

"clientCorrelationId" : "4cc86e25641048788d87fad7df010601",

"beneficiaryAddress" : "0x8477ea99492c285712adaE90939B9D1311ADbE0d",

"tokenType" : "ETH",

"noOfTokens":"0.1",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

*ERC20:*

{

"clientCorrelationId" : "4cc86e25641048788d87fad7df010601",

"beneficiaryAddress" : "0x8477ea99492c285712adaE90939B9D1311ADbE0d",

"tokenType" : "ERC20",

"tokenSymbol" : "XYZ",

"noOfTokens":"10",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

***CLM To User Account:***

{

"userCorrelationId": "67c86e25641048748687fa77df015601",

"tokenType": "ERC20",

"tokenSymbol": "CLM",

"noOfTokens": "20",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

***User Account To User Address:***

*BTC:*

{

"userCorrelationId" : "67c86e25641048748687fa77df015601",

"beneficiaryAddress" : "mzmPq51AAbQVW3bbqmxBUncLDZnJpP5tKK",

"tokenType" : "BTC",

"noOfTokens":"0.1"

}

*ETH:*

{

"userCorrelationId" : "67c86e25641048748687fa77df015601",

"beneficiaryAddress" : "0x8477ea99492c285712adaE90939B9D1311ADbE0d",

"tokenType" : "ETH",

"noOfTokens":"0.1",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

*ERC20:*

{

"userCorrelationId" : "67c86e25641048748687fa77df015601",

"beneficiaryAddress" : "0x8477ea99492c285712adaE90939B9D1311ADbE0d",

"tokenType" : "ERC20",

"tokenSymbol" : "XYZ",

"noOfTokens":"15",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

*CLM:*

{

"userCorrelationId" : "67c86e25641048748687fa77df015601",

"beneficiaryAddress" : "0x8477ea99492c285712adaE90939B9D1311ADbE0d",

"tokenType" : "ERC20",

"tokenSymbol" : "CLM",

"noOfTokens":"10",

"gasPrice":"21000000000",

"gasLimit":"21000"

}

**Sample Response (JSON Payload):**

* ***Success***

{

"status": 200,

"code": 100,

"message": "Success",

"accountDetails": [

{

"tokenType": "ERC20",

"tokenSymbol": "XYZ",

"address": "0x64aD6313dd83Fd4D074C78D5f291bD264c317510",

"balance": "300"

}

],

"transactionHash": "0xa640483d8450f42b6c70c35d295e17ec49cd57de1e74c13f6f8aa1b96c17dfea"

}

* ***Error:***

{

"status": 400,

"code": 900,

"errors": [{

"code": 806,

"message": "Invalid Token Amount : -1"

}],

}

{

"status": 500,

"code": 900,

"message": "Internal Server Error",

"errors": [{

"code": 500,

"message": "Internal Server Error"

}]

}

{

"status": 400,

"code": 818,

"message": "Internal Server Error",

"errors": [{

"code": 818,

"message": "Exceeds block gas limit"

}],

}

{

"status": 400,

"code": 819,

"message": "INTERNAL\_SERVER\_ERROR",

"errors": [{

"code": 819,

"message": "Intrinsic gas too low"

}],

}

{

"status": 500,

"code": 820,

"message": "INTERNAL\_SERVER\_ERROR",

"errors": [{

"code": 820,

"message": "Insufficient funds to transfer"

}],

}

## Account Balance

**URL:** {URL}/cc-blockchain-api/balance?access\_token=<access-token>

**Request Type** : POST

**Headers:**

* **Content-Type :** application/json

**Parameter List**:

* clientCorrelationId
* clientType (Values – COMPANY, USER)

**Sample Request (JSON Payload):**

{

"clientCorrelationId": "COMPANY1",

"clientType": "COMPANY"

}

**Sample Response (JSON Payload):**

* ***Success:***

{

"status": 200,

"code": 100,

"accountDetails": [{

"tokenType": "BTC",

"address": "mnNpnxsS7QVFHeR9XdB5RaNuorrPGPmPNA",

"balance": "0.55"

},

{

"tokenType": "ETH",

"address": "0x64aD6313dd83Fd4D074C78D5f291bD264c317510",

"balance": "2.7"

}

]

}

***- Error:***

{

"status": 400,

"code": 900,

"message": "Internal Server Error",

"errors": [{

"code": 804,

"message": "Company doesn’t exists for CorrelationId – COMPANY1"

}]

}

{

"status": 500,

"code": 900,

"message": "Internal Server Error",

"errors": [{

"code": 500,

"message": "Internal Server Error"

}]

}

## Save Token Details API

**URL:** {URL}/cc-blockchain-api/company/token?access\_token=<access-token>

**Request Type** : POST

**Headers:**

* **Content-Type :** application/json

**Parameter List**:

* clientCorrelationId
* tokenName
* tokenSymbol
* tokenDecimals
* tokenContractAddress
* tokenContractBinary

**Sample Request (JSON Payload):**

{

"clientCorrelationId": "Company1",

" tokenContractAddress": "<contract-address>"

}

**Sample Response (JSON Payload):**

* ***Success***

{

"status": 200,

"code": 100,

"message": "Token Details for token 'CMP1' successfully saved",

}

* ***Error***

{

"status": 400,

"code": 900,

"message": "Internal Server Error",

"errors": [{

"code": 804,

"message": "No company exists with Correlation Id – Company1"

}]

}

{

"status": 400,

"code": 900,

"message": "Internal Server Error",

"errors": [{

"code": 810,

"message": "Token with token symbol 'CMP1' already exists for Client Correlation Id – Company1"

}]

}

# Response Codes

Following are the different response codes returned by Rest APIs based on internal processing and response received from blockchain:

|  |  |  |
| --- | --- | --- |
| **Response Code** | **Description** | **Remarks** |
| 100 | TRANSACTION\_SUCCESS |  |
| 801 | GENERAL\_ERROR | Internal server error |
| 802 | INVALID\_REQUEST | Request data sent to Rest API is invalid or incomplete |
| 803 | MISSING\_OR\_INVALID\_CLIENT\_CORRELATION\_ID | Company with correlation id already exists |
| 804 | CLIENT\_DOES\_NOT\_EXISTS | Client/Company with correlation Id doesn’t exists |
| 805 | CLIENT\_ALREADY\_EXISTS | Client/Company with correlation id already exists |
| 806 | INVALID\_TOKEN\_AMOUNT | Token amount is invalid |
| 807 | MISSING\_OR\_INVALID\_BENEFICIARY\_ADDRESS | Invalid beneficiary address |
| 808 | USER\_DOES\_NOT\_EXISTS | User account doesn’t exist for given correlation id |
| 809 | USER\_ALREADY\_EXISTS | User account already exist for given correlation id |
| 810 | TOKEN\_ALREADY\_EXISTS | Token details already exists with given token Symbol/code |
| 811 | TOKEN\_DOES\_NOT\_EXISTS | Token details doesn’t exist with given token symbol/code |
| 812 | INVALID\_CLIENT\_TYPE | Invalid client type received in request |
| 813 | INVALID\_TOKEN\_TYPE | Invalid token type received in request |
| 814 | INVALID\_TOKEN\_DETAILS | Invalid token details received |
| 815 | FAILED\_RETRIEVING\_TOKEN\_BALANCE | Fail to retrieve token balance |
| 816 | TOKEN\_TRANSFER\_FAILED | Transfer of tokens failed |
| 817 | TOKEN\_CONTRACT\_ADDRESS\_INVALID | Invalid Token contract address received in request |
| 818 | GAS\_LIMIT\_EXCEEDS | Gas Limit exceeds for the transaction |
| 819 | INTRINSIC\_GAS\_TOO\_LOW | Gas is too low to process the transaction |
| 820 | INSUFFICIENT\_FUNDS | Insufficient funds to perform transfer operation |