Open Programming Interface

API Version 2.04

Release date: 1 June 2019

Document History

Release	Author	Contribution	Date	Notes
v1.0	Llew Morkel llew@Fraxeum.com	Ernst Naude	25/04/2019	API Release 2.01

Postman API Version: 2.04

Prev version	**New Version**	**Feature**	**Change**	**API Endpoints Affected**	**Notes**
1.0	2.01	Added API Authentication	API users require a key to access the Fraxeum API.	ALL	A script has been implemented that generates a
2.01	2.01	Released for EE DD	EE secret added in pre-script		
2.01	2.02	Added general APIs and KYC API			
2.02	2.03	Added blockchain APIs, MemberList API and Customer Support API	Note, we added a change in Postman to accommodate testing without IP address lockdown. Please upgrade the Postman API to version 2.03.		
2.03	2.04	Finance distribution API	Updated the finance distribution API, optimised the processing procedures.	Distributionsave Distributionlist Distributionapprove Distribute Clientdeposit Dissolve	

Table of contents

Introduction

Fraxeum offers a JSON/HTTP-REST api with more than 100-end points that allows the client to integrate every part of the Fraxeum platform.

Connecting to the API

The API requires client authentication. Authentication is done via encrypted key which is a combination of variables encrypted and signed with a generated signature key.

NOTE:

1. The signature key can only be used once and must be renewed with every request.
2. Requests are made with HTTP POST (over SSL) unless otherwise indicated.

Variable	**Value**	**Notes**
Target	http://api.fraxeum.org/demo v1	TESTNET
Target	**http://api.fraxeum.org/v1**	**MAINNET**

Generating the signature key

The signature key requires a *client secret* that is provided to the client by Fraxeum.

ACCOUNT SECURITY NOTE:
The client secret MUST NEVER be used in source code that is publicly visible. DO NOT embed in client side javascript (website or mobile apps).

Javascript example:

```
var nonce = parseInt((new Date()).getTime() / 1000);
var sigString = nonce + command + user + ipAddress;
var signatureKey = CryptoJS.HmacSHA256(sigString,key).toString();
```

NOTE: ipAddress must match the registered server IP address range that has been specified. **For test purposes the ipAddress is fixed. Please use: 10.0.0.1**

Making API request calls

Global input variables

Variable	Value	Type	Notes
user	Username	String	Provided with the client secret
nonce	nonce	String	Same nonce used to generate the last signatureKey
sign	**signatureKey**	String	As generated above

Global output variables

Variable	Value	Notes
success	true/false	True if successfully executed or false with message if failed to execute
message	String	Error message, only populated when success= failed

NOTE: Variable names marked in BOLD indicates that the variable is required in the API call. Non-bold variables are optional.

Implementing the Fraxeum API

Utility functions

Utility functions offer a range of services that return supplementary data.

List Values

List values returns an array of the specified data element. Example, when CountryId is specified

API: `listvalues`
Request type: GET

REQUEST

Variable	Value	Type	Notes
c	`listvalues`	String	Same command used to encrypt the string
sign	signatureKey	String	
field	CountryId, Designation,	String	See field descriptors below
value	*Internal use only*		
key	Deprecated.		

Field value	Returns
CountryId	Returns a list of country names and abbreviations. { "success": true, "msg": "", "data": [{ "Name": "Afghanistan", "value": "4" },...]

Designation	Returns a list of management titles created on the system. { "success": true, "msg": "", "data": [{ "Name": "CEO", "value": "1" }, ...]
currency	Returns a list of currencies created on the system. { "success": true, "msg": "", "data": [{ "Name": "INR", "value": "2" }, ...]

fees	Returns a list of names and values for specific fee classes as configured on the system.
	
	{
	"success": true,
	"msg": "",
	"data": [
	{
	"Name": "Credit card ",
	"value": "3"
	},
	{
	"Name": "EFT",
	"value": "1"
	},
	{
	"Name": "Wallet",
	"value": "0"
	}
	]
	}

IdType	Returns a list of valid personal identification types that a user can submit. { "success": true, "msg": "",

	"data": [
	{
	"Name": "Government Identity document",
	"value": "G"
	},
	{
	"Name": "Passport",
	"value": "P"
	}
	]
	}

LRType	Returns the legal and regulatory target type (Client, SPV, Consumer) that's been created on the system.
	{
	"success": true,
	"msg": "",
	"data": [
	{
	"Name": "Client",
	"value": "1"
	},
	}

MemberStatus	Returns a list of all the SPV states that is used to manage the legal structure journey through the system.
	{
	"success": true,
	"msg": "",
	"data": [
	{
	"Name": "Closed",
	"value": "C"
	}, ...
	]
	}

PropertyType	Returns a list of property types created for the system.
	{
	"success": true,
	"msg": "",
	"data": [
	{
	"Name": "Commercial",
	"value": "18"
	},
	…
	]
	}

StructureType	Returns a list legal structure types that has been defined in the system
	{
	"success": true,
	"msg": "",
	"data": [
	{
	"Name": "Limited Liability Company (Ltd)",
	"value": "2"
	},
	…
	]
	}

role	Returns a list of user roles created in the system.
	{
	"success": true,
	"msg": "",
	"data": [
	{
	"Name": "Content administrator",
	"value": "1"
	},
	…
	]
	}

supporttopic	Returns a list of support topics defined in the system.
	{
	"success": true,
	"msg": "",
	"data": [
	{
	"Name": "Assets",
	"value": "2"
	},
	{
	"Name": "KYC",
	"value": "3"
	},
	{
	"Name": "Other",
	"value": "4"
	},
	{
	"Name": "Problem signing in",
	"value": "1"
	}
	]
	}

usertype	Returns a list of system user types. { "success": true, "msg": "", "data": [{ "Name": "Client Director", "value": "6" }, { "Name": "Client Manager", "value": "7" }, { "Name": "Client Webadmin", "value": "8" }....]}

List Properties

Returns a list of properties that have been created on the platform.

API: `propertylist`
Request type: GET

REQUEST

Variable	Value	Type	Notes
c	`propertylist`	String	Same command used to encrypt the string
sign	signatureKey	String	
user	Username or email address	String	The Client whose properties should be included in the response.
state	*Deprecated (State now part of SPV)*		

RESPONSE


```
{
   "success": true,
   "msg": "",
   "data": [
      {
         "PlaceId": "1",
         "MemberId": "2",
         "Name": "Unit 3 Strelitzia Place",
         "Clientname": "PROPEXTRA SA ",
         "CompanyId": "1",
         "LegalId": "1",
         "ClassBshares": null,
         "Lat": null,
         "Long": null,
         "Description": "Two bedroom, luxury unit within walking distance from the medical
centre.",
         "Anchortenants": null,
         "VacancyM2": "0",
         "VacancyPer": "0",
         "PassingRent_m2": "0",
         "WeightedAve": "0",
         "LeaseExpiry": "0",
         "NettIncome": "5500",
         "AskingYield": "11",
         "NetAskingprice": "570000",
         "Lastvaluation": "578000",
         "Lastvaluationdate": null,
         "Currenttenancy": "Yes",
         "Contractperiod": null,
         "GLA": "15",
         "A_T_Name": [
            "Tannie Rita van Wyk"
         ],
         "RemainingContractPeriod": [
            "3"
         ],
         "Images": [
            {
               "MediaId": "4",
               "linkid": "1",
               "LinkTo": "Place",
               "FieldName": "images",
               "name": "uploads/Fraxeum/images_201907/31zqgn1410422207.jpg",
               "oldname": "item1.jpg",
```


```
    "Description": "",
    "Group": "images",
    "Type": "img",
    "Status": "N",
    "TimeStamp": "2019-02-14 10:42:31"
},
{
    "MediaId": "5",
    "linkid": "1",
    "LinkTo": "Place",
    "FieldName": "images",
    "name": "uploads/Fraxeum/images_201907/31bgzr1410429147.jpg",
    "oldname": "item2.jpg",
    "Description": "",
    "Group": "images",
    "Type": "img",
    "Status": "N",
    "TimeStamp": "2019-02-14 10:42:31"
},
{
    "MediaId": "6",
    "linkid": "1",
    "LinkTo": "Place",
    "FieldName": "images",
    "name": "uploads/Fraxeum/images_201907/31dfsp1410424061.jpg",
    "oldname": "item3.jpg",
    "Description": "",
    "Group": "images",
    "Type": "img",
    "Status": "N",
    "TimeStamp": "2019-02-14 10:42:31"
},
{
    "MediaId": "7",
    "linkid": "1",
    "LinkTo": "Place",
    "FieldName": "images",
    "name": "uploads/Fraxeum/images_201907/31wtpb1410421070.jpg",
    "oldname": "item4.jpg",
    "Description": "",
    "Group": "images",
    "Type": "img",
    "Status": "N",
    "TimeStamp": "2019-02-14 10:42:31"
},
{
```

```
                "MediaId": "8",
                "linkid": "1",
                "LinkTo": "Place",
                "FieldName": "images",
                "name": "uploads/Fraxeum/images_201907/31qvbp1410421993.jpg",
                "oldname": "item5.jpg",
                "Description": "",
                "Group": "images",
                "Type": "img",
                "Status": "N",
                "TimeStamp": "2019-02-14 10:42:31"
            }
        ],
        "profile": [
            {
                "MediaId": "2",
                "linkid": "1",
                "LinkTo": "Place",
                "FieldName": "Propertyprofile",
                "name": "uploads/Fraxeum/profile_201907/31jwmn1410425298.pdf",
                "oldname": "2 Bedroom Apartment _ flat for sale in Strand South -
P24-106997874.pdf",
                "Description": "",
                "Group": "profile",
                "Type": "doc",
                "Status": "N",
                "TimeStamp": "2019-02-14 10:42:31"
            }
        ],
        "logo": [
            {
                "MediaId": "3",
                "linkid": "1",
                "LinkTo": "Place",
                "FieldName": "logoimage",
                "name": "uploads/Fraxeum/logo_201907/31fzbv1410424290.png",
                "oldname": "promo1.png",
                "Description": "",
                "Group": "logo",
                "Type": "img",
                "Status": "N",
                "TimeStamp": "2019-02-14 10:42:31"
            }
        ]
    },
```

Ping

Checks if the user defined by the value of the `email` variable exists.

API: `ping`
Request type: GET

REQUEST
See Postman for detailed field list and descriptions.

Variable	Value	Type	Notes
c	`ping`	String	Same command used to encrypt the string
sign	signatureKey	String	
email	Username or email address	String	

RESPONSE

```
{
"success":true | false, //true if user exists - false if not.
"msg":"",
"data":[]
}
```

Uploading documents

The docuplload API allows administrators and users to upload documents like a prospectus, financial statement or photos into the system.

Valid file types are: PDF, JPEG, PNG, JPG

API: `docuplload`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

Variable	Value	Type	Notes
c	`docuplload`	String	Same command used to encrypt the string
sign	signatureKey	String	
token	Session key	String	If this is a user document that is being uploaded then a session key is required.

RESPONSE

{
"success":true | false,
"msg":"",
"data":[]
}

Viewing documents

The docview API returns all documents uploaded with the docupload API.

API: `docview`
Request type: GET

REQUEST

Variable	Value	Type	Notes
c	`docview`	String	Same command used to encrypt the string
sign	signatureKey	String	
token	Session key	String	If this is a user document that is being uploaded then a session key is required.
linkid			
linkto			
group	“fica”, “...”	String	The document group that is requested.

RESPONSE

XXXXXXXXXXX

API: `docview`
Request type: GET

Retrieves all files linked to any entity in the system (Client, Legal Structure or Investor).

REQUEST
See Postman for detailed field list and descriptions.

Variable	Value	Type	Notes
c	`docview`	String	Same command used to encrypt the string
sign	signatureKey	String	
email	Username or email address	String	

REQUEST

```
{
"success":true | false,
"msg":"",
"data":[]
}
```

List all users

The memberlist API lists all system users.

API: `memberlist`
Request type: GET

REQUEST
See Postman for detailed field list and descriptions.

Variable	Value	Type	Notes
c	`memberlist`	String	Same command used to encrypt the string
sign	signatureKey	String	
email	Username or email address	String	
include	docs		Includes links to all docs for each member in the response.
MemberType	Investor Client Legal		Limits response data to this member type only.
CompanyId	Company number	int	Required if members listed are part of specific company.

RESPONSE


```
{
   "success": true,
   "msg": "",
   "data": [
      {
         "MemberId": "23",
```

```
            "FirstName": "Test User",
            "LastName": "",
            "IdNo": "",
            "IdType": null,
            "Gender": null,
            "UserName": "FraxTest",
            "Email": "FraxTest@Fraxeum.co.za",
            "DOB": null,
            "CountryId": "710",
            "Role": "1",
            "MemberType": null,
            "CompanyId": null,
            "Level": null,
            "NodeAddress": "1PnjQs4BKyNYB6hrnD6uth1MLqH7cAsx1Phpx6",
            "NodeAddressEscrow": "16n2HxeyJHNrozq6WfHnw4EiV9at3ugMXMdjW1"
        },
        {
            "MemberId": "26",
            "FirstName": "Test User",
            "LastName": "",
            "IdNo": "",
            "IdType": null,
            "Gender": null,
            "UserName": "FraxTestUser",
            "Email": "FraxTestUseer@Fraxeum.co.za",
            "DOB": null,
            "CountryId": "710",
            "Role": "9",
            "MemberType": null,
            "CompanyId": null,
            "Level": null,
            "NodeAddress": "1JKYtYdYp2TmVLtWARugfNEkhJ9Uq7VkkGK52c",
            "NodeAddressEscrow": "1KFFzBvQfjbbAdAqaS59SXUzb3oHHo2HLtNd1S"
        }
    ]
}
a":[]
}
```

User functions

The user functions are grouped under the “General folder”.

User Registration

Users are required to register to use the platform. User registration requires the following parameters.

API: `signup`
Request type: GET

REQUEST:

Variable	Value	Type	Notes
c	`signup`	String	Same command used to encrypt the string
sign	signatureKey	String	
email	String	String	User’s email address. NOTE: See `autoconfirm` variable below.
password	String	String	Case sensitive string of letters. No restriction has been implemented.
seed	Array	String	Pipe separated string of words Restrictions: min 9 words max 12 words ORDER is important
countryId	Number ***=710 (RSA)***	Int	“Value” item as selected by the user from the country list. To get country list: Command: `listvalues` Field: `CountryId`

group	number ***Frax=3***	Int	The client supergroup ID. The system allows a user to belong to a client. The client is in a group. The client group id is provided in the onboarding email (along with secret and username).
r		String	Referral key provided when signing up. Not implemented
autoconfirm	true|false	String	See notes

NOTES:
`seed`: **Account recovery keywords**
The user is required to provide 9 to 12 keywords that enables Fraxeum to secure the user's account. The keywords can be any length, not case sensitive and may be repeated. Order of words are important. To recover a lost password the user is required to provide all words in the correct order.

`autoconfirm`: If auto confirm is true then a user doesn't have to confirm the email address (no email sent).
False requires an email address to be confirmed before account is activated.

RESPONSE:
(See notes inline - We will document this properly in due course)

```
{
   "success": true,
   "msg": "",
   "data": {
      "general": {
         "MemberId": "24", //Unique in system userID
         "FirstName": "", //to be provided later in profile update
         "LastName": "",//to be provided later in profile update
         "IdNo": "",//to be provided later in profile update
         "IdType": null,//to be provided later in profile update
         "Gender": null,//to be provided later in profile update
         "UserName": "demo2@testuser.co.za",//can be changed to a generic username
```

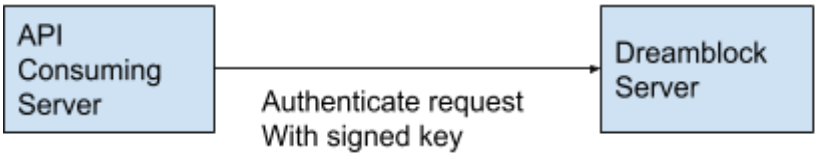

```
        "Email": "demo2@testuser.co.za",//used as primary identfier for a user
        "RefBy": "44", //referrer userID
        "DOB": null,//to be provided later in profile update
        "CountryId": "710",
        "Role": "0",//user type (0 is normal, else admin roles)
        "MemberType": null,
        "RoleName": null//to be provided later in profile update
      },
      "address": [],//user physical address
      "contacts": [],//to be provided later in profile update
      "VerificationLevel": 2,//FICA verification level (0=not verified, 1=basic verification(docs submitted), 2=full verification (externally verified)
      "ip": "",//user's current ip address
      "referral": "Z9M",//user's referral code to use when referring others
      "bank": null,//user's personal banking details
      "BankingLoaded": false,//to be provided later in profile update
      "UserImage": "https://www.w3schools.com/w3images/avatar2.png",
      "2fa": 0,//has this user setup google auth for 2FA?
      "state": 1,//login state (0=not logged in, 1=logged in)
   }
}
```

User Login

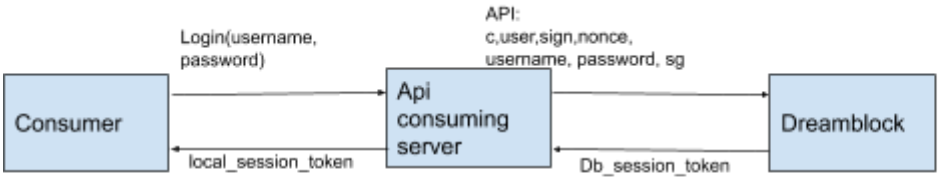
The system defines two types of users that will log into the system namely:

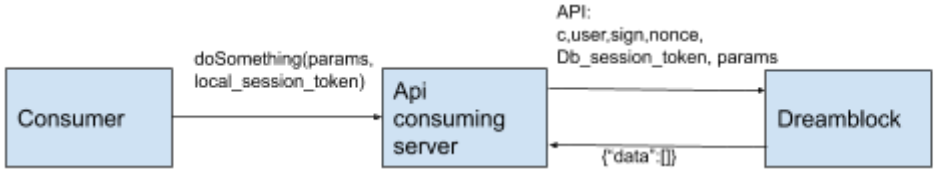
- Mid Tier Systems - API Consumer
- Consumers - Connects to system via Mid Tier Systems

Mid Tier Systems authenticate using the signatureKey.



Mid Tier Systems authenticate consumers by sending the global parameters to the Fraxeum system authenticated with the signatureKey.





The system then returns a session token (“token”) which identifies the consumer’s session. The mid tier system should return this token to the Fraxeum system with every consumer account transaction.

Users are required to log into the system to transact. The resulting user session times out after 5 minutes of inactivity.

API: `signin`
Request type: POST

Variable	Value	Type	Notes
c	`signin`	String	Same command used to encrypt the string
sign	signatureKey	String	
username	String	String	User's email address or username if set
password	String	String	User's password
sg	number ***Frax=3***	String	Client's super group ID

RESPONSE:
(See notes inline - Only commented on new variables - not commented above in registration)

```
{
   "success": true,
   "msg": "",
   "data": {
      "general": {
         "MemberId": "8",
         "FirstName": "Llew",
         "LastName": "Morkel",
         "IdNo": "",
         "IdType": null,
         "Gender": null,
         "UserName": "Llew",//Username now set
         "Email": "llewellynmorkel@gmail.com",
         "RefBy": null,
         "DOB": null,
         "CountryId": "710",
         "Role": "9", //see role list below
         "MemberType": null,
         "RoleName": "Investor"
      },
```

```
        "address": [],
        "contacts": [
            {
                "ContactId": "1",
                "ContactType": "Cell",
                "ContactVal": "0823313232"
            }
        ],
        "VerificationLevel": 2,
        "ip": "",
        "referral": "LM4",
        "bank": null,
        "BankingLoaded": false,
        "UserImage":
"https://Fraxeum.co.za/uploads/Fraxeum/Docs_201908/16sjqx2307023571.png",
        "2fa": 0,
        "state": 1
          },
 “token”:”6a4d6ca4e897368788aaed7cd45e9e91f3c646df94985a23e4cdcb26a87b8cec”
}
```


USER Role ID list

"Role": "9" maps to $RoleMap[Role] i.e. $RoleMap[9]

0: Guest
1: Power Admin
2: db_admin
3: db_manager
4: db_finadmin
5: db_fundman
6: client admin
7: client manager
8: client finadmin
9: User / member
10: Client Support admin

User Info (Get User Profile Data)

Lists all the profile information pertaining to a specific user.

Request type: POST

Variable	Value	Type	Notes
c	userinfo	String	Same command used to encrypt the string
sign	signatureKey	String	

RESPONSE:


```
{
   "success": true,
   "msg": "",
   "data": {
      "general": {
         "MemberId": "8",
         "FirstName": "Llew",
         "LastName": "Morkel",
         "IdNo": "",
         "IdType": null,
         "Gender": null,
         "UserName": "Llew",
         "Email": "llewellynmorkel@gmail.com",
         "RefBy": null,
         "DOB": null,
         "CountryId": "710",
         "Role": "9",
         "MemberType": null,
         "RoleName": "Investor"
      },
      "address": [],
      "contacts": [
         {
            "ContactId": "1",
            "ContactType": "Cell",
            "ContactVal": "0823313232"
         }
      ],
      "VerificationLevel": 2,
      "ip": "",
```

```
        "referral": "LM4",
        "bank": null,
        "BankingLoaded": false,
        "UserImage":
"https://Fraxeum.co.za/uploads/Fraxeum/Docs_201908/16sjqx2307023571.png"
    }
}
```

Reset Password

Request type: POST

Variable	Value	Type	Notes
c	userreset	String	Same command used to encrypt the string
sign	signatureKey	String	
email		String	User's registered email address
seed	Array	String	Seed words separated with pipes
pass		String	New password to be applied if seed words match

RESPONSE:

```
{
   "success": true | false
}
```

User Security - 2FA (Setup,Verify, Enable, Disable)

API: `usersecurity`
Request type: POST

Variable	Value	Type	Notes
c	usersecurity	String	Same command used to encrypt the string
sign	signatureKey	String	
type	set2fa	String	Used to initiate setup of Google Auth. Returns 2FA secret and image
type	verify2fa	String	Confirm current 2FA code. REQUIRES `code` variable
	code*	String	code
type	enable2fa	String	Checks if 2FA is enabled for this account. SEE NOTE BELOW.
type	disable2fa	String	Cancels 2FA REQUIRES `seed` OR `code`
	seed	String	
	code	String	

NOTE: `enable2fa` returns **TRUE** if 2FA is not enabled (you can enable 2FA) or FALSE if enabled (logic: can I enable2fa? True, it isn't enabled. False, it is already enabled).

RESPONSE:

Type: set2fa

```
{
   "success": true,
   "data": {
      "secret": "RBWIPYHYW6YJRKI6J3RMV3W3ARRSTMC7",
      "image":
"
BmJLR0QA/wD/AP+gvaeTAAAFW0lEQVR4nO3d0W7juBJF0WQw///LPa8BAjVIs7Ys37vWY
0OWlfQBUyCp4vefP3+++YNo/734A/jcJFgnBIiFYJASLhGCRECwgkVCsEgIFgnBIiFYJASLh
GCRECwgkVCsj8+9rHvr+/Z5No/734A/jcJFgnBIiFYJASLhGCREDCwSgkVCsEgIFgnBIiFYJASLh
GCRECwgkVCsEj8+9rHvr+/Z5No/734A/jcJFgnBIiFYJASLhGCRECwgkVCsEj8B/Uxw4c2zIEIAAAAAElFTkSuQmCC
"
   },
   "msg": ""
}
```


Type: verify2fa

```
{
   "success": true | false,
   "msg": ""
}
```

Type: enable 2fa

```
{
   "success": true | false,
```

"msg": ""
}

Type: disable 2fa

{
"success": true | false,
"msg": ""
}

Managing KYC

The system manages know your client requirement in two ways. Firstly it allows users to upload the documents specified by a specific jurisdiction to the system. Once the documents have been uploaded they can be reviewed by an administrator and approved or rejected. The rejection process is managed through the built-in support API.

Secondly, the system makes provision for external KYC verification. The Fraxeum blockchain uses Civic as its external KYC verification platform. Miners are required to create a Civic account and verify their identity through the Civic processes. For purposes of miner verification the Fraxeum system requests nothing more than confirmation from the Civic platform that the miner in question has been validated.

For purposes of automating the KYC and AML compliance process in each jurisdiction it may be required to integrate into a paid service like Jumio.

“KYC documents” is a term that the systems uses to refer to all documents associated with a user (Client, legal structure or investor) as mandated by a specific jurisdiction for purposes of legal and regulatory compliance. The requirements are configured in the system, per jurisdiction, as detailed in the Jurisdiction section of this document.

Uploading KYC documents

Uploaded KYC documents are stored and linked to the user’s profile. If the Status modifier has been set the list returned is filtered to include only selected state.

API: `KYCupload`
Request type: POST

Variable	Value	Type	Notes
c	`KYCupload`	String	Same command used to encrypt the string
sign	signatureKey	String	
token		String	Session key
Status	A, R, N	String	[A]pprove [R]refuse [N]ew

RESPONSE:

```
{
   "success": true | false
}
```

View KYC documents

The KYCList API returns a list of all documents associated with the user. Setting the status parameter filters the output.

API: `KYCList`
Request type: GET

Variable	Value	Type	Notes
c	`KYCList`	String	Same command used to encrypt the string
sign	signatureKey	String	
token		String	
status	A, R, N		[A]pprove [R]refuse [N]ew

RESPONSE:
XXXXXXXXX

Manual KYC document review

The system uses one API to approve or reject KYC documents. Each document is verified separately and as such requires a mediaId obtained from the `KYCList` API.

API: `KYCApprove`
Request type: GET

Variable	Value	Type	Notes
c	`KYCList`	String	Same command used to encrypt the string

sign	signatureKey	String	
token		String	
Status	A, R, N		
MediaId		Int	Image identifier

RESPONSE:


```
{
   "success": true,
   "msg": "",
   "data": null
}
```

Managing Jurisdiction

Fraxeum is made to be multi-jurisdictional. A single implementation of the platform supports multiple countries each with its own currency and legal as well as legal and regulatory framework.

Legal and regulatory is divided into legal and regulatory requirements for the Client (the entity creating the investment product), the SPV (the entity that owns and manages the assets) and the consumer (investor who buys the item).

Compliance with legal and regulatory requirements are enforced through document upload forms. For example, an investor may be required to FICA and as such is required to submit a copy of his or her ID and proof of residence and optionally, proof of a place of work. These requirements are created and setup in the jurisdiction:

Juristic person	**Document required (description)**	**Requirement (optional/mandatory)**
Investor	Copy of ID	mandatory
Investor	Proof of residence	mandatory
Investor	Place of work	optional

Create and Edit Jurisdiction

Adds a new country to the system.

NOTE: *Create* and *Edit* functions both use the same API. The only difference is the "jurisdictionId" variable. If a jurisdictionId is present then the system looks for that jurisdiction and updates the data with the data in the fields provided. Else if a jurisdictionId is not present then a new jurisdiction is created.

API: `jurisdictionsave`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE


```
{
   "success": true,
   "msg": "",
   "data": {
      "Name": "Jurisdiction Angola",
      "Country": "Angola",
      "CountryId": "24",
      "CurrencyName": "bgt4", //Internal mirror asset for country currency
      "CurrencySymbol": "$", //Juristic currency symbol
      "CurrencySymbolPos": "Left",
      "CurrencyId": null, //If existing currency was selected this would be populated with the currency ID
      "Extra": "{\"NodeAddress\":\"1T3aWJK48ruqc6rL2ZRRzMzCVDmQgS74Jvw2A\",\"ENodeAddress\":\"1JcW9maXTzPoDcrqPbAYXZ5UajYqLLSm6g519S\"}", //See note below
      "Type": "",
      "id": "7"
   }
}
```


NOTE:
Extra.NodeAddress: Jurisdiction currency normal wallet on the blockchain.
Extra.ENodeAddress: Jurisdiction currency escrow wallet address on the blockchain.

View Jurisdiction

Returns a list of all the parameters defined for a specified country.

API: `jurisdictionlist`
Request type: GET

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE

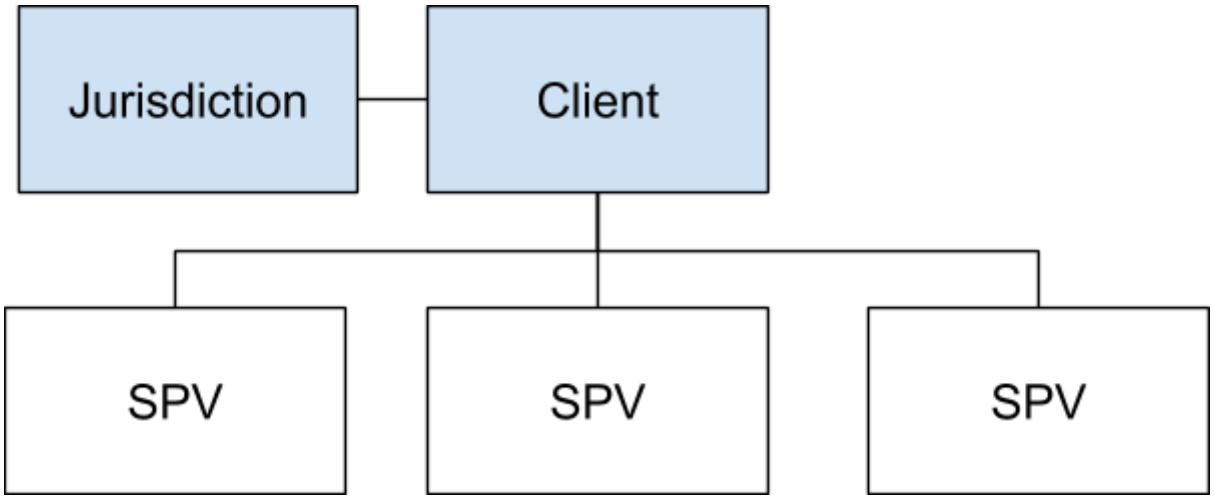
```
{
   "success": true,
   "msg": "",
   "data": [
      {
         ….
         "L_R": [ //Legal&Regulatory requirements
            {
               "SetType": "1",//L&R types - SEE NOTES BELOW
               "L_R_Name": "Memorandum of incorporation",
               "L_R_Mandatory": "1",
               "L_R_ValidTypes": [
                  [
                     "PDF"
                  ],
                  [
                     "JPEG"
                  ],
                  [
                     "JPG"
                  ],
                  [
                     "PNG"
                  ]

               ],
               "L_R_AdminApproved": null,//Does this doc require admin approval?
               "L_R_FileSize": null //Uploaded doc filesize
            },
            ….
      …..
   ]
}
```

Notes:
1. Legal & Regulatory requirement types: 1 = Client, 2 = SPV, 3 = Investor

Managing Clients

The system supports one to many clients in each jurisdiction. A client creates one or more investment products. An investment product is contained in some sort of legal structure (SPV) that allows the creation of various types of shares (class A, class B etc).



Create and Edit Client

A client in this system is defined as the company that is setting up and offering the investment product.

A client operates within a specific jurisdiction under that jurisdiction's laws.

NOTE: *Create* and *Edit* functions both use the same API. The only difference is the "clientId" variable. If a clientId is present then the system looks for that client and updates the data with the data in the fields provided. Else if a clientId is not present then a new client is created.

API: `clientsave`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE

```
{
   "success": true | false,
   "msg": ""
}
```

View Client

Request the client information for one or more clients.

API: `clientlist`
Request type: GET

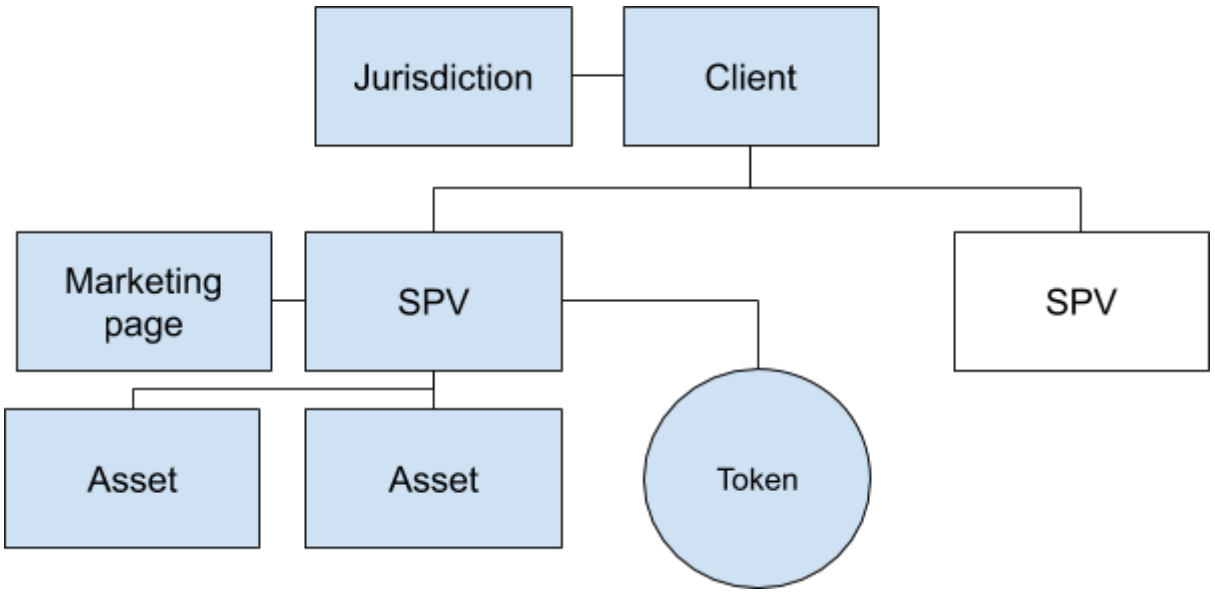
REQUEST
See Postman for detailed field list and descriptions.

RESPONSE

```
{
   "success": true | false,
   "msg": ""
}
```

Managing Legal Structures

A legal structure (SPV) is a legal entity in the specified jurisdiction.



Assets
An SPV can own one to many assets. It is important to note that fractional investment happens at SPV level not at asset level.

Crypto token association
One internal crypto asset is created for each SPV. The number of sale shares in the SPV determines the number of crypto tokens that is generated. It is possible that the SPV authorised a number of shares but only issues a portion of that. In that case, the system will generate only the issued portion of shares as tokens and generate the balance as and when they are issued.

The number of assets held by the SPV does not have any effect on the crypto token association with the SPV. One SPV, one token.

Marketing page
Each legal structure has its own, highly configurable marketing page. Read more in the Marketing page section below.

Create and Edit Legal Structure

Once an SPV has been created it is only visible to users with specific roles (all administrative roles 1 through 8 and 10). An SPV must be published before the crypto token is generated and the page becomes visible to the public.

API: `legalsave`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

NOTE:

1. DELETE SPV: To delete a legal structure send a POST request to this API with the following parameters only:
   `LegalId`: *SPV_LegalId*
   `RecStatus`: D
2. PUBLISH SPV: To publish a newly created SPV send a POST request to this API with the following parameters only:
   `LegalId`: *SPV_LegalId*
   `Status`: A

RESPONSE

```
{
   "success": true | false,
   "msg": ""
}
```

View Legal Structures

Returns a list of legal structures present on the system.

API: `legallist`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE


```
"success": true,
"msg": "",
"data": [
  {
    "LegalId": "2",
    "Name": "Retirement Prop Fund",
    "Clientname": "Central Developments Property Group",
    "CompanyId": "2",
    "StructureType": "2",
    "Registrationnumber": "123456789",
    "Country": "",
    "CountryId": "710",
    "CountryCode": "ZA",
    "Industry": "5",
    "StructureValue": "150000000.70000000", //Made up single or multiple structures
    "currency": "", //replaced by SPV currency - deprecated
    "CryptoCurrencyName": "Ret001",  //internal crypto asset name
    "CryptoCurrencySymbol": "Ret00",  //internal five digit crypto asset symbol
    "ClassAshareholding": "0.00000000", //irrelevant deprecated
    "ClassAshare": "0.00000000",//irrelevant deprecated
    "Fractions": "1000000.00000000", //the number of tokens that will be created when asset is published - this is === to ClassBshare variable
    "ClassBshareholding": "1000000.00000000", //the total class B shares authorised
    "ClassBshare": "1000000.00000000", //class B shares issued (may be <= to authorised)
    "PriceperFraction": "150.00000070", //price to charge per token
    "FractionsSold": 0, //number of tokens sold
    "MemberId": "4",
    "Extra": "{\"Prospectus\":null}", //this variable would contain the prospectus item
    "cExtra": //a list of extra field (contains all the marketing fields - See Marketing below)
"{\"NodeAddress\":\"1A3errVq45UYWn3Rymkx5mjZxetLkUUW63gH1B\" //asset wallet address
```

,\"ENodeAddress\":\"1TJoxhy9gmsu8JNKiZqPnUWxX4H5TKGJATvf8o\"}",//asset escrow wallet address
"TimeStamp": "2019-02-18 08:35:35", //created timestamp
"TokensAvailable": 1000000,//initial capacity
"substructures": "1",//number of structures that make up the StructureValue
"premium": "0.00",//percentage or amount added on top of token value to get price Percentage denoted by 0.xx and fixed by >0.
"TokenPrice": "150.00000070",
"TokensSold": 0,//number of tokens sold
"Status": "A",//publishing state - N = new, A = available, C = closed, D = deleted
"Landingpagefile": "uploads/Fraxeum/Legal_201908/35hnvw1808354805.png",//summary image for mobile app
"PerTokensSold": 0, //percentage of tokens sold
"tradestats": [] //trade statistics
}
]
}

Marketing page

Each SPV has its own marketing page. The marketing page hosts all the information for that SPV that the client wants to show investors, including but not limited to:

- Information (descriptive text)
- Images
- Videos
- PDFs like -
    - Prospectus
    - Business plan
    - Financials
- Graphs like -
    - Financial projection
    - Token sale progress
- Buy button

The marketing page is template based - all documents, data, images and media that makes up the marketing page is loaded into the system via the API and tagged with keyword identifiers.

The marketing team supplies the layout map. A layout map is simply the page design with relevant documents, media, images and data in specific positions.

Marketing page creator

A marketing page creator form allows an administrator to quickly and easily load data received from the marketing team.The marketing page creator implements the `spvmarketing` API

API: `spvmarketing`
Request type: POST

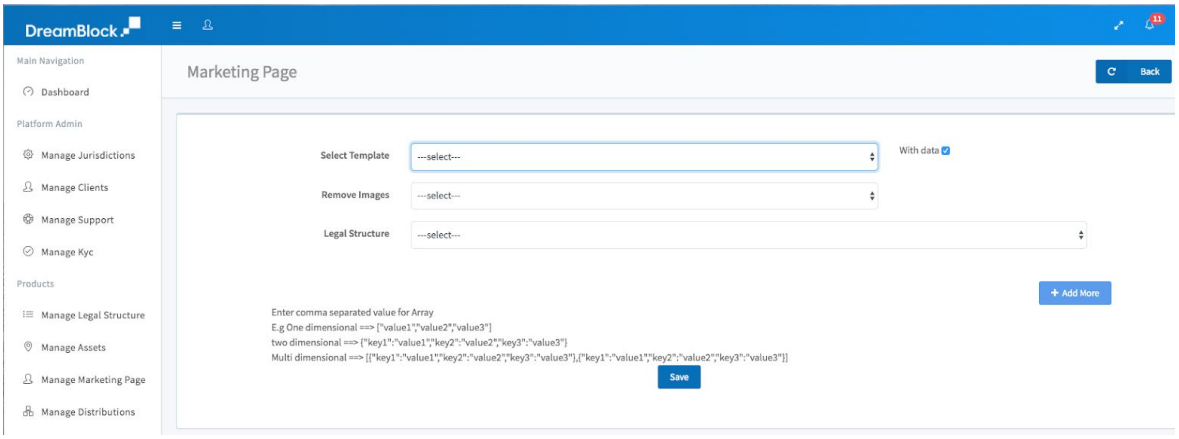
REQUEST
See Postman for detailed field list and descriptions.

RESPONSE

```
{
   "success": true | false,
   "msg": ""
}
```

Marketing page creator location:
http://Fraxeum.co.za/admin-dev/addmarketingpage

Marketing page creator overview



The administrator has two options when working with marketing page templates:

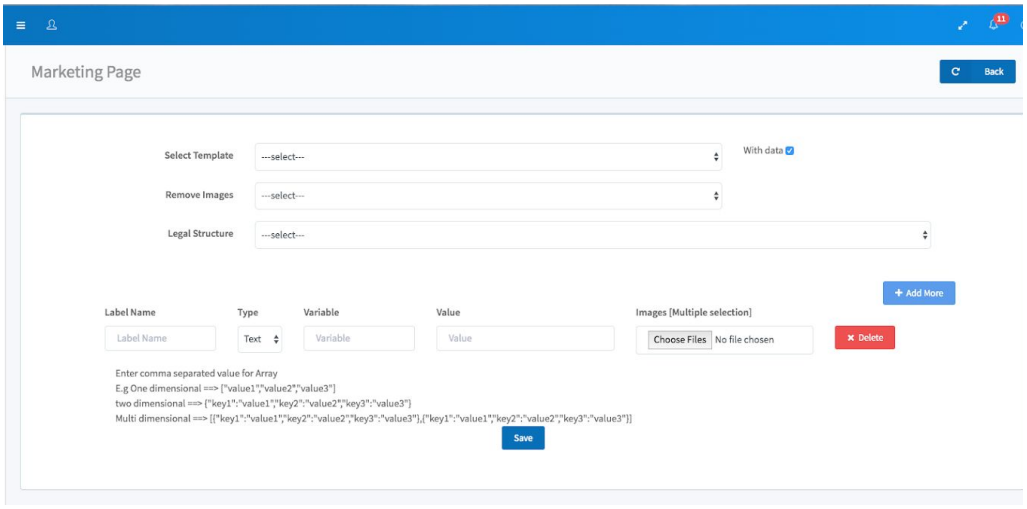
1. Start from scratch.

2. Start a marketing page by copying the structure and optionally all data from an existing template.

Create a marketing template from scratch

Press the “Add more” button.

A new template appears:



A template is simply a design structure with placeholders for data. The marketing page creator lets the administrator specify each placeholder according to:

1. Label name
   Label name is short descriptor that describes this template field.
2. Field
   Is a selector that allows the administrator to define the type of data that will be loaded into this field.
3. Variable
   The variable field allows the administrator to create a variable key that will match this field in the database to a specific field in the html/app template.

4. Value [text/images]
   Text:
   The value field when populated is the textual content that will replace a variable in a template.

   Images:

Allows the administrator to select and upload one or more images.

Create a marketing template from a template

The marketing page offers three drop down menus:

1. **Select template**
   This dropdown allows the administrator to select a previously created template and the “with data” toggle to the right of the drop down allows the administrator to return the template with or without data.
   “With data” will return the all the data for the selected template, except for images.
   “Without data” will return only the fields and tags, no data.

   NOTE: To edit data for a an existing marketing page (except images), the administrator will select the marketing page template “With data” - make required changes and save.

2. **Remove images**
   This drop down allows the administrator to load a new set of images to a marketing page. If images should not be changed, select “Remove images”: NO. If images should be removed and only data returned select “Remove images”: YES.

3. **Legal structure**
   This option allows the administrator to apply the currently selected template data (with data and images - if selected) to be applied to the legal structure selected.

Adding fields to a new or existing template

New fields can be added to a selected marketing page template by clicking the “Add more” button.

Marketing page data structures

In certain cases it may be required to provide data in an array format - for example, financial projections for a graph where labels are required.

Arrays are specified as follows:

One dimensional array: ["value1","value2","value3"]
Two dimensional: {"key1":"value1","key2":"value2","key3":"value3"}
Multi dimensional: [{"key1":"value1","key2":"value2","key3":"value3"}
{"key1":"value1","key2":"value2","key3":"value3"},{"key1":"value1","key2":"value2","key3":"value3"
}]

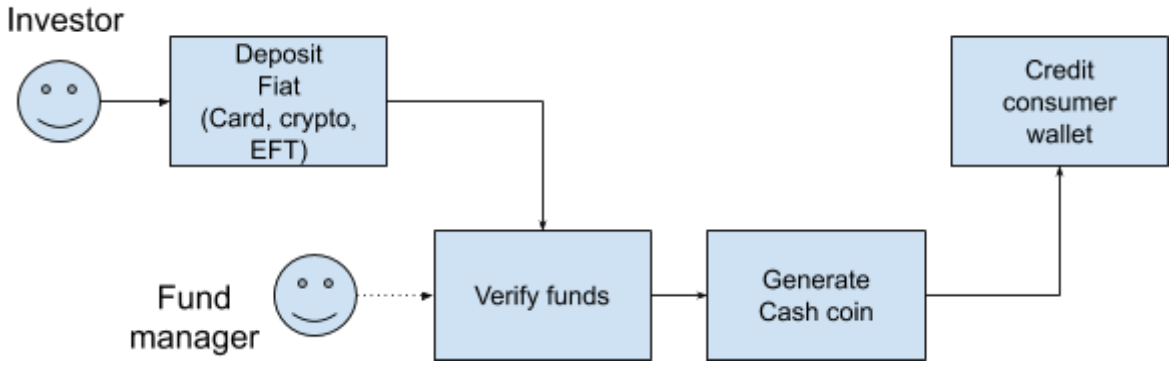
Managing investment

Investment in the Fraxeum system is when a consumer buys shares in an SPV. The shares are represented by the asset token associated with that SPV.

Investor: Depositing funds

The system makes provision for three types of deposit:

- EFT
- Credit Card
- Crypto (Ten major currencies)



EFT deposits

Fiat is transferred from a user's bank account to the fund manager's nominee bank account. The nominee bank account details are provided by the `XXXX` . A fund manager verifies the deposit and loads the deposit into the system via the `depositscommit` API. Plans are in place to automate this process through integration with a bank's API.

Credit card deposits

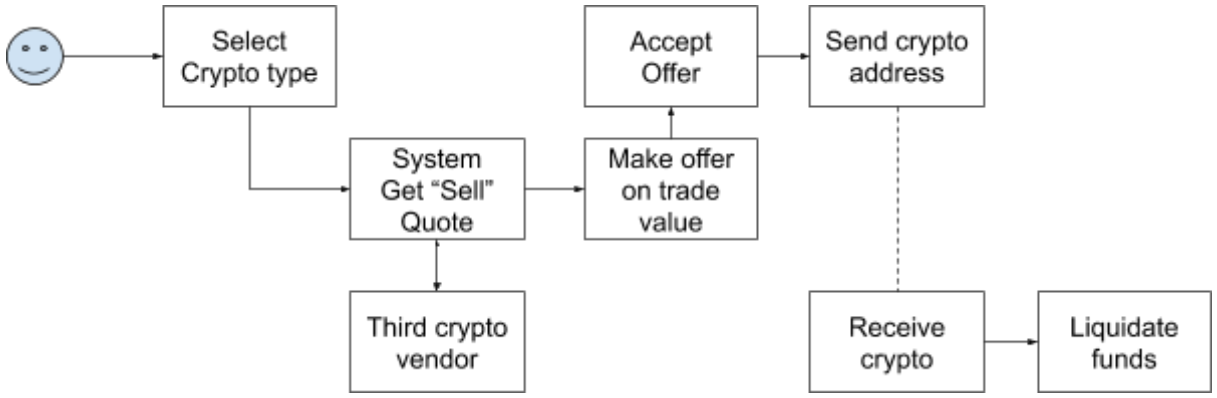
Credit cards are supported via integration with a third party payment gateway like Paygate.

Cryptocurrency deposits

The system em allows a user to deposit cryptocurrencies by offering an instant settlement quote on the cryptocurrency tendered.

A third party system generates a secure deposit address where the tendered crypto type is received. The crypto is immediately converted to BTC and transferred to the local company holding account where it is converted into the local currency.

The user's account is automatically credited with the precise amount of fiat received.



Deposit API call flow

1. Setting up a deposit
The user lets the system know that he/she wants to make a deposit. The system generates a reference number to match the deposit to the user.

API: `userdeposit`
Request type: GET
Type modifier values: EFT, CreditCard, Crypto

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE
{
"success": true | false,


```
    "msg": "",
   “RefNo”: ”REFERENCE”
}
```

2. Committing the deposit in the system
NOTE: The funds verification step is currently a manual process that can and should be automated through integration with the bank that runs the nominee account.

Committing is a two step process: Load deposit and Approve deposits

Loading deposits:
A funds manager administrator loads deposits from the bank statement into the system using the reference number and the amount deposited. The system matches the reference number to the user and assigns the deposit value to be approved.

API: `depositscommit`
Request type: GET

REQUEST
See Postman for detailed field list and descriptions. Uses RefNo.

RESPONSE
```
{
   "success": true | false,
   "msg": ""
}
```

Approving deposits:
The system requires a second person to approve deposits before they are allocated to the user's wallet. Once allocated, the system generates an internal cash coin - a crypto asset unique to this specific bank account (tracking the balance of the bank account 1:1). The crypto asset is transferred to the user's wallet as "cash".

API: `depositslist`
Returns a list of deposits that need to be approved.

Request type: GET

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE
```
{
   "data": [],
   "success": true,
   "msg": []
}
```

API: `depositsapprove`

Request type: GET

REQUEST

See Postman for detailed field list and descriptions. Use deposit ID.

RESPONSE

```
{
   "success": true | false,
   "msg": ""
}
```

API: `depositsreject`

This API simply rejects a deposit that cannot be verified.

Request type: GET

REQUEST

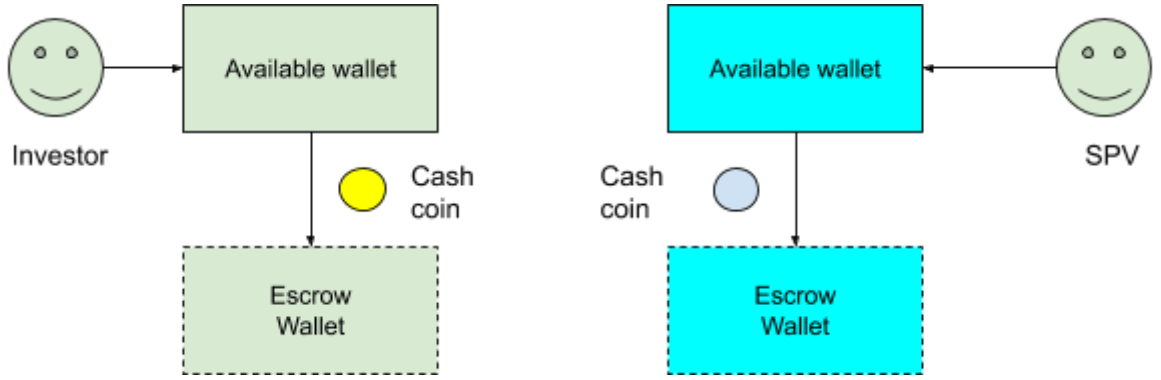
See Postman for detailed field list and descriptions. Use deposit ID.

RESPONSE

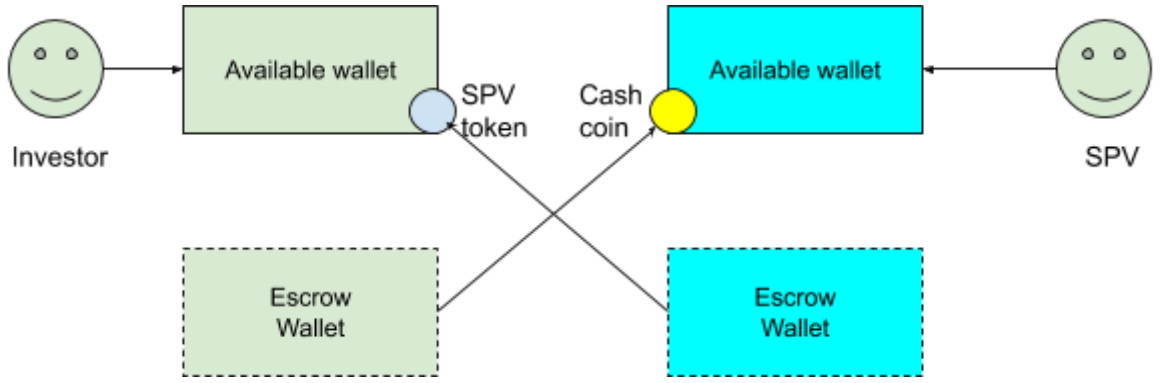
```
{
   "success": true | false,
   "msg": ""
}
```

Investor: Buy shares

To buy shares the investor selects the SPV and number of shares required. The system checks if the investor has enough funds and it checks if there are enough shares available to complete the transaction. Once the transaction has been validated cash coin tokens are moved from the investor's available wallet to the investor's system controlled escrow wallet. At the same time spv tokens are moved from the SPV's available wallet to the SPV's system controlled escrow wallet.



Tokens remain in escrow until such time as the funding requirements for the SPV has been met. If the SPV sales requirements are not met, the Client closes the SPV at which time the tokens roll back to the available wallet of each party. If the SPV sales requirement is met, the token move across to the respective parties' available wallets.



Investor: List Assets

All assets and asset transactions are stored on the Fraxeum blockchain. Certain transactions like `depositsapprove` and `userlistassets` write and read from the blockchain.

API: `userlistassets`
Request type: GET

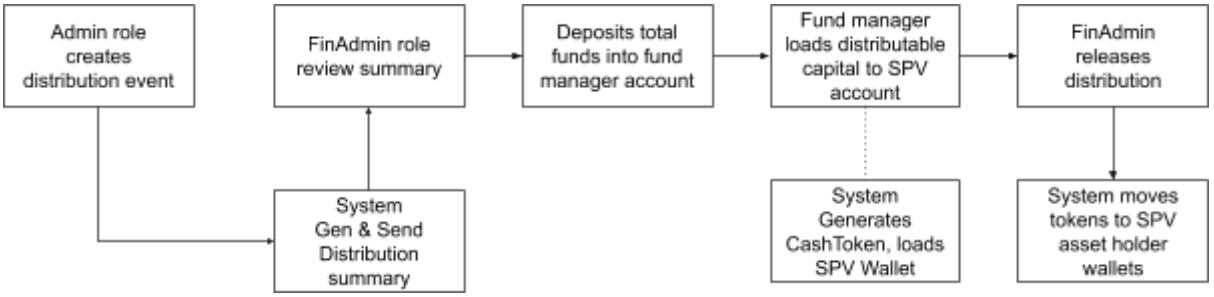
REQUEST
See Postman for detailed field list and descriptions. Requires user for whom asset balance is required.

RESPONSE
XXXXXXXX

Managing profit distribution

A revenue generating legal structure (SPV) has the potential of generating profit and dividends. Dividends are distributed to shareholders.

Funds distribution process



Starting a distribution

Once the SPV board of directors have declared a dividend the cash value of the dividend needs to be distributed to the shareholders. Typically, the CFO or financial director (financial administrator role (FinAdmin)) will create a new funds distribution event on the Fraxeum system.

Distribution review

The system creates a distribution report that details the payment breakdown to each shareholder. Once reviewed, the FinAdmin *approves* or *declines* the distribution. A *decline event* cancels the distribution completely.

Once approved, the system checks if the SPV wallet contains at least the minimum amount required to complete the distribution.

Loading funds

Funds needs to be deposited via Cryptocurrency or fiat.

Cryptocurrency deposit:

The depositor receives a spot quote for the selected cryptocurrency and two additional quotes for upper and lower limit variance. The crypto is converted to a dollar based stable coin as soon as it is received at the current dollar price[1] of the cryptocurrency at the time of receipt. As soon as the last transaction has been confirmed the Fraxeum system generates local Cash Coin equivalent to the reigning local currency to US Dollar exchange rate.

[1] Current price is obtained from an API request to the largest online cryptocurrency exchange by volume at the time.

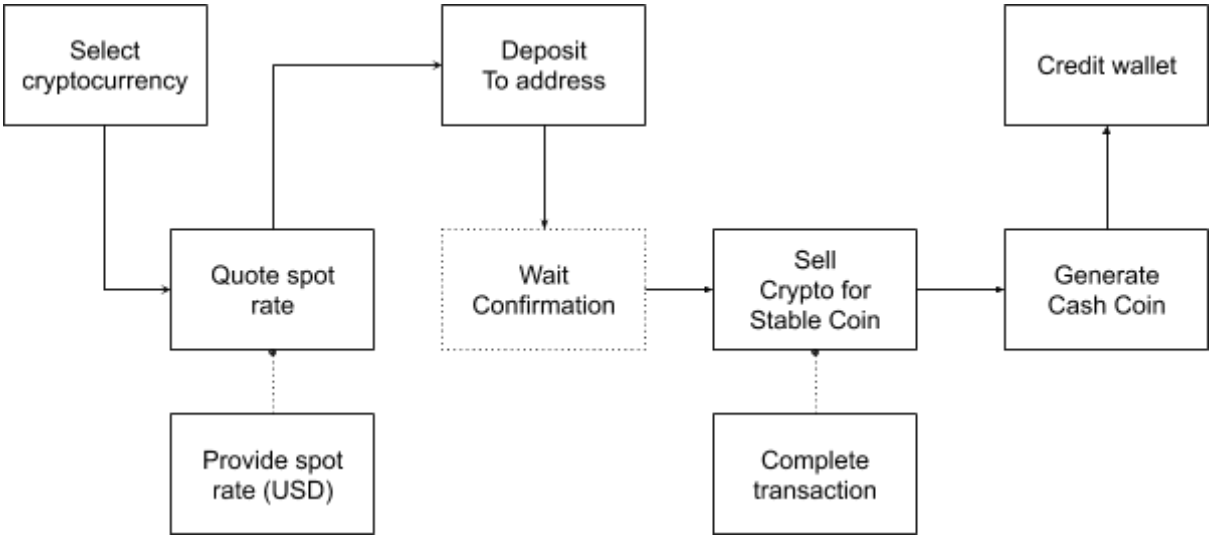


Diagram: Loading wallet using cryptocurrency

In the case of fiat, an accredited Fraxeum fund manager needs to load funds into the Fraxeum system. As soon as the transaction is received, the system generates local Cash Coin.

Once funds have been loaded the distribution is runnable.

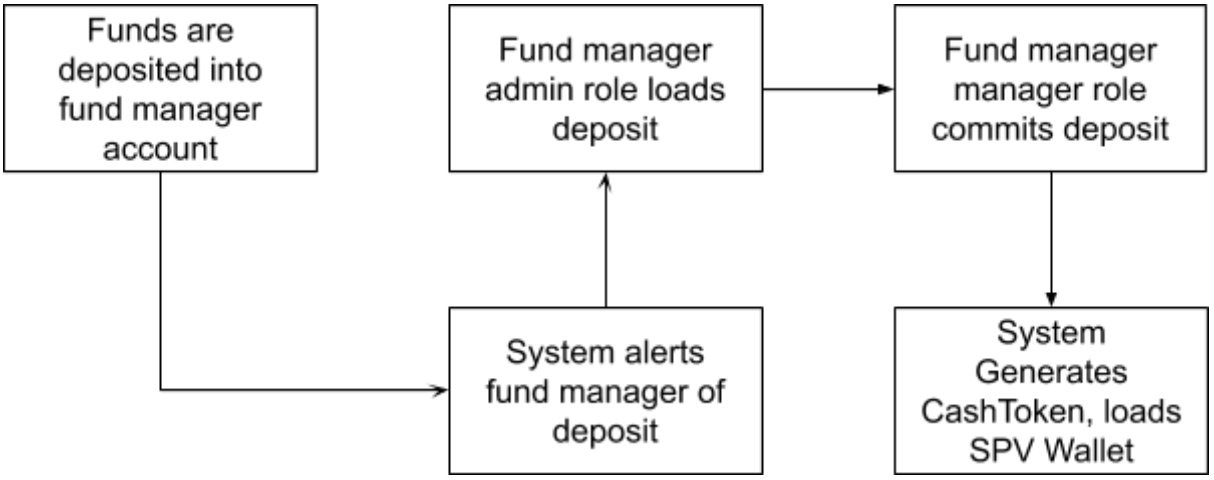
Dissolving an SPV
When an SPV reaches end of life it gets dissolved. Dissolving an SPV is a two step process:

1. `distributionsave`: Create a new distribution (can be zero). Set variable *`DissolveAfterDistribution`*=1. The system returns a legalId and a distributionId.
2. `dissolveconfirm`: Send legalId and distributionId to confirm dissolution.

When an SPV is marked for dissolution the relevant crypto token will be transferred from the consumer's wallet to a "burn address" with an information stating the details of the transaction.

Loading Distributable Funds

This process starts with the financial administrator loading a new deposit request on the system - this process is the same as the `userdeposit` API (but the API is different, see NOTE below).
Call Flow: `clientdeposit`, `depositscommit`, `depositsapprove`)



NOTE: The `clientdeposit` API and `userdeposit` API are NOT interchangeable. Use `clientdeposit` for loading SPV fund distributions and `userdeposit` when loading an investor's deposits.

Deposit distributable funds

API: `clientdeposit`
Request type: POST

REQUEST

See postman for detailed field descriptions.

RESPONSE:

Create a new distribution event

API: `distributionsave`
Request type: POST

REQUEST

See postman for detailed field descriptions.

RESPONSE:

Review distribution summary

API: `distributionlist`
Request type: POST

REQUEST

See postman for detailed field descriptions.

RESPONSE:

Approve and distribute tokens

API: `distributionapprove`
Request type: POST

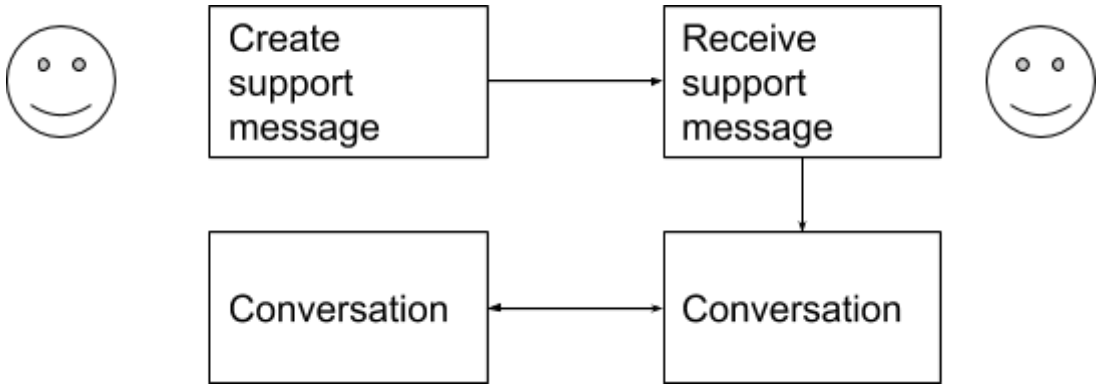
REQUEST

See postman for detailed field descriptions.

RESPONSE:

Managing Investor Support

The system allows investors to interact with Clients (first line support) and system administrators (third line support) via a simple text message interface.



Create a new support message

API: `addsupport`
Request type: POST

REQUEST

See postman for detailed field descriptions.


```
{
   "success": true,
   "data": {
      "Header": "Cannot log in",
      "Status": "N",
      "AssignMem": "5",
      "Rate": "0",
      "SenderId": "26",
      "Text": "[{\"msg\":\"Hi there, I haven't been able to log in since last week Thursday. Please check my account status.\",\"datetime\":\"2019-04-29 14:48:25\",\"SenderId\":\"26\",\"email\":null,\"state\":0,\"canReply\":1,\"MessageId\":1}]",
      "id": "2"
   }
}
```


RESPONSE:


```
{
```


```
    "success": true,
    "data": {
        "Header": "Cannot log in",
        "Status": "N",
        "AssignMem": "5",
        "Rate": "0",
        "SenderId": "26",
        "Text": "[{\"msg\":\"Hi there, I haven't been able to log in since last week Thursday.
Please check my account status.\",\"datetime\":\"2019-04-29
14:48:25\",\"SenderId\":\"26\",\"email\":null,\"state\":0,\"canReply\":1,\"MessageId\":1}]",
        "id": "2"
    }
}
```

Get support topics

See list values API => field=supporttopic

View support messages

API: `listsupport`
Request type: GET

REQUEST

See postman for detailed field descriptions.

{{url}}?c=listsupport&token={{token}}&Open=True&Status=N&user=FraxTest&nonce={{nonce}}&sign={{sign}}

RESPONSE:

{ [...
{
"supportId": "2",
"Header": "Cannot log in",
"CompanyId": null, //id of the company (whitelabel) that implemented the support API
"TopicId": null, //the id of an item from listvalues API with field=supportopic
"SenderId": "26",//user ID
"MemberId": null,
"AssignMem": "5",//ClientId of the client who is responsible to provide first line support
"AssignCom": null,//Agent Id when support request is assigned to a support agent
"Status": "N",//Status: [N]ew, [P]ending, [C]losed
"Email": null,//Set if user cannot log in and needs support (thus no token available)
"RecStatus": "A",
"TimeStamp": "2019-04-29 14:48:25",
"LastEditBy": "26",
"ModifyDate": "2019-04-29 14:48:25",
"Settings": null,
"rate": "0",
"Extra": null,
"msg": [
{
"msg": "Hi there, I haven't been able to log in since last week Thursday. Please check my account status.",
"datetime": "2019-04-29 14:48:25",
"SenderId": "26",
"email": null,


```
                "state": 0,
                "canReply": 1, //Support messages can be one way broadcasts to which no
resply is possible
                "MessageId": 1
            }
        ]
    }
… ] }
```

View support messages

API: `listsupport`
Request type: GET

REQUEST

See postman for detailed field descriptions.

{{url}}?c=listsupport&token={{token}}&Open=True&Status=N&user=FraxTest&nonce={{nonce}}&sign={{sign}}

RESPONSE:


```
{ [...
{
        "supportId": "2",
        "Header": "Cannot log in",
        "CompanyId": null,
        "TopicId": null,
        "SenderId": "26",
        "MemberId": null,
        "AssignMem": "5",
        "AssignCom": null,
        "Status": "N",
        "Email": null,
        "RecStatus": "A",
        "TimeStamp": "2019-04-29 14:48:25",
        "LastEditBy": "26",
        "ModifyDate": "2019-04-29 14:48:25",
        "Settings": null,
        "rate": "0",
        "Extra": null,
        "msg": [
          {
            "msg": "Hi there, I haven't been able to log in since last week Thursday. Please check my account status.",
            "datetime": "2019-04-29 14:48:25",
            "SenderId": "26",
            "email": null,
            "state": 0,
            "canReply": 1,
            "MessageId": 1
          }
        ]
```

}
…] }

View support messages

API: `listsupport`
Request type: GET

REQUEST

See postman for detailed field descriptions.

{{url}}?c=listsupport&token={{token}}&Open=True&Status=N&user=FraxTest&nonce={{nonce}}&sign={{sign}}

RESPONSE:

{ [...
{
"supportId": "2",
"Header": "Cannot log in",
"CompanyId": null,
"TopicId": null,
"SenderId": "26",
"MemberId": null,
"AssignMem": "5",
"AssignCom": null,
"Status": "N",
"Email": null,
"RecStatus": "A",
"TimeStamp": "2019-04-29 14:48:25",
"LastEditBy": "26",
"ModifyDate": "2019-04-29 14:48:25",
"Settings": null,
"rate": "0",
"Extra": null,
"msg": [
{
"msg": "Hi there, I haven't been able to log in since last week Thursday. Please check my account status.",
"datetime": "2019-04-29 14:48:25",
"SenderId": "26",
"email": null,

```
            "state": 0,
            "canReply": 1,
            "MessageId": 1
        }
    ]
}
… ] }
```

OTC Trading: Internal secondary market place

The Fraxeum platform offers an internal market place where shares can be offered for sale and purchased.

The internal market place allows users to buy and sell directly from other users (over the counter (OTC)).

Interested buyers make offers for the sale shares which the seller can accept or reject.

Create sell order

The action of selling shares requires the user to create a bulletin with the following information:

1. Selected asset to sell
2. Amount of asset to sell
3. Desired selling price
4. Min selling price (used to auto reject offers below, but should not be advertised)

API: `createSellOffer`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE
Xxxxxxxx

Making an offer

Much like selling shares, making an offer requires the buyer to specify a number of details:

1. Selected asset to buy
2. Amount of asset to buy
3. Desired purchase price

API: `makeBid`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE
xxxxxxxx

Respond to a bid

A seller can only accept or reject a bid. The seller cannot engage in a negotiation.

API: `respondBid`
Request type: POST

REQUEST
See Postman for detailed field list and descriptions.

RESPONSE
xxxxxxxx

Managing the secondary marketing place

The Fraxeum API offers a number of functions that manages the OTC market place.

- listOwnSellOffers
- cancelSellOffer
- listAllSellOffers
- listOwnBids
- cancelBid
- getClosedTransactions
- tradeHistory

Fraxeum blockchain API

Fraxeum implements a permissioned blockchain forked from the Multichain core source code.

Like most popular blockchains, mining the Fraxeum blockchain offers miner rewards paid for every block that a miner's server mines. Unlike other blockchains, the cost of mining the Fraxeum blockchain is as low as US$10.00 per month.

Mining requirements

1. Fraxeum App - Download from Android and iOS stores.

2. Fraxeum user registration.

3. Civic user registration.

4. Ubuntu 18.04

5. Server with unique IP address for each node.

Blockchain admin server API

Each blockchain node administrator is able to query the blockchain from the node's command line interface. As a private blockchain, nodes have certain permissions. Therefore only a subset of API would be accessible by node administrators.

Node parameters can be found here:

https://www.multichain.com/version-1/json-rpc-api/