|  |  |
| --- | --- |
| 产品名称Product Name | 密级Confidentiality Level |
| BN | 内部 |
| 产品版本Product Version Release | Total 15 Pages共 15 页 |
| V1.0 |

**身份认证内部接口**

|  |  |
| --- | --- |
| Prepared By  拟制人 | Date  日期 |
| Reviewed By  评审人 | Date  日期 |
| Approved By  批准人 | Date  日期 |
| Authorized By  签发人 | Date  日期 |

|  |
| --- |
| Earthledger Technologies Co., Ltd. |
| 易链科技(深圳)有限公司 |
| All rights reserved  版权所有 侵权必究 |
| System Interface V1.0/for internal use only |
| System Interface V1.0/仅供内部使用 |

目录

[1 目的 1](#_Toc507246028)

[2 范围 2](#_Toc507246029)

[3 术语、缩略语 3](#_Toc507246030)

[4 身份认证接口 5](#_Toc507246031)

[4.1 用户注册 5](#_Toc507246032)

[4.1.1 注册请求 5](#_Toc507246033)

[4.1.2 注册响应 5](#_Toc507246034)

[4.2 Auth Token 获取 6](#_Toc507246035)

[4.2.1 认证请求 6](#_Toc507246036)

[4.2.2 挑战challenge 6](#_Toc507246037)

[4.2.3 响应response 6](#_Toc507246038)

[4.2.4 返回Token 6](#_Toc507246039)

[4.3 资源访问授权 8](#_Toc507246040)

[4.3.1 授权验证请求 8](#_Toc507246041)

[4.3.2 授权验证响应 8](#_Toc507246042)

[5 结果码 8](#_Toc507246043)

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 发布日期 | 拟制人 | 修订描述 |
| v0.1 | 20170425 | 易链科技(深圳)有限公司 | 首稿 |
| v0.2 | 20170524 | 易链科技(深圳)有限公司 | 修改了“账单信息查询”的BlockLabel查询条件 |
| v0.3 | 20170602 | 易链科技(深圳)有限公司 | 新增若干条接口细节描述 |
| v0.4 | 20170808 | 易链科技(深圳)有限公司 | 未确认，不清晰公共接口信息细化 |
| v0.5 | 20171011 | 易链科技(深圳)有限公司 | 接口重新审查，修正，分类 |
| v0.6 | 20171026 | 易链科技(深圳)有限公司 | 新增接口，重新描述相关概念 |
|  |  |  |  |

# 1 目的

为了更好的提供接口功能到「客户侧平台」，在分层架构的指引下，明确各层职责，避免耦合设计，从而达到有效地层间交互。

本文档主要面向功能接口，监控接口的规范化描述。

# 2 范围

交互设备为：BAAR，FBAAR

交互协议为：SOAP Web Service，~~HTTPS~~，HTTP

# 3 术语、缩略语

|  |  |  |
| --- | --- | --- |
| Block Router | BR | 易链设备简称 |

|  |  |  |
| --- | --- | --- |
| Block Aggregation Access Router | BAAR | 区块链**数据聚合**路由设备 |
| Block Storage Distribution Router | BSDR | 区块链**存储分发**路由设备 |
| Block Consensus Router | BCR | 区块链**共识**路由设备 |

|  |  |  |
| --- | --- | --- |
| Block Operating System | BOS | 区块链操作系统 |
| Block Networking | BN | 区块链网络 |
| Block Devices | BDs | 区块链设备组 |
| Block Protocols | BPs | 区块链协议 |
| Block Platform | BP | 区块链平台 |

|  |  |
| --- | --- |
| BlockLabel | BOS区块链系统，数据(资源)的唯一标识，需要客户侧平台构建，用于标识「数据登记确权」唯一的数据 |
| Market | 表示市场参与者，特指某台BSDR或者某一组BSDR的统一名称 |
| ProviderMarket | ProviderMarket表示市场参与者的提供者职责 |
| ConsumerMarket | ConsumerMarket表示市场参与者的消费者职责 |
| Price | 用于标识「数据登记确权」唯一的数据的定价，定价不上链 |
| Content | 「数据登记确权」的数据，面向易链BOS，不论繁简的客户数据，都统一作为Content，  Content类型可以是XML、**JSON**、PB、YML等 |
| SrcMarket | SrcMarket表示市场参与者的转出方职责 |
| DstMarket | DstMarket表示市场参与者的接收方职责 |
| DeviceGroupId | 一整套BN设备组的标识 |

|  |  |
| --- | --- |
| ★ | 被该符号标注的接口表示支持批量查询，未明确标注的接口表示不支持批量查询 |

|  |  |
| --- | --- |
| User | 用户，或者称端用户，例如张三 |
| Customer | 客户，例如法信 |
| Market | 市场参与者，例如某公证处 |

|  |  |
| --- | --- |
| s3host | 服务端主机IP和端口 |
| s3access | 服务端分配和管理的用户账户接入码 |
| s3secret | 服务端分配和管理的用户账户接入码对应密码 |
| s3bucket | 服务端分配给某用户所属的桶空间 |
| s3srcfile | S3客户端原始路径和文件名 |
| s3dstfile | 服务端保存的文件名 |

关于BlockLabel的构建，需要业务场景(business scenario)、实际用况(use case)进行驱动。

客户层、应用层可按照如下3类进行BlockLabel构造

# 4 身份认证接口

## 4.1 用户注册

### 4.1.1 注册请求

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 注册请求  asym-auth/register/ | curl -X POST -d "user=yilian333&publickey=LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlHZk1BMEdDU3FHU0liM0RRRUJBUVVBQTRHTkFEQ0JpUUtCZ1FDSTdVZE9WVkx5Vk5XRnhlc1dmRWcwNWxacQplYitCVzRCR1NoV0tZOUQ0Sk05QVFON3U1OVhPdW1OTnFKdHduT04zOVJ1MHE5SlFGbC9MZjZocE4yU0laTEtmCm5ISU4rc3l3V1BaNkNkWEswNmxyMzJDNW5UdEFFL3drNjlsamNNMVgyNHpRR21QeEhBeC9DaE5RcUVGcEExVzEKNmFGc0FrZTJSNmpCSlN3bVNRSURBUUFCCi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==" http://localhost:8000/asym-auth/register/ |

### 4.1.2 注册响应

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 注册成功 | {"result":“success”,”message”:{“name”:"Register Success"},”error”:”null”} |
| 注册失败 | {“result”:”error”,”message”:”null”,"error":{"code":-101,"Info":"User already registered "}}  -1 没有使用POST方法  -101用户已经注册  -102公钥格式不正确 |

## 4.2 Auth Token 获取

### 4.2.1 认证请求

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 认证请求  asym-auth/authRequest/ | curl -X POST -d "user=yilian333" http://localhost:8000/asym-auth/authRequest/ |

### 4.2.2 挑战challenge

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 挑战challenge | {"result":“success”,”message”:{”name”:"challenge",”random”:24156},”error”:”null”} |
| 失败 | {“result”:”error”,”message”:”null”,"error":{"code":-201,"Info":"User already registered "}}  -1 没有使用POST方法  -201用户名不存在 |

### 4.2.3 响应response

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 响应response  asym-auth/response/ | curl -X POST -d "user=yilian333&signature=EaefQrB/uEwYokRjtJPvHJpfdJe6IjijKub1mAmY8uUvZQQiIRDLtvvno2NeveM3m7ge+97ZpOw3Ah1BfeYih3YWmIB+SppHIrLC6NM/Tpz1oCqydcVIUTHZO4fpJHunriGrBIncyf2t/j3Rr+QyVW4yjhsjAjXqAVzHdHk5pkI=" http://localhost:8000/asym-auth/response/ |

### 4.2.4 返回Token

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 成功 | {“result”:”success”,"message":{“name”:"Return","JWT":jwt\_token},”error”:”null”} |
| 失败 | {“result”:”error”,”message”:”null”,"error":{"code":-301,"Info":" Verify the signature fail "}}  -1 没有使用POST方法  -201用户名不存在  -301验证签名失败  -302 Challenge值不存在 |

## 4.3 资源访问授权

### 4.3.1 授权验证请求

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 授权验证请求  asym-auth/verify/ | curl -X POST -d "user=yilian333&Authorization=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJFYXJ0aGxlZGdlciBBVVRIIiwidXNlcm5hbWUiOiJ5aWxpYW4zMzMiLCJleHAiOjE1MTgzMzc4MDF9.LEpif7MQvp1DkO9ac8\_h7wfRGgSziZUnzQyxhmWqkvY" http://localhost:8000/asym-auth/verify/ |

### 4.3.2 授权验证响应

|  |  |
| --- | --- |
| 功能 | 格式举例 |
| 成功 | {“result”:”success”,"message":{“name”:"Auth Success" },”error”:”null”} |
| 失败 | {“result”:”error”,”message”:”null”,"error":{"code":-401,"info":" JWT verify fail "}}  -1 没有使用POST方法  -201用户名不存在  -401 JWT验证失败  -402 JWT过期  -403用户没有权限 |

# 5 结果码

|  |  |  |
| --- | --- | --- |
| Code | 说明 | 备注 |
| -1 | 没有使用POST方法 | Not use POST method |
| -101 | 用户已经注册 | User already registered |
| -102 | 公钥格式不正确 | Public key format error |
| -201 | 用户名不存在 | User not exist |
| -301 | 验证签名失败 | Verify the signature fail |
| -302 | Challenge值不存在 | Challenge number not exist |
| -401 | JWT验证失败 | JWT verify fail |
| -402 | JWT过期 | JWT expired |
| -403 | 用户没有权限 | User has no permissions |

# 6 数据格式

## 6.1 数据表

### 6.1.1 公钥表

|  |  |
| --- | --- |
| 键1（unique） | 键2 |
| User | Publickey |

### 6.1.2 challenge表

|  |  |
| --- | --- |
| 键1（unique） | 键2 |
| User | Challenge |

## 6.2 公钥格式

用openssl命令生成的公钥，有pkcs#1和pkcs#8两种密钥信息格式，存储上可以有der和pem两种格式，pem格式如下所示。直观上看pkcs#1在头部和尾部比pkcs#8格式多一个RSA字段。

### 6.2.1 pkcs#1格式

-----BEGIN RSA PUBLIC KEY-----

...

-----END RSA PUBLIC KEY-----

### 6.2.2 pkcs#8格式

-----BEGIN PUBLIC KEY-----

...

-----END PUBLIC KEY-----

### 6.2.3 pkcs#1和pkcs#8转换

可以通过openssl命令完成两种信息格式的转换：

#生成pkcs1格式RSA私钥

openssl genrsa -out rsa\_private\_key.pem 1024

#由pkcs1私钥生成RSA pkcs8公钥

openssl rsa -in rsa\_private\_key.pem -out rsa\_public\_key.pem –pubout

#将pkcs1格式RSA私钥转换成 pkcs8 格式RSA私钥

openssl pkcs8 -topk8 -inform PEM -in rsa\_private\_key.pem -outform pem -nocrypt -out pkcs8.pem

Public key: PKCS#8 -> PKCS#1

openssl rsa -pubin -in public.pem -RSAPublicKey\_out -out public\_pkcs1.pem

Public key: PKCS#1 -> PKCS#8

openssl rsa -RSAPublicKey\_in -in public\_pkcs1.pem -out public\_pkcs8.pem

## 6.3 签名验证

Publickey可以是pkcs1格式注册或者pkcs8格式注册，如果客户用pkcs1格式的公钥注册，就要用pkcs1相应的签名函数；如果客户用pkcs8格式的公钥注册，就要用pkcs8相应的签名函数。

## 6.4 传输格式

公钥、签名以及JWT都是转换成base64格式传输的。