Backend API Specification

( v1.0.0 )

**Table Of Contents**

[1. Unauthorize API 3](#_Toc176531114)

[1) GetVendorGames 3](#_Toc176531115)

[2) GetWagerList 4](#_Toc176531116)

[3) GetTransactionList 5](#_Toc176531117)

[2. Authorize API 6](#_Toc176531118)

[1) CreateAffiliater 6](#_Toc176531119)

[2) GetAffiliaterInfo 7](#_Toc176531120)

[3) ClaimRefferalBonus 8](#_Toc176531121)

[4) GetUnreadNoticeList 9](#_Toc176531122)

[5) ReadNotice 10](#_Toc176531123)

[6) CheckBalance 11](#_Toc176531124)

[7) GetStakingInfo 11](#_Toc176531125)

[8) Stake / Unstake 13](#_Toc176531126)

[9) ClaimStakingBonus 14](#_Toc176531127)

[10) GetBalanceModalInfo 15](#_Toc176531128)

[11) GetVirtualBalanceModalInfo 16](#_Toc176531129)

[3. GET Request(Initial login) 17](#_Toc176531130)

[1) Response code 17](#_Toc176531131)

1. Unauthorize API

* For authorize API integration
* const response = await fetch(`${backendUrl}/backend/unauthorizeapi`, {
* method: "POST",
* headers: {
* "Content-Type": "application/json",
* },
* body: JSON.stringify({
* method: "GetVendorGames",
* affiliaterCode: affiliateCode,
* }),
* });

1. GetVendorGames

You must call this API to get vendor game list.

vendorCode must be a code obtained from the GetVendors API.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| field | category | explanation | essential |
| method | String | GetVendorGames | Y |
| agentCode | String | agent code | Y |
| gameType | Integer | Game type 1-slot, 9- minigame | Y |

* Request example:

{

“method” : “ GetVendorGames”,

“agentCode ” : “ testAgent ”,

“gameType” : 9

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| vendorGames | List<VendorGame> | Game list |

* Response example:
* success

{

"status" : 0,

" msg" : "SUCCESS",

"vendorGames": [

 {

             "vendorName": "{\"en\":\"Spribe Gaming\"}",

            "vendorCode": "mini-spribe",

            "gameType": 9,

            "gameCode": "Aviator",

            "imageUrl": "{\"en\":\"https://app.roogsino.io/resources/image/games/Av-new@2x.png\"}",

            "gameName": "{\"en\":\"Aviator\",\"ko\":\"Aviator\"}"

        }

]

}

* error

{

“status”: 12,

" msg ” : "INVALID\_VENDOR"

}

1. GetWagerList

You must call this API to get wager list.

If userCode is empty, it returns transactions of all users.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | category | explanation | essential |
| method | String | GetTransactionList | Y |
| agentCode | String | agent code | Y |
| userCode | String | User code | N |
| Offset | Integer |  | Y |
| Length | Integer |  | Y |

* Request example:

{

“method” : “ GetTransactionList”,

“agentCode ” : “ testAgent ”,

“userCode ” : “orai11111”,

“offset”:0,

“length”:10

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | category | explanation |
| Status | Integer | Response code (see Appendix 4.1) |
| Msg | String | response message |
| Wagers | List | Game list |

* Response example:
* success

{

"status" : 0,

" msg" : "SUCCESS",

"wagers": [

 {

             "UserCode": "orai111",

            "AgentCode": "testAgent",

            "BetAmount": 10,

“PayoutAmount”:20

“CreatedAt”:” 2024-08-29 10:00:12”,

        }

]

}

* error

{

“status”: 12,

" msg ” : "INVALID\_VENDOR"

}

1. GetTransactionList

You must call this API to get transaction list.

If userCode is empty, it returns transactions of all users.

If txnType equals 0, it returns all transaction types.

public enum UserMoneyChangeType : byte

{

Deposit = 1,

Withdraw = 2,

WithdrawAll = 3,

CallApply = 4,

CallCancel = 5,

DepositCoin = 10,

WithdrawCoin = 11,

WithdrawCoinFail = 12,

DepositBonus = 13,

RefferalBonus = 14,

}

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| field | category | explanation | essential |
| method | String | GetTransactionList | Y |
| agentCode | String | agent code | Y |
| userCode | String | User code | N |
| txnType | Integer | Transaction type | Y |
| offset | Integer |  | Y |
| length | Integer |  | Y |

* Request example:

{

“method” : “ GetTransactionList”,

“agentCode ” : “ testAgent ”,

“userCode ” : “orai11111”,

“txnType” : 0,

“offset”:0,

“length”:10

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| transactions | List | Game list |

* Response example:
* success

{

"status" : 0,

" msg" : "SUCCESS",

"transactins": [

 {

             "UserCode": "orai111",

            "AgentCode": "testAgent",

            "Amount": 10,

            "Type": 10,

“CreatedAt”:” 2024-08-29 10:00:12”,

"UserPrevMoney": 120,

        }

]

}

* error

{

“status”: 12,

" msg ” : "INVALID\_VENDOR"

}

1. Authorize API

* For authorize API integration, you must put token to header below.
* const response = await fetch(`${backendUrl}/backend/authorizeapi`, {
* method: "POST",
* headers: {
* "X-Access-Token": accessToken,
* "Content-Type": "application/json",
* },
* body: JSON.stringify({
* method: "CreateAffiliater",
* affiliaterCode: affiliateCode,
* }),
* });

1. CreateAffiliater

You calls this API to create affiliater.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| field | category | explanation | essential |
| method | String | CreateAffiliater | Y |
| affiliaterCode | String | Affiliater code | Y |

* Request example:

{

“method” : “ CreateAffiliater”,

“affiliaterCode” : “xyz”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS"

}

* error

{

“status”: 1,

" msg ” : "Already exist"

}

1. GetAffiliaterInfo

You calls this API for get information for affiliate page.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| field | category | explanation | essential |
| method | String | GetAffiliaterInfo | Y |
| currencyCode | String | Currency code | Y |

* Request example:

{

“method” : “ GetAffiliaterInfo”,

“currencyCode” : “ROOG”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| affiliateCodes | List<String> |  |
| lastSettleTime | DateTime |  |
| currencyCode | String |  |
| totalBetCount | Integer |  |
| totalBetAmount | Decimal |  |
| totalPayoutAmount | Decimal |  |
| totalIncome | Decimal |  |
| totalReferralCount | Integer |  |
| referralInfos | List<ReferralInfo> |  |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

"affiliateCodes" : [“xyz”],

"currencyCode" : “ROOG”,

“lastSettleTime”:” 2024-08-29 10:00:12”,

“totalBetCount”:100,

“totalBetAmoutn”:12000.56,

“totalPayoutAmount”:13000,

“totalIncome”:20.6,

“totalReferralCount”:3,

“refferalInfos”:[

{

“affiliaterCode”:”xyz”,

“userCode”:”serrat”

“betCount”:2,

“betAmount”:60,

“payoutAmount”:100,

“income”:0.5

}

]

}

* error

{

“status”: 5,

" msg ” : "No affiliater"

}

1. ClaimRefferalBonus

You calls this API to claim referral bonus.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| field | category | explanation | essential |
| method | String | ClaimRefferalBonus | Y |
| currencyCode | String | Currency | Y |

* Request example:

{

“method” : “ CreateAffiliaterCode”,

“currencyCode” : “ROOG”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| bonus | Decimal |  |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

“bonus”:5

}

* error

{

“status”: 1,

" msg ” : "Already exist"

}

1. GetUnreadNoticeList

You calls this API to get unread notice list.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| field | category | explanation | essential |
| method | String | GetUnreadNoticeList | Y |

* Request example:

{

“method” : “ GetUnreadNoticeList”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| data | List<Notice> |  |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

“data”:[

{

“Id”:1234,

“Title”:”Deposit bonus”,

“CreatedAt”:” 2024-08-29 10:00:12”

}

]

}

* error

{

“status”: 1,

" msg ” : "Already exist"

}

1. ReadNotice

You calls this API to get notice and read.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | category | explanation | essential |
| method | String | ReadNotice | Y |
| Id | Integer |  | Y |

* Request example:

{

“method” : “ReadNotice”,

“id”:1234

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| data | Notice |  |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

“data”:[

{

“Id”:1234,

“AgentCode”:”serrat”,

“UserCode”:”123123123”,

“Title”:”Deposit bonus”,

“Content”:” You received a referral game play bonus between January 1st and January 5th”

“CreatedAt”:”2024-08-29 10:00:12”

}

]

}

* error

{

“status”: 1,

" msg ” : "Already exist"

}

1. CheckBalance

You calls this API to check balance.

This API is used in Solana chain

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | category | explanation | essential |
| method | String | CheckBalance | Y |
| chain | String |  | Y |
| coinType | String |  | Y |

* Request example:

{

“method” : “CheckBalance”,

“chain”:”Solana”,

“coinType”:”ROOG”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| Msg | String | response message |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

}

* error

{

“status”: 1,

" msg ” : "Fail"

}

1. SetupTrustLineOnXrp

You calls this API to setup trust line on xrp.

This API is used in Solana chain

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | category | explanation | essential |
| method | String | SetupTrustLineOnXrp | Y |
| chain | String |  | Y |
| coinType | String |  | Y |

* Request example:

{

“method” : “SetupTrustLineOnXrp”,

“chain”:”Xrpl”,

“coinType”:”ARTI”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

}

* error

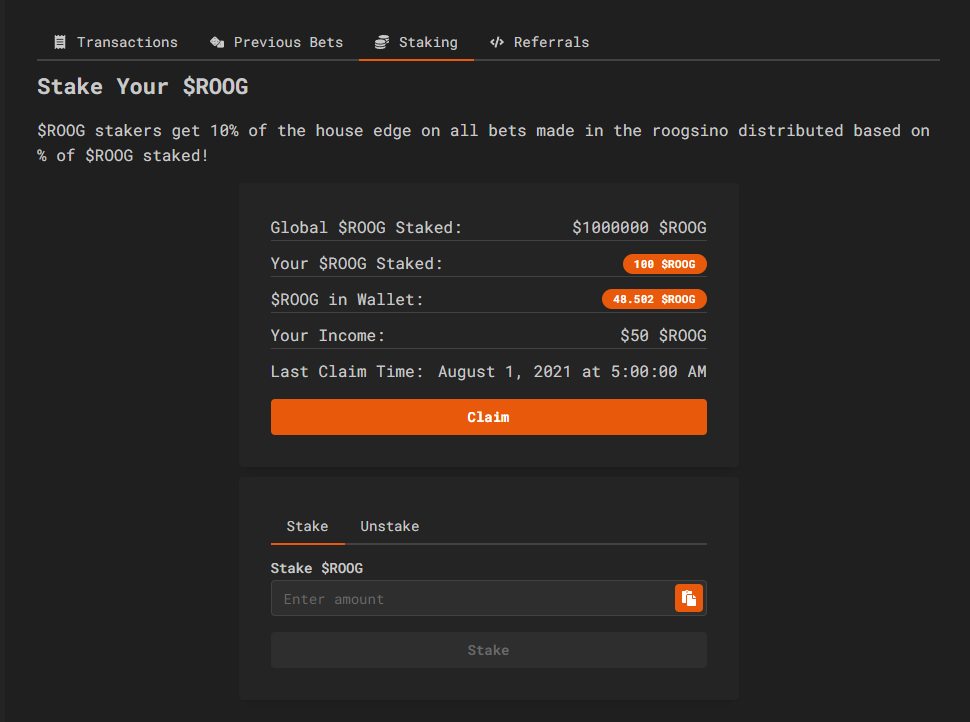
{

“status”: 1,

" msg ” : "Fail"

}

1. GetStakingInfo



This method returns the amount of $token a user has staked, the total amount of $token staked and the amount of earnings the user has.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | Explanation | essential |
| Method | String | GetStakingInfo | Y |
| currencyCode | String | Coin type | Y |

* Request example:

{

“ method” : “ GetStakingInfo”,

“ currencyCode” : “xrpl”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| stakedAmount | Decimal | User's staking money |
| totalStakedAmount | Decimal | Total staking money |
| income | Dictionary | Profit |
| lastSettleTime | DateTime |  |

* Response example:
* success

{

"status" : 0,

" msg " : "SUCCESS",

"stakedAmount" : 10000,

“totalStakedAmount”:1000000,

“income”:{“USDC”:30,”ROOG”:10,”NANA”:50}, //10% of bet amount,

“lastSettleTime”: ”2024-08-29 10:00:12”

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. Stake / Unstake

This method stakes $token for profit share, the profit share should be 10% of house edge on all bets devided by the % of each persons stake.

If amount > 0, stake

If amount < 0, unstake

After API call, increase stake amount, decrease balance.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | Explanation | essential |
| Method | String | Stake | Y |
| currencyCode | String | Coin type | Y |
| Amount | Decimal | amount |  |

* Request example:

{

“method” : “ Stake”,

“currencyCode” : “xrpl”,

“amount”:1000

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | Category | explanation |
| Status | Integer | Response code (see Appendix 4.1) |
| Msg | String | response message |
| Balance | Decimal | User's balance |
| stakedAmount | Decimal | User’s staking money |
| Income | Dictionary |  |
| lastSettleTime | DateTime |  |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

"balance" : 10000,

“stakedAmount”:1000000,

“income”:{“USDC”:30,”NANA”:20,”ROOG”:10}, //10% of bet amount,

“lastSettleTime”: ”2024-08-29 10:00:12”

}

* error

{

“status”: 5,

" msg ” : "Xrpl is not staking currency"

}

1. ClaimStakingBonus

This method returns the amount of $token a user has staked, the total amount of token staked and the amount of earnings the user has.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | Explanation | essential |
| method | String | ClaimStakingBonus | Y |
| currencyCode | String | Coin type | Y |

* Request example:

{

“method” : “ ClaimStakingBonus”,

“currencyCode” : “xrpl”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| Field | Category | explanation |
| Status | Integer | Response code (see Appendix 4.1) |
| Msg | String | response message |
| balance | Decimal | User's balance |
| stakedAmount | Decimal | User’s staking money |
| income | Dictionary |  |
| lastSettleTime | DateTime |  |

* Response example:
* success

{

"status" : 0,

"msg" : "SUCCESS",

"balance" : 10000,

“stakedAmount”:1000000,

“income”:{“USDC”:30,”NANA”:20,”ROOG”:10}, //10% of bet amount,

“lastSettleTime”: ”2024-08-29 10:00:12”

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. GetBalanceModalInfo

When user launch Balance modal dialog, you calls this API to get modal information.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | Explanation | essential |
| method | String | GetBalanceModalInfo | Y |
| coinType | String | Coin type | Y |

* Request example:

{

“ method” : “ GetBalanceModalInfo”,

“ coinType” : “xrp”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| balance | Decimal | User's money |
| depositAddress | String | Deposit address |
| modalMessage | String | Message shown in modal |
| withdrawalMaxLimit | Decimal | Withdrawl max limit |
| depositMinLimit | Decimal | Minimum deposit limit |
| depositable | Bool | When using xrp token, first setup trust line |

* Response example:
* success

{

"status" : 0,

"msg " : "SUCCESS",

"balance" : 10000,

“depositAddress”:”orai111111”,

“modalMessage”:”The first deposit amount must be larger than 10 xrpl”,

“withdrawalMaxLimit”:30, //10% of bet amount,

“depositMinLimit”:10xrpl,

“depositable”:true,

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. GetVirtualBalanceModalInfo

When user launch Balance modal dialog, you calls this API to get modal information.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | explanation | essential |
| Method | String | GetVirtualBalanceModalInfo | Y |
| coinType | String | Coin type | Y |

* Request example:

{

“ method” : “ GetVirtualBalanceModalInfo”,

“ coinType” : “USO”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| virtualBalance | Decimal | User's money |
| balances | Dictionary |  |
| depositConvertRatio | Dictionary | If coinType is virtual |
| depositMinLimit | Decimal |  |
| withdrawConvertRatio | Dictionary |  |
| withdrawMaxLimit | Decimal |  |

* Response example:
* success

{

"status" : 0,

"msg " : "SUCCESS",

"virtualBalance" : 10000,

“balances”:{“xrpl”:500}

“modalMessage”:”The first deposit amount must be larger than 10 xrpl”,

“depositConvertRatio”:{

“sol”:0.5,

“USDC”:1

},

“withdrawConvertRatio”:{ //1% of bet amount

“sol”:1,

“USDC”:100

}

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. DepositVirtualCoin

When user deposit virtual coin from real coin, you calls this API.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | explanation | essential |
| method | String | DepositVirtualCoin | Y |
| chain | String |  | Y |
| coinType | String | Real Coin type | Y |
| virtualCoinType | String | Virtual Coin type | Y |
| amount | Decimal |  | Y |

* Request example:

{

“method” : “ DepositVirtualCoin”,

“coinType” : “USO”,

“virtualCoinType”:”Xrp”,

“chain”:”Xrpl”,

“amount”:100

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| virtualBalance | Decimal | User's virtual balance |
| balance | Decimal | User’s real balance |

* Response example:
* success

{

"status" : 0,

"msg " : "SUCCESS",

"virtualBalance" : 10000,

“balance”:100

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. WithdrawVirtualCoin

When user withdraw virtual coin to real coin, you calls this API.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | explanation | essential |
| method | String | WithdrawVirtualCoin | Y |
| chain | String |  | Y |
| coinType | String | Real Coin type | Y |
| virtualCoinType | String | Virtual Coin type | Y |
| amount | Decimal |  | Y |

* Request example:

{

“method” : “WithdrawVirtualCoin”,

“coinType” : “USO”,

“virtualCoinType”:”Xrp”,

“chain”:”Xrpl”,

“amount”:100

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| virtualBalance | Decimal | User's virtual balance |
| balance | Decimal | User’s real balance |

* Response example:
* success

{

"status" : 0,

"msg " : "SUCCESS",

"virtualBalance" : 10000,

“balance”:100

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. GetEventList

When user is going to get event list, you calls this API.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | explanation | essential |
| method | String | GetEventList | Y |
| offset | Integer |  | Y |
| limit | Integer |  | Y |
| currencyCode | String |  | Y |

* Request example:

{

“method” : “GetEventList”,

“offset” : 0,

“limit”:10,

“currencyCode”:”xrp”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| data | List<Event> | event list |

* Response example:
* success

{

"status" : 0,

"msg " : "SUCCESS",

"data" : [

{

“id”:123,

“title”:”Weekly Big Win Bonus”,

“content”:”…”,

“createdAt”:” 2024-08-29 10:00:12”,

“status”:true, //true-active, false-end “startTime”:” 2024-08-29 10:00:12”,

“endTime”:” 2024-08-29 10:00:12”,

“info”:{

“type”:1,

“bonusList”:[

{“currencyCode”:”xrp”,”amount”:500},

{“currencyCode”:”xrp”,”amount”:250},

{“currencyCode”:”xrp”,”amount”:100}

]

}

}

]

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. GetEvent

When user is going to get specific event, you calls this API.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | explanation | essential |
| method | String | GetEvent | Y |
| eventId | Integer |  | Y |

* Request example:

{

“method” : “GetEvent”,

“eventId” : 123

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |
| data | Object | user list |

* Response example:
* success

{

"status" : 0,

"msg " : "SUCCESS",

"data" : {

“type1”:[

{

“rank”:1,

“userCode”:”saasdas”,

“nickName”:”timon”,

“vendorCode”:”mini-spribe”,

“gameCode”:”Dice”,

“betAmount”:100,  
 “payoutAmount”:10000

}

]

}

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. UpdateNickName

When user withdraw virtual coin to real coin, you calls this API.

* Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Category | explanation | essential |
| method | String | UpdateNickName | Y |
| nickName | String |  | Y |

* Request example:

{

“method” : “UpdateNickName”,

“nickName” : “xxx”

}

* Response parameters

|  |  |  |
| --- | --- | --- |
| field | Category | explanation |
| status | Integer | Response code (see Appendix 4.1) |
| msg | String | response message |

* Response example:
* success

{

"status" : 0,

"msg " : "SUCCESS"

}

* error

{

“status”: 5,

" msg ” : "INVALID\_USER"

}

1. GET Request(Initial login)
2. Response code

|  |  |  |
| --- | --- | --- |
| response code | response message | explanation |
| 0 | SUCCESS | success |
| 1 | INTERNAL\_ERROR | Server internal error |
| 2 | INVALID\_ACTION | request error |
| 3 | INVALID\_AGENT | agent error |
| 4 | BLOCK\_AGENT | blocked agent |
| 5 | INVALID\_USER | user error |
| 6 | BLOCK\_USER | blocked user |
| 7 | DUPLICATE\_USER | duplicate users |
| 8 | INSUFFICIENT\_MONEY | Insufficient money error |
| 12 | INVALID\_VENDOR | vendor error |
| 13 | INVALID\_PARAMETER | Request parameter error |
| 14 | NETWORK\_ERROR | network error |
| 15 | MAINTENANCE | Under maintenance |
| 18 | INVALID\_WAGER | Transaction details ID |
| 20 | INVALID\_TIME | time error |
| 21 | DUPLICATE\_REQUESTKEY | Duplicate prevention request key error |
| 22 | TIMEOUT\_ERROR | Timeout error |