FUNDIST.ORG API v2.23 @2015-11-04

Contents

Change History

2.00	13 Feb 2013	Documentation is split into several documents: • Generic API • System-specific parameters
2.01	26 Mar 2013	Extended language list
2.02	01 May 2013	Extended user authentication with “IsMobile” parameter support Added color coding
2.03	29 Aug 2013	Added clarification of [TID]
2.04	11 Sep 2013	Added optional Timezone parameter for User/Add and User/[Direct]Auth Added previously not documented Nick parameter to User/Add Fixed User/DirectAuth pattern (added missing optional parameters)
2.05	24 Sep 2013	Extended User/Add with number of new parameters.
2.06	25 Sep 2013	Added “Funds Transfer Safety” section
2.07	09 Nov 2013	Dropped deprecated User/Auth request
2.08	13 Nov 2013	Clarified format of Amount parameter. Added previously not documented RegistrationIP parameter for User/Add Updated error codes Misc. formatting
2.09	26 Jan 2014	Minor correction of User/DirectAuth Hash calculation due to issue introduced in Rev 2.07
2.10	01 Apr 2014	Added optional ExtParam paramter to User/DirectAuth
2.11	04 May 2014	Corrected links to include more proper path starting from /System/. Added previously not documented UserIP parameter for User/DirectAuth Added User/KillAuth request. Corrected references to Appendix and product-specific Addendum Added Holistic Picture
2.12	07 Jul 2014	Added User/AuthHTML API call
2.13	11 Aug 2014	Added support for Demo play without user creation in User/*Auth* requests for supported systems
2.14	11 Aug 2014	Finally, internal dynamic game list interface got exposed as Game/* family of public API.
2.15	23 Sep 2014	Changed format of Games/List: “HasMobile” is replaced with “MobilePageCode” field
2.16	16 Oct 2014	Added “AffiliateID” support to Stats/BetsSummary. Added previously not documented “User/Update” request.
2.17	06 Nov 2014	New error codes related to payment system integration. Lao language added.
2.18	18 Dec 2014	Added ExtParam support to User/KillAuth
2.19	20 Jan 2015	Changed optional parameters to become mandatory: User/Add: Language, Gender, Country, DateOfBirth, RegistrationIP User/*Auth*: Page, UserIP
2.20	20 May 2015	Added ExternalCode and MobileExternalCode to Game/List
2.21	14 Jul 2015	Added games jackpots API request
2.22	20 Aug 2015	100 error code added

2.23	04 Nov 2015	666 code change to 600 for non financial errors

Introduction

1. The main provisions

- This document describes API of FUNDIST.ORG system
- For access setup and technical support please contact FUNDIST.ORG
- This document is intended only for authorized partners of FUNDIST.ORG, and it's forbidden to distribute the document without written permission of FUNIDST.ORG representatives.
- Testing environment is available for API integration development and verification.
- Before entering production, please make sure your system has successfully passed all integration tests, especially for error handling. Please contact FUNDIST.ORG technical support to get the latest integration test script.
- API setup on FUNDIST.ORG side requires full list of IPs for white-listing

2. System definitions

[SERVER] – API server address
[KEY] – API access key
[PWD] – API access password
[IP] – API client IP or '0.0.0.0'
[HASH] – hash value for current request
[TID] – Unique transaction ID (for whole history of requests, 32 chars max)

[URL] – URL for API request
[SYSTEM] – System ID of target Merchant (see product-specific Addendum)

{Param} – optional parameter (without brackets)

To make API call, client calls a specially defined [URL]. All requests are case-sensitive and require the exact format, described in Chapter 2. Every request has a defined response. Usually, response consists of response codes and additional information, separated by commas. Response codes are defined in Appendix 1.

A simple hint: if created URL includes space characters then it must be properly escaped with %20

Query string note: API parameters start from "?&", but not traditional "?".

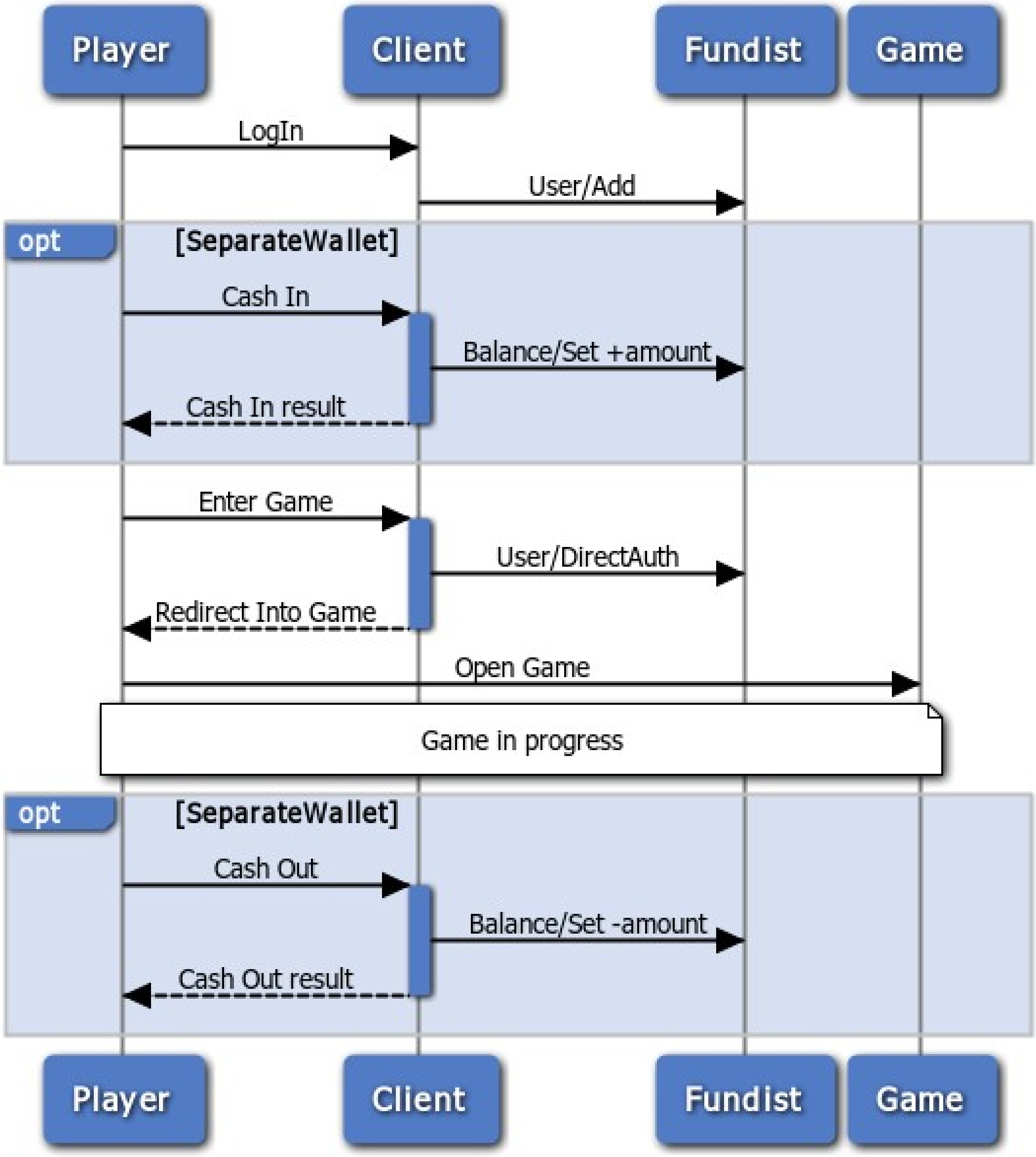
Holistic Picture

Player – actual gambler's web browser

Client – client system with own website

Fundist – eGamings financial system, hidden from Player

Game – one of game product implementations, visible to Player



Funds transfer safety

This sections covers important client-side program logic flow for financial transaction safety.

1. **Cash In / Crediting user account**
   - … Cash In is initiated …
   - Optional step:
     - Perform Balance/Get API request
     - if request fails, abort further processing
     - *Note: this step prevents generation of too many “manual verification” requests*
   - PERFORM internal funds transfer
     - Remove transaction amount from user balance
   - STORE Fundist request in database
     - *Note: this step should be performed in same DB transaction as internal funds transfer, if possible*
     - *COMMIT DB transaction*
   - Optional step, if possible to perform separately:
     - Establish connection to Fundist server
     - if connection fails:
       - transaction can be rolled back with failure status or connection retried.
   - Send API request ==> Fundist server
     - Make sure automatic retries are DISABLED
   - Receive API response <== Fundist server
   - Process API response
     - if valid response: “XX,YY”, **where XX are digits and YY is arbitrary string**
       - **STORE** API response in database
       - if XX equal 1:
         - ***COMMIT** DB TRANSACTION*
         - **FINISH PROCESSING, Success**
       - if XX **equal** (0 or 666):
         - TRIGGER manual verification
         - ***COMMIT** DB TRANSACTION*
         - FINISH PROCESSING, Manual Verification
       - else
         - PERFORM internal funds transfer (return amount to user balance)

        - **COMMIT** combined DB TRANSACTION
        - **FINISH PROCESSING,** Failure
    - In all other cases, **DO NOT** return transaction amount to user balance
      - Do manual verification and/or automatic error recovery

2. **Cash Out/ Debiting user account**
   - ➔ … Cash Out is initiated …
   - ➔ PERFORM Balance/Get API request
     - if request fails, abort further processing
     - check if current game balance >= cash out amount
   - ➔ STORE Fundist request in database
     - *COMMIT DB transaction*
   - ➔ Optional step, if possible to perform separately:
     - Establish connection to Fundist.ORG server
     - if connection fails:
       - transaction can be rolled back or connection retried.
   - ➔ Send API request ==> Fundist.ORG server
     - Make sure automatic retries are DISABLED
   - ➔ Receive API response <== Fundist.ORG server
   - ➔ Process API response
     - if valid response: “XX,YY”, **where XX are 1-2 digits and YY is arbitrary string**
       - Store API response in database
       - if XX equal 1:
         - PERFORM internal funds transfer (add amount to user balance)
         - COMMIT combined DB TRANSACTION
         - **FINISH PROCESSING, Success**
       - if XX **equal** (0 or 666):
         - TRIGGER manual verification
         - ***COMMIT*** *DB TRANSACTION*
         - FINISH PROCESSING, Manual Verification
       - else
         - ***COMMIT*** *DB TRANSACTION*
         - FINISH PROCESSING, Failure
     - In all other cases, **DO NOT** add transaction amount to user balance
       - **Do manual verification and/or automatic error recovery**

3. Automatic error recovery

➔ Identify MAX Fundist API request time (max_req_time):

- ○ (Connection timeout + Send timeout + Receive timeout + DB/misc lock timeouts) x attempts
- ○ And multiple by x2 for extra safety

➔ On regular schedule (e.g. once in X minutes):

- ○ SELECT API_requests from database where response is null and request_time <= (current_time – max_req_time)
- ○ FOR EACH PERFORM "Request/Info" API request (specification 5.2)
    - ▪ if response is "NOT_FOUND" OR "Request" and "Arguments" fields match:
        - • if response is "NOT_FOUND":
            - ○ Set response as "19,Not found"
        - • Continue with "Process API response step" in Cash In or Cash Out flows
            - ○ *Note: "0" response code, means Fundist.ORG is still processing transaction, but it must be handled as manual verification, according to flow above*
    - ▪ In all other cases:
        - • Contact Fundist.ORG support for manual verification and further directions
    - ▪ CONTINUE WITH NEXT

API Request specification

1. New user creation and/or update

[URL] has a form of:

```
https://[SERVER]/System/Api/[KEY]/User/Add/?
&Login=[LOGIN]
&Password=[PASSWORD]
&TID=[TID]
&Currency=[CURRENCY]
&Hash=[HASH]
&Language=[LANGUAGE]
&Gender=[GENDER]
&Country=[COUNTRY]
&DateOfBirth=[DOB]
&RegistrationIP=[REGIP]
{&Nick=[NICK]}
{&Timezone=[TIMEZONE]}
{&Name=[NAME]}
{&LastName=[LASTNAME]}
{&Phone=[PHONE]}
{&AlternativePhone=[ALTPHONE]}
{&City=[CITY]}
{&Address=[ADDRESS]}
{&Email=[EMAIL]}
{&AffiliateID=[AFFILIATEID]}
```

where:

[LOGIN] – User Login in API system. Allowed symbols: 0-9, a-z, A-Z и _. Maximal length: 29 symbols. The login can consist only from digits, what allows using client's system unique user ID. In our back office, the user names are prefixed with special string NN_, where NN_ is always the same per API setup.

[PASSWORD] – User's password. It's an internal password shared only between FUNIDST.ORG and client's system. Normally, this password is a random generated string. Do NOT use the real password of user in client's system! Minimal length is 6. It should not contain the [LOGIN].
[CURRENCY] –The currency of user's account. It cannot be changed!

[HASH] – MD5 checksum of the following data:

```
User/Add/[IP]/[TID]/[KEY]/[LOGIN]/[PASSWORD]/[CURRENCY]/[PWD]
```

Optional parameters:
[LANGUAGE] – language code (Appendix 3). If present, user's language in database is updated.
[NICK] – nick name, which prevents prompting on first entry in some merchants

[TIMEZONE] – timezone shift in minutes relative to UTC. Valid range is from -720 till +720.
[NAME] – first name
[LASTNAME] – last name
[GENDER] – male or female
[PHONE] – primary phone
[ALTPHONE] – alternative phone
[COUNTRY] – ISO 2 or 3 symbol code
[CITY] – city name
[ADDRESS] – home address
[EMAIL] – e-mail. Must not be changed or user may be rejected by some merchants until restored
[DOB] – YYYY-MM-DD
[AFFILIATEID] – third-party affiliate ID
[REGIP] – valid IPv4 address (e.g. 12.34.56.78)

Server response:

1. On success: 1

2. On error, and error code and description is returned, according to Appendix 1

3. On duplicate login, database is updated with new information and success status is returned.

1.1. User information update

User/Update – is an alias for User/Add. Please use this request for semantically more correct naming.

Note:

[HASH] – MD5 checksum of the following data:

User/Update/[IP]/[TID]/[KEY]/[LOGIN]/[PASSWORD]/[CURRENCY]/[PWD]

2. Balances and Transfers

2.1. User cash in/out to/from specified merchant

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/Balance/Set/?&Login=[LOGIN]&System=[SYSTEM]&Amount=[AMOUNT]&TID=[TID]&Currency=[CURRENCY]&Hash=[HASH]

where:

[LOGIN] - user's login in API
[AMOUNT] – Amount of transaction. Positive amount – cash in, negative amount – cash out. The final balance = current balance + amount.
Note: Integer part must be always present with 2 digits after point. Like "-1234.56", "1234.00", "0.70", etc.
[CURRENCY] – Transaction current, which must match user's currency
[HASH] – MD5 checksum of the following data:

Balance/Set/[IP]/[TID]/[KEY]/[SYSTEM]/[AMOUNT]/[LOGIN]/[CURRENCY]/[PWD]

Server response:

1. On success: 1,XX – where XX is user balance after transaction is complete

2. On error, and error code and description is returned, according to Appendix 1
   All errors, except code 666 and unknown response, should rollback the transaction in client's system.

3. When client's network runs out of balance, our technical support is immediately notified.

2.2. Get user balance in specified merchant

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/Balance/Get/?&Login=[LOGIN]&System=[SYSTEM]&TID=[TID]& Hash=[HASH]

where:

[LOGIN] - user's login in API
[SYSTEM] – the target merchant account (see product-specific Addendum)
[HASH] – MD5 checksum of the following data:

Balance/Get/[IP]/[TID]/[KEY]/[SYSTEM]/[LOGIN]/[PWD]

Server response:

1. On success: 1,XX,YY – where XX - current balance, YY – currency of the balance

2. On error, and error code and description is returned, according to Appendix 1

2.3. User balance withdrawal from specified merchant

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/Balance/Withdraw/?
&Login=[LOGIN]&System=[SYSTEM] &TID=[TID]&Hash=[HASH]

where:

[LOGIN] - user's login in API
[SYSTEM] – the target merchant account (see product-specific Addendum)
[HASH] – MD5 checksum of the following data:

Balance/Withdraw/[IP]/[TID]/[KEY]/[SYSTEM]/[LOGIN]/[PWD]

Server response:

1. On success: 1,XX,YY – where XX is withdrawn balance, YY its currency
   Important: never rely on previous „get balance" calls. Always use the value of XX!
   Note: if user balance is zero, the call still succeeds.

2. On error, and error code and description is returned, according to Appendix 1

3. User redirection into merchant system

Historically, it was a two step process: authorization and then redirection. Currently, we support ONLY a single step process with direct authorization, which does not require redirecting users to FUNIDST.ORG

3.1. Direct authorization

[URL] has a form of:

```
https://[SERVER]/System/Api/[KEY]/User/DirectAuth/?
&Login=[LOGIN]
&Password=[PASSWORD]
&System=[SYSTEM]
&TID=[TID]
&Hash=[HASH]
&Page=[PAGE]
&UserIP=[USERIP]
{&Language=[LANGUAGE]}
{&Nick=[NICK]}
{&Timezone=[TIMEZONE]}
{&Demo=[DEMO]}
{&IsMobile=[ISMOBILE]}
{&ExtParam=[EXTPARAM]}
```

where:

[LOGIN] - user's login in API

> Note: use "Login=$DemoUser$&Demo=1" for demo play without user registration, if supported by system

[PASSWORD] – user's password in API
[SYSTEM] – the target merchant account (see product-specific Addendum)
[HASH] – MD5 checksum of the following data:

```
User/DirectAuth/[IP]/[TID]/[KEY]/[LOGIN]/[PASSWORD]/[SYSTEM]/[PWD]
```

Optional parameters:
[PAGE] – page code for redirection (see product-specific Addendum). Main lobby by default. Please contact Egamings support for valid game codes.
[LANGUAGE] – language code (Appendix 3). User language is saved in database.
[NICK] – nick name, which prevents prompting on first entry
[TIMEZONE] – timezone shift in minutes relative to UTC. Valid range is from -720 till +720.
[DEMO] – either 1 or 0. Enables demo play mode for certain systems. It's ignored for lobby page.
[ISMOBILE] – either 1 or 0. Enables mobile device support. It's ignored for lobby page.
[EXTPARAM] – optional parameter, which will be present in OneWallet callbacks
[USERIP] – IP address of user. May be required by same products with extra security and/or geo-optimization.

Server response:

1. On success: 1,XXXXX – where XXXXX is a direct link to merchant system, which can contain commas (only the first comma should be processed).
2. On error, and error code and description is returned, according to Appendix 1

3.2. Kill authorization (end active sessions)

This request forcibly kills all active user sessions. It may not work for all systems.

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/User/KillAuth/?
&Login=[LOGIN]
&TID=[TID]
&Hash=[HASH]
{&ExtParam=[EXTPARAM]}

where:

[LOGIN] - user's login in API
[HASH] – MD5 checksum of the following data:
[EXTPARAM] – optional parameter. If set, kills only sessions associated with ExtParam value

User/KillAuth/[IP]/[TID]/[KEY]/[LOGIN]/[PWD]

Server response:

1. On success: 1
2. On error, and error code and description is returned, according to Appendix 1

3.3. Authorization with HTML fragment

Due to integration approach of some service providers, User/DirectAuth approach with plain URL for <iframe> cannot be used in all cases.

User/AuthHTML is absolutely identical to User/DirectAuth, except that HTML fragment is returned instead of URL.

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/User/AuthHTML/?
&Login=[LOGIN]
&Password=[PASSWORD]
&System=[SYSTEM]
&TID=[TID]
&Hash=[HASH]
&Page=[PAGE]
&UserIP=[USERIP]
{&Language=[LANGUAGE]}
{&Nick=[NICK]}

{&Timezone=[TIMEZONE]}
{&Demo=[DEMO]}
{&IsMobile=[ISMOBILE]}
{&ExtParam=[EXTPARAM]}

where:

[HASH] – MD5 checksum of the following data:

User/AuthHTML/[IP]/[TID]/[KEY]/[LOGIN]/[PASSWORD]/[SYSTEM]/[PWD]

Please see User/DirectAuth for description of other parameters.

Server response:

3. On success: 1,XXXXX – where XXXXX is a HTML fragment to be inserted into website, which can contain commas (only the first comma should be processed).
4. On error, and error code and description is returned, according to Appendix 1

4. Statistics

Based on a heavy demand of customers, we have redesigned and re-enabled statistics API. ***Please note that any legacy still working calls will be disabled as soon as the new API fully replaces the old one.***

4.1. Universal Bets Summary

[URL] has a form of:

```
https://[SERVER]/System/Api/[KEY]/Stats/BetsSummary/?
&Date=[DATE}
&TID=[TID]
&Hash=[HASH]
{&Login=[LOGIN]}
{&System=[SYSTEM]}
{&Format=[FORMAT]}
{&AffiliateID=[AFFILIATEID]}
```

where:

[DATE] – YYYY-MM-DD date for selection. Range selections are currently disabled.
[HASH] – MD5 checksum of the following data:

```
Stats/BetsSummary/[IP]/[TID]/[KEY]/[DATE]/[PWD]
```

Optional parameters:
[LOGIN] – apply filter based on user's login in API
[SYSTEM] – apply filter based on the merchant system (see product-specific Addendum)
[FORMAT] – output format: “csv” or “json”. Default: “csv”
[AFFILIATEID] – apply filter based on AffiliateID parameter passed to User/Add

Server response:

1. On success, either CSV or JSON formatted statistical data is returned
2. On error, and error code and description is returned, according to Appendix 1

CSV response:

First line: field names and their order
All other lines: user x system x date bets summary

JSON response:

```
{
        “fields” : [ field order list for data ],
        “data” : [
                [ user x system x date bets summary ],
                …
        ]
}
```

Possible fields:

'user_name' – user login as in API requests
'system_id' – merchant's system

'currency_name' – user's currency name (ISO code)
'bet_total' – total amount of bets
'bet_count' – total count of bets
'win_total' – total amount of wins
'game_desc' – game description
'gambling_date' – date of the summary (added only in range selections)

5. Error handling

5.1. Retrieve response of previously executed request

This functionality can be used to workaround connection problems.

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/Response/Get/?&TID=[TID]

Server response:

- The original response, if found
- NOT_FOUND, if not found
- KEY_ERROR, if [KEY] is invalid
- IP_ERROR, if requesting IP is not allowed for [KEY]

5.2. Retrieve information of previously executed request

The request is identical to Response/Get, but provides additional information for better robustness of automatic error handling and recovery. Response/Get has a potential race condition, if duplicated [TID] requests are received.

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/Response/Info/?&TID=[TID]

Server response:

If request [TID] is found then response consists of several lines separated by newline ('\n') character:

1. Request (Balance/Get, Balance/Set, etc.)
2. Arguments (?&=....)
3. Response code
4. Full response
5. Date and time of request
6. IP of request

On error:

- NOT_FOUND, if not found
- KEY_ERROR, if [KEY] is invalid
- IP_ERROR, if requesting IP is not allowed for [KEY]

6. Dynamic Game Lists

FUNDIST.ORG provides unique dynamic game lists, ready to be used directly in web sites. Of course, sane result caching for performance and load reasons is strongly recommended. We recommended to update the game list on daily basis.

The major advantage over all competitors is that categories and games are presorted in recommended order. However, each client can override the default sorting order, using our FUNDIST.ORG backoffice operator account.

The game list is continuously maintained by our staff.

6.1. Get game categories

[URL] has a form of:

https://[SERVER]/System/Api/[KEY]/Game/Categories/?
&TID=[TID]
&Hash=[HASH]

where:

[HASH] – MD5 checksum of the following data:

> Game/Categories/[IP]/[TID]/[KEY]/[PWD]

Server response:

1. On success, JSON formatted data is returned
2. On error, and error code and description is returned, according to Appendix 1

JSON response:

```
[
        {
                "ID" : "globally unique numeric category ID",
                "Trans" : {
                        "en" : "Name in English, always present",
                        ...
                },
                "Tags" : [
                        "main",
                        ...
                ],
        },
        ...
]
```

Note: "main" tag identifies primary category type. Non-primary categories are not necessary needed to be used/displayed.

6.2. Get available games

[URL] has a form of:

```
https://[SERVER]/System/Api/[KEY]/Game/List/?
&TID=[TID]
&Hash=[HASH]
```

where:

[HASH] – MD5 checksum of the following data:

```
        Game/List/[IP]/[TID]/[KEY]/[PWD]
```

Server response:

1. On success, JSON formatted data is returned
2. On error, and error code and description is returned, according to Appendix 1

JSON response:

```
        [
                {
                        "System" : "Refers to System parameter of User/*Auth* requests",
                        "PageCode" : "Refers to Page parameter of User/*Auth* requests",
                        "MobilePageCode" : "Refers to Page parameter of User/*Auth* requests",
                        "Trans" : {
                                "en" : "Name in English, always present",
                                ...
                        },
                        "ImageURL" : "Game logo location at public servers",
                        "Categories" : [
                                 globally unique numeric category ID,
                                ...
                        ],
                        "ExternalCode" : "Refers to game_desc field of Stats/*Auth*",
                        "MobileExternalCode" : "Refers to game_desc  field of Stats/*",
                },
                ...
        ]
```

Note: for performance reasons, we recommend to retrieve image (ImageURL) and cache on client web servers for players

7. Games jackpots

FUNDIST.ORG provides functionality to get jackpots for following game operators: Netent, Microgaming, Betsoft. The API request for obtaining jackpot information is as follows:

[URL] has a form of:

```
https://[SERVER]/System/Api/[KEY]/WLCGames/Jackpots/?
&TID=[TID]
&Hash=[HASH]
&System=[SYSTEM]
{&Currency=[CURRENCY]}
```

where:

[SYSTEM] – the target merchant account (see product-specific Addendum)
[CURRENCY] – if optional parameter is provided (ISO currency code) then jackpot amounts are returned in this currency
[HASH] – MD5 checksum of the following data:

```
WLCGames/[IP]/[TID]/[KEY]/[PWD]
```

Server response:

1. On success, JSON formatted data is returned
2. On error, and error code and description is returned, according to Appendix 1

JSON response:

```
[
        {
                "game" : "Games name",
                "amount" : "Jackpot amount"
        },
        ...
]
```

Note: response format can vary depending on requested merchant id, this is because different jackpot systems from different merchants.

Appendix 1. Error codes

Error code	Description
11,Wrong authorization	Invalid API [KEY] specified
12,Wrong authorization ip	Request IP is not allowed for specified API [KEY]
13,Wrong action	Wrong action is requested
14,Wrong incoming params	Wrong incoming arguments are specified
15,Wrong authorization hash	Request hash is not properly calculated
16,Login exists	In older versions, it was returned on duplicate User/Add
17,User not found	Unknown user login
18,Wrong merchant access	Unsupported action for specified merchant
19,Transaction fail, [desc]	Internal transaction failure with description
20,Merchant transaction fail,[desc]	External transaction failure with description
21,Can't get balance	Failed to get balance from merchant
22,Wrong user password	Wrong user password is supplied
23,User disabled	Request for disabled/blocked user
24,Redirect error	Redirection error
25,Key error	Failure to create a redirection key
26,User creation error	Failure to create a user
27,Your stall disabled	Request to disabled stall
28, Your net disabled	Request to disabled network
29, Your API disabled	Request to disabled API setup
30,Transactions ID failed value	Duplicate [TID]
31,Wrong currency	Either specified currency is not enabled for stall or user's currency differs.
32,Wrong start or end dates	Start and end dates are not in specified format
33,This site is on maintenance	System is temporary unavailable
34,Invalid Request	Invalid API request (unsupported function)
35,[<MESSAGE>]	Payment system referred in the request is not on the Fundist system, JSON object contains general descriptive message under index 0.
36,[<MESSAGE>]	Payment system has been disabled. JSON object contains general descriptive message under index 0.
37,[<MESSAGE>]	API setup is missing config for referred payment system. JSON object contains general descriptive message under index 0.
38,[<MESSAGE>]	Handler for particular payment system is not implemented. JSON object contains general descriptive message under index 0.
39,[<MESSAGE>, {Min: NN.NN, Max:	Payment amount is out of allowed range for particular API setup and currency.

MM.MM, Currency: <CUR>}]	JSON object contains general descriptive message under index 0, and an array of data (Min, Max, Currency) under the last index, that could be used to generate custom translation.
40,[<MESSAGE>]	Payment init has failed for some reason. JSON object contains general descriptive message under index 0.
41,[<MESSAGE>]	Configuration error. JSON object contains general descriptive message under index 0.
42,[<MESSAGE>]	One of data fields is missing to proceed with payment in particular payment system. JSON object contains general descriptive message under index 0.
43,[<MESSAGE>]	Provided merchant is not on the Fundist system. JSON object contains general descriptive message under index 0.
44,[<MESSAGE>]	Withdraw payment failed because of loyalty system block. JSON object contains general descriptive message under index 0.
45,[<MESSAGE>]	Amount requested for withdrawal is greater than available balance minus already pending withdraws if any. JSON object contains general descriptive message under index 0.
46,[<MESSAGE>]	One of data fields, sent in withdraw request, is invalid. JSON object contains general descriptive message under index 0.
47,[<MESSAGE>]	Amount requested for withdrawal is greater than limit specific for payment system. JSON object contains general descriptive message under index 0.
100,[<MESSAGE>]	An error while requesting an external source. Message contains unexpected (faulty) answer from external source.
600,[<MESSAGE>]	Any unexpected critical error not related to financial transactions.
666,Manual Verify 666,Transient Error	Transaction error, when it's not allowed to rollback a transaction. This situation should be manually process by technical support.
NN,...	Any other code is an error

Appendix 2. Currency codes

Currency code	**Currency name**
AED	Emirati Dirham
AMD	Armenian Dram
AUD	Australian Dollar
CAD	Canadian Dollar
CHF	Swiss Franc
DKK	Danish Krone
EUR	Euro
GBP	British Pound
KZT	Kazakhstani Tenge
LTL	Lithuanian Litas
NOK	Norwegian Krone
PLN	Polish Zloty
RUB	Russian Ruble
SEK	Swedish Krona
UAH	Ukrainian Hryvna
USD	US Dollar

Appendix 3. Language codes

Language code	**Language**
en	English
ru	Russian
de	German
tr	Turkish
nl	Dutch, Netherlands
pl	Polish
ro	Romanian
gr	Greek
fr	French
it	Italian
al	Albanian
bp	Brazilian Portuguese
bg	Bulgarian
ca	Catalan
cn	Chinese
hr	Croatian
cs	Czech
dk	Danish
ee	Estonian
fi	Finnish
he	Hebrew
hu	Hungarian
ja	Japanese
no	Norwegian
pt	Portuguese
sr	Serbian
sk	Slovakian
sl	Slovenian
es	Spanish
sv	Swedish
th	Thai
la	Lao