Connecting to Fundist.org API – One Wallet

Contents

Change History

1.00	10 Feb 2014	First official version of Fundist One Wallet protocol
1.01	28 Feb 2014	Clarified usage of {extra_field} in successful response. Added “tid” and “balance” as mandatory response fields for debit and credit.
1.02	27 Mar 2014	Added information-only “i_gameid” and “i_rollback” fields to debit and credit. Updated introduction section with paragraph on transaction rollbacks.
1.03	01 Apr 2014	Added information-only “i_extparam” field to debit and credit.
1.04	03 Apr 2014	Added information-only “i_extparam” field also to balance request
1.05	06 Apr 2014	Added information-only “i_gamedesc” field to credit, debit and balance requests
1.06	03 May 2014	Added Holistic Picture
1.07	15 Jul 2014	Added clarification for error checking of requests with duplicated TID. Added hmac field in response examples Added type of ID fields description.
1.08	26 Nov 2014	Added “roundinfo” request from Lazy Mode spec.

1. Introduction to One Wallet

The default separate wallet implementation assumes that Client's system and Fundist.ORG system have two independent wallets. The Client system is responsible for adding and withdrawing funds from game account.

In One Wallet implementation, there is a single wallet on Client's system side. Fundist.ORG system performs real-time requests to credit and debit a single user's account in Client's system.

Fundist.ORG performs requests to Client's system through HTTPS as JSON-encoded object in POST data.

There is no concept of transaction rollback on protocol level. Instead, a standard credit request is sent to revert previous debit operation. Such special credit requests can be identified by presence of “i_rollback” field, which holds “tid” of related debit.

Note: Client's system must never depend on i_extparam for validation of OneWallet requests

Response is expected as JSON-encoded object in HTTP response message body.

Note: UTF-8 encoding is assumed, if non-ASCII symbols are used

For optimal performance, please configure HTTPS Keep-Alive of 120 seconds and enable SSL session caching.

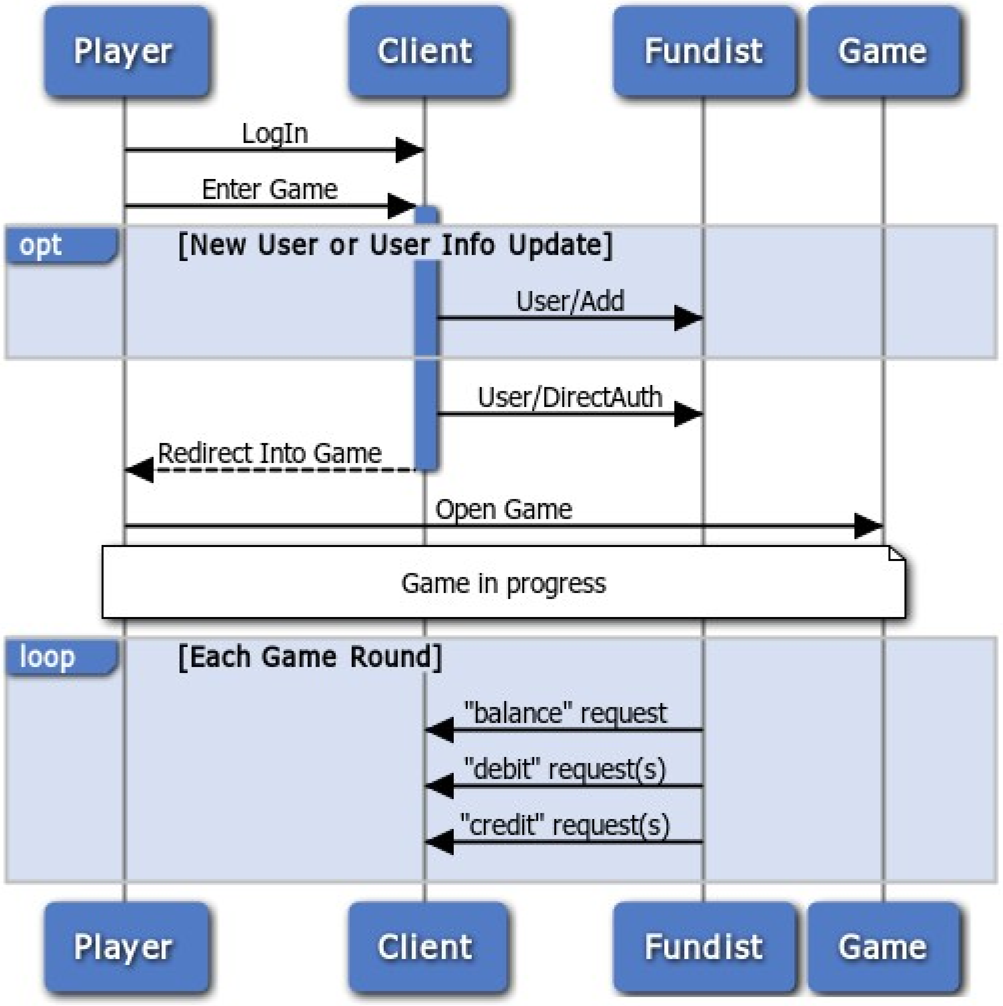
2. Holistic Picture

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



3. Message Authentication

Every message (request and response) should have a special field “hmac”, which value is generated using HMAC[1] with SHA-256 algorithm. HMAC input is generated as:

1. Sort message fields in case-sensitive ascending order
2. Concatenate value of all fields into single string
3. Run HMAC with SHA-256 using secret key shared between Client's and Fundist.ORG system

*Note: **Client's system must explicitly whitelist all Fundist.ORG IP addresses**. All other IPs must be rejected. Please contact eGamings support for actual list of outgoing IP addresses.*

Example:

```
{
        "bbb" : "Val1",
        "aaa" : "Val2",
        "hmac" : "...."
}

hmac = HMAC_SHA256("Val2Val1", SHA256("ASCII_Shared_Secret"))
```

1 HMAC - http://en.wikipedia.org/wiki/Hash-based_message_authentication_code

4. Message Types

Please note: "amount" and "balance" field must always have 2 digits after decimal point and at least one digit in integer part. Decimal point sign must always be a period (“.”). No other symbols are allowed.

4.1. Ping

This request is used to check Client's system availability.

```
{
	"type" : "ping",
	"hmac" : "..."
}
```

4.2. Get balance

Retrieve current available balance of user in Client's system.

Note: this information is used only for in-game display and basic bet amount checking

```
{
	"type" : "balance",
	"userid" : "User/DirectAuth Login parameter",
	"currency" : "User/Add Currency parameter",
	"i_extparam" : "External parameter, passed to User/DirectAuth",
	"i_gamedesc" : "{SystemID}:{GameType}",
	"hmac" : "..."
}
```

Response must contain extra "balance" field.

4.3. Debit from user account

```
{
	"type" : "debit",
	"tid" : "Unique Transaction ID – alphanumeric maxlen 32",
	"userid" : "User/DirectAuth Login parameter",
	"currency" : "User/Add Currency parameter",
	"amount" : "0.00",
	"i_gameid" : "Game ID. Can be used to identify related transactions – alphanumeric
maxlen 32",
	"i_extparam" : "External parameter, passed to User/DirectAuth",
	"i_gamedesc" : "{SystemID}:{GameType}",
	"hmac" : "..."
}
```

1. Client's system must check, if request with the same "tid" has been performed before:
   a) if any of essential fields ("tid", "userid", "currency", "amount") mismatch:
      - return error "Transaction parameter mismatch"

   b) if yes and response is known, return original response
      - Note: "balance" field must contain current value

    c) if yes and response is NOT known, fail with HTTP error “408 Request Timeout”
    d) otherwise, continue
2. Check if currency matches user balance currency
    a) Respond with error, if not
3. Perform in single DB transaction:
    a) Check if user balance is greater or equal to requested amount
    b) If the check is passed, decrease user balance by requested amount
    c) If the check is NOT passed, fail with error
    d) Store request and response in DB
    e) Commit DB transaction
4. Send response message
    a) Response must contain extra "tid" field of original request and <u>current</u> "balance" field.

4.4. Credit to user account

```
{
        "type" : "credit",
        "tid" : "Unique Transaction ID – alphanumeric maxlen 32",
        "userid" : "User/DirectAuth Login parameter",
        "currency" : "User/Add Currency parameter",
        "amount" : "0.00",
        "i_gameid" : "Game ID. Can be used to identify related transactions – alphanumeric maxlen 32",
        "i_extparam" : "External parameter, passed to User/DirectAuth",
        "i_rollback" : "Optional. Can be used to link related debit request.",
        "i_gamedesc" : "{SystemID}:{GameType}",
        "hmac" : "..."
}
```

1. Client's system must check, if request with the same "tid" has been performed before:
    a) if any of essential fields ("tid", "userid", "currency", "amount") mismatch:
        - return error "Transaction parameter mismatch"
    b) if yes and response is known, return original response
        - Note: "balance" field must contain <u>current</u> value
    c) if yes and response is NOT known, fail with HTTP error “408 Request Timeout”
    d) otherwise, continue
2. Check if currency matches user balance currency
    a) Respond with error, if not
3. Perform in single DB transaction:
    a) Increase user balance by requested amount
    b) Store request and response in DB

   c) Commit DB transaction

4. Send response message

   a) Response must contain extra "tid" field of original request and current "balance" field

4.5. Round Info

Provide real-time round information after round is over. Useful for bonus & loyalty analytics.

This request is e re-send until Client system responds with successful reply.

NOTE: this is a configurable option for backward compatibility purposes.

Request format:

```
{
        "type" : "roundinfo",
        "gameid" : "Unique Game ID",
        "userid" : "User/DirectAuth Login parameter",
        "actions" : [
                {
                        "actid" : "Unique Action ID",
                        "type" : "bet|win",
                        "amount" : "0.00",
                        "timestamp" : "YYYY-MM-DD hh:mm:ss"
                },
                ...
        ],
        "i_gamedesc" : "{SystemID}:{GameType}",
        "hmac" : "..."
}
```

4.6. Successful response

Simple response:

```
{
        "status" : "OK",
        "hmac" : "..."
}
```

Extended response:

```
{
        "status" : "OK",
        "{extra_field}" : "Additional Response field",
        "hmac" : "..."
}
```

4.7. Error response

```
{
        "error" : "Human readable description",
        "hmac" : "..."
}
```