Description of OneWallet Integration Principles v1.3

Introduction

OneWallet implementation uses a single wallet on the client system side. The integration service performs real-time requests to credit and debit funds from a single user account in the client’s system. The integration service sends requests to the client’s system via HTTPS as a JSON-encoded object in the POST request body. The response is expected as a JSON-encoded object in the HTTP response body.

Note: If non-ASCII characters are used, UTF-8 encoding is assumed.

Change History

Version	Date	Description
1.0	01/10/2024	First version of the protocol
1.1	16/10/2024	• Added multi-currency support • Added description of retrieving data from the game catalog • Redesigned the mechanism for obtaining a game link • Added rollback event • Added a table with error codes and their descriptions
1.2	17/10/2024	Minor edits and clarifications in the text
1.3	05/11/2024	• Fixed example code for generating the signature field • Removed currency from **getBalance** response • Added error code 607 Internal Error

Glossary

Player – The user's web browser.

Client – The client’s system with its own website.

Integration Service – A service that interacts with the client and is hidden from the player.

Game – One of the game implementations visible to the player.

UID – A signature generated by the Integration Service to ensure user authentication and data security.

Token – An access token used to authenticate the client when initializing the game (transmitted to the client via secure communication channels).

Secret key – A secret key used to generate message signatures, ensuring secure data transmission between the Integration Service and the client.

Interaction Stages

1. Game Initialization

To retrieve a list of available games, the client system must send a request to the Integration Service at the following URL:

https://b.misuerte.bet/api/integration/wallet/api/v1/game/getGamesCatalog/

passing the following parameters:

```
{
  "token": "41025659200f421e14d178c51dce7303a11472",
  "currency": "USD"
}
```

Where:

token – client access token,

currency – Currency. Format: 3-letter code according to ISO 4217.

If the request is successfully processed, the response will return the following data set:

```
[
    {
        "id": 1,
        "name": "Game 1",
        "demo_availability": false,
        "preview_url": "https://url_to_preview_pic/pic_game1.jpg"
    },
    {
        "id": 3,
        "name": "Game 2",
        "demo_availability": true,
        "preview_url": " https://url_to_preview_pic/pic_game2.jpg "
    }
]
```

Where:

id – Game ID in the catalog,

name – Display name of the game,

demo_availability – Flag indicating the availability of demo mode,

preview_url – URL of the game preview image for website display

If the request fails, the response will return an HTTP status code in the 6xx range (refer to the error codes table in the corresponding documentation section). The response body will contain an error description.

```
{
    "status": "ERROR",
    "error": "Unauthorized"
}
```

To retrieve a game link, the client system must send a request to the Integration Service at the following URL:

https://b.misuerte.bet/api/integration/wallet/api/v1/game/getGameUrl/

passing the following parameters:

```
{
    "catalog_game_id": 1,
    "is_demo": false,
    "currency": "USD",
    "player_login": "test1",
    "player_lang": "en",
    "player_ip": "8.8.8.8",
    "token": "41025659200f421e14d178c51dce7303a11472"
}
```

Where:

catalog_game_id – The game ID obtained from the game catalog,

is_demo – Flag indicating whether demo mode is enabled,

currency – Currency. Format: 3-letter code according to ISO 4217,

player_login – The player's login in the client's system,

player_lang – The player's language,

player_ip – The player's IP address,

token – The client's access token

If the request is successfully processed, the response will return a link to the game.

```
{
    "url": "https://url_with_get_parameters"
}
```

If the request fails, the response will return an HTTP status code in the 6xx range (refer to the error codes table in the corresponding documentation section). The response body will contain an error description:

```
{
    "status": "ERROR",
    "error": "Unauthorized"
}
```

2. Gameplay Process

Once the player opens the initialized game, the Integration Service will send requests to the Client's system. The API endpoint must be provided by the Client and registered in the Integration System.

- Balance request – Sent every 10 seconds.
- Other requests – Sent depending on the player's actions in the game.

Each request and response must be authenticated.

Message Authentication

Every message (both request and response) must include a special field called **"signature"**, whose value is generated using the **HMAC with SHA-256** algorithm.

The HMAC input data is generated as follows:

1. **Sort** the message fields in case-sensitive ascending order.
2. **Concatenate** all field values into a single string.
3. **Run HMAC with SHA-256**, using the **Secret Key**, which is shared between the Client's system and the Integration Service.

Example:

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

Example code for generating the "signature" field in Node.js

```
const crypto = require('crypto');

const base = Object.assign({}, requestBody);
delete base.hmac;

const hash = crypto.createHash('sha256');
const hmac = crypto.createHmac('sha256', hash.update(secretKey).digest('buffer'));

let hmacBase = '';
Object.keys(base).sort().forEach(key => hmacBase += base[key]);

const signature = hmac.update(hmacBase).digest('hex');
```

Message Types

The types of messages that the Integration Service sends to the Client's system:

getBalance – Retrieve the user's balance

Request Parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
type	string	Type of request	Required
user	string	Player login obtained from the Client system during game initialization	Required
currency	string	Currency (3-letter ISO 4217 code)	Required
signature	string	Digital signature for message authentication	Required

Response parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
status	string	Message Processing Status HTTP Code 200 and OK in the body – means the request has been processed successfully. HTTP Code other than 200 – indicates an error.	Required
balance	string	The balance of the requested user	Required
signature	string	Digital signature for message authentication	Required

debitBalance - request for deducting funds from the user's balance

Request Parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
type	string	Type of request	Required
user	string	Player login obtained from the Client system during game initialization	Required
game_id	string	The game ID within which the balance deduction occurs	Required
transaction_id	string	Transaction ID in the Integration Service	Required
amount	string	Amount to be deducted from the user's balance	Required
currency	string	Currency for the deduction (3-letter ISO 4217 code)	Required
signature	string	Digital signature for message authentication	Required

Response parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
status	string	Message Processing Status HTTP Code 200 and OK in the body – means the request has been processed successfully. HTTP Code other than 200 – indicates an error.	Required
balance	string	User balance after transaction	Required
transaction_id	string	Transaction ID in the Integration Service	Required
signature	string	Digital signature for message authentication	Required

creditBalance – Request for depositing funds into the user's balance

Request Parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
type	string	Type of request	Required
user	string	Player login obtained from the Client system during game initialization	Required
game_id	string	The game ID within which the balance is being deposited	Required
transaction_id	string	Transaction ID in the Integration Service	Required
amount	string	Amount to be added to the user's balance	Required
currency	string	Currency for the deposit (3-letter ISO 4217 code)	Required
signature	string	Digital signature for message authentication	Required

Response parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
status	string	Message Processing Status HTTP Code 200 and OK in the body – means the request has been processed successfully. HTTP Code other than 200 – indicates an error.	Required
balance	string	User balance after transaction	Required
transaction_id	string	Transaction ID in the Integration Service	Required
signature	string	Digital signature for message authentication	Required

rollback – Request for cancelling a deposit or deduction operation

Request Parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
type	string	Type of request	Required
user	string	Player login obtained from the Client system during game initialization	Required
game_id	string	The game ID within which the balance change (deposit or deduction) occurs	Required
transaction_id	string	Transaction ID in the Integration Service	Required
amount	string	The amount by which the balance is being changed (either deposited or deducted)	Required
currency	string	Currency (3-letter ISO 4217 code)	Required
rb_ transaction_id	string	The unique transaction ID of the transaction being canceled	Required
rb_type	string	Type of the operation being canceled ("debit" for deduction, "credit" for deposit)	Required
signature	string	Digital signature for message authentication	Required

Operation Logic Based on rb_type:

- When rb_type = "credit":
  The funds should be deducted from the player's balance.
- When rb_type = "debit":
  The funds should be credited to the player's balance.

Response parameters:

Parameter Name	Parameter Type (JSON)	Description	Required / Optional
status	string	Message Processing Status HTTP Code 200 and OK in the body – means the request has been processed successfully. HTTP Code other than 200 – indicates an error.	Required
balance	string	User balance after transaction	Required
transaction_id	string	Transaction ID in the Integration Service	Required
signature	string	Digital signature for message authentication	Required

Number Format for amount and balance Fields

The values for the amount (sum) and balance fields must always adhere to the following format:

- Two digits after the decimal point.
- At least one digit before the decimal point.
- The decimal separator must always be a period ("."), and no other symbols are allowed.
- The data type for these fields in JSON must be string.

Examples: "0.00", "0.01", "1.00", "2.19"

Error Message Format

In case an error occurs while processing a message, the response should include:

- An HTTP code 6xx (from the table below).
- The response body may also include a human-readable error message in the following format:

```
{
  "status": "ERROR",
  "error": "error text"
}
```

Error Code	Message	Description
601	Unauthorized	Authorization failed
602	Invalid game id	The game ID does not exist in the game catalog
603	Demo unavailable	Demo mode is not supported for this game
604	Invalid currency	The specified currency is not supported
605	Invalid user	The player's login contains invalid characters
606	Insufficient funds	The player’s balance is insufficient
607	Internal error	An internal service error occurred

Example Messages

Below are examples of messages generated using a secret key:

Secret key = QsOK6QyWFWY7oGeWZTVoauxTIyu8mgW5drae3TMsp6zge6w0QZg1hnWglVPlqDrx

getBalance

Request:

```
{
  "type": "getBalance",
  "user": "test1",
  "currency": "COP",
  "signature": "17df160f296f269444db6fbdbd862aa87474a6c06d8c71568d5dd16aa08258f8"

}
```

Response:

```
{
  "status": "OK",
  "balance": "10000.00",
  "signature": " 0c9311cd721a6996b8d2667bfb7e488212842b820baf5b55f24815f22bbbdfec"

}
```

debitBalance

Request:

```
{
   "type": "debitBalance",
   "user": "test1",
   "game_id": "50",
   "transaction_id": "644",
   "amount": "5.00",
   "currency": "COP",
   "signature": "4b00697b65c695b8479477e5b3690ecba2ae672f6434b500c34448ad3e681d99"

}
```

Response:

```
{
   "status": "OK",
   "balance": "9995.00",
   "transaction_id": "644",
   "signature": "98b5481baf0cae75a793c903ab24398b1af0434bc83ff28dfd046c11c04b36ec"

}
```

creditBalance

Request:

```
{
   "type": "creditBalance",
   "user": "test1",
   "game_id": "50",
   "transaction_id": "647",
   "amount": "20.00",
   "currency": "COP",
   "signature": "f2b8674ea105b2672a0dff05d8d751f35f69a26c4fb09257087d130f07bf8e5b"

}
```

Response:

```
{
   "status": "OK",
   "balance": "10015.00",
   "transaction_id": "647",
   "signature": "6a602dd6b33ca6443d499b28c60b88960c49de62f4c442ebfaa18ad6f6b6e25d"

}
```

rollbackTransaction

Request:

```
{
   "type": "rollbackTransaction",
   "user": "test1",
   "game_id": "50",
   "transaction_id": "648",
   "amount": "20.00",
    "currency": "COP",
    "rb_ transaction_id ": "647",
    "rb_type": "credit",
    "signature": "5abbc314f5211bfb475a79c0dfeb110b8d2271350ddc4d27952c274be6d58cd8"
}
```

Response:

```
{
   "status": "OK",
   "balance": "9995.00",
   "transaction_id": "648",
   "signature": "25730a16a8e217b08e7b7c57f6a5967713706e8293d95604a8b0391551d39fad"

}
```