Push Balance

Financial Transactions

Revision History

Date	Changes	Version	Author
September, 2017	First Release	V2.0.1	Marianna Reva
February, 2022	Server address update	V2.0.2	Razvan Andrei

Contents

1. Document Overview

The purpose of this document is to describe the push balance for the perspective programmer and integration experts.

2. General

The push balance is an informative action that does not carry direct financial changes or risks. It does not change the actual financial status of the player.

3. Functionality

The Push Balance API provides the operators with an ability to update player's balance on Ezugi servers. The operator will use the API only if the player's balance was updated due to player's activity on other game providers than Ezugi or due to funds withdraw/deposit.

4. Security

The hash signature will be created by using SHA256 hash algorithm and will create a signature by using a secret key and the push balance message. An output format is Base64 encoding. The value of the hash is unique to the hashed data. Any change in the data, even changing or deleting a single character, results in a different value. In addition, Ezugi will whitelist all range of IPs provided by operator.

5. Setup

Ezugi integrator will provide the hash in UUID format. It's the same Hash Signature Key provided at the beginning of the integration.
For Example: **6f2a6e62-fb91-4fb7-8e71-83a82f1fabae**

6. Connecting with Ezugi Servers

Integration server:
https://engineint.tableslive.com/GameServer/OperatorGate

7. Protocol:

Https POST communication based on Json string representation message type.

8. Fields Description Message

Parameter Name	Parameter Format	Description
operatorId	int	Defines the operator that the player belongs to. This is allocated by the Live Casino system in advance.
uid	string	An operator's unique identification for each player.
balance	double	Player's new balance
timestamp	long	Time representation in UNIX milliseconds format. To see the system time in Ezugi system you can check https://www.epochconverter.com or http://currentmillis.com/ (Here you can find integration instructions)
MessageType	Varchar	Message type "PushBalance" will be sent by operator

HTTP Header

Header Name	Header Value
hash	An HmacSHA256 signature that was created by hashing the push balance request message and a secret key that was issued to operator by Ezugi. Signature creation examples can be found p.11 Creating base64 Hash Using HMAC SHA256

9. Response

The response on push balance request will be returned in a JSON format as follows:

```
{
      "status": "Success"/"Failed"
}
```

10. Message Example

```
{
    "operatorId": 131245278,
    "uid": "12345",
    "balance": 100.0,
    "timestamp": 12879437712,
    "MessageType": "PushBalance"
}
```

11. Creating base64 Hash Using HMAC SHA256

11.2 Implementation methods for different languages

Java

```
public static String encode (String key, String data) throws Exception {

        Mac sha256_HMAC = Mac.getInstance("HmacSHA256");

        SecretKeySpec secret_key = new SecretKeySpec(key.getBytes(),
        "HmacSHA256"); sha256_HMAC.init(secret_key);

        return
        Base64.encodeBase64String(sha256_HMAC.doFinal(data.getBytes()))
        ;

        }
```

PHP

Use the standard function hash_mac

```
$s = hash_hmac('sha256', 'Message', 'secret', true);

echo base64_encode($s);
```

JavaScript

Use the CryptoJS library.

```
var hash = CryptoJS.HmacSHA256("Message", "secret"); var

hashInBase64 = CryptoJS.enc.Base64.stringify(hash);
```

Other Examples

Information about other languages you may find by here:

https://www.jokecamp.com/blog/examples-of-creating-base64-hashes-using-hmacsha256in-different-languages/#csharp

11.2 External Hashing Tool

We are using SHA-256 algorithm with Base 64 Encoding for output. Below is the tool which can help you to create manually hash signature.

https://www.devglan.com/online-tools/hmac-sha256-online

Ensure SHA-256 algorithm and Base64 are chosen: