Overview

This document describes an API based on HTTP/1.1 protocol [RFC 2616].

Document version

1.1.5

Links

RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1
ISO 4217, Currency codes
ISO 8601, Date and time format

Changelog

1.0.0 (2017-12-26, ma)
documentation initialized
1.1.0 (2018-02-07, ma)
exit request added
1.1.1 (2018-05-07, ma)
exit request - add response fields
1.1.2 (2018-10-12, ma)
init request added demo flag, get log request added
1.1.3 (2018-10-22, ma)
addAmount request added
1.1.4 (2019-09-03)
getGamesSeries, getGamesInSerie requests added
1.1.5 (2020-03-20)
games list - big images

GIS

Overview

Integration data provided by GIS

1. Merchant ID
2. Merchant Key
3. Base API URL

Endpoints and Base API URL

For example: If base API URL is *https://gis.com/api/gisv1* and Endpoint is /games/lobby, then calls from integrator to GIS should be *https://gis.com/api/gisv1/games/lobby*

Request format

Query parameters should be passed with `application/x-www-form-urlencoded` content type

Response format

Default response format is `json` with `Content-Type: application/json` header

List of used HTTP codes

- `200`: OK. Everything worked as expected.
- `201`: A resource successfully created in response to a POST request. The Location header contains the URL pointing to the newly created resource.
- `204`: The request handled successfully and the response contains no body content (like a DELETE request).
- `304`: The resource was not modified. You can use the cached version.
- `400`: Bad request. This could be caused by various actions by the user, such as providing invalid JSON data in the request body, providing invalid action parameters, etc.
- `401`: Authentication failed.
- `403`: The authenticated user is not allowed to access the specified API endpoint.
- `404`: The requested resource does not exist.
- `405`: Method not allowed. Please check the Allow headers for the allowed HTTP methods.
- `415`: Unsupported media type. The requested content type or version number is invalid.
- `422`: Data validation failed (in response to a POST request, for example). Please check the response body for detailed error messages.
- `429`: Too many requests. The request was rejected due to rate limiting.
- `500`: Internal server error. This could be caused by internal program errors.

Error response

Generic error response contains a single object with following attributes:

- `name, string` exception name
- `message, string` exception message
- `code, integer, default: 0` exception code
- `status, integer` HTTP status code

Response example:

```
HTTP/1.1 404 Not Found
...
{
```

```
"name": "Not Found Exception",
"message": "The requested resource was not found.",
"code": 0,
"status": 404
}
```

Game launch flow

Games should be stored/cached on the client side after retrieval. Game could be launched in several steps according to scenario based on lobby availability.

1. Call `/init`
2. Launch game by redirecting player to the provided URL

Security

All requests should contain authorization headers (except Launch phase with player redirection).

Authorization headers

- `X-Merchant-Id`: Merchant ID provided by integration manager
- `X-Timestamp`: Request timestamp. If differ from current timestamp for more than 30 seconds - request considered expired
- `X-Nonce`: Random string
- `X-Sign`: Sign calculated with sha1 hmac

X-Sign calculation

1. Merge request array with authorization headers array
2. Sort resulting array by key in ascending order
3. Generate a URL-encoded query string from this array
4. Use sha1 hmac algorithm with Merchant Key (provided by integration manager) for signing

PHP example of the X-Sign calculation

```
$merchantKey = 'Merchant Key provided by integration manager';
$headers = [
    'X-Merchant-Id' => 'value',
    'X-Timestamp'   => time(),
    'X-Nonce'       => md5(uniqid(mt_rand(), true)),
];
$requestParams =  [
    'game_uuid' => $gameId,
    'player_id'  => $playerId,
    'credit_price'  => 1,
    'balance'  => $playerBalance,
    'currency'  => 'USD',
    'session_id'  => 'game_session_id',
    'return_url'  => $return_url,
    'exit_url'  => $exit_url,
    'language'  => 'ENG',
];
$mergedParams = array_merge($requestParams, $headers);
ksort($mergedParams);
$hashString = http_build_query($mergedParams);
$XSign = hash_hmac('sha1', $hashString, $merchantKey);
```

Example

Request:

```
GET /games
...
X-Merchant-Id: ff955b5759b3885f08cf125d4454ceb4
X-Timestamp: 1471857411
X-Nonce: e115cf0f66a645aca08225c9c1b20b80
X-Sign: 1bb7e4cd5c43f9885ba6a1758ad30fc562f88821
...
```

Games

Endpoint URL

`/games`
`[ GET / ]` Retrieving games list
You will receive games collection available for your Merchant ID

Game item fields

- `uuid: string`, Game UUID that will be used in `/init`
- `name: string`, Game name
- `image: string`, Game image url
- `image_big`: `string`, Game image big url (370 x 185 pixels)
- `type: string`, Game type
- `provider: string`, Game provider name
- `is_mobile: integer`, 1 or 0 - indicates if game used for mobile devices and should be opened in new window (not in iframe or some <div> container)

Example

Request:

```
GET /games HTTP/1.1
...
```

Response:

```
HTTP/1.1 200 OK
...
{
"items": [
      {
            "uuid": "abcd12345",
            "name": "Book of Ra",
            "image": "https://image-url.com",
            "image_big": "https://image-url.com/big_img_path/"
            "type": "Slots",
            "serie": "gaminator",
            "provider": "abcd12345",
            "is_mobile": 0
      }, {
            "uuid": "abcd12345",
            "name": "Baccarat",
            "image": "https://image-url.com",
            "type": "Baccarat",
            "serie": "live",
            "provider": "abcd12345",
            "is_mobile": 0
```

```
        }
]
}
```

Init

This action will prepare game for launch and return final url where player should be redirected to start playing.

Endpoint URL

`/init`
`[ POST / ]` Initializing game session

Request fields

- `game_uuid: string, required`, Game UUID provided in /games
- `player_id: string, required`, Unique player ID on the integrator side
- `currency: string, required`, Player currency that will be used in this game session
- `balance: double, required`, Player's balance
- `credit_price: double, required`, Credit price for game session
- `session_id: string, required`, Game session ID on the integrator side
- `return_url: string, required`, Redirect player to this url after game ends
- `exit_url: string, required,` Send `/exit` request from GIS to integrator to this url after player finish the game
- `language: string, optional`, Player language
- `demo: boolean, optional`, Set **true** to run game session in demo mode (give player 100 000 demo credits without the possibility of collect)

Response fields

`url: string`, redirect player to this url for start selected game

Example

Request:
```
POST /init HTTP/1.1
...
game_uuid=abcd12345&player_id=abcd12345&player_name=abcd12345&curre
ncy=USD& ....
```
Response:
```
HTTP/1.1 200 OK
...
{
    "url": "https://gis-url.com/endpoint"
}
```

Game launch

To launch the game redirect player to the URL returned by `/init`.

Add amount

This action will add specified credits amount to game session balance

Endpoint URL

`/addAmount`
`[ POST / ]` Add credits amount

Request fields

- `session_id: string, required`, Game session ID on the integrator side
- `amount: double, required`, Amount of credits that will be added to player's balance

Example

Request:
```
POST /addAmount HTTP/1.1
...
session_id=abc&amount=20
```
Response:
```
HTTP/1.1 200 OK
...
```

Exit

This action will reset and exit the active game.

Endpoint URL

`/exit`
`[ POST / ]` Exit the game

Request fields

- `session_id: string, required`, Game session ID on the integrator side

Response fields

- `balance: double`, Player’s balance
- `session_id: string`, Game session ID on the integrator side

Example

Request:

```
POST /exit HTTP/1.1
...
session_id=abc
```

Response:

```
HTTP/1.1 200 OK
...
```

Get Log

This action will request game logs.

Endpoint URL

`/getLog`
`[ POST / ]` get game logs

Request fields

- `session_id: string, required`, Game session ID on the integrator side
- `from: int, required`, Unix Timestamp to get log "from date"
- `to: int, required`, Unix Timestamp to get log "to date"

Response fields

list of rows:

- `date: int`, game turn date
- `game: string`, game name
- `line: int`
- `bet: int`
- `amount: double`, terminal balance
- `cp: double`, game turn denominator
- `pr: double`, game turn prize

Example

Request:

```
POST /getLog HTTP/1.1
...
session_id=abc&from=1100&to=1200
```

Response:

```
HTTP/1.1 200 OK
...
```

Get Games Series

Endpoint URL

`/getGamesSeries`
`[ GET / ]` Retrieving games series list
You will receive collection of all games series available for your Merchant ID

Response fields

list of rows `string`, game series

Example

Request:

```
GET /getGamesSeries HTTP/1.1
...
```

Response:

```
HTTP/1.1 200 OK
...
{
"items": [“gaminator1”, “global”]
}
```

Get Games In Serie

This action will request games in specified serie.

Endpoint URL

`/getGamesInSerie`
`[ POST / ]` get games in serie

Request fields

- `serie: string, required`, games serie

Game item fields

- `uuid: string`, Game UUID that will be used in `/init`
- `name: string`, Game name
- `image: string`, Game image url
- `type: string`, Game type
- `provider: string`, Game provider name
- `is_mobile: integer`, 1 or 0 - indicates if game used for mobile devices and should be opened in new window (not in iframe or some <div> container)

Example

Request:

```
POST /getGamesInSerie HTTP/1.1
…
serie=global
```

Response:

```
HTTP/1.1 200 OK
...
{
"items": [
        {
                "uuid": "abcd12345",
                "name": "Book of Ra",
                "image": "https://image-url.com",
                "type": "Slots",
                "serie": "gaminator",
                "provider": "abcd12345",
                "is_mobile": 0
        }, {
                "uuid": "abcd12345",
                "name": "Baccarat",
                "image": "https://image-url.com",
                "type": "Baccarat",
        "serie": "live",
        "provider": "abcd12345",
        "is_mobile": 0
    }
]
}
```

Integrator

Overview

Integrator should provide endpoint URL to communicate with GIS during the game session
GIS could send 1 type of calls to integrator

- Exit

Request format

All calls from GIS to integrator will be done via `POST` and parameters will be passed with `application/x-www-form-urlencoded` content type

Response format

All integrator responses should have `Content-Type: application/json` header, `json` format and `HTTP/1.1 200 OK` status code.

Security

All requests should contain authorization headers (except Launch phase with player redirection).

Authorization headers

- `X-Merchant-Id`: Merchant ID provided by integration manager
- `X-Timestamp`: Request timestamp. If differ from current timestamp for more than 30 seconds - request considered expired
- `X-Nonce`: Random string
- `X-Sign`: Sign calculated with sha1 hmac

X-Sign calculation

1. Merge request array with authorization headers array
2. Sort resulting array by key in ascending order
3. Generate a URL-encoded query string from this array
4. Use sha1 hmac algorithm with Merchant Key (provided by integration manager) for signing

PHP example of the X-Sign calculation

```
$merchantKey = 'Merchant Key provided by integration manager';
$headers = [
    'X-Merchant-Id' => 'value',
    'X-Timestamp'   => time(),
    'X-Nonce'       => md5(uniqid(mt_rand(), true)),
];

$XSign = 'Get header value'

$requestParams =  [
    'game_uuid' => 'abcd12345',
    'currency'  => 'USD',
];
$mergedParams = array_merge($requestParams, $headers);
ksort($mergedParams);
$hashString = http_build_query($mergedParams);
$expectedSign = hash_hmac('sha1', $hashString, $merchantKey);
if ($XSign !== $expectedSign) {
     throw new Exception ('Invalid sign');
}
```

Exit

When player finish the game and want to return to integrator's site, GIS will send this action before redirect to 'return_url'.

Endpoint URL

`/exit`
`[ POST / ]` Initializing game session

Request fields

- `session_id: string, required`, session ID that GIS was received from integrator with `/init` request
- `balance: double, required`, new actual player's balance after playing the game

Example

Request:
```
POST /exit HTTP/1.1
session_id=abcd12345&balance=200.00
```
Response:
```
HTTP/1.1 200 OK
...
```