Game Server Integration

Introduction

This document describes integration between *RGS* (*Remote Game Server*) and *Game Server*.

The key words `MUST`, `MUST NOT`, `REQUIRED`, `SHALL`, `SHALL NOT`, `SHOULD`, `SHOULD NOT`, `RECOMMENDED`, `MAY`, and `OPTIONAL` in this document are to be interpreted as described in RFC 2119.

Integration

Prerequisites:

- Communication `SHOULD` be done via `HTTP/1.1` and encrypted with `SSL/TLS`.
- All API requests `MUST` be passed with `Content-Type: application/json`.
- All API responses `MUST` be passed with `Content-Type: application/json`.
- Any additional response or request data, parameters, fields or codes not described in documentation `SHOULD NOT` be used.
- Character encoding utf-8 `SHOULD` be set to `UTF-8`
- All time and date fields `SHOULD` be in `UTC` timezone

Response codes

All responses return appropriate response's codes.

Code	Status
`2XX`	Successful
`4XX`	Application Error
`5XX`	Server Error

Port

Server should listen on the port specified in environment variable `PORT`.

Errors

All errors follow unified structure.

Field `message` on non-production environments gives more verbose description of the problem to help troubleshooting. On production environment it is replaced with very generic statement for security reasons.

Field `code` is the same across all environments and should be used to display localised message to the end user. Available codes: `APPLICATION_ERROR`, `SERVER_ERROR`, `PLAYER_UNAUTHORIZED`, `SERVER_UNAUTHORIZED`, `INSUFFICIENT_FUNDS`, `LOSS_LIMIT`, `UNKNOWN`.

```
{
  "error": {
    "message": "Error description",
    "code": "APPLICATION_ERROR"
```

```
    }
  }
```

API

Path: `/api/games` **Method**: `GET` **Authorization**: `NO`

Get list of games

Response

Name	Type		Example	Description
N/A	`{[key:string]: string[]}`	`REQUIRED`	`{myProvider: ["game1", "game2", "game3"]}`	List of games

Path: `/api/games/{game}/cheats` **Method**: `GET` **Authorization**: `NO`

Returns list of cheats per action.

Query Parameters

Name	Type		Example	Description
`game`	`string`	`REQUIRED`	`game1`	Game Id

Response

Name	Type		Example	Description
N/A	`object`	`REQUIRED`	`{main: ["win", "bigWin"], bonus: ["feature1"]}`	List of cheats per action

Path: `/api/games/{game}/config?variant={variant}` **Method**: `GET` **Authorization**: `NO`

Returns game config for specific variant

Query Parameters

Name	Type		Example	Description
`game`	`string`	`REQUIRED`	`game1`	Game Id
`variant`	`string`	`OPTIONAL`	`lowRtp`	variant Id

Response

Name	Type		Example	Description
N/A	`object`	`REQUIRED`	`{paytable: 123}`	Game config to be returned to Game Client

Path: `/api/games/{game}/bets` **Method**: `GET` **Authorization**: `NO`

Returns configuration and other information

Query Parameters

Name	Type		Example	Description
`game`	`string`	`REQUIRED`	`game1`	Game Id

Response

Name	Type		Example	Description
`bets`	`Record<string, BetConfig>`	`REQUIRED`	`{main: {available: [0.1, 1, 10], maxWin: 100, default: 1, coin: 10}, bonus: {available: {min: 100, max: 500, step: 25}, maxWin: 10, default: 100, coin: 100}}`	Bet config per action. Base bet action `SHOULD` be called `main`. See BetConfig

Example:

```
{
  "bets": {
    "main": {
      "available": [
        1,
        2,
        3
      ],
      "default": 2,
      "maxWin": 20000,
      "coin": 1
    },
    "bonus": {
      "available": [
        100,
        200,
        300
      ],
      "default": 200,
      "maxWin": 20,
      "coin": 100
    }
  }
}
```

Path: `/api/games/{game}/play` **Method**: `POST` **Authorization**: `NO`

Generates new wager

Body

Name	Type		Example	Description
`game`	`string`	`REQUIRED`	`mySlot`	Game Id

Body

Name	Type		Example	Description
`bet`	`number`	`REQUIRED`	`0.1`	Bet amount
`coin`	`int`	`REQUIRED`	`10`	Bet coin
`action`	`string`	`REQUIRED`	`main`	Bet action
`cheat`	`string`	`OPTIONAL`	`bigWin`	Cheat
`params`	`object`	`OPTIONAL`	`{extraValue: 1}`	Extra parameters for game server
`state`	`object`	`OPTIONAL`	`{myCollection: 3}`	Player's persistent state
`variant`	`string`	`OPTIONAL`	`lowRtp`	Game's variant
`betLimits`	`BetLimits`	`REQUIRED`		See BetLimits

Response

Name	Type		Example	Description
N/A	`Wager & { feed }`	`REQUIRED`		See Wager decorated with game feed to store

Path: `/api/games/{game}/action` **Method**: `POST` **Authorization**: `NO`

Determines next action in case unfinished round with `next` options available.

Body

Name	Type		Example	Description
`game`	`string`	`REQUIRED`	`mySlot`	Game Id

Body

Name	Type		Example	Description
N/A	`Wager`	`REQUIRED`		See Wager

Response

Name	Type		Example	Description
action	`string`	`REQUIRED`	`{action: "green"}`	Next action

Path: `/api/games/{game}/validate` **Method**: `POST` **Authorization**: `NO`

Optional validation used only for side bets and bets marked with `validate: true`.

It `MUST NOT` return `true` by default as that can lead to unwanted side bets.

Body

Name	Type		Example	Description
`bet`	`number`	`REQUIRED`	`0.1`	Bet amount
`coin`	`int`	`REQUIRED`	`10`	Bet coin
`action`	`string`	`REQUIRED`	`main`	Bet action
`params`	`object`	`OPTIONAL`	`{extraValue: 1}`	Extra parameters for game server
`state`	`object`	`OPTIONAL`	`{myCollection: 3}`	Player's persistent state
`variant`	`string`	`OPTIONAL`	`lowRtp`	Game's variant
`betLimits`	`BetLimits`	`REQUIRED`		See BetLimits

Response

Name	Type		Example	Description
`valid`	`boolean`	`REQUIRED`	`true`	Indicates if validation was successful

Path: `/api/games/{game}/evaluate` **Method**: `POST` **Authorization**: `NO`

Optional evaluation of the given round used for specific platform features. It is necessary to implement if the game aims to incorporate these functionalities.

Body

Name	Type		Example	Description
`type`	`string`	`REQUIRED`	`achievement`	Evaluate request type
`wagers`	`IWager[]`	`REQUIRED`		List of round's wagers. See Wager
`data`	`any`	`OPTIONAL`	`{jackpotWon: wagers[0].win > 0}`	Data specific for a given evaluate request type

Response

Name	Type		Example	Description
N/A	any	`REQUIRED`	`true`	Evaluation result specific to a given request type

Portugal regulatory evaluation

To ensure compliance with regulations for game launches in Portugal, it is mandatory to include `sm_result` and `descr_ap` strings that describe each game round (other "AJOG file" entries might also be included, but are optional). The platform can transmit these values to Portuguese operators, provided the `regulatory-pt` evaluation is implemented and the data adheres to the following format:

Body

Name	Type	Value
`type`	`string`	`regulatory-pt`
`wagers`	`IWager`	

Response

Name	Type	Example
N/A	`{ "sm_result"?: string, "descr_ap"?: string, [key: string]: string }`	`{ "sm_result": "0:1;3;1;1;1#5;0;7;2;1#5;0;5;2;2#", "descr_ap": "60GoldenCoinsPro" }`

Path: `/api/criticalFileChecksum?criticalFilePath={criticalFilePath}` **Method**: `GET` **Authorization**: `NO`

Returns checksum of the file specified by the path within the file system. It's RECOMMENDED to use SHA1 algorithm. It's RECOMMENDED that path is relative to the working directory of the game server process.

Query Parameters

Name	Type		Example	Description
`criticalFilePath`	`string`	`REQUIRED`	`games/my-game/math/calculateWins.ts`	file to calculate checksum on

Response

Name	Type		Example	Description
checksum	`string`	`REQUIRED`	`3d10b225b47322829cdb6b9acd69921ffd3a1e87`	sha1 checksum of the specified file

Path: `/health` **Method**: `GET` **Authorization**: `NO`

Health check

Response

If application is initialized correctly it should respond with plain text `OK` with status `200`.

After process receives `SIGTERM` signal it should respond with plain text `QUITING` with status `500`.

If any other case (i.e. application not responding) it should respond with plain text `ERROR` with status `500`.

Wager

Name	Type		Example	Description
`data`	`any`	`OPTIONAL`	`[{"spin": 1}, {"spin": 2}]`	Game outcome
`state`	`any`	`OPTIONAL`	`{"collection": 10}`	Player's game state persistent between rounds
`win`	`number`	`REQUIRED`	`16.45`	Cash win in Player's currency
`next`	`string[]`	`OPTIONAL`	`["gamble", "take"]`	Next actions for multi-wager rounds. If response contains `next` array then in next play request `action` should be set to one of the array's elements

BetConfig

Name	Type		Example	Description
`available`	`number[]` or `{min: number, max: number, step: number}`	`REQUIRED`	`[0.01, 0.5, 1, 5, 10]` or `{min: 100, max: 500, step: 25}`	Available bets in base currency
`default`	`number`	`REQUIRED`	`0.5`	Default bet in base currency
`maxWin`	`number`	`REQUIRED`	`25`	Maximum theoretical win
`coin`	`number`	`REQUIRED`	`25`	Coin passed to play function. Bet is not affected by coin value. It is only used to calculate win multiplier of base bet (`main` action). So important to keep proportion i.e. coin of `main` is 1 and coin of `bonus` 100 (if `bonus` is x100 base bet).
`validate`	`boolean`	`OPTIONAL`	`true`	If `true` it custom validation is called also for initial bets

BetLimits

Name	Type		Example	Description
`minBet`	`number`	`REQUIRED`	`1`	Min bet in player's currency
`maxBet`	`number`	`REQUIRED`	`100`	Max bet in player's currency
`maxBonusBet`	`number`	`REQUIRED`	`1000`	Max bonus in player's currency
`maxExposre`	`number`	`REQUIRED`	`10000`	Max exposure in player's currency
`currencyRate`	`number`	`REQUIRED`	`5`	Currency rate to base currency (typically `eur`)

Name	Type		Example	Description
`exchangeRate`	`number`	`REQUIRED`	`4.21342`	Exchange rate (from Currency feeds) to base currency (typically `eur`)
`currencyDecimals`	`number`	`REQUIRED`	`2`	Amount of decimal places for the currency
`currencyUnit`	`number`	`REQUIRED`	`0.01`	Minimal monetary value of a currency (e.g. 0.01 for "cent")