Changelog

Version	**Description**
1.1.0	Added AuthToken endpoint.
	Added GameLink endpoint.
1.1.1	Added Fetch Balance, Withdraw, Deposit, Rollback endpoint for operator

	reference and implement api.
1.1.2	Added target_transaction_id parameter to Rollback api.
1.1.3	Added error list for wallet api. Added note for protocol description.
1.1.4	Fix rollback error list, delete the 10208 and 10209 and then added 10210 plus 10211.
1.1.5	Added retry strategy.
1.1.6	Added exitUrl parameter to GameLink api.
1.1.7	Added error code 10110 to GameLink api to remind the game is maintained.
1.1.8	Update Retry Strategy content. "If a transaction API call fails, we will retry immediately five times, but only the deposit API applies to the Retry Strategy mentioned below."
1.1.9	Update description of Retry Strategy.
1.1.91	Added mode parameter to GameLink api.
1.1.92	Remove the currency parameter of Withdraw and Deposit API.
1.1.93	Added the JP Deposit API.
1.1.94	Added the roundended parameter to Withdraw , Deposit and JP Deposit API.
1.1.95	Added jp_win parameter to Deposit API and added jpcontrib parameter to Withdraw API.
1.1.96	Update the comment of the jpcontrib parameter in Withdraw API.
1.1.97	Update the jpcontrib to the jp_contrib for consistency.

1.1.98	Added a comment of GameLink API about the parameter exitUrl.
1.1.99	Added quitHide parameter to GameLink API.
1.1.100	Update the withdraw jp_contrib to the jpcontrib

Operator identification

The word platform mentioned in the doc means Casino operator which one integrates with us and launches our game for their user. We will give each platform operator a set of verification parameters (apikey and platformId) after filling in and returning the application form.

apikey

The apikey usually is string plus case sensitive and always is put in the request header.

Please do not disclose the apikey to anyone at any time!

platformId

It's like the name of the platform operator.

The platformId is used for calling Auth_token and must be provided in the endpoint of api.

sessionId (from operator)

The sessionId comes from the platform operator and is used for Auth_token calling first.

After that, it would also be required for GameLink api calling.

We will also bring it into the platform operator’s wallet api request, so that the platform operator can do identity verification, etc.

Protocol description

- The common header contains two parameters, one is content-type and one is content-length, because our post uses raw-data form.
- Content-type always is “application/json”.

Auth_token

The api will authenticate for the platform before calling GameLink api.

For token of response, the token should be one-time and will be void after it is used to request GameLink api.

If an account does not exist, the api will auto register here.

Request Endpoint
HTTP POST https://{host}/agent/specify/{platformId}/authenticate/auth_token

HTTP HEADERS

Parameter	*Type*	*Required*	*Description*
apiKey	String	Required	Provided apikey.

Request

Parameter	*Type*	*Required*	*Description*
id	String	Required	Provided platformId.
account	String	Required	Unique player identifier.
currency	String	Required	ISO 4217
sessionId	String	Required	Platform user`s token or session.
channel	String	Optional	Accept empty value.

Request Example

```
{
    “id”:”test_platform”,
    “account":"test_account",
    “currency”:”TWD”,
    “sessionId”:”eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
…”,
    “channel":""
}
```

Responses

Status	Code	Body	Description
200	none	{ “token”:{token} }	Token is used for game login.
200	10100	{ “errorCode”:10100, “message”: ”Server is not ready!” }	Server error.
	10101	{ “errorCode”:10101, “message”: ”Post data is empty!” }	Post data is empty.
	10102	{ “errorCode”:10102, “message”: ”Post data is invalid!” }	Post data is missing some necessary parameters.

	10103	{ “errorCode”:10103, “message”: ”Account is invalid!” }	Account is invalid!
	10104	{ “errorCode”:10104, “message”: ”Platform is not exist!” }	Platform id is wrong.
	10105	{ “errorCode”:10105, “message”: ”Authenticate failed!” }	Apikey is wrong or empty in the header.
	10106	{ “errorCode”:10106, “message”: ” Currency numbers do not exist or are not supported on your platform.” }	Currency numbers do not exist or are not supported on your platform.
	10107	{ “errorCode”:10107, “message”: ”Session id exists!” }	Session id exists.

GameLink

The api is like the game launcher in that it will respond to game links. Please call the api in ten minutes when called Auth_token api done.

Because the sessionId is attached to the platform info, so the platformId will be not necessary in the request body here.

Request Endpoint
HTTP POST https://{host}/agent/specify/{platformId}/gameLink/link

HTTP HEADERS

Parameter	***Type***	***Required***	***Description***
apiKey	String	Required Optional	Be Required if mode is “real_play”. Provided apikey.
token	String	Required Optional	Be Required if mode is “real_play”. Provided token from auth_token response.

Request

Parameter	***Type***	***Required***	***Description***
mode	String	Optional	The mode of the gameplay, and its default value is “real_play”. “real_play” | “demo”
language	String	Required	Our supported language at the last second page.
gameId	String	Required	Provided game id.
account	String	Required Optional	Be Required if mode is “real_play”. Unique player identifier.
sessionId	String	Required Optional	Be Required if mode is “real_play”. Platform user`s token which is sent in AuthToken.
exitUrl	String	Optional	Players will be redirected to this url when they quit the game or encountered some error. Default url comes from operator’s requirement, or we will decide by ourselves. Example: ● exitUrl=”close” ○ close game window ● exitUrl=”event” ○ It will trigger a postMessage to the top window with the string “{\”event\”:\”exit\”}” when the player presses the home button. ● exitUrl=”https://example.com” ○ redirect to

			https://example.com ● exitUrl="" ○ use default url ○ if the default url is empty, and the behavior of the quit will be closing the game window. TIPs: 1. Equal signs need to be converted to '%3D' in the parameters of the URL.
quitHide	Integer	Optional	The gameplay will hide the quit button once quitHide is 1, and default is 0.

Request Example

```
{
  “mode”: “real_play”,
  “account": "test_account",
  "sessionId":   "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "language": “th-TH“,
  "gameId": “123",
  “exitUrl”: “”
}
```

Response

Status	***Code***	***Body***	***Description***

200	none	{
		  “url”:{url}
		}
			Url is used in game launches.

200	10100	{
		  “errorCode”:10100,
		  “message”:
		  ”Server is not ready!”
		}
			Server error.

	10101	{
		  “errorCode”:10101,
		  “message”:
		  ”Post data is empty!”
			Post data is empty.

		}	
	10102	{ "errorCode":10102, "message": "Post data is invalid!" }	Post data is missing some necessary parameters.
	10103	{ "errorCode":10103, "message": "Account is invalid!" }	Account is invalid!
	10104	{ "errorCode":10104, "message": "Platform is not exist!" }	Platform id is wrong.
	10105	{ "errorCode":10105, "message": "Authenticate failed!" }	Token or apikey is wrong or empty in the header.
	10108	{ "errorCode":10108, "message": "Session id is wrong!" }	Session id is wrong.
	10110	{ "errorCode":10110, "message": "The game is currently maintained!" }	The game is currently maintained!
	10204	{ "errorCode":10204, "message": "Account does not exist!" }	Account does not exist!

Fetch Balance (Implemented By Operator)

FetchBalance call is implemented by the operator. POST request sample is made to provide Fetch Balance url with parameters below.

Usually, this is to get the player balance when the player enters the game.

Request Endpoint
HTTP POST https://domain/fetchBalance

HTTP HEADERS

Parameter	***Description***
apiKey	Operator provided apikey.

Parameter	***Type***	***Required***	***Description***
account	String	Required	Unique player identifier.
sessionId	String	Required	Platform user`s token which is sent in AuthToken.

Request

Request Example

```
{
    “account”:”test_account”,
    “sessionId”:”eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
…”
}
```

Responses

Status	***Code***	***Body***	***Description***

200	none	{ “balance”:{balance} }	Response balance is the user’s balance amount.
200	10100	{ “errorCode”:10100, “message”: ”Server is not ready!” }	Server error.
	10101	{ “errorCode”:10101, “message”: ”Post data is empty!” }	Post data is empty.
	10102	{ “errorCode”:10102, “message”: ”Post data is invalid!” }	Post data is missing some necessary parameters.
	10105	{ “errorCode”:10105, “message”: ”Authenticate failed!” }	SessionId or apikey is wrong or empty.
	10204	{ “errorCode”:10204, “message”: ”Account does not exist!” }	Account does not exist!

Withdraw (Implemented By Operator)

Withdraw call is implemented by the operator. POST request sample is made to provide Withdraw url with parameters below.

This is caused by a player placing a bet in a game.

Request Endpoint
HTTP POST https://domain/withdraw

HTTP HEADERS

Parameter	*Description*
apiKey	Operator provided apikey.

Parameter	*Type*	*Required*	*Description*
account	String	Required	Unique player identifier.
sessionId	String	Required	Platform user`s token which is sent in AuthToken.
amount	Decimal	Required	Amount to withdraw.
game_id	String	Required	Provided game id.
round_id	String	Required	The game round id.
transaction_id	String	Required	The transaction id of this transaction.
jpcontrib	Decimal	Required	Contributor to the jackpot game and the source of the contributor is our RTP. Please do not deduct this amount to the player's balance, It is just used to record.
roundended	Bool	Required	If it is true, and it means the round ends. And it will be true only when Deposit API calling.

Request

Request Example

```
{
        "account":"test_account",
        "sessionId":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
        "amount":100,
        "game_id":"123",
        "round_id":"round123456789",
        "transaction_id":"tran123456789",
        “roundended”:false
}
```

Responses

Status	***Code***	***Body***	***Description***
200	none	{ “transaction_id”:”tran123456789”, “balance”:100.00 }	Response balance is the user’s balance amount.
200	10100	{ “errorCode”:10100, “message”: ”Server is not ready!” }	Server error.
	10101	{ “errorCode”:10101, “message”: ”Post data is empty!” }	Post data is empty.
	10102	{ “errorCode”:10102, “message”: ”Post data is invalid!” }	Post data is missing some necessary parameters.
	10105	{ “errorCode”:10105, “message”: ”Authenticate failed!” }	SessionId or apikey is wrong or empty.

	10106	{ “errorCode”:10106, “message”: ”Currency numbers do not exist!” }	Currency numbers do not exist!
	10109	{ “errorCode”:10109, “message”: ”Game not found!” }	Game not found!
	10201	{ “errorCode”:10201, “message”: ”Warning value must not be less 0.” }	Warning value must not be less 0.
	10203	{ “errorCode”:10203, “message”: ”Balance value error. Insufficient balance” }	Balance value error. Insufficient balance
	10204	{ “errorCode”:10204, “message”: ”Account does not exist!” }	Account does not exist!
	10208	{ “errorCode”:10208, “message”: ”Transaction id exists!” }	Transaction id exists!
	10209	{ “errorCode”:10209, “message”: ”Round id exists!” }	Round id exists!

Deposit (Implemented By Operator)

Deposit call is implemented by the operator. POST request sample is made to provide a Deposit url with parameters below.

Normally this is caused by a player winning in a game but could also be related to a promotional free round pay-out.

The operator can also add the optional parameter “jp_win“ for jackpot winning, and please do not implement JP_Deposit API if the operator chooses FS_Deposit and Deposit API to deal with the jackpot.

Request Endpoint
HTTP POST https://domain/deposit

HTTP HEADERS

Parameter	***Description***
apiKey	Operator provided apikey.

Request

Parameter	***Type***	***Required***	***Description***
account	String	Required	Unique player identifier.
sessionId	String	Required	Platform user`s token which is sent in AuthToken.
amount	Decimal	Required	Amount to deposit.
game_id	String	Required	Provided game id.
round_id	String	Required	The game round id.
transaction_id	String	Required	The transaction id of this transaction.
jp_win	Decimal	Optional	Jackpot wins.
roundended	Bool	Required	If it is true, and it means the round ends.

			And it will be true only when Deposit API calling.

Request Example

```
{
        "account":"test_account",
        "sessionId":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
        "amount":100,
        "game_id":"123",
        "round_id":"round123456789",
        "transaction_id":"tran123456789",
        “jp_win”: 0.0000
        “roundended”:true
}
```

Responses

Status	**Code**	**Body**	**Description**
200	none	{ “transaction_id”:”tran123456789”, “balance”:100.00 }	Response balance is the user’s balance amount.
200	10100	{ “errorCode”:10100, “message”: ”Server is not ready!” }	Server error.
	10101	{ “errorCode”:10101, “message”: ”Post data is empty!” }	Post data is empty.
	10102	{ “errorCode”:10102, “message”: ”Post data is invalid!” }	Post data is missing some necessary parameters.

	10105	{
		“errorCode”:10105,
		“message”:
		”Authenticate fail!”
		}
			SessionId or apikey is wrong or empty.

	10109	{
		“errorCode”:10109,
		“message”:
		”Game not found!”
		}
			Game not found!

	10201	{
		“errorCode”:10201,
		“message”:
		”Warning value must not be less 0.”
		}
			Warning value must not be less 0.

	10204	{
		“errorCode”:10204,
		“message”:
		”Account does not exist!”
		}
			Account does not exist!

	10207	{
		“errorCode”:10207,
		“message”:
		”The balance limit has been reached.”
		}
			The balance limit has been reached.

	10208	{
		“errorCode”:10208,
		“message”:
		”Transaction id exists!”
		}
			Transaction id exists!

	10209	{
		“errorCode”:10209,
		“message”:
		”Round id exists!”
		}
			Round id exists!

JP_Deposit (Implemented By Operator)

JP_Deposit call is implemented by the operator. POST request sample is made to provide a Jackpot Deposit url with parameters below.

Normally this is caused by a player winning jackpot in a game but could also be related to a promotional free round pay-out.

Request Endpoint
HTTP POST https://domain/jp_deposit

HTTP HEADERS

Parameter	***Description***
apiKey	Operator provided apikey.

Request

Parameter	***Type***	***Required***	***Description***
account	String	Required	Unique player identifier.
sessionId	String	Required	Platform user`s token which is sent in AuthToken.
jp_win	Decimal	Required	Amount to deposit.
game_id	String	Required	Provided game id.
round_id	String	Required	The game round id.
transaction_id	String	Required	The transaction id of this transaction.
roundended	Bool	Required	If it is true, and it means the round ends. And it will be true only when Deposit API calling.

Request Example

```
{
    "account":"test_account",
    "sessionId":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "jp_win":1000000000.0000,
    "game_id":"123",
    "round_id":"round123456789",
    "transaction_id":"tran123456789",
    “roundended”:true
}
```

Responses

Status	***Code***	***Body***	***Description***
200	none	{ “transaction_id”:”tran123456789”, “balance”:100.00 }	Response balance is the user’s balance amount.
200	10100	{ “errorCode”:10100, “message”: ”Server is not ready!” }	Server error.
	10101	{ “errorCode”:10101, “message”: ”Post data is empty!” }	Post data is empty.
	10102	{ “errorCode”:10102, “message”: ”Post data is invalid!” }	Post data is missing some necessary parameters.
	10105	{ “errorCode”:10105, “message”: ”Authenticate fail!” }	SessionId or apikey is wrong or empty.
	10109	{	Game not found!

		“errorCode”:10109, “message”: ”Game not found!” }	

	10201	{
		“errorCode”:10201,
		“message”:
		”Warning value must not be less 0.”
		}
			Warning value must not be less 0.

	10204	{
		“errorCode”:10204,
		“message”:
		”Account does not exist!”
		}
			Account does not exist!

	10207	{
		“errorCode”:10207,
		“message”:
		”The balance limit has been reached.”
		}
			The balance limit has been reached.

	10208	{
		“errorCode”:10208,
		“message”:
		”Transaction id exists!”
		}
			Transaction id exists!

	10209	{
		“errorCode”:10209,
		“message”:
		”Round id exists!”
		}
			Round id exists!

Rollback (Implemented By Operator)

Rollback call is implemented by the operator. POST request sample is made to provide Rollback url with parameters below.

Used to rollback a previous withdraw. Deposit can not be rolled back.

Request Endpoint
HTTP POST https://domain/rollback

HTTP HEADERS

Parameter	***Description***
apiKey	Operator provided apikey.

Request

Parameter	***Type***	***Required***	***Description***
account	String	Required	Unique player identifier.
sessionId	String	Required	Platform user`s token which is sent in AuthToken.
game_id	String	Required	Provided game id.
currency	String	Required	ISO 4217
round_id	String	Required	The game round id is used for rollback.
transaction_id	String	Required	The transaction id of this transaction.
target_transaction_id	String	Required	The transaction id is used for rolling back.

Request Example

```
{
        "sessionId":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
        "account":”test”,
        "game_id":"123",
        "round_id":"round123456789",
        “transaction_id”:"new_tran123456789",
        “target_transaction_id”:”tran123456789”
}
```

Responses

Status	***Code***	***Body***	***Description***
200	none	{ “transaction_id”:”new_tran 123456789”, “balance”:100.00 }	Response balance is the user’s balance amount.
200	10100	{ “errorCode”:10100, “message”: ”Server is not ready!” }	Server error.
	10101	{ “errorCode”:10101, “message”: ”Post data is empty!” }	Post data is empty.
	10102	{ “errorCode”:10102, “message”: ”Post data is invalid!” }	Post data is missing some necessary parameters.
	10105	{ “errorCode”:10105, “message”: ”Authenticate failed!” }	SessionId or apikey is wrong or empty.

	10109	{ “errorCode”:10109, “message”: ”Game not found!” }	Game not found!
	10204	{ “errorCode”:10204, “message”: ”Account does not exist!” }	Account does not exist!
	10210	{ “errorCode”:10210, “message”: ”Target transaction id not found!” }	Target transaction id not found!
	10211	{ “errorCode”:10211, “message”: ”Transaction id not found!” }	Transaction id not found!
	10212	{ “errorCode”:10212, “message”: ”Round was not found!” }	Round was not found!

Error List for Wallet Api

Error response format:

“errorCode”:”{errorCode}”

“message”:”{message}”

ErrorCode	***Message***
10100	Server error.
10101	Post data is empty.
10102	Post data is missing some necessary parameters.
10105	SessionId or apikey is wrong or empty.
10106	Currency numbers do not exist!
10109	Game not found!
10200	Please try it later.
10201	Warning value must not be less 0.
10203	Balance value error. Insufficient balance
10204	Account does not exist!
10207	The balance limit has been reached.
10208	Transaction id exists!
10209	Round id exists!
10210	Target transaction id not found!
10211	Transaction id not found!
10212	Round was not found!

Retry Strategy

The only thing that the retry system should do is re-request the specified API until its response success or the deadline is reached.

- If a transaction API call fails, we will re-request immediately five times, but only the Deposit API will be sent to the retry system.
- Our process of calling API is always to do the next action after its response.
- The operator can provide us with the retry rules regarding the following format.
- We can negotiate if the operator wants to change or add the process.

The API calling in the whole spinning process is divided into two step (retry system in the second step):

1. It should be re-request immediately 5 times not only Both The Withdraw API but also the Deposit API when one of them fails to respond. In this step, the player is still waiting for a spinning result until the five API calls are completed, and then when the Deposit still response fails, it will proceed to the next step.
2. In the second step, the failed deposit request will be sent to the retry system that relies on our Retry Strategy and the player will also receive an error message return here. Please refer to the description below for details.

Normally our retry system is divided into two phases.

1. In the first phase, we would quickly re-request 5 times in case of temporary glitch.
2. In the second phase, we retry every 10 minute until the end of 48 hours, and then when 48 hours has passed, it will be abandoned.

The format of the retry parameters:

- quick_retry_phase : the count for the first phase to retry.
- schedule_retry_phase : the interval duration between two retries.
- deadline : the duration of the whole retry process.

Default:

```
{
        "quick_retry_phase":5,
        "schedule_retry_phase":300,  // (seconds)
        "deadline":172800   // (seconds)
}
```

Supported Languages

This our game supports language code now.

Description	***LanguageCode***
English (United States)	en-US
Traditional Chinese (繁體中文)	zh-TW
Simplified Chinese (简体中文)	zh-CN
Turkish (Türkçe)	tr-TR
Vietnamese (Việt Ngữ)	vi-VN
Korean (한국어)	Ko-KR
Japanese (にほんご)	ja-JP
Thai (ภาษาไทย)	th-TH

List of common error codes

Code	***Description***
10100	Server is not ready!
10101	Post data is empty!
10102	Post data is invalid!
10103	Account is invalid!
10104	Platform is not exist!
10105	Authenticate failed!
10106	Currency numbers do not exist or are not supported on your platform.
10107	Session id exists!
10108	Session id is wrong!
10110	The game is currently maintained!
10204	Account does not exist!