Casino Integration API 3.1.3 rev1.08

Help

Exported on 03/17/2023

Table of Contents

BETCONSTRUCT

Casino Integration for Games API 3.1.3 Quick Reference Document Rev 1.08

Revision History

Rev #	Revision Date	Description	Approved by	Corrected by
1.0	29/07/2016	Proofreading. Ch. Basic Concept added	Geham Yegoryan	Igor Mouhsian
1.01	1/08/2016	Ch.3 Getting Skins added	Ashot Boyakhchyan	Igor Mouhsian
1.02	9/08/2016	Ch. 2 modified and united w/ Ch. 3	Ashot Boyakhchyan	Igor Mouhsian

1.03	12/10/2017	Ch. 2 was written a new. Ch 3 was created instead	E. Mkrtchyan	Igor Mouhsian
1.04	22/02/2018	Global replacement token varchar(250) by varchar(50) – bugfix in prod. rel 3.1.2	E. Mkrtchyan	Igor Mouhsian
1.05	04/05/2018	Ch 2, Front End Integration	E. Mkrtchyan	Igor Mouhsian
1.06	June 4, 2020	A new point added to the Important notes in 1.1.6 Rollback	Artyom Dalibaltyan	Anahit Korkotyan
1.07	05/11/2022	Changes in 1.1.4, in Ch. 2; Removed 1.1.7 RefreshToken, 1.2.6 BonusFailed and 1.2.9 BonusAccepted; added AddClientToBonus to 1.2	Lilit Hayrapetyan	Gohar Kocharyan
1.08	11.08.2022	Changes: 1.2 Error description title remove, Change ordering in 1.2, changes in Players, AddClientToBonus, BonusActivated	Lilit Hayrapetyan	Gohar Kocharyan

Contents

Basic Concept

Remote Gaming Server (RGS) is designed to support integration with third-party partner systems (Operators). Operators usually have their own site (casino, poker, sports betting, etc.) with existing integrations with other game providers and wish to integrate RGS into their own site.
Operators wish to manage their players and balances/wallets separately in their own systems and have them seamlessly integrated. For this purpose, the Operators have to implement the Partner API to be called by RGS. This API is a service contract with a collection of methods/calls with request and response messages. The format and details of the messages are described below in this document. The calls are accomplished during normal processing of RGS when needed to exchange or notify some information to Operator. The integration consists of two parts: web (iFrame) and back-end.
Web integration is used to provide UI with modules of games. The back-end integration provides sending and receiving messages between RGS and Operator (for example, placed bets, winning information, etc.).
All the back-end calls are verified by PublicKey which is the Sha256 value of the message body and Shared Key. The message body is the JSON string value of the request object (properties are ordered by their names).

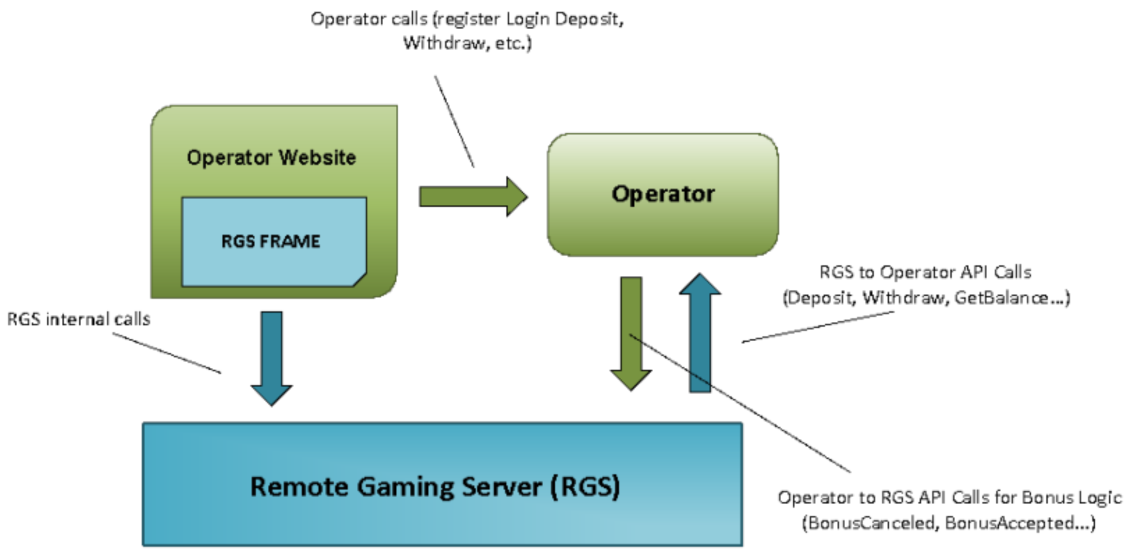
Operator calls (register Login Deposit, Withdraw, etc.)
Operator Website
RGS FRAME
Operator
RGS to Operator API Calls (Deposit, Withdraw, GetBalance...)
RGS internal calls
Remote Gaming Server (RGS)
Operator to RGS API Calls for Bonus Logic (BonusCanceled, BonusAccepted...)

Fig 1 BC Casino Integration for Games concept

Integration Workflow

This document contains detailed information, which allows the operator to integrate with Betconstruct Remote Gaming Server.
The Integration consists of 2 parts: implementing Front End and Back End.
With Front End integration, the Operator is allowed to get the Product, Game list, and initialize the Game launching process.
Back End integration manages the game-playing process.

All the backend calls are verified by PublicKey which is the Sha256 value of the message body and Shared Key. The message body is the JSON string value of the request object (properties are ordered by their names).

1. Back End Integration (Single Wallet)

1.1 Wallet Integration

Message protocol: http/https POST
Message format: JSON
Security: Shared key security and IP whitelisting

In each call of API, the PublicKey parameter is presented, which is the Sha256 hash of the message body and Shared Key.

1.1.1 Authentication

- Authenticates a user in the game by username and password (Downloadable client).
- Authenticates a user in the game by Token (integration with iFrame).

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of the operator (provided by BetConstruct)	Int32
Token	Initial player's token to launch the protocol	Varchar(50)
ClientRfid	Players RFId Code (Optional for Land Based Operations)	VarChar(250)
UserName	The UserName of the player is used when the Token is empty. Authentication with username and password is used mainly in case the games have downloadable clients. (See FAQ for details)	Varchar(50)
Password	The password of the player is used when the Token is empty. Authentication with username and password is used mainly when the games have downloadable clients. (See FAQ for details)	Varchar(50)
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response Parameters

Parameter	**Purpose**	**Validation**

OperatorId	Id of the operator (provided by BetConstruct)	Int32
Name	Name of the authenticated user	Varchar(50)
NickName	The nickname of the authenticated user. The NickName must be unique for each user	Varchar(50)
UserName	Username of the authenticated user. The UserName must be unique for each user	Varchar(50)
Token	This token should not be confused with an input token. This token is a session token only, a unique identifier generated by the operator to identify the session's interactions	Varchar(50)
TotalBalance	The available balance of authenticated user's account	Decimal(18,4)
Gender	Gender of the authenticated user	Boolean
Currency	Currency of authenticated user	Char(3)
Country	Country of the authenticated user	Char(2)
PlayerId	User Id of the authenticated user. User Id must be unique for each user	Int32
UserIP	User IP of the authenticated user	Varchar(30)
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)

1.1.2 GetBalance

This method returns the available balance into the player's account. It is called when games are loaded and while finishing uncompleted game rounds. It may also be called during the other events. It returns an object of type **GetBalanceOutput**.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerId	Id of player for who the balance is retrieved for	Int32
Token	A unique identifier is generated by the operator to identify the session's interactions	Varchar(50)
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response Parameters

Parameter	Purpose	Validation
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
PlayerId	Id of player for whom the balance is retrieved	Numeric
TotalBalance	Player's current balance amount	Decimal(18,4)
BonusWin	Money won by player during bonus (optional)	Decimal(18,4)
BonusMoney	Bonus wallet money (optional)	Decimal(18,4)
FrozenMoney	Frozen money during bonus (optional)	Decimal(18,4)
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)

- TotalBalance - includes FrozenMoney and Withdrawable money
- BonusWin, BonusMoney, and FrozenMoney fields must be available if the player has a bonus (see more in 1.1.3 Withdraw and 1.1.4 Deposit call descriptions)

1.1.3 Withdraw

This method withdraws money from the player's account and returns the transaction reference and player's account balance after the transaction was made. This method is used to place a bet. It does return an object of type **WithdrawOutput**.

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerId	Id of player who the withdrawal is made for	Int32
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)
WithdrawAmount	Amount to withdraw from player's account	Decimal(18,4)
Currency	Currency of withdrawal transaction	Char(3)
GameId	Id of game	Int32
RGSTransactionId	A unique key to indicate a specific financial activity. This key guarantees that transactions are processed only once.	Int64
TypeId	Describes the reason for withdrawal. You can see the list of possible reasons in the section 1.1.7	Int32
BonusDefId	Optional parameter for Bonus logic	int?
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response Parameters

Parameter	**Purpose**	**Validation**

HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
PlayerId	Id of player who the withdrawal is made for	Int32
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)
TotalBalance	Player's balance amount after the transaction	Decimal
PlatformTransactionId	TransactionId of the platform for the withdrawal operation	Int64

- If the BonusDefId is available, then money should be taken off the wallets in the following order: 1. BonusWin2. FrozenMoney3. BonusMoney
  Example: Suppose the player has in the wallet BonusWin=100 USD, FrozenMoney=200 USD, BonusMoney=500 USD. If operator receives withdraw request with 400 USD WithdrawAmount then it should be taken off the way as described below: 400 -> 100(BonusWin) + 200(FrozenMoney) + 100(BonusMoney)
  After withdraw: BonusWin : 0 USD FrozenMoney: 0 USD BonusMoney: 400 USD
- If BonusDefId is available but there is not enough money in bonus wallets (BonusWin, FrozenMoney, BonusMoney) to process the withdrawal, then the corresponding error must be returned (error code 21 Not Enough Balance)
- If the BonusDefId is available then:

TotalBalance = BonusWin + FrozenMoney + BonusMoney

Otherwise, TotalBalance is the **withdrawable** money

1.1.4 Deposit

This method provides depositing on the player's account and returns the transaction reference and player's account balance after the transaction is made. This method is intended for collecting wins or collecting prizes in a tournament. It returns an object of the type **DepositOutput**.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of the operator (provided by BetConstruct)	Int32

PlayerId	Id of player whose deposits are made for	Int32
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)
DepositAmount	Amount to deposit on player's account	Decimal(18,4)
Currency	Currency of deposit transaction	Char(3)
GameId	Id of the game (null if cashback bonus)	Int32
RGSTransactionId	A unique key to indicate a specific financial activity. This key guarantees that the transaction was processed only once Note: Multiple deposits with the same RGSRelatedTransactionId but different RGSTransactionId are allowed.	Int64
RGSRelatedTransactionId	RGSTransactionId of the corresponding withdrawal request. This allows connecting the withdrawal and deposit requests. It is null if there is no connection. Note: Multiple deposits with the same RGSRelatedTransactionId but different RGSTransactionId are allowed.	Int64
TypeId	Describes a reason for the deposit. You can see the list of possible reasons in the section 1.1.7	Int32
BonusDefId	Optional parameter for Bonus logic	int?
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response Parameters

Parameter	**Purpose**	**Validation**
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32

ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
PlayerId	Id of player for whom the deposit is done	Int32
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)
TotalBalance	Player's balance amount after transaction done	Decimal
PlatformTransactionId	TransactionId of the platform for deposit operation	Int64

Important!

1. The Platform must successfully process this even with an expired token (simply checking if such a token exists)
2. In some cases, the platform can get requests with the same RGSRelatedTransactionId but different RGSTransactionId's. This happens because some game providers do it by themselves.

- If BonusDefId is available then money must be put in the BonusWin wallet
- If BonusDefId is available then:

TotalBalance = BonusWin + FrozenMoney + BonusMoney
Otherwise, TotalBalance is the **withdrawable** money

1.1.5 WithdrawAndDeposit

This method is a combination of Withdrawal and Deposit methods. This enables reducing the number of API calls to as many as possible. It returns an object of the type **WithdrawAndDepositOutput**.

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerId	Id of player for making a withdrawal	Int32
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)
WithdrawAmount	Amount to withdraw from player's account	Decimal(18,4)
DepositAmount	Amount to deposit on player's account	Decimal(18,4)

Currency	Currency of transaction	Char(3)
GameId	Id of game	Int32
RGSTransactionId	A unique key to indicate a specific financial activity. This key guarantees that a transaction was processed only once	Int64
TypeId	Describes a reason for withdrawal. You can see the list of possible reasons in the section 1.1.7	Int32
BonusDefId	Optional parameter for Bonus logic	int?
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response Parameters

Parameter	**Purpose**	**Validation**
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
PlayerId	Id of player for whom withdrawal was made	Int32
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)
TotalBalance	Player's balance amount after transaction made	Decimal
PlatformTransactionId	A unique key to indicate a specific financial activity. This key guarantees that the transaction was processed only once	Int64

See Withdraw and Deposit methods for bonus logic details

1.1.6 Rollback

If a need to reimburse the player's already-placed bet has come up, then a try to roll back the withdrawal using this method is applied until it succeeds. It returns an object of type **RollbackOutput**.

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerId	Id of player for who the balance is retrieved for	Int32
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)
RGSTransactionId	Id of transaction	Int64
GameId	Id of game	Int32
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response Parameters

Parameter	**Purpose**	**Validation**
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
TotalBalance	Player's balance amount after transaction made	Decimal
Token	A unique identifier is generated by the operator to identify interactions during the session	Varchar(50)

Important!

1. If rollback succeeded, the Platform is getting the rollback request with the same RGSTransactionId, then the platform responds without error.
2. The Platform must return an error with ErrorId = 107 when the transaction with the RGSTransactionId is not found.
3. The Platform must successfully process the rollback call with the expired token.
4. If the bet has won and a rollback occurred, then the win should also be rolled back, and the win amount should be deducted from the player's balance.

1.1.7 Transaction types

TypeId	**Description**
-9	Tip (Mainly for Live Dealer games)
-4	Join a tournament
-2	Create or sit behind the skill game table (Buy in)
-1	Standard bet (Spins in slots, bets in virtual and live games, etc.)
0	Standard DoBetWin (slots wins)
1	Standard win (slots wins, wins in virtual and live games, etc)
2	Win on skill game table
3	Tournament Win
4	Unregister from tournament
9	CashbackBonus

1.1.8 Response error codes

ErrorId	**Description**
8	Wrong Player Id
21	Not Enough Balance

29	Player Is Blocked
102	Invalid Token
107	Transaction Not Found
109	Wrong Transaction Amount
110	Transaction Already Complete
111	The Deposit Transaction Already Received
125	Invalid Bonus Definition Id
130	General Error

Important!

1. The Platform must return an error with ErrorId = 110 when the transaction with the same RGSTransactionId is already processed except for a RollBack.
2. The Platform must return an error with ErrorId = 111 when the deposit transaction with the same RGSTransactionId is already processed.

In some cases, RGS may send you the same request several times until it receives a positive answer about the transaction process succeeded, otherwise an error with id code 110/111 appears.

1.2 Bonus Logic (Optional)

To enable activating the bonus support, the partners need to implement the corresponding function. In essence, this consists of two parts. One part includes methods, which are provided by Remote Gaming Server, and another part involves methods provided by the Operator.

1.2.1 SyncBonusDefinition

Once the bonus is defined, the RGS replicates it to the Operator.

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of the operator (provided by BetConstruct)	Int32
BonusDefId	Id of bonusdefinition	Int32
Name	Bonus Name	Varchar(255)

TriggerId	**The Id of Trigger that platform must handle**	Int32
TriggerDetails	**{"PaymentSystemId":67,"Count":0}**	String JSON
BeginDate	Date when bonus starts (When it becomes visible/ available)	datetime
EndDate	Date when the bonus ends (When it becomes invisible/not available)	datetime
BonusShedule	Schedule of bonus (null by default)	string JSON
MaxPlayerCount	Max count of users who can accept the bonus	Int32
ExpirationDate	The Date when all accepted but not completed bonuses will expire (for all clients of that bonus)	datetime
ExpirationDays	Number of days after accepting the bonus when not completed bonus is canceled (for one client) Counting down from bonus activation date, i.e. a number of days after which the bonus is deemed expired	Int32
MaxBudget	Max budget of bonus	decimal(18,4)
BonusDetails	Bonus details information (max or min deposit, bet amount, wagering factor (mandatory)) related to bonus type	string JSON
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	string
TypeId	Type of bonus. Currently, the following 4 types of bonuses are supported 1- Deposit, 2- No Deposit, 3- Freespin, 4- Cash	Int32
Description	Bonus Description text	Varchar(255)
IsVisibleForAllPlayers	0-false, 1- true	bit

Response Parameters

Parameter	**Purpose**	**Validation**
HasError	If the response contains errors, then the HasError value is true, else false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)

1.2.2 BonusSchedule

Parameter	**Purpose**	**Validation**
Name	Name of schedule	Varchar(255)
StartDate	Schedule start date (null by default)	datetime
EndDate	Schedule end date (null by default	datetime
PeriodType	The Period: type None,1 day, 2 weeks, 3 months, 4 years	Int32
Period	Number of cycles to repeat per PeriodType (i.e. if PeriodType is a day-long and Period equals 2, then the number of cycle is 2 times per day)	Int32
Count	Number of times to repeat the PeriodType within StartDate -EndDate	Int32

1.2.3 BonusDetails

Bonus details are wager information (max or min deposit, bet, amount, and wagering factor (mandatory) related to bonus type. A structure of such an object is based on bonus type. The table below shows the object fields for each type of bonus. Each field represents an array of BonusAmount **objects** (amounts for different currencies).

Name	**Description**	Deposit	NoDeposit	Cash	Freespin
Money Requirement	• Fixed amount • Percentage	+ +	+ +	+	

Minimum Deposit	The BonusAmount object array	+		+	
Maximum Deposit	The BonusAmount object array	+		+	
Minimum Amount	The BonusAmount object array		+		
Maximum Amount	The BonusAmount object array		+		
Bonus Wagering Factor (mandatory)	The Integer. A wagering factor is a multiplier that represents the number of times the player has to wager the bonus funds	+	+		
Deposit Wagering Factor (mandatory)	The Integer. A wagering factor is a multiplier that represents the number of times the player has to wager the deposit funds	+			
Maximum Bet	The BonusAmount object array		+	+	
Maximum Payout	The BonusAmount object array		+	+	

1.2.4 BonusAmount

Parameter	**Purpose**	**Validation**
Amount	Amount value	decimal(18,4)
Currency	Currency of the amount	Char(3)

1.2.5 BonusSucceeded

RGS calls this method in order to notify the Platform that the player has succeeded with a bonus.

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of operator (provided by BetConstruct)	Int32
PlayerID	Id of player for whom the balance is retrieved	Int32

BonusDefId	Id of bonus definition	Int32
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	string

Response Parameters

Parameter	**Purpose**	**Validation**
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
Token	Security token associated with this player's partner	Varchar(50)

1.2.6 BonusExpired

RGS calls this method to notify the Platform that the bonus has expired for the player.

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerID	Id of player for who the balance is retrieved for	Int32
BonusDefId	Id of bonus definition	Int32
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	string

Response Parameters

Parameter	**Purpose**	**Validation**

HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
Token	Security token associated with this player's partner	Varchar(50)

1.2.7 BonusCanceled

The operator calls this method in order to notify RGS that the bonus was canceled for the player.

Request Parameters

Parameter	**Purpose**	**Validation**
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerID	Id of the player to retrieve the balance for	Int32
BonusDefId	Id of bonus definition	Int32
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	string

Response Parameters

Parameter	**Purpose**	**Validation**
HasError	If the response contains errors, then the HasError value is true, else false	Boolean
ErrorId	Identifies whether the request was processed successfully. If no error is detected, this code value is 0	Int32

ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
Token	The security token associated with this player's partner	Varchar(50)

Parameter	Purpose	Validation
PlayerId	Id of player	int32
PlayerBonusId	player bonus Id in operator side	int32
FreeRoundCount	count of freerounds	int32
CurrencyId	the currency code	Char(3)

1.2.8 BonusActivated

The operator calls this method in order to notify RGS that the player has activated the bonus. The parameters shown below are general for all types of bonus definitions.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerID	Id of player to retrieve the balance for	Int32
BonusDefId	Id of bonus definition	Int32
Amount	Amount value (nullable).	decimal(18,4)
Currency	Currency of the Amount (nullable)	Char(3)
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	string

Response Parameters

Parameter	Purpose	Validation
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
BonusAmount	Bonus amount granted by RGS (nullable)	decimal(18,4)
FrozenAmount	The deposit money gets frozen until the wagering requirement is fulfilled.	
Currency	Currency of the Amount (nullable)	Char(3)
PlayerId	Id of Player	Int32
Token	Security token associated with this player's partner	Varchar(50)

1.2.9 AddClientToBonus

The operator calls this method to notify RGS to give free spins to the player.

Request Parameters

Parameter	Purpose	Validation
BonusDefId	Id of bonus definition	int32
PartnerId	Id of the operator (provided by BetConstruct	int32
Players	Players information	string JSON
Hash	An MD5 hash of message body and Shared Key.	string

Response Parameters

Parameter	Purpose	Validation
Result	OK/ERROR	string
Partnerid	Id of the operator (provided by BetConstruct)	int32
HasError	If the response contains errors, then the HasError value is true, else false.	Boolean
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Int32
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected this code value is null.	Varchar(250)

1.2.10 Players

Parameter	Purpose	Validation
PlayerId	Id of player	int32
CurrencyId	the currency code	Char(3)
FreeRoundCount	count of freerounds	int32
PlayerBonusId	player bonus Id in operator side	int32

1.2.11 GetCurrencies

RGS calls this method in order to retrieve the partner's currency list.

Request Parameters

Parameter	Purpose	Validation
OperatorId	The Id of the operator (provided by BetConstruct)	Int32

PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	string

Response Parameters

Parameter	Purpose	Validation
HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
Data	Currency list	List of string

Response example:

```
{"HasError": false, "ErrorId":0, ErrorDescription:"", "Data": \["USD","EUR",\]\}"
```

1.2.12 GetPaymentSystems

RGS calls this method in order to retrieve the partner's payment system list.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	string

Response Parameters

Parameter	Purpose	Validation

HasError	If the response contains errors, then the HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error is detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to the player but rather gives details of the error	Varchar(250)
Data	**PaymentSystem** object list	List of objects

1.2.13 PaymentSystem object

Name	**Description**	**Validation**
Id	Payment system identifier	Int32
Name	Payment system name	string

Response example

```
{ "HasError": false, "ErrorId":0, ErrorDescription:"", "Data": \[
{
  "Id": 1,
  "Name": "Webmoney"
 },
 {
  "Id": 2,
  "Name": "Skrill"
 },
{
  "Id": 3,
  "Name": "Paypal"
  }]
```

2. Front End integration

After implementing the backend side, the operator should start with implementing the frontend. As a result, the BC platform connects to Operator games.
To launch every single game, the partner should perform the following steps:

1. Create a game: yourwebsitedomain.com and point it to 185.106.82.84, 188.42.197.100 IP addresses. Provide the SSL certificate to BetConstruct.
2. Create a unique token when the user clicks on a game icon.
3. Call the game to launch the URL

For Fun

http://games.yourwebistedomain/authorization.php?partnerId=&gameId=&openType=fun&language=en&devicetypeid=

For Real
http://games.yourwebistedomain.com/authorization.php?gameId=&token=&partnerId=&language=en&openType=real%20&devicetypeid=
After the above steps are completed, our server calls back to the partner's server for getting the user details. If the response contains no errors, the game is launched.

Parameter	Purpose	Validation
gameId	Id of the game for which information has taken	Int32
token	A unique token	Varchar(50)
partnerId	OperatorId (Assigned by Betconstruct)	Int32
language	2 character standard (ISO) (en, fr, ru, etc.)	Varchar(50)
openType	real/fun	Varchar(50)
devicetypeid	The type of the device, from which the game was launched.	INT 32 (1 for web, 2 for mobile web, 3 for IOS app, 4 for android)
isMobile	If isMobile: true, the game will launch for the mobile version of the website	Boolean
exitURL	An URL to exit the game. To exit the homepage	String
deposit_url	An URL to exit to the deposit page.	String

3. Getting Skins

3.1 Get Skin All Games Based On Parameters

Request URL - https://www.cmsbetconstruct.com/casino/getGames
Request Parameters

- partner_id - Partner site_id.
- category - Skin game category id.
- Provider- Skin game provider id.
- offset - Offset count. By default 0.
- limit - Limit count. By default 100.
- search - Fame name for searching.
- Count - Game count.
- is_mobile - Type for mobile games
- except - Excluded game ids.
- game_id- Game id for getting one game.
- external_id - Id from backend for getting one game.

Examples https://www.cmsbetconstruct.com/casino/getGames?partner_id=198&offset=0&limit=30
https://www.cmsbetconstruct.com/casino/getGames?partner_id=198&category=35&offset=0&limit=30
https://www.cmsbetconstruct.com/casino/getGames?partner_id=198&category=35&provider=1X2&offset=0&limit=30
For getting all games
https://www.cmsbetconstruct.com/casino/getGames?partner_id={partner_id}&count=all
Response Parameters
type - JSON
data - All games with categories, providers, and other properties of the game objects.

Error	**Error Description**.
emptyRequest	partner_id is missing
noSuchPartner	partner_id is wrong
wrongGameIdsType	type of game_id is not string
wrongGameId	game_id is not a number
noSuchGame	wrong game_id
wrongExternalIdsType	game_id is missing and the type of external_id is not a string
noSuchGame	game_id is missing, and there are no mobile games with such external_id
wrongCategoryType	type of the category is not string

wrongCategories	when the category is not valid
wrongProviderType -	type of the provider is not string
wrongProviders	when the provider is not valid
searchParamIsLonger	search parameter is longer than 50 characters
wrongTypeOfExcept	except parameter is not an array

3.2 Get Skin All Categories And Providers

Request URL - https://www.cmsbetconstruct.com/casino/getOptions
Request Parameters
partner_id - Partner site_id.
is_mobile - Type for mobile games.
Example - https://www.cmsbetconstruct.com/casino/getJeckpots?partner_id=198

Response Parameters
type - JSON
data - All categories and providers.

3.3 Get Skin Games Jackpots

Request URL - https://www.cmsbetconstruct.com/casino/getJeckpots
Request Parameters
partner_id - Partner site_id.
offset - Offset count. By default 0.
limit - Limit count. By default 100.
count - Game count.
is_mobile - Type for mobile games.
Example - https://www.cmsbetconstruct.com/casino/getJeckpots?partner_id=198

Response Parameters
type - JSON
data - All games jackpots (local jackpots | BetConstruct jackpots | global jackpots).

3.4 Get Skin Game Description

Request URL - https://www.cmsbetconstruct.com/casino/getSkinGameDesc?
Request Parameters
game_skin_id - Skin game ID.
Example - https://www.cmsbetconstruct.com/casino/getSkinGameDesc?game_skin_id=727

Response Parameters
type - JSON
data - Skin game description

4. FAQ

Question: What's the use of PublicKey in all methods?

Answer: PublicKey is used for security reasons to verify the caller. The JSON body of the request plus the SharedKey(given by Betconstruct) is used to compute PublicKey. var PublicKey= ComputeSHA(String.Format("{0}{1}", MessagejsonBody,SharedKey));

```
public static string ComputeSHA(string data)
{
using (var sha = SHA256.Create())
{
var hashArray = sha.ComputeHash(Encoding.UTF8.GetBytes(data));
return string.Concat(Array.ConvertAll(hashArray, b => b.ToString("x2")));
}
}
```

The above code is C#

Question: What is the difference between total and virtual balances?

Answer: VirtualBalance will be used for bonus bets in the future. For VirtualBalance the value is set to null.