Casino API
Integration for Games
(Version 3.1.0)

Rev 1.1

Revision History

Rev #	Revision Date	Description	Approved by	Corrected by
1.0	29/07/2016	Proofreading. Ch. Basic Concept added	Geham Yegoryan	Igor Mouhsian
1.1	1/08/2016	Ch.3 Getting Skins added	Ashot Boyakhchyan	Igor Mouhsian

Table of 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, sport betting, etc.) with existing integrations to other game providers and wish to integrate RGS into their own site.

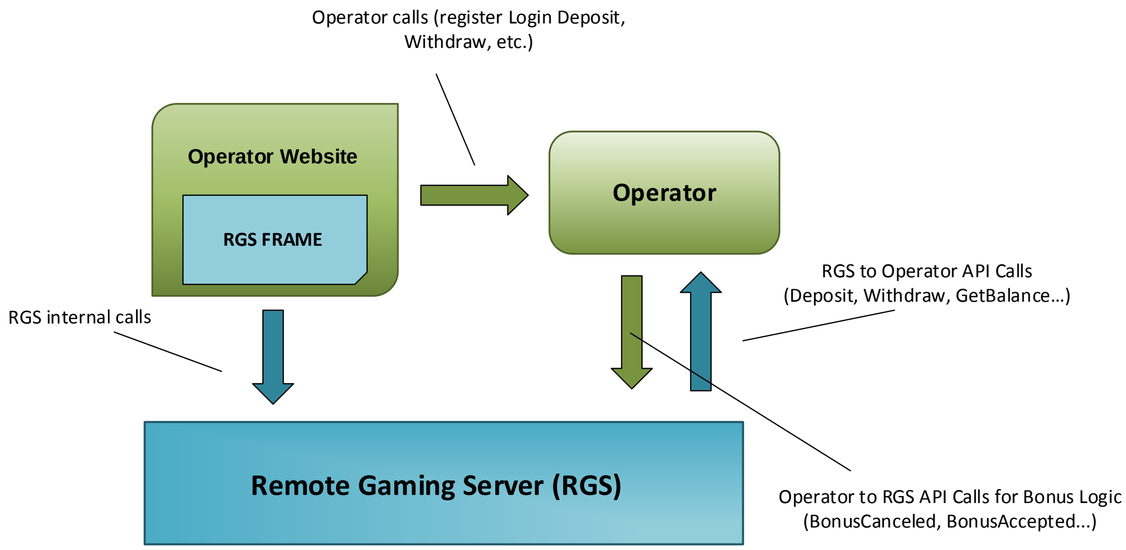
Operators can manage their players and balances/wallets separately in their own systems and seamlessly integrate. For this purpose Operators have to implement the Partner API to be called by RGS.

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 for modules of games. The back-end integration is implemented to send and receive messages between RGS and Operator (for example, placed bets, winning information, etc.).

All the backend calls are verified by PublicKey which is Sha256 value of message body and Shared Key.

Message body is the Json string value of request object (properties are ordered by their names).



Integration Workflow

This document contains detailed information which allows an operator to integrate with Betconstruct Remote Gaming Server.

The Integration consists of 2 parts: Front end and Back end.

With Front end integration 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 Sha256 value of message body and Shared Key. Message body is the Json string value of 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 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 operator (provided by BetConstruct)	Int32
Token	Initial player's token to launch the protocol	Varchar(250)
ClientRfid	Players RFId Code (Optional for Land Based Operations)	VarChar(250)
UserName	The UserName of the player, which is used when 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	Password of player, which is used when Token is empty. Authentication with username and password is used mainly in case the games have been downloaded by clients.(See FAQ for details)	Varchar(50)
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response

Parameter	Purpose	Validation
OperatorId	Id of operator (provided by BetConstruct)	Int32
Name	Name of authenticated user	Varchar(50)
NickName	Nickname of authenticated user. NickName must be unique for each user	Varchar(50)
UserName	Username of authenticated user. UserName must be unique for each user	Varchar(50)
Token	Be not confused with input token. This token is a session token only - A unique identifier, which is generated by operator to identify the session's interactions	Varchar(250)
TotalBalance	Available balance of authenticated user's account	Decimal(18,4)
Gender	Gender of authenticated user	Boolean
Currency	Currency of authenticated user	Char(3)
Country	Country of authenticated user	Char(2)
PlayerId	User Id of authenticated user. User Id must be unique for each user	Int32
UserIP	User IP of authenticated user	Varchar(30)
HasError	HasError: If response contains errors, then HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error detected, this code value is 0	Int32

ErrorDescription	A string that describes the response. This string is not a message to 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 operator (provided by BetConstruct)	Int32
PlayerId	Id of player who the balance is retrieved for	Int32
Token	A unique identifier which is generated by 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

Parameter	Purpose	Validation
HasError	If response contains errors, then HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to player but rather gives details of error	Varchar(250)
PlayerId	Id of player who the balance is retrieved for	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 which is generated by operator to identify interactions during the session	Varchar(50)

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

1.1.3 Withdraw

This method withdraws money from player's account and returns the transaction reference and player's account balance after 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 operator (provided by BetConstruct)	Int32
PlayerId	Id of player who the withdrawal is made for	Int32
Token	A unique identifier, which is generated by 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 transaction was processed only once.	Int64
TypeId	Describes the reason of withdrawal. You can see the list of possible reasons in the section 1.1.1	Int32
BonusDefId	Optional parameter for Bonus logic	int ?
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response

Parameter	Purpose	Validation
HasError	If response contains errors, then HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to player but rather gives details of error	Varchar(250)
PlayerId	Id of player who the withdrawal is made for	Int32
Token	A unique identifier which is generated by operator to identify interactions during the session	Varchar(50)
TotalBalance	Player's balance amount after 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. BonusWin
  2. FrozenMoney
  3. 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 no enough money in bonus wallets (BonusWin,FrozenMoney, BonusMoney) to process the withdrawal then the corresponding error must be returned (errorcode 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 player's account and returns the transaction reference and player's account balance after the transaction made. This method can be used to collect a win or to collect a prize in a tournament. It returns an object of type DepositOutput.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of the operator (provided by BetConstruct)	Int32
PlayerId	Id of player who deposit is made for	Int32
Token	A unique identifier which is generated by 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 game (null if cashback bonus)	Int32
RGSTransactionId	A unique key to indicate a specific financial activity. This key guarantees that transaction was processed only once	Int64
RGSRelatedTransactionId	RGSTransactionId of the corresponding withdrawal request. This provides capability to connect withdrawal and deposit requests. It is 0 if there is no connection. It is allowed having multiple deposits with same RGSRelatedTransactionId but different RGSTransactionId	Int64
TypeId	Describes a reason of deposit. You can see the list of possible reasons in the section 1.1.1	Int32
BonusDefId	Optional parameter for Bonus logic	int ?
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response

Parameter	Purpose	Validation
HasError	If response contains errors, then HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to player but rather gives details of error	Varchar(250)
PlayerId	Id of player for whom deposit is done	Int32
Token	A unique identifier which is generated by 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 expired token (checking out only if such a token just exists)
2. There can be cases when the platform can get requests with the same RGSRelatedTransactionId but different RGSTransactionId's. This happens because of some game providers do it by themselves.

- If the BonusDefId is available then money must be put in BonusWin wallet
- If the 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. It allows to reduce an amount of API calls as many as possible. It returns an object of type WithdrawAndDepositOutput.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of operator (provided by BetConstruct)	Int32
PlayerId	Id of player for making withdrawal	Int32
Token	A unique identifier which is generated by 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 of withdrawal. You can see the list of possible reasons in the section 1.1.1	Int32
BonusDefId	Optional parameter for Bonus logic	int ?
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response

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

ErrorDescription	A string that describes the response. This string is not a message to player but rather gives details of error	Varchar(250)
PlayerId	Id of player for whom withdrawal was made	Int32
Token	A unique identifier which is generated by 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 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 with using this method is applied until it succeeds. It returns an object of type RollbackOutput.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of operator (provided by BetConstruct)	Int32
PlayerId	Id of player who the balance is retrieved for	Int32
Token	A unique identifier which is generated by 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

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

Important

1. If rollback succeeded, the Platform is getting the rollback request with same RGSTransactionId, then platform responds without error.
2. The Platform must return error with ErrorId = 107 when the transaction with the RGSTransactionId was not found.
3. The Platform must successfully process the rollback call with expired token.

1.1.7 RefreshToken

Sometimes it could be necessary to refresh current token. The RefreshToken function returns an object of type RefreshTokenOutput.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of operator who was assigned to player	Int32
Token	A unique identifier which is generated by operator to identify interactions during the session	Varchar(50)
PublicKey	Sha256 hash of message body and Shared Key. (See FAQ for details)	Varchar(64)

Response

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

1.1.8 Transaction types

TypeId	Description
-9	Tip (Mainly for Live Dealer games)
-4	Join to 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

1.1.9 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 error with ErrorId = 110 when the transaction with the same RGSTransactionId already processed except RollBack.
2. The Platform must return error with ErrorId = 111 when the deposit transaction with the same RGSTransactionId already processed.

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

1.2 Bonus Logic (Optional)

To enable activating the bonus support, the partner’s need to implement corresponding functions. In essence, this consists of 2 parts. One part includes methods which are provided by Remote Gaming Server and another part involves methods which should be provided by Operator.

1.2.1 SyncBonusDefinition

RGS will replicate it to Operator Side once bonus is defined.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of 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 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	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 are cancelled (for one client) Counting down from bonus activation date, i.e. an amount of days after which the bonus is deemed as expired	Int32
MaxBudget	Max budget of bonus	decimal(18,4)
BonusDetails	Bonus details information (max or min deposit, bet amount, wagering factor (mandatory)) relative 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		Varchar(255)
IsVisibleForAllPlayers	0-false, 1- true	bit

Response

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

1.2.2 BonusSchedule

Parameter	Purpose	Validation
Name	Name of schedule	Varchar(255)
StartDate	Schedule Starting Date (null by default)	datetime
EndDate	Schedule Ending 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 to 2 then cycling figure 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)) relative to bonus type. A structure of such an object is based on bonus type. Below are shown the mentioned fields, which are contained in this object for each type of bonus. Each field represents array of BonusAmount **objects** (amounts for different currencies).

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

	• Percentage	+	+		
Minimum Deposit	BonusAmount object array	+		+	
Maximum Deposit	BonusAmount object array	+		+	
Minimum Amount	BonusAmount object array		+		
Maximum Amount	BonusAmount object array		+		
Bonus Wagering Factor (mandatory)	Integer. How many times needed to turn over bonus money	+	+		
Deposit Wagering Factor (mandatory)	Integer. How many times needed to turn over Deposit money	+			
Maximum Bet	BonusAmount object array		+	+	
Maximum Payout	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 Platform that player has succeeded with bonus.

Request Parameters

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

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

1.2.6 BonusFailed

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

Request Parameters

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

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

1.2.7 BonusExpired

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

Request Parameters

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

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

1.2.8 BonusCanceled

Operator calls this method in order to notify RGS that bonus was canceled for 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

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

1.2.9 BonusAccepted

Operator calls this method in order to notify RGS that bonus was taken by the player.

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
TournamentId	Tournament ticket Id in RGS	Int32

Response

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

1.2.10 BonusActivated

Operator calls this method in order to notify RGS that bonus was activated for player. The parameters shown below are general for all types of bonus definition.

Request Parameters

Parameter	Purpose	Validation
OperatorId	Id of 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

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

1.2.11 GetCurrencies

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

Request Parameters

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

Response

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

Request Parameters

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

Response

Parameter	Purpose	Validation
HasError	If response contains errors, then HasError value is true, else it is false	Boolean
ErrorId	Identifies whether or not the request was processed successfully. If no error detected, this code value is 0	Int32
ErrorDescription	A string that describes the response. This string is not a message to player but rather gives details of 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 backend side Operator should start with implementing frontend. The BC platform will be connected for Operator games, and Operator may get list of game by following step:

//casinoapi.betconstruct.com/api.php?action=getGamesList&partnerId={OperatorId}&apiKey={secretKey}

Where :
partnerId = OperatorId (Assigned by Betconstruct)
apiKey = Secret key (Assigned by Betconstruct)

Response is in JSON format where Operator can find each connected game with following parameters:

Parameter	Purpose	Validation
gameName	Name of game	String
gameIcon2	URL of game picture placed on BC side	String
externalID	Game ID, which will be used in game launch call	Int32
gameBackground	URL of game background placed on BC side (NOT ALL GAMES HAVE BACKGROUND)	String
providerFilter	3 letter code of game provider. This is optional. Operator may create provider filter at his side	String
isMobile	Is the game mobile or not	Enum (Y/N)
jackpotGame	Is the game jackpot or not	NOT SPECIFIED YET
masterCategory	Game category type	String
gameType	Additional information about game type (OPTIONAL)	NOT SPECIFIED
gameOptions	Additional parameters (should be always added after constructing main LAUNCH URL)	String
gameCategory	Category of game suggested from BC (Operator may change this as he wants by using ADMIN TOOL provided by BC)	String
ordering	value of game order (Operator may change this on his own by using ADMIN TOOL provided by BC)	Int32

To launch every single game the partner must perform the following steps:

1) Create a unique token when user clicks on game icon.

2) Call the game to launch URL by sending the Id of partner, Id of game, Id of the table (if this is, for example, live casino game and partner wishes to open table directly) and generating a token.

After abovementioned steps done, our server calls back to partner's server for getting user details. If response is without errors then game is launched.

The following URL is for launching the Games:

//casinoapi.betconstruct.com/authorization.php?gameId={gameId}&token={token}&partnerId={partnerId}[&tableId={tableId}]&language={lang}&openType={openType}

Parameter	Purpose	Validation
gameId	Id of game for which information has taken	Int32
token	A unique token	Varchar(50)
partnerId	OperatorId (Assigned by Betconstruct)	Int32
tableId	Live Dealer games table Id (need just for LiveCasino games)	Int32
language	2 character standard (ISO) (en, fr, ru etc …)	Varchar(50)
openType	real/fun	Varchar(50)

3. Getting Skins

3.1 Get skin all games based on parameters

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

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

type - Json

data - All games with categories, providers and other properties of game object.

3.2 Get skin all categories and providers

Request URL - https://www.cmsbetconstruct.com/casino/getOptions

Parameters

partner_id - Partner site_id.

is_mobile - Type for mobile games.

Example - https://www.cmsbetconstruct.com/casino/getOptions?partner_id=198

Response

type - Json

data - All categories and providers.

3.3 Get skin games jackpots

Request URL - https://www.cmsbetconstruct.com/casino/getJeckpots

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

type - Json

data - All games jackpots(locale jackpots | betconstruct jackpots | global jackpots).

3.4 Get skin game description

Request URL - https://www.cmsbetconstruct.com/casino/getSkinGameDesc?

Parameters

game_skin_id - Skin game ID.

Example - https://www.cmsbetconstruct.com/casino/getSkinGameDesc?game_skin_id=727

Response

type - Json

data - Skin game description

4. FAQ

1. **Question:** What's the use PublicKey in all methods

   Answer: PublicKey is using for security reasons to verify the caller. The json body of request plus the SharedKey(given by Betconstruct) are 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")));
    }
}
```

Note: The above code is C#

---

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

   Answer: VirtualBalance will be used for bonus bets in the future.
   For VirtualBalance to be set 0.