



+Play API

for Seamless and Transfer Wallet integration

Last Updated 21 April 2023

Important Links and Latest versions:

This Document:

<https://haba88.com/go/apidoc>

Postman examples:

<https://haba88.com/go/postman>

Game Events Javascript API:

<https://haba88.com/go/jsapidoc>

Integrations Tests:

Seamless: <https://haba88.com/go/seamless>

Transfer: <https://haba88.com/go/transfer>

Document / API Changes:

21 April 2023

General

Added Show Dependency parameter for Critical Files API

14 April 2023

General

Add GetGameReplayUrl api method

General

Add Hide Replay parameter on Game Launch to hide the history Replay column

9 March 2023

General

Update WebHooks Bonus Event Messages, add parameters in Bonus_GameAction

30 January 2023

General

Update Supported ProductExternal in Addendum I

14 October 2022

General

ReportJackpotWinner – added BrandGameId

30 September 2022

General

Added new method GetJackpotsByGroup which returns all Brand Jackpots for a specific groupid

6 September 2022

General

Added new method ExpireAllBonusBalance which will expire all bonus balances for the brand via CouponCode or CouponId

Seamless

BuyfeatureId added to seamless request indicating which feature a player purchased

27 July 2022

General

Added BuyFeatureId Mapping in Addendum L

14 June 2022

General

GetBrandIncompleteGames – added BuyFeatureId in the response

06 April 2022

General

ReportJackpotWinner – added GameStake and GameTotalPayout

24 March 2022

General

Update Alternative Fund Credits parameter description

15 March 2022

General

Deprecated Webhook Game and POS event messages

07 March 2022

General

Add examples for different tournament prize types

04 March 2022

General

Added Partner Prize into addendumK TournamentPrizeType

23 February 2022

General

Added FeatureCount and BuyFeatureId to GetBrandCompletedGameResultsV2, GetGroupCompletedGameResultsV2

11 February 2022

General

Added PlayerTypeId in GetBrandIncompleteGames API response

07 February 2022

General

Added “ig” (include games) parameter on Jackpot API ticker

13 January 2022

General

Added Defer Credits information for Seamless clients

06 January 2022

General

Updated test page for client-side jackpot ticker api

13 December 2021

General

Moved Regular Bonus from Seamless examples to a new section (Bonus (Regular))

22 November 2021

Seamless

Added nullable "MaxPayLimit" to seamless wallet fundrequest gamedetails

05 November 2021

General

Rename Alternative Fund Credits response from fundtransferresponse to altfundsresponse

05 October 2021

General

Added new API Method – GetBrandCCWinners

Rename GetBrandCCTournamentEvent to GetBrandCCEvents

04 October 2021

General

Added new API Method - GetBrandCCTournamentEvent

01 October 2021

General

Added new fields to Alternative Fund Credits

Added Tournament Prize Type and Event Types in Addendum (J,K)

27 September 2021

Seamless

Added nullable "ProductExternalId" identifier to seamless wallet fundrequest

22 September 2021

General

Added "coins" field to GetGames API Response

8 September 2021

Seamless

Added "initialdebittransferid" to the QueryRequest

3 August 2021

General

Add Config Detail Request to API doc

Added new Game Launch Flags (rsw, hnw, cnw)

Mark GetPlayerGameTransactions method as deprecated

Updated Client Ticker section

Updated supported currency list

11 May 2021

General

Added section for Habanero Game Lobby usage

20 April 2021

General

Added new field for Certification Files

14 April 2021

General

Added new response parameter increment, offset, persec for client-side jackpot api

12 March 2021

General

Added new showpaytable parameter for Graphical game result

23 February 2021

General

Added client-side jackpot api / ticker info

Added SupportBonusFS to getgames() to indicate the game supports bonus free spins

10 February 2021

General

Added new viewtype for Graphical game result (gameevents) and featureno to seamless messages.

25 January 2021

Seamless

Added freespin count and value to BonusDetails sent in fundtransferrequest

18 November 2020

General

Added link to "Habanero - Game Javascript Events API" on iframe page

18 September 2020

General

Added ReportBrandCouponRedemptions method

9 September 2020

General

Added game client reality check options to launch url

15 April 2020

Seamless

Added jptypeid and jpname to seamless wallet win message

25 March 2020

General

Added JackpotWin and JackpotContribution to ReportPlayerStakePayout()

19 March 2020

General

Added info for completed gamestatemode for bonus payouts. Update currency support list

11 March 2020

General

Added info for Alternate Fund Credits (altfundsrequest). Used for Seamless wallet Tournament payout.

4 March 2020

General

Added more info for playerendsessionrequest

5 November 2019

General

Added "Send-per-Spin" Bonus Information for Seamless transactions

14 October 2019

General

Added "Service Error Codes" for JSON/soap webservice error responses.

10 September 2019

General

Added GetGameParameters method

30 August 2019

General

Added GetBrandTournamentEvent beta method

3 July 2019

General

Added new info for Certification Files

Added new method in API ExpireBrandGames

18 June 2019

General

Added new section for Certification Files

14 June 2019

General

Added Jurisdiction to Launch Url to specify jurisdiction to use.

Seamless

- Added Jurisdiction option to return the Jurisdiction of the player from the Auth request

5 June 2019

General

Added new method – GetBrandCompletedGameResultsV2 and GetGroupCompletedGameResultsV2 and deprecated old ones (GetBrandCompletedGameResults and GetGroupCompletedGameResults)

Add new field (srij_smdata) to fundtransferrequest.gamedetail

9 May 2019

General

Add new API Message Type - playerendsession

25 March 2019

General

Add new field GameInFeature in fundtransferrequest.fundinfo

11 March 2019

General

Add new field AccountTransactionType in fundtransferrequest. Please see Addendum G

7 March 2019

General

Missing information about ExpireDays added for CreateAndApplyBonus

19 February 2019

General

Added QueueUnregisteredPlayers and CreatePlayerIfNotExist parameters to Coupon/Bonus methods that adds control to player and bonus redemption

8 January 2019

General

Added HistoryUrl parameter to Launch Url so you may set a custom game history url / view

Added new Reality Check information in Custom Game Dialog Messages

14 December 2018

General

Added GetAllJackpotsInAllBrands() method to retrieve Jackpot information for all the brands in the group

21 November 2018

General

Added optional BrandId parameter that will show Game RTP information in game help

25 October 2018

General

Added note on date response time format (ISO 8601) and updated some examples

24 October 2018

Seamless and Transfer

Added `GetPlayerResumeGames()` and `ExpirePlayerGames()` method to retrieve list of incomplete games for a player and ability to expire it

17 October 2018

Seamless

Added `GetPlusPlayErrors()` and `ResolvePlusPlayError()` to the JSON webservice for reconciling Seamless errors via API

7 September 2018

General

Added `GetBrandIncompleteGames` service method to retrieve list of incomplete games in a brand including those in Seamless wallet error state.

26 July 2018

Seamless

`QueryRequest` – added `BaseGame` info, `GameInstanceId` and `FriendlyGameInstanceId` to request

10 May 2018

General

Added nullable decimal `PointBalance` to player response for `CreateOrLoginPlayer` and `QueryPlayer`

11 April 2018

General

Added new fields (`GameKeyNames` and `BrandGameIds`) to `CreateAndApplyBonusMulti` method – free spin bonuses can now be assigned to more than one game of the same coin configuration.

Added new fields (`GameNames` and `GameKeyNames`) to `GetBonusAvailablePlayer()` and `GetBonusBalances()` to indicate whether multiple games are configured for this bonus

Added new Jackpot Type "Race" (`JackpotTypeId` 8) and relevant field details to `GetJackpots()`

Added "Position" of the winners in a Jackpot Race payout to `ReportJackpotWinner()`

07 March 2018

General

Added `FriendlyId` to game history so either `GameInstanceId` or `FriendlyId` can now be used to view game details.

23 February 2018

Seamless

Added `gamelaunch <bool>` to player detail request so you can distinguish if the call is for validating the token on game launch versus a balance refresh.

12 February 2018

Seamless

Added couponcode to BonusDetails in fund transfer request

22 December 2017

Reporting

Added BalanceAfter to GetPlayerGameResults(), GetBrandCompletedGameResults(), GetGroupCompletedGameResults() indicating the players balance after a game was completed. This will only be populated for records after 22 December.

General

Added new Game Types in Section 4 - GameTypeID of 18 (Other Table Games)

13 December 2017

General

Added information for MaxRedemptionIntervalId in Coupon methods that will set/get the Redemption Interval of Coupons (0 = All Time, 2 = Daily, 3 = Weekly, 4 = Monthly). This is added for the following functions: CreateAndApplyBonusMulti, CreateAndApplyBonus, GetBonusAvailablePlayer

23 Novemebr 2017

General

Added information about Rate Limits for reporting webservice

27 October 2017

General

Re-arranged document into new sections and general cleanup

Bonus related API changes to enable toggling/deleting bonuses by couponId
Webhooks added to for bonus info.

20 October 2017

General

Added game keynames and brandgameids to GetJackpots() service method.

13 October 2017

General

Added new option for configuring bonus values according to game bet parameters in **CreateAndApplyBonusMulti**. This eases multi-currency setups

2 October 2017

General

Added **CreateAndApplyBonusMulti** and **ApplyBonusToPlayerMulti** to replace previous methods. These new methods are better suited for multicurrency bonuses and supports multiple players in one request

14 Aug 2017

General

Added JSON Web Service information in section 7. You can now use JSON instead of SOAP for webservice requests.

27 July 2017

Seamless API

Added "**initialdebittransferid**" which indicates on a fundsinfo node what the first transferid for the game was.

11 July 2017

Seamless API

QueryRequest method now has its own Endpoint which can be set in the Habanero Backoffice. If the endpoint is not specified then the Transaction endpoint will be used as per the current method.

QueryRequest – added "queryamount" field which is the positive (in case of credit) or negative (in case of debit) amount in question for the specified transferid which is being queried.

FundTransferRequest – "customplayertype" is now sent on all api calls. Value of 0 indicates regular player. Value of 1 indicates player marked as Tester in Habanero Backoffice. Discuss with your account manager for further information

-- this page intentionally left blank --

Table of Contents

Document / API Changes:	2
Table of Contents	12
Introduction	15
Seamless Wallet	15
Transfer Wallet.....	15
Whitelisting	15
Game and Logo Import	16
Game Identifiers.....	16
Game Help/Rules.....	17
Launching Games (Transfer and Seamless)	18
Using iFrames with game	20
Using PostMessage with game close / return to lobby	21
Game Events API	21
Habanero Game Lobby.....	22
Habanero Web Service (SOAP or JSON)	24
Using SOAP or JSON?.....	25
Service Error Codes	26
API Methods.....	27
LoginOrCreatePlayer (Transfer Wallet Only)	27
UpdatePlayerPassword (Transfer Wallet Only)	29
DepositPlayerMoney (Transfer Wallet Only)	30
WithdrawPlayerMoney (Transfer Wallet Only)	31
QueryTransfer (Transfer Wallet Only)	32
QueryPlayer (Transfer Wallet Only).....	33
LogOutPlayer (Transfer and Seamless).....	34
SetMaintenanceMode (Seamless only)	35
GetBrandIncompleteGames (Transfer and Seamless).....	36
GetPlayerResumeGames (Transfer and Seamless).....	37
ExpirePlayerGames (Transfer and Seamless Wallet)	38
ExpireBrandGames (Transfer and Seamless Wallet).....	39
GetGames (Transfer and Seamless Wallet)	40
GetGameParameters (Transfer and Seamless Wallet)	41
GetGameReplayUrl (Transfer and Seamless Wallet)	42

GetJackpots (Seamless and Transfer Wallet).....	43
GetAllJackpotsInAllBrands (return all data for Group)	43
GetJackpotsByGroup (Seamless and Transfer Wallet).....	45
GetBrandTournamentEvent (Transfer and Seamless Wallet)	46
GetBrandCCEvents (Transfer and Seamless Wallet)	47
GetBrandCCWinners (Transfer and Seamless Wallet)	48
GetPlusPlayErrors (Seamless Wallet).....	49
ResolvePlusPlayError (Seamless Wallet).....	50
Jackpot Contribution Reports	51
ReportJackpotContribution()	51
ReportJackpotContributionPerGame()	51
Bonus/Coupon Methods	52
CreateAndApplyBonusMulti (Transfer and Seamless)	52
GetBonusAvailablePlayer (Transfer and Seamless)	59
ApplyBonusToPlayerMulti (Transfer and Seamless)	60
GetBonusBalancesForPlayer (Transfer and Seamless).....	61
SetPlayerBonusBalanceActive (Transfer and Seamless)	62
DeletePlayerBonusBalance (Transfer and Seamless).....	63
ExpireAllBonusBalance (Transfer and Seamless)	63
Reporting Methods	66
GetBrandCompletedGameResultsV2 (Transfer Wallet).....	67
GetGroupCompletedGameResultsV2 (Group Report).....	67
GetBrandTransferTransactions (Transfer Wallet).....	68
GetPlayerTransferTransactions (Transfer Wallet)	69
GetPlayerGameTransactions (Transfer Wallet) DEPRECATING SOON	70
GetPlayerGameResults (Transfer Wallet)	71
GetPlayerStakePayoutSummary (Transfer Wallet).....	72
ReportGameOverviewPlayer (Transfer Wallet)	73
ReportPlayerStakePayout (Transfer Wallet).....	74
ReportJackpotWinner (Transfer and Seamless Wallet)	75
ReportGameOverviewBrand (Transfer Wallet)	76
ReportBrandCouponRedemptions (Transfer and Seamless Wallet)	77
SEAMLESS WALLET API	80
Player Detail Request (For Authentication and Balance).....	83
Fund Transfer Request (Game actions)	86
Alternative Fund Credits (Tournament and Prize Drops).....	96

Bonus (Regular)	101
Bonus Send-Per-Spin (Bonus transaction)	102
Seamless Wallet Error Resolution	106
Refund Request	108
Re-Credit Requests	110
QueryRequest (used for Dual D&C failure resolution)	111
Custom Game Dialog Messages (Reality Checks, Maintenance and more)	115
Game Bet/Config Parameters (Optional)	117
Player Game End Session (Optional)	119
Seamless Wallet Checklist / Signoff	120
Graphical Game Result for Customer Support / Backoffice	122
View Player history	123
View a Game Event result	124
Show game Paytable	125
Client-side Jackpot Ticker	126
WebHooks	129
Certification Files	134
Addendum	136
Addendum A - Languages	136
Addendum B - Currencies	137
Addendum C – ChannelTypeId	141
Addendum D – GameTypeId	142
Addendum E – CouponTypeId	143
Addendum F – JackpotTypeId	144
Addendum G – AccountTransactionType	145
Addendum H – CouponStatusId	147
Addendum I – ProductExternalId	148
Addendum J – TournamentEventType	149
Addendum K – TournamentPrizeType	150
Addendum L – BuyFeatureId	151

Introduction

Habanero supports 2 different integration modes:

Seamless Wallet

The Seamless Wallet integration allows you to add Habanero games to your platform using your own wallet and authentication systems. You may optionally provide game configuration/parameters to Habanero. You maintain full control over the player and wallet and all debits/credits are performed against your wallet in real-time.

You will need to implement the following:

Section 1: Game and logo import

Section 2: Launching Games

Section 4: Implement the API on your own server. We call your server.

Section 3: Bonusing and other features exposed via the Habanero JSON service

Section 5: Graphical game result retrieval

Test your Seamless Integration here:

<https://haba88.com/go/seamless>

Transfer Wallet

The Transfer Wallet integration provides web service methods for you to create/login/logout players and deposit/withdraw money to a player's Habanero based Wallet. The player and wallet exist in Habanero along with all other configuration options. You simply transfer money from your system into the Habanero wallet which is then used for debits/credits.

You will need to implement the following:

Section 1: Game and logo import

Section 2: Launching Games

Section 3: Player creation, authentication, deposit and withdraw, wagering/game reports

Bonusing and other features via the Habanero JSON service. **You call our server.**

Section 5: Graphical game result retrieval

Test your Transfer Integrations here:

<https://haba88.com/go/transfer>

**** WARNING REGARDING STRICT DATA CONTRACTS ****

Do NOT implement strict contracts for any of the described API's. We add additional fields to existing methods in a way that does not break the API unless you have strict contracts!

Whitelisting

- Seamless wallet users are provided with the Habanero IP addresses which will call your API.
- If using our webservice, you may setup the IP access list in our Backoffice
- Backoffice users do not need whitelisting but this can be setup per User in the Backoffice
- Players whitelisting can be done by editing each player in the Habanero backend and set them to Excluded from Geo IP checking. This is strictly for CS accounts only. In addition, Group Super Admins can add a list of whitelisted IP's in the Backoffice.

Game and Logo Import (Transfer and Seamless)

Game Identifiers

Each Habanero game has a static **KeyName** and a dynamic BrandGameId (unique per Brand) which identifies a game.

Use the static Keyname and import this into your game list from the Excel sheet provided by Habanero.

If you implement a dynamic system to retrieve games from our webservice then you may consider using the BrandGameId as this will allow you more programmatic control to switch RTP's for a game. This is only recommended for advanced integrations. If you are in doubt – then use the Keyname or discuss with us.

Detailed Explanation on Keyname vs BrandGameId

The KeyName is a **static** identifier for a game, which is the same on every Habanero installation.

The BrandGameId is '**dynamic**' and only assigned when a game is linked to a brand in the Backoffice. The BrandGameId therefore differs between brands and your TEST and the LIVE server.

The BrandGameId exists to support different RTPs in slot games - When you assign a Slot game to your brand, you select which RTP to use... If a game has 3 different RTP's and you assign all of them to your brand you will have 3 different BrandGameId for the same Game. The benefit of having all 3 active and ready in Habanero is so that you may decide in your code / backend which to offer the player.

Example of 1 slot game added to a brand with 3 RTPs:

BrandGameId	KeyName	ReportName	RTP
66E7B8AA...	SGTheBigDeal	The Big Deal – 92% RTP	92.00
3ED6C358...	SGTheBigDeal	The Big Deal – 94% RTP	94.00
B6A440CD...	SGTheBigDeal	The Big Deal – 96% RTP	96.00

Notice: 3 different BrandGameId but the same KeyName for the same game

Habanero reporting in the Backoffice, or via the webservice methods will always report and group data by BrandGameId but will also provide the KeyName for the game data, so you are free to decide which to use.

Retrieving BrandGameId's (if using brandgameid instead of keyname)

Use the GetGames() webservice method described later in this document to retrieve a list of brand games.

Game Help/Rules

Game payable information is available online and can be accessed by BrandGameID or Keyname as shown in the URLs below:

https://app-test.__.com/help/game?keyname=<GameKeyname>&locale=<locale>
https://app-test.__.com/help/game?brandgameid=<BrandGameID>&locale=<locale>

OPTIONAL: **BrandId** – adding this on the URL, together with the **KeyName**, will show game RTP information

https://app-test.__.com/game?keyname=<GameKeyname>&locale=<locale>&brandId=<BrandId>

Game Logos, Backgrounds

Please download logos and host them on your own site.

To download all current logos: visit <https://haba88.com/go/logopack>

Alternately - full assets such as game backgrounds, characters, symbols, PSD of logos etc are available at <http://client.habanerosystems.com> – the username/password is provided in your welcome email.

Hot linking game logo from Habanero (not recommended)

Game logo images may be dynamically loaded from the Habanero server using the following URL. They are resized to the width requested in the route.

https://app-test.__.com/img/<type>/<width>/<brandgameid>OR<keyname>_<locale>.png

type of logo	Value
Oval (with a gold border)	o / oval
Oval Flat	of / ovalflat
Rectangle	r / rectangle
Square	s / square
Circle (with a gold border)	c / circle
Circle Flat	cf / circleflat
Paytable	paytable

Locale (English and Chinese):

Locale is optional and defaults to English (en). See the Addendum for locale codes. If no localised image exists, the English version will be displayed.

Examples:

https://app-test.insvr.com/img/oval/200/SGArcticWonders_zh-CN.png
<https://app-test.insvr.com/img/square/400/SGArcticWonders.png>
<https://app-test.insvr.com/img/rect/300/SGArcticWonders.png>
<https://app-test.insvr.com/img/circle/300/SGArcticWonders.png>

Hot Linking Game Backgrounds

<https://app-test.insvr.com/bgimg/{gameKeyName}.jpg>

example: <https://app-test.insvr.com/bgimg/SGArcticWonders.jpg>

Launching Games (Transfer and Seamless)

Launch a game by redirecting your player to Habanero with a Token. You will need to familiarise yourself with Section 3 or 4 first (depending on wallet type) >> **See your welcome email for the game launch endpoint to use** <<<

Request parameters

Variable	Description	Example	Notes
brandid	Your brandid	6af6f2f8-0ecd-4829-9bb7-e78abcffe6ef	Required
brandgameid -OR- keyname	guid	c07552ae-a65c-4d7d-93dd-10add80817be	See Gamelidentifiers section
	string	SGAllForOne	
token	Player Token	VBvBXNXhW26opsr4dWiWE If the token is blank, fun mode will be forced.	Seamless API – token is created by you. Max length is 250 chars. Transfer wallet – token returned from Habanero LoginOrCreatePlayer() web method
mode	Fun / Real	fun or real	Note: fun play will not query your API. Any value that is not "fun" will result in real play.
locale	Game UI language	en,zh-CN, ja, fr	See Addendum A for locales
Optional parameters:			
lobbyurl	A URL to return to Or a special postMessage	https://www.yoursite.com or pm-XXX	Games show a lobby button if this parameter is set . Use URLEncode for urls. You may send "pm-XXX" and the game will do a postMessage("pm-XXX") to the parent frame containing the param sent in LobbyUrl. This is useful when loading game in iframe. https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage
historyUrl	A URL to external game history	https://www.yoursite.com	Internal Habanero History will be shown if ShowHistory is enabled in the BO otherwise specify your own history page/url
cashierUrl	A URL to external cashier		If set a cashier button is shown in the game and displayed when insufficient funds occur
segmentkey	Segmenting of players in one brand	"Powercasino360"	Seamless API only. If using one brand on Habanero but your players are from different operators then specify a segmentation key here or in playerdetailresponse. For Transfer wallet specify this in LoginOrCreatePlayer() method
rcinterval	Reality Check interval (seconds)	120	The interval at which to show reality check message to the player. Specified in seconds.
rcsessionelapsed	Length of session so far (seconds)	600	Indicate how long the player has been active on the current session. Specified in seconds.
hideCS	hide Currency Symbol	0 (default) 1	Add hideCS=1 to hide the real money currency symbol from game
sfb	Starting Fun Balance	500	Specify starting balance to use for fun mode
fpl	Fun Play Limit	50	Limit number of fun play game rounds
nofs	No Full Screen	nofs=1	Disable auto fullscreen on android
rsw	Iframe Redirect to self instead of top	rws=1	Can be set in BO, default is to redirect top
hnw	open History in new window	hnw=1	Can be set in BO, default is same window. If it is not set, the history will display in an iframe overlay inside game
cnw	open Cashier in new window	cnw=1	Can be set in BO – default is same window. If it is not set, the cashier will redirect out of game to cashier
hhr	Hide History Replay	hhr=1	Hides the Replay column in the game history.

Game Launch Examples:

Launching a game using **keyname**

```
<habanero_launch_url>?brandid={0}&keyname={1}&token={2}&mode={3}&locale={4}&lobbyurl={5}
```

Launching a game using **fun** (demo) mode (**omit the token and set mode=fun**)

```
<habanero_launch_url>?brandid={0}&keyname={1}&mode=fun&locale={2}&lobbyurl={3}
```

Launching a game using **brandgameid**

```
<habanero_launch_url>?brandid={0}&brandgameid={1}&token={2}&mode={3}&locale={4}&lobbyurl={5}
```

Using iFrames with game

- 1) Consider using a postMessage in the LobbyUrl and subscribing to game events. See the next page.
- 2) If you need to trigger a refresh of the game balance you may add a postMessage
window.frames['youriframeName'].contentWindow.postMessage("refreshBalance", "*");

OR

You can make use of the [Game Javascript API](#) and refer to JS Events "Receive Events" section

- 3) Use the html and script below and:
 - a. Add 'allowfullscreen' or 'allow="fullscreen"' to your iframe tag.
 - b. Add the javascript to: focus the iframe after load so we can capture keyboard input;
 - c. resolve orientation/sizing issues for mobile browsers.

```
<html>
<head>
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1, minimum-
scale=1, maximum-scale=1, user-scalable=no">
<style>
  body {
    margin: 0;
  }
</style>
<script>
  window.onload = function () {
    document.getElementById('embedgameIframe').focus();
    window.addEventListener('resize', resizeIframe);
    window.addEventListener('orientationchange', resizeIframe);

    function resizeIframe() {
      var iframe = document.getElementById('embedgameIframe');
      var parent = iframe.parentNode;
      if (parent) {
        var rect = parent.getBoundingClientRect();
        iframe.style.width = rect.width + 'px';
        iframe.style.height = rect.height + 'px';
        iframe.style.left = '0px';
        iframe.style.top = '0px';
        iframe.style.position = 'absolute';
      }
    }
    resizeIframe();

    document.documentElement.style.width = "100%";
    document.documentElement.style.height = "100%";
    document.documentElement.style.overflow = 'hidden';
    document.body.style.width = "100%";
    document.body.style.height = "100%";

    var viewport = document.querySelector('meta[name=viewport]');

    if (!viewport) {
      var metaTag = document.createElement('meta');
      metaTag.name = 'viewport';
      metaTag.content = 'width=device-width, height=device-height, initial-scale=1, minimum-scale=1,
maximum-scale=1, user-scalable=no';
      document.getElementsByTagName('head')[0].appendChild(metaTag);
    }
    else {
      viewport.setAttribute('content', 'width=device-width, height=device-height, initial-scale=1,
minimum-scale=1, maximum-scale=1, user-scalable=no');
    }
  };
</script>
</head>
<body>
  <div style="position:relative; width:100%; height:100%;">
    <iframe id="embedgameIframe" src="HABANERO_LAUNCH_URL" allowfullscreen allow="fullscreen"
scrolling="no" frameBorder="0" style="margin: 0; padding: 0; white-space: nowrap; border: 0; width:100%;
height:100%"></iframe>
  </div>
</body>
```

Using PostMessage with game close / return to lobby

Postmessages are used with iframes. When a postmessage (pm-XXX) is set in the LobbyUrl, it will show the home/lobby button in-game and, once clicked, will send your postmessage on the page containing the iframe. You need to listen and execute commands. Below is an example:

Example Launch URL:

<habanero_launch_url>?brandid={0}&keyname={1}&token={2}&mode={3}&locale={4}&lobbyurl=pm-CloseMyGame

```
<script>
window.addEventListener("message", receiveMessage, false);
function receiveMessage(event)
{
  if (event.data == "pm-CloseMyGame")
  {
    //custom code here
    // eg. close iframe, etc
  }
}
</script>
```

Game Events API

Download the PDF here: [Game Javascript API](#)

When loading a game in an iframe you may subscribe to events that the game sends such as when the game starts, ends, balance changes, win amounts, bet amounts etc.

Habanero Game Lobby

Habanero offers a game lobby site, where all the games configured for the brand is listed.

How to Render/Go to the Lobby site:

Players can go to the Lobby using 2 approaches:

1. **Launching with special keyname: HBGAMELOBBY** - If a game is launched using the keyname=HBGAMELOBBY or brandgameid=HBGAMELOBBY we will redirect the player to lobby.

example:

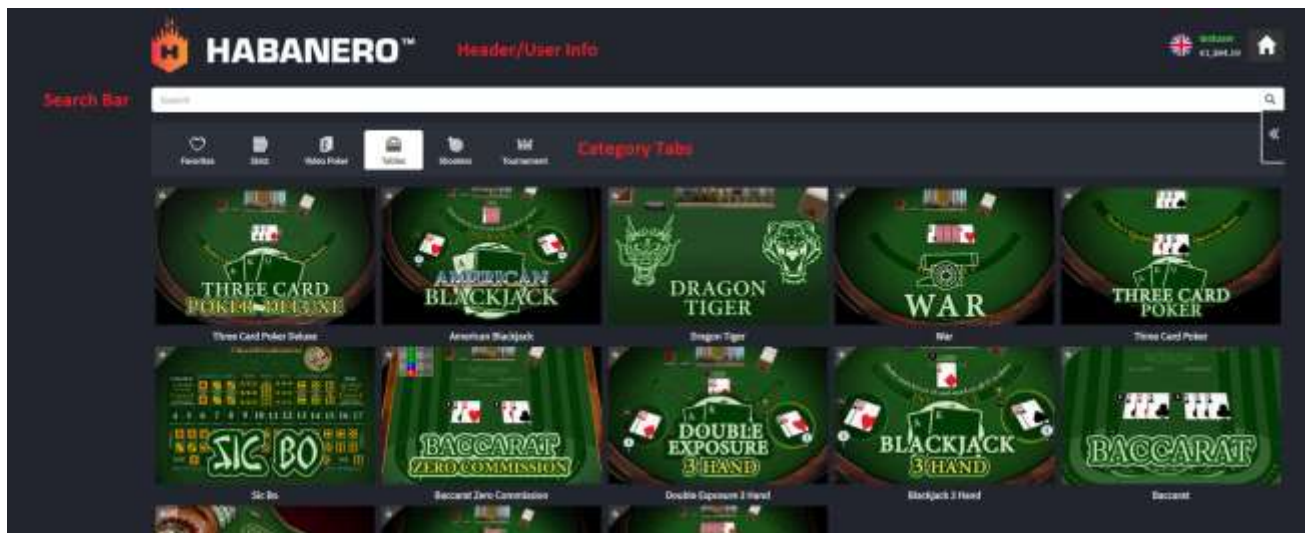
<habanero_launch_url>?brandid=yourbrandid&keyname=HBGAMELOBBY&token=yourtoken&mode=real&locale=en&lobbyurl=yourencodedurl

2. **Force Redirection from Backoffice** - If configured from the Backoffice (default is off), players will be redirected to the Habanero Game Lobby when clicking the home button in-game. The lobby will have a button to redirect players to the original lobby (if lobbyurl is set in launchurl by the casino - more details in Habanero API, page 11, Launching Games).

Optional Parameters for Lobby Layout Adjustment:

These url parameters can be used to adjust lobby layout:

Name	Description
hidesearch	Hide Lobby site Search bar
hideinfo	Hide Lobby site User Info/header
hidetab	Hide the Game Category tabs
lobbyindex	Specify the default tab to open on load. Works only when HideTab = 1



URL Example:

<habanero_launch_url>??brandid=yourbrandid&keyname=HBGAMELOBBY&hidesearch=1&hideinfo=1&hidetab=1&lobbyindex=1&token=yourtoken&mode=real&locale=en&lobbyurl=yourencodedurl

-- this page intentionally left blank --

Habanero Web Service (SOAP or JSON)

Purpose of this Web service:

Transfer Wallet - create players, authentication, deposit, withdraw, game results, and more.

Seamless Wallet – first implement the Seamless Wallet API in the next section. Return later for bonusing and other methods which may be of use.

Connecting to service:

Credentials:

Please use the **BrandId** and **APIKey** which has been sent to you in your Welcome Email.

IP Whitelist:

Please provide a list of IP Addresses which will be white listed to access the web service. Any IP Address not configured in Habanero will be rejected unless you setup your Brand to skip the IP check.

Endpoint:

Please see your welcome email. The end points are as follows:

SOAP webservice: <https://ws-{domainname}/hosted.asmx?WSDL>

JSON service: <https://ws-{domainname}/jsonapi/{methodname}>

Recommended Tools:

For testing and exploring the JSON API we recommend using POSTMAN - <https://www.getpostman.com/>

For POSTMAN JSON Documentation- <https://haba88.com/go/postman>

Notes:

DO NOT USE A STRICT DATA CONTRACT SERIALIZER

We may add additional fields at any time. If you reject additional or undocumented fields, your implementation *will* break.

This typically applies to Java.

Using SOAP or JSON?

If you are using the Microsoft .NET stack, then the webservice is the easiest to implement. For other languages using the JSON service is recommended.

JSON service usage:

Perform an HTTP POST with the JSON request object and we will respond with a JSON response object.

NOTE: All response dates use ISO 8601 format

PHP Example: GetGames() request

```
<?php
//get important fields in array then convert to JSON string.
$jsonRequestBodyObject = array("BrandId"=><your brandId>, "APIKey"=><your APIKey>);
$data_string = json_encode($jsonRequestBodyObject);

//POST data to the JSON Web service URL
$ch = curl_init("https://ws-test.insvr.com/jsonapi/getgames");
//url format is <webservice url>/jsonapi/<methodname>

curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
curl_setopt($ch, CURLOPT_POSTFIELDS, $data_string);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json'));
$result = curl_exec($ch);
//convert JSON string response to object
$methodResult = json_decode($result);
foreach ($methodResult->Games as $game)
{
    echo "Keyname: ".$game->KeyName.", Game Name: ".$game->Name."<br/>";
}
?>
```

Eg: GetGames() response:

```
{
  "Games": [
    {
      "BrandGameId": "887fa4dd-8c19-48fa-82d2-1af42c7adc64",
      "Name": "Double Bonus Poker 100 Hand",
      "KeyName": "DoubleBonusPoker100Hand",
      "IsNew": false,
      "DtAdded": "2009-08-03T12:35:38.953Z",
      "DtUpdated": "2009-08-03T12:35:38.953Z",
      "GameTypeId": 6,
      ...<additional information hidden>
    }
  ]
}
```

Service Error Codes

If the request fails, the service throws a server 400 or 500 http error with additional info in the following Response Headers:

"X-HABA-ErrorCode" - numeric code detailed in grid below

"X-HABA-ErrorName" - Name detailed in grid below

"X-HABA-ErrorDetail" - Additional information about the error

Code	Name	
1	InvalidRequest	See the ErrorDesc for details on missing/wrong parameters
2	SecurityError	IP Whitelisting error, wrong credentials etc
3	SystemMaintenanceMode	System is in Maintenance Mode
4	RateLimited	Your requests are rate limited. Please slow down rate of requests
5	GeneralError	See ErrorDesc for details
6	DataDelayedRetry	There is a delay in reporting data. Please retry the same request later
7	GroupApiQueryDisabled	Contact support to enable this method
8	ConfigurationRequired	Contact support to enable feature
9	PlayerNotFound	The player not found for the method

Body Cookies (1) Headers (10) Test Results		
KEY	VALUE	
Cache-Control	no-cache	
Pragma	no-cache	
Content-Type	application/x-javascript; charset=utf-8	
Expires	-1	
Server	Microsoft-IIS/10.0	
X-HABA-ErrorCode	1	
X-HABA-ErrorName	InvalidRequest	
X-HABA-ErrorDetail	Service: ws.local.insvr.com - BrandId '7cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef' does not exist on this server.	
Date	Mon, 14 Oct 2019 05:39:11 GMT	
Content-Length	147	

API Methods

LoginOrCreatePlayer (Transfer Wallet Only)

[POSTMAN EXAMPLE](#)

In Transfer Wallet mode you must create players in the Habanero Database and/or Log them into Habanero before launching a game. This is NOT for Seamless wallet – see the Player Detail Request in seamless section.

- a) Use the **LoginOrCreatePlayer()** method which will create a player if the username does not exist. If the username does exist and the password matches, the player's details will be updated with provided information. The player will then be logged in and a session Token is returned. You will use this returned token to launch a game.

LoginOrCreatePlayerRequest Object:

REQUIRED FIELDS:		
BrandId	String	(ALWAYS REQUIRED FOR ALL WEBSERVICE CALLS)
APIKey	String	(ALWAYS REQUIRED FOR ALL WEBSERVICE CALLS)
PlayerHostAddress	String	Player IP Address (PLEASE PROVIDE)
UserAgent	String	Send the players browser useragent string so we can display the Device in reports.
KeepExistingToken	Bool	Default True : If the player is already logged in, do not create a new session token. Set to False if you want to invalidate other sessions
Username	String	Max length of 150 characters
Password	String	
CurrencyCode	String	(See Addendum of Currency Codes)
OPTIONAL FIELDS:		
PlayerRank	Int	Integer matching the rank of a Habanero Playerclass configured in Backoffice
FirstName	String	
LastName	String	
IdentityNumber	String	Passport, IC etc..
DOB	String	Format yyyyMMdd
Address1	String	
Address2	String	
State	String	
City	String	
PostalCode	String	
CountryCode	String	
LanguageCode	String	(See Addendum of Locale codes)
Gender	String	M, F or U (unknown)
EmailAddress	String	
TelNumber	String	
SegmentKey	String	NB! Use this field to specify an operatorid or casino for the player when you only have 1 Habanero BrandId but have players from different casinos inside this one brand. Up to 40 chars string.

LoginUserResponse Object:

Authenticated	Bool	Indicates if the user is logged in / authenticated after the request. False if password is incorrect
PlayerId	String	The internal <guid> Primary Key in Habanero database (may be of use)
BrandId	String	The brandid of the player
BrandName	String	The name of the brand
Token	String	The session Token for the player. Use this to launch game
RealBalance	String	The player's Real money balance
CurrencyCode	String	The player's Currency Code
CurrencySymbol	String	The player's currency symbol
PlayerCreated	String	True if IsNewPlayer, else False (player existed already)
HasBonus	Bool	Indicated if player has an active Bonus
BonusBalance	Decimal	The bonus balance if an active bonus is present
BonusSpins	Int	If bonus is of type Free Spins, then this is how many free spins left
BonusGameKeyName	String	The keyname of the game which has the bonus
BonusPercentage	Decimal	The % progress to completion of the bonus wagering
BonusWagerRemaining	Decimal	How much needs to be wagered on this bonus till completion
PointBalance	Decimal	<nullable> decimal indicating player points
Message	String	Informational message

UpdatePlayerPassword (Transfer Wallet Only)

[POSTMAN EXAMPLE](#)

Update an existing player's password

[UpdatePlayerPasswordRequest](#) **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player username
NewPassword	String	New password to use

[UpdatePlayerPasswordResponse](#) **Object:**

Success	String	Indicates if Successful
Message	String	Informational message

DepositPlayerMoney (Transfer Wallet Only)

POSTMAN EXAMPLE

Deposit money into player's wallet. **If the username does not exist, then a new player will be created.** This means you do not need to create a player before they can Transfer into the Habanero wallet. This will not log the player in.

If you wish to update the player's record, then populate the fields in LoginOrCreatePlayer before launching a game.

Tip: Use the RequestId field to ensure that a Deposit is only made once. RequestId must differ between Deposit and Withdrawals.

DepositPlayerMoneyRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player username
Password	String	Player password
CurrencyCode	String	See Addendum of Currency Codes
Amount	Decimal	
RequestId	String	(optional) Specify a GLOBALLY UNIQUE string up to 50 characters for this request. Habanero will only process the deposit once. Must also differ from Withdraw RequestId value so prefix if required.

MoneyResponse Object:

Success	Bool	Indicates if Deposit Successful
Amount	Decimal	Amount deposited
RealBalance	Decimal	Player's balance after Deposit
TransactionId	String	The unique Habanero <guid> representing the deposit
CurrencyCode	String	The player's Currency Code
Message	String	Informational message

WithdrawPlayerMoney (Transfer Wallet Only)

POSTMAN EXAMPLE

Withdraw money from player's wallet by specifying a Negative amount.

Tip: Use the RequestId field to ensure that a Withdrawal is only made once. RequestId must differ between Deposit and Withdrawals.

Tip: Easily remove all funds by specifying WithdrawAll = true. The response object will include the amount that was deducted.

WithdrawPlayerMoneyRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player username
Password	String	Player password
CurrencyCode	String	See Addendum of Currency Codes
Amount	Decimal	Specific amount (negative) or:
WithdrawAll	Bool	Set true to withdraw all funds
RequestId	String	(optional) Specify a GLOBALLY UNIQUE string up to 50 characters for this request. Habanero will only process the withdrawal once. Must also differ from Deposit RequestId value so prefix if required.

MoneyResponse Object:

Success	Bool	Indicates if Withdrawal Successful
Amount	Decimal	The amount withdrawn (useful if using WithdrawAll=true)
RealBalance	Decimal	Player's balance after Withdraw
TransactionId	String	The unique Habanero <guid> representing the withdrawal
CurrencyCode	String	The player's Currency Code
Message	String	Informational message

QueryTransfer (Transfer Wallet Only)

POSTMAN EXAMPLE

Query a RequestId from DepositPlayerMoney() or WithdrawPlayerMoney() to determine status of a transfer.

QueryTransferRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player username
RequestId	String	The requestid specified in DepositPlayerMoney() or WithdrawPlayerMoney()

QueryTransferResponse Object:

Success	String	Boolean indicating if transaction was received
PlayerId	String	The internal <guid> Primary Key in Habanero database for player
Username	String	The username of the player
CurrencyCode	String	Currency of player
DtAdded	String	The Date of the transaction
Amount	Decimal	Negative for a withdrawal, Positive for deposit
BalanceAfter	Decimal	The player balance after the transaction was committed
TransactionId	String	The unique Habanero <guid> representing the deposit

QueryPlayer (Transfer Wallet Only)

POSTMAN EXAMPLE

Query a player to return the player's current balance, token and bonus info.

QueryPlayerRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	
Password	String	

QueryPlayerResponse Object:

Found	Bool	Boolean indicating if record found (Player Exists). If token is empty, player is not logged in.
PlayerId	String	The internal <guid> Primary Key in Habanero database (may be of use)
BrandId	String	The brandid of the player
BrandName	String	The name of the brand
Token	String	The session Token for the player. Use this to launch game. If empty then player is not logged in.
RealBalance	Decimal	The player's Real money balance
CurrencyCode	String	The player's Currency Code
CurrencySymbol	String	The player Currency Symbol
HasBonus	Bool	Indicated if player has an active Bonus
BonusBalance	Decimal	The value of the active bonus balance
BonusSpins	Int	If bonus is of type Free Spins, then this is how many free spins left
BonusGameKeyName	String	The keyname of the game which has the bonus
BonusPercentage	Decimal	The % progress to completion of the bonus wagering
BonusWagerRemaining	Decimal	How much needs to be wagered on this bonus till completion
Message	String	Message in case Found = false

LogOutPlayer (Transfer and Seamless)

[POSTMAN EXAMPLE](#)

Logout a player. Should not be required for Seamless wallet. Additionally, players can be logged out after a configured session timeout period (set in the Backoffice)

LogoutPlayerRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player username
Password	String	Player password

LogoutPlayerResponse Object:

Success	Bool	Indicates if player was logged out.
Message	String	Informational message

LogoutAllPlayersInBrand (Transfer and Seamless)

[POSTMAN EXAMPLE](#)

All logged in players for the BrandId will be logged out.

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	

SetMaintenanceMode (Seamless only)

You can enable maintenance mode for ALL the brands in your Group. This will gracefully end all play and prevent launching of games. If you have more than 1 Brand in your configured Group, all of them will be disabled. Group WS Reporting must be enabled in Backoffice to use this method.

Request:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Enabled <bool>	Bool	True = Maintenance ON false = Maintenance OFF

Response:

Message	String	You should inspect this message to determine if which services have accepted your command
---------	--------	---

GetBrandIncompleteGames (Transfer and Seamless)

POSTMAN EXAMPLE

Get a list of games which are In Progress or have an Error due to Seamless wallet issue.

Request:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	

Response:

PlayerId	guid	Internal Habanero PlayerId
Username	string	Your user/id for player
GameInstancelId	guid	
FriendlyId	Int64	Friendly Game InstancelId
GameName	string	
GameKeyName	string	
Provider	string	Name of game provider
BrandGameId	guid	
DtStarted	date	Date the game started
Stake	money	
Payout	money	
GameStateId	int	2 = In Progress (player can resume this themselves) 5 – Transaction Debit Error (Seamless wallet error) 6 – Transaction Credit Error (Seamless wallet error) 9 – Transaction DebitCredit Error (Seamless wallet error)
GameStateName	string	Description of GameStateId
PlayerTypeId	Int	1 – Transfer Player 3 – Seamless Player 4 – Guest Player
BuyFeatureId	int	A non 0 number means a player used the "Buy Feature" option. If you want a mapping of the BuyFeatureId, please refer to Addendum L

GetPlayerResumeGames (Transfer and Seamless)

POSTMAN EXAMPLE

Get a list of incomplete games for a specific player.

Request:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player's username

Response:

GameName	String	Name of the game
Keyname	String	Keyname for the game
GameInstanceId	String	The internal Habanero record for the gameinstance
FriendlyId	Int64	Integer reference to GameInstanced (This is visible to player in game)
GameTypeId	Int	SEE ADDENDUM D

Example:

```
[
  {
    "FriendlyId": 1300257,
    "GameInstanceId": "176ed6a8-47d7-e811-8469-f44d30070f6d",
    "Keyname": "BlackJack3H",
    "GameName": "Blackjack (3 Hand)",
    "GameTypeId": 4,
    "DtStarted": "2018-10-24T04:45:48.257"
  }
]
```

ExpirePlayerGames (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Expire all games for a specific player

Request:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player's username

Response:

Success	Bool	Indicates if request was successful
Message	String	Request Information
GamesExpired	Int	Total number of games expired

Example:

```
{
  "Success": true,
  "Message": "Player incomplete games expired succesfully",
  "GamesExpired": 1
}
```

ExpireBrandGames (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Expire all games for a brand

Request:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	

Response:

Message	String	Request Information
GamesExpired	Int	Total number of games expired

Example:

```
{
  "Message": "Incomplete games expired succesfully",
  "GamesExpired": 1
}
```

GetGames (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Retrieve information about the games configured in your brand. This should not be needed if you import our list of games from the excel sheet provided.

GameRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	

Response: (some fields omitted)

FieldName	Type	Description
BrandGameId	Guid	See GameIdentifiers on previous page
KeyName	String	See GameIdentifiers on previous page
Name	String	English Name of the Game
ReportName	String	For your INTERNAL usage only - (includes the Name + RTP and lines detail) – Do not display to player
LineDesc	String	For slots, a description of the lines – "25 Lines, 50 Lines, 243 Ways" etc.
DtAdded	Datetime	The Date the game was added to the brand
DtRTM	DateTime	The Date the game was released to the market
GameTypeId	Int	GameType of the game as per GetGameTypes() or SEE ADDENDUM D
RTP	Decimal	The expected RTP of the game
MobileCapable	Boolean	Indicates if the game supports mobile (HTML5) All games will respond true
TranslatedNames	List< GameTranslationDTO >	List of translated names for the game where GameTranslationDTO contains: Locale – string (e.g. "zh-CN") Translation – string
SupportBonusFS	Boolean	If the game supports free spins bonus type
Coins	Int	Number of Coins of the game. Defaults to 1 for non-video slot games.
ExProv	Boolean	External Provider? false = Habanero game true = External Provider with a specified ProductExternalId
ProductExternalId	String <nullable>	blank/null = Habanero otherwise as per addendum SEE ADDENDUM I

IMPORTANT NOTE: See Addendum D for GameTypes and note: GameType 7 / Gamble cannot be launched from outside a game. It is provided in the GetGames() list so that you may associate data to the game. Do not show the Game as an option for a player to Launch. We also do not provide the game logos gamble games.

GetGameParameters (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Retrieve game parameter information (NOTE: this is only available for Slot Games)

GetGameParameterRequestObject:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
KeyName	String	Habanero Game Keyname
CurrencyCode	String	(OPTIONAL) 3 Letter ISO-Code

GameParametersDTO[] Response:

FieldName	Type	Description
CurrencyCode	String	CurrencyCode
KeyName	String	See GameIdentifiers on previous page
GameCoinValue	Int	The specified Game Coin Value
GameTypeId	Int	GameTypeId of the game SEE ADDENDUM D NOTE: Only GameTypeId 11 is supported
StakeIncrement	List<decimal>	Game Stake Increment
StakeDefault	decimal	Default stake for the game parameter
MaxPayLimit	decimal	Maximum Pay Limit of the parameter – 0 is unlimited
CoinsIncrement	List<int>	Coins increment used for this game
IsGeneric	Boolean	Indicates if the game parameter is the base generic configuration in the brand (no currency and no category)

GetGameReplayUrl (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Retrieve game replay URL for a given GameInstanceId or FriendlyId (NOTE: this may return a null ReplayUrl if the game does not support Replay)

GameReplayRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Either:		
GameInstanceId	String	Habanero GUID Game Round Id
FriendlyId	String	Habanero Friendly Game Round Id

GameReplayDTO Response:

FieldName	Type	Description
ReplayUrl	String	Replay URL for the game round. Can be null if the game does not support replays.

GetJackpots (Seamless and Transfer Wallet)

GetAllJackpotsInAllBrands (return all data for Group)

POSTMAN EXAMPLE

All jackpots are setup in configurable **base currency**. If EUR is your most popular currency, then use EUR as the base currency. Habanero will do a forex conversion to/from the player's currency and display the correct value/currency in the player's game.

GetAllJackpotsInAllBrands

Type of Jackpot

JackpotTypeId	Jackpot Type Name
1	Random Mini
2	Random Minor
3	Random Major
4	Random Grand
8	Race
15	Coin Size Jackpot

JackpotInfoRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	

JackpotInfoDTO[] **Result:**

FieldName	Type	Description
brandid	String	Your brandid
jackpotid	String	Unique primary key for Jackpot
jackpottypeid	Int	Type of Jackpot
jackpottypename	String	String name of type of jackpot
currencycode	String	3 Letter ISO currency code for the base currency of jackpot
currencysymbol	String	The currency symbol for the base currency of the jackpot
currentvalue	Decimal	The value of the jackpot
clawtotal	Decimal	Amount of money clawed back towards the jackpot. When this amount > the currentvalue, then the jackpot is profitable.
startingvalue	Decimal	The starting value of the jackpot.
consumeperc	Decimal	Percentage RTP of the jackpot
randfreqperc	Decimal	Random event frequency as per jackpot setup
markupperc	Decimal	Jackpot Markup percentage
blockuntilfunded	Bool	Specifies if this jackpot is set to only pay after funded
Conversions	[array]	The jackpot base currency converted into other currencies <ul style="list-style-type: none"> • Currencycode <string> • Currencysymbol <string> • Currencyexponent <decimal> • Currentvalue <decimal> • Exrate <decimal> • Increment <decimal>
gamebrandgameids	String[]	List of game BrandGameId's which are in this jackpot
gamekeynames	String[]	List of game Keynames which are in this jackpot
dtlastwin	String?	(NULLABLE) Last date when jackpot has been won

The following are Race specific fields (JackpotTypeId = 8)

FieldName	Type	Description
dtracestartutc	String?	(NULLABLE) Date when Race will start
dtraceendutc	String?	(NULLABLE) Date when Race will end
raceopen	Bool?	(NULLABLE) indicates if the jackpot race is currently open

Example:

```
[
  {
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef",
    "jackpotid": "92b64147-8c3c-e811-842e-f44d30070f6d",
    "jackpotgroupid": "46e0fa05-8c3c-e811-842e-f44d30070f6d",
    "jackpottypename": "Race",
    "jackpottypeid": 8,
    "dtlastwin": "2018-04-10T07:01:36.947",
    "cid": 3,
    "currencycode": "EUR",
    "currencysymbol": "€",
    "currencyexponent": 2,
    "currentvalue": 0.0625,
    "increment": 0.1,
    "bettotal": 12.5,
    "clawtotal": 0.0625,
    "startingvalue": 0,
    "consumeperc": 0.5,
    "randfreqperc": 0,
    "markupperc": 0,
    "blockuntilfunded": false,
    "raceopen": false,
    "dtracestartutc": "2018-04-11T06:00:00Z",
    "dtraceendutc": "2018-04-11T11:00:00Z",
    "conversions": [
      {
        "currencycode": "CNY",
        "currencysymbol": "¥",
        "currencyexponent": 2,
        "currentvalue": 0.484446169375,
        "exrate": 7.75113871,
        "increment": 0.775113871
      },
      ...
    ],
    "gamebrandgameids": [
      "51ff1b3c-a6c3-496f-bf18-fe9f881110d4"
    ],
    "gamekeynames": [
      "SGLondonHunter"
    ]
  },
  ...
]
```

GetJackpotsByGroup (Seamless and Transfer Wallet)

POSTMAN EXAMPLE

Returns all brand's configured jackpot for a specific Jackpot Group ID, regardless of status (Active or Inactive)

JackpotsByGroupRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
JackpotGroupId	Guid	Specific Jackpot Group ID to query

JackpotInfoDTO[] **Result:**

FieldName	Type	Description
brandid	String	Your brandid
jackpotid	String	Unique primary key for Jackpot
jackpottypeid	Int	Type of Jackpot
jackpottypename	String	String name of type of jackpot
currencycode	String	3 Letter ISO currency code for the base currency of jackpot
currencysymbol	String	The currency symbol for the base currency of the jackpot
currentvalue	Decimal	The value of the jackpot
clawtotal	Decimal	Amount of money clawed back towards the jackpot. When this amount > the currentvalue, then the jackpot is profitable.
startingvalue	Decimal	The starting value of the jackpot.
consumeperc	Decimal	Percentage RTP of the jackpot
randfreqperc	Decimal	Random event frequency as per jackpot setup
markupperc	Decimal	Jackpot Markup percentage
blockuntilfunded	Bool	Specifies if this jackpot is set to only pay after funded
Conversions	[array]	The jackpot base currency converted into other currencies <ul style="list-style-type: none"> • Currencycode <string> • Currencysymbol <string> • Currencyexponent <decimal> • Currentvalue <decimal> • Exrate <decimal> • Increment <decimal>
gamebrandgameids	String[]	List of game BrandGameld's which are in this jackpot
gamekeynames	String[]	List of game Keynames which are in this jackpot
dtlastwin	String?	(NULLABLE) Last date when jackpot has been won

The following are Race specific fields (JackpotTypeId = 8)

FieldName	Type	Description
dtracestartutc	String?	(NULLABLE) Date when Race will start
dtraceendutc	String?	(NULLABLE) Date when Race will end
raceopen	Bool?	(NULLABLE) indicates if the jackpot race is currently open

GetBrandTournamentEvent (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Get a list of tournament events and winners.

BrandTournamentEventRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
EventKey	String	Unique Event ID
OR		
DtStartUTC	String	UTC Start date (yyyyMMddHHmmss format)
DtEndUTC	String	UTC End date (yyyyMMddHHmmss format)

List<TournamentsDTO>

FieldName	Type	Description
EventKey	String	Unique Event ID
Name	String	Name/Description of promotion
DtStart	String	Event start
DtEnd	String	Event end
IsCompleted	Boolean	Indicates if tournament is completed and winners are announced
Winners <Array>	Array of Winners	

Winners Array

FieldName	Type	Description
WinnerId	String	Unique Winner ID
Username	String	Player username
AmountAwarded	Decimal	Amount awarded to player (in player currency)
PlayerCurrency	String	Currency code of the player and the AmountAwarded
OriginalAmount	Decimal	The original prize amount (in base currency of tournament)
OriginalCurrency	String	Currency code of the tournament
DtAwarded	String	Date of award
Rank	Int	The position of the player in the event

Example:

```
[
  {
    "EventKey": "18",
    "Name": "XYZ CNY 2018",
    "DtStart": "2018-01-23T00:00:00Z",
    "DtEnd": "2018-01-29T00:00:00Z",
    "IsCompleted": true,
    "Winners": [
      {
        "Username": "YVPCB9ZB",
        "AmountAwarded": 7800.04,
        "PlayerCurrency": "CNY",
        "OriginalAmount": 1000,
        "OriginalCurrency": "EUR",
        "DtAwarded": "2018-01-30T08:38:34.147Z",
        "Rank": 1
      }
    ]
  }
]
```

GetBrandCCEvents (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Get a list of central tournament events and winners.

NOTE: Dates reference the DtStart of events

BrandCCEventRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
EventKey	String	Unique Event ID
OR		
DtStartUTC	String	UTC Start date (yyyyMMddHHmmss format)
DtEndUTC	String	UTC End date (yyyyMMddHHmmss format)

List<CCTournamentsDTO>

FieldName	Type	Description
EventKey	String	Unique Event ID
Name	String	Name/Description of promotion
DtStart	String	Event start
DtEnd	String	Event end
IsCompleted	Bool	Indicates if tournament is completed and winners are announced
Winners <Array>	Array of Winners	
TournamentEventTypeId	short	Refer to Addedum J
TournamentPrizeTypeId	short	Refer to Addedum K

Winners Array

FieldName	Type	Description
WinnerId	String	Unique Winner ID
Username	String	Player username
AmountAwarded	Decimal	Amount awarded to player (in player currency)
PlayerCurrency	String	Currency code of the player and the AmountAwarded
OriginalAmount	Decimal	The original prize amount (in base currency of tournament)
OriginalCurrency	String	Currency code of the tournament
DtAwarded	String	Date of award
Rank	Int	The position of the player in the event
BetMultiplier	Int?	Bet multiplier if winner is from Bet Multiplier prize type (-1 if not valid)
FSCount	Int?	Free Spin Count (-1 if not valid)
FSValue	Decimal?	Free Spin value per spin (-1 if not valid)

GetBrandCCWinners (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Get a list of central tournament winners.

NOTE: Dates reference the winning date (DtAwarded)

BrandCCWinnersRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date (yyyyMMddHHmmss format)
DtEndUTC	String	UTC End date (yyyyMMddHHmmss format)

List< CCTournamentWinnerDTO>

FieldName	Type	Description
WinnerId	String	Unique Winner ID
Username	String	Player username
AmountAwarded	Decimal	Amount awarded to player (in player currency)
PlayerCurrency	String	Currency code of the player and the AmountAwarded
OriginalAmount	Decimal	The original prize amount (in base currency of tournament)
OriginalCurrency	String	Currency code of the tournament
DtAwarded	String	Date of award
Rank	Int	The position of the player in the event
BetMultiplier	Int?	Bet multiplier if winner is from Bet Multiplier prize type (-1 if not valid)
FSCount	Int?	Free Spin Count (-1 if not valid)
FSValue	Decimal?	Free Spin value per spin (-1 if not valid)
TournamentInfo	TournamentInfo	Details of the event

TournamentInfo Object

FieldName	Type	Description
EventKey	String	Unique Event ID
Name	String	Name/Description of promotion
DtStart	String	Event start
DtEnd	String	Event end
IsCompleted	Bool	Indicates if tournament is completed and winners are announced
TournamentEventTypeId	short	Refer to Addedum J
TournamentPrizeTypeId	short	Refer to Addedum K

GetPlusPlayErrors (Seamless Wallet)

POSTMAN EXAMPLE

Get a list of Errors from Seamless wallet issues

GetPlusPlayErrorsRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)
ShowResolved	Bool	Include resolved errors or not

PlusPlayTransactionsDT0[] **Result:**

```
[
  {
    "ThirdPartyTransactionId": "85eb1cd7-06d1-e811-8289-cc2f711c5021",
    "PlayerUsername": "153",
    "GameInstanceId": "84eb1cd7-06d1-e811-8289-cc2f711c5021",
    "FriendlyId": "1449152",
    "DtCompleted": "2018-10-16T05:56:17.333Z",
    "DtStarted": "2018-10-16T05:46:35.48Z",
    "IsDC": false,
    "DebitTransferId": "84eb1cd706d1e8118289cc2f711c5021",
    "DebitAmount": -100,
    "AfterDebitGameStateMode": 1,
    "GameStateId": 5,
    "GameState": "Transaction Debit Error",
    "AttemptCount": 5,
    "Message": " http://seamless.ap.ngrok.io/api.ashx?auths - The remote server returned an error: (500) Internal Server Error.",
    "IsResolved": false
  }
]
```

Important Fields:

ThirdPartyTransactionId <string> - This must be used with the ResolvePlusPlayError() method

GameInstanceId/FriendlyId – The usual identifiers for a game round

IsDC <bool> - Indicates if this is a Debit & Credit – if so there will be a DebitTransferId, DebitAmount and CreditTransferId and Credit Amount in the response.

DebitTransferId <nullable string> - The transferId for the Debit (if present)

DebitAmount <nullable decimal> - The debit amount in Negative

CreditTransferId <nullable string> - The transferId for the Credit (if present)

CreditAmount <nullable decimal> - The credit amount

GameStateId <int> - The status of the game – this is important to determine how to resolve the game

- 2 = In Progress (only possible if resolved)
- 3 = Completed (only possible if resolved)
- 8 = Refunded Debit Reversed (only possible if resolved)
- 10 = Void Debit not done (only possible if resolved)
- 5 = Transaction Debit Error (there was a problem with a debit)
- 6 = Transaction Credit Error (there was a problem with the credit)
- 9 = D&C Error (there was a problem during a Debit&Credit call)

ResolvePlusPlayError (Seamless Wallet)

POSTMAN EXAMPLE

Specify a ThirdPartyTransactionId as received from GetPlusPlayErrors() method

PlusPlayResolveRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
ThirdPartyTransactionId	String	
RefundStatus	int<null>	1 = Refunded, 2 = Refund not required since debit was not done
DebitStatus	int<null>	0 = debit NOT success, 1 = debit SUCESS
CreditStatus	int<null>	1 = credit WAS success (this is only resolution possible)

According to the **GameStateId** from GetPlusPlayErrors() you should specify the RefundStatus or DebitStatus or CreditStatus.

5 - Transaction Debit Error

Check if the DebitTransferId was executed on your side

- if the debit **was performed**, you must **reverse** it and send RefundStatus = 1
- if the debit was **not** performed send RefundStatus = 2

6 - Transaction Credit Error

Check if the CreditTransferId was executed on your side

- You must commit the credit on your side and resolve with CreditStatus = 1

9 - D&C Error

Check if the DebitTransferId was done. If not done send DebitStatus = 0 and the game will be voided.

Continue rolling forward the Credit. Check if the CreditTransferId was done and ensure that you commit / perform the credit on your side.

Then send DebitStatus = 1 and CreditStatus = 1

Note: If all three values are sent as null or "-1" - we will resend the transaction to your seamless API and try to resolve it as per the usual retry scenario.

Jackpot Contribution Reports

ReportJackpotContribution() and ReportJackpotContributionPerGame()

POSTMAN EXAMPLE

Habanero jackpots are self-funding – no reconciliation is required.

This method returns a dataset contribution information for the requested BrandId and Date Range.

The DtStartUTC and DtEndUTC are string parameters and must be in the format of **yyyyMMddHHmmss** in UTC time zone. The DtStartUTC is inclusive and DtEndUTC is exclusive. (eg. where date >= dtStartUtc and date < dtEndUtc)

JackpotId	GUID identifier of the Jackpot
JackpotName	Configured name of Jackpot
JackpotTypeId	See 7.1
BrandId	
BrandName	
FundedCurrency	The currency of the player
FundedIncrement	Increment amount in player currency
FundedClaw	The clawback (contribution) in player currency
ConvertedCurrency	The base currency of the Jackpot
ConvertedIncrement	Increment amount in jackpot currency
ConvertedClaw	The clawback (contribution) in jackpot currency

Definitions:

Clawback (Contribution): The Clawback is the full contribution amount from the bet. E.g. if \$10 bet is made and the jackpot consumes 0.5%, then the Clawback is \$0.05. The operator is billed for the Clawback.

Increment: The Increment is a percentage of the Clawback, which is added to the current value of the Jackpot and is provided for informational purposes only.

The Clawback is always more than the Increment, allowing the jackpot to recover the initial Seed/Starting amount. Once the clawback is equal or greater than the jackpots current value, the WAN jackpot can be won. At this point onwards the jackpot is "profitable" since it has now clawed back the initial Seed value as well as the amount the jackpot was incremented by.

Example:

Assume a brand contains 2 players, one using CNY currency and another USD. The Jackpot is always based in one currency. Assume here it is configured in CNY.

<p>The USD player would generate the following data:</p> <p>FundedCurrency: USD (the player's currency)</p> <p>FundedIncrement: 10</p> <p>FundedClaw: 20</p> <p>ConvertedCurrency: CNY (the WAN jackpot currency)</p> <p>ConvertedIncrement: 60</p> <p>ConvertedClaw: 120</p> <p>The USD amounts are converted into CNY using the exchange rate at the time of the contribution.</p>	<p>The CNY player would generate the following data:</p> <p>FundedCurrency: CNY (the player's currency)</p> <p>FundedIncrement: 50</p> <p>FundedClaw: 100</p> <p>ConvertedCurrency: CNY (the WAN jackpot currency)</p> <p>ConvertedIncrement: 50</p> <p>ConvertedClaw: 100</p> <p>Since the FundedCurrency is the same as the JackpotCurrency, the conversion is 1 to 1.</p>
---	---

Bonus/Coupon Methods (Only coupontypeid=5 and coupontypeid=3 currently supported)

TIP: see Bonus Webhooks in the webhook section for notifications that may be useful

CreateAndApplyBonusMulti (Transfer and Seamless)

POSTMAN EXAMPLE

You may create coupons in the Backoffice or use this API method. If you want to apply the created bonus to a player specify a list of usernames. If a player does not exist, the bonus will be queued.

CreateAndApplyBonusMultiRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
CouponCodeToCreate (optional)	String	If you wish to specify a coupon code use this field, otherwise a random code will be generated. Must be unique if set.
ReplaceActiveCoupon	Bool	If you specify a username, then this field will indicate if you want to set this new coupon as the active bonus balance
CouponTypeId	Int	Use value of : 5 for No Deposit: Free Spins coupon. 3 For No Deposit : Free Money
DtStartUTC	String	Valid from UTC date – yyyyMMddHHmmss NOTE: when specifying a player to redeem a coupon for - the Start date is ignored and will be immediately available to player
DtEndUTC	String	Valid until UTC date – yyyyMMddHHmmss - If no date supplied then validity is 2 months from creation.
ExpireAfterDays	Int	Number of days to expire the BonusBalance of the player once the coupon has been applied (Defaults to 14 days). Set to -1 to use DtEndUTC of the Coupon
MaxRedemptionsPerPlayer	Int	How many times may an individual player redeem this coupon
MaxRedemptionsForBrand	Int	How many times this may be redeemed Brand wide. This must be more than the list of Players you wish to redeem (if specified)
MaxRedemptionIntervalId	Int	Settings for Coupon Redemption (0 = All Time, 2 = Daily, 3 = Weekly, 4 = Monthly)
WagerMultiplierRequirement	Int	The wagering required on the amount the player has 'collected' after free spins are completed. This wagering must be completed to convert the bonus to real money. Set to 0 to accept all funds won from Free spins with no further wagering.
MaxConversionToRealMultiplier	Int	The multiplier for max conversion to real. If the bonus is worth \$50 and this field is set to 5 then the player can convert no more than 50x5 into real money. Set to 0 for no maximum.
NumberOfFreeSpins	Int	Number of free spins to award (if CouponTypeId is 5)
GameKeyNames Or BrandGameIds	String[] OR Guid[]	The list of Games to enable the free spins on (if CouponTypeId 5) If specifying more than 1 item in the list Note that the coins per game must be the same for all games. E.g. you cannot mix a 25 coin and a 50 coin game.
Deprecated single game format GameKeyName <string> Or BrandGameId <guid>	String OR Guid	

continued on next page...

CouponCurrencyData	[array]	<div>Array of either: a money amount for each spin</div> <table><tr><td>CurrencyCode <string></td><td>Currency code</td></tr><tr><td>Amount <decimal></td><td>Total amount per spin (Type 5) or Free Money amount (Type 3)</td></tr><tr><td>CancelAtAmount <decimal> <i>(only applicable if WagerMultiplierRequirement is set)</i></td><td>value to cancel the coupon at during the wagering phase of the game. This prevents player from being stuck in the coupon and having to reduce line bet to complete bonus balance.</td></tr></table> <div>Or a position in game bet config for the currency (Type 5)</div> <table><tr><td>CurrencyCode <string></td><td>Currency code</td></tr><tr><td>CoinPosition <int> Zero-based index</td><td>The position of the coin as setup for the currency in Game configs Eg: if USD coin params are 0.01 0.05 0.10 1 5 CoinPosition 0 will be 0.01 cents per line. CoinPosition 2 will be 0.10 cents per line. NOTE: You can also set this value in Game Parameters in BO which will override any value sent here for specific game config.</td></tr></table>		CurrencyCode <string>	Currency code	Amount <decimal>	Total amount per spin (Type 5) or Free Money amount (Type 3)	CancelAtAmount <decimal> <i>(only applicable if WagerMultiplierRequirement is set)</i>	value to cancel the coupon at during the wagering phase of the game. This prevents player from being stuck in the coupon and having to reduce line bet to complete bonus balance.	CurrencyCode <string>	Currency code	CoinPosition <int> Zero-based index	The position of the coin as setup for the currency in Game configs Eg: if USD coin params are 0.01 0.05 0.10 1 5 CoinPosition 0 will be 0.01 cents per line. CoinPosition 2 will be 0.10 cents per line. NOTE: You can also set this value in Game Parameters in BO which will override any value sent here for specific game config.
CurrencyCode <string>	Currency code												
Amount <decimal>	Total amount per spin (Type 5) or Free Money amount (Type 3)												
CancelAtAmount <decimal> <i>(only applicable if WagerMultiplierRequirement is set)</i>	value to cancel the coupon at during the wagering phase of the game. This prevents player from being stuck in the coupon and having to reduce line bet to complete bonus balance.												
CurrencyCode <string>	Currency code												
CoinPosition <int> Zero-based index	The position of the coin as setup for the currency in Game configs Eg: if USD coin params are 0.01 0.05 0.10 1 5 CoinPosition 0 will be 0.01 cents per line. CoinPosition 2 will be 0.10 cents per line. NOTE: You can also set this value in Game Parameters in BO which will override any value sent here for specific game config.												
Players	[array]	<table><tr><td>Username <string></td><td>Username of player (In seamless this is the AccountId PK)</td></tr><tr><td>CurrencyCode <string></td><td>3 letter currency code of the player (used if player does not exist)</td></tr></table>		Username <string>	Username of player (In seamless this is the AccountId PK)	CurrencyCode <string>	3 letter currency code of the player (used if player does not exist)						
Username <string>	Username of player (In seamless this is the AccountId PK)												
CurrencyCode <string>	3 letter currency code of the player (used if player does not exist)												
QueueUnregisteredPlayers	bool	Controls whether to Queue bonus to unregistered players or skip them. Default = true											
CreatePlayerIfNotExist	bool	Specifies whether to register unregistered players to the system. (Default = false) NOTE: Available for SEAMLESS WALLETS only. Will require CurrencyCode for player if set to TRUE											

CreateAndApplyBonusMultiResponse Object:

CouponId <guid>	String	The internal Habanero guid for the coupon. Also sent in bonusdetails node for Seamless wallet transfers on final transfer
Created <bool>	Bool	Indicates if the coupon was created successfully
Message<string>	String	Contains info when a Created= false (due to invalid request)
CouponCodeCreated <string>	String	The CouponCodeToCreate or the randomly generated code if this was not specified.
Players	[array]	Array of:
		Username <string> Username of requested player
		CurrencyCode <string> Currency of the player
		Redeemed <bool> If you specified a username then this indicates if the coupon was successfully applied to the player
		IsQueued <bool> If the player did not exist, then the coupon will be queued for redemption.
		FailMessage <string> Information if Created or Redeemed is false
		BonusBalanceId <guid> If Redeemed = true, then this is the bonus balanceid created for the player. Use this to toggle active state or delete the bonus
		SetToActive<bool> Indicated is the requested coupon was toggled to Active
		FreeSpinsGiven <int> The number of free spins granted
		BonusValue <decimal> The total value of the entire bonus in player currency
		ValuePerFreeSpin <decimal> The money value per spin in player currency
		CoinSize <decimal> The size of coin in player currency

Example: CreateAndApplyBonusMulti() request using a money amount per spin.

USD – we are requesting 3 USD for each spin, cancelling once the bonus is less than 1 USD.

CNY – we are requesting 30 CNY for each spin, cancelling once bonus is less than 5 CNY

```
{
  BrandId: "..",
  APIKey: "...",
  CouponTypeId: 5,
  ExpireAfterDays: 7,
  MaxRedemptionsPerPlayer: 1,
  MaxRedemptionsForBrand: 10,
  WagerMultiplierRequirement: 0,
  MaxConversionToRealMultiplier: 10,
  NumberOfFreeSpins: 15,
  GameKeyNames:
  [
    "SGQueenOfQueens243",
    "SGFireRooster"
  ],
  CouponCurrencyData:
  [
    {
      CurrencyCode: "USD",
      Amount: 3.00,
      CancelAtAmount: 1
    },
    {
      CurrencyCode: "CNY",
      Amount: 30.00,
      CancelAtAmount: 5
    }
  ],
  Players:
  [
    {
      Username: "tt"
    },
    {
      Username: "cny500"
    }
  ]
}
```

Response for **money amount per spin.**

```
{
  "CoupondId": "078d65fc-33a7-e711-822c-cc2f711c5021",
  "Created": true,
  "CouponCodeCreated": "API-S6akSjEap0aLL3CQN2GUBA",
  "Players": [
    {
      "Username": "tt",
      "CurrencyCode": "USD",
      "Redeemed": true,
      "Queued": false,
      "BonusBalanceId": "69a17578-b5af-e711-8235-cc2f711c5024",
      "SetToActive": false,
      "BonusValue": 45,
      "FreeSpinsGiven": 15,
      "ValuePerFreeSpin": 3,
      "CoinSize": 0.1
    },
    {
      "Username": "cny500",
      "CurrencyCode": "CNY",
      "Redeemed": true,
      "Queued": false,
      "BonusBalanceId": "6ba17578-b5af-e711-8235-cc2f711c5024",
      "SetToActive": false,
      "BonusValue": 450,
      "FreeSpinsGiven": 15,
      "ValuePerFreeSpin": 30,
      "CoinSize": 1
    }
  ]
}
```

Note:

The USD player has received a bonus worth a total of 45 USD (3USD per spin * 15 free spins). The game will use a coin size of 10 cents. (if you divide ValuePerFreeSpin/CoinSize (3/0.10) you can calculate the number of coins/lines a game uses)

Example: CreateAndApplyBonusMulti() request using a coin position

USD – we are requesting coin position 0 to calculate the bonus value.

CNY – we are requesting coin position 2 to calculate the bonus value

Tip: use CoinPosition -1 to use the Default coin size

WARNING: there are risks associated with using this since it depends on the configured game parameters as set in the Backoffice. It is therefore advised to check the response values.

```
{
  BrandId: "..",
  APIKey: "...",
  CouponTypeId: 5,
  ExpireAfterDays: 7,
  MaxRedemptionsPerPlayer: 1,
  MaxRedemptionsForBrand: 10,
  WagerMultiplierRequirement: 0,
  MaxConversionToRealMultiplier: 10,
  NumberOfFreeSpins: 15,
  GameKeyNames:
  [
    "SGQueenOfQueens243",
    "SGFireRooster"
  ],
  CouponCurrencyData:
  [
    {
      CurrencyCode: "USD",
      CoinPosition:0,
    },
    {
      CurrencyCode: "CNY",
      CoinPosition:2,
    }
  ],
  Players:
  [
    {
      Username: "tt"
    },
    {
      Username: "cny500"
    }
  ]
}
```

Response for coin position

```
{
  "CoupondId": "078d65fc-33a7-e711-822c-cc2f711c5021",
  "Created": true,
  "CouponCodeCreated": "API-S6akSjEap0aLL3CQN2GUBA",
  "Players": [
    {
      "Username": "tt",
      "CurrencyCode": "USD",
      "Redeemed": true,
      "Queued": false,
      "BonusBalanceId": "70f3eed1-b6af-e711-8235-cc2f711c5024",
      "SetToActive": false,
      "BonusValue": 90,
      "FreeSpinsGiven": 15,
      "ValuePerFreeSpin": 6,
      "CoinSize": 0.2
    },
    {
      "Username": "cny500",
      "CurrencyCode": "CNY",
      "Redeemed": true,
      "Queued": false,
      "BonusBalanceId": "72f3eed1-b6af-e711-8235-cc2f711c5024",
      "SetToActive": false,
      "BonusValue": 2250,
      "FreeSpinsGiven": 15,
      "ValuePerFreeSpin": 150,
      "CoinSize": 50
    }
  ]
}
```

Note:

Backoffice settings for this example:

USD:

Stake Increment: **0.20|0.50|1|5**

Default Stake: 0.50

CNY:

Stake Increment: **5|10|50|200**

Default Stake: 10

As can be seen from the response:

USD coinposition 0 = 0.20

CNY coinposition 2 = 50

Tip: if CoinPosition -1 was requested:

USD would be set to 0.50 which is Default Stake

CNY would be set to 10 which is Default Stake

GetBonusAvailablePlayer (Transfer and Seamless)

Get a list of configured coupons/promos which are valid and usable for this player. The player can then redeem any of these coupons using ApplyBonusToPlayerMulti().

BonusAvailablePlayerRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	

List<CouponInfoDTO> Object:

CouponId	String	The internal couponId
Code	String	The coupon code requested. Use this with ApplyBonusToPlayer() method
DtExpire	String	Expiry date of coupon
FreeSpinCount	Int?	(Optional) Number of free spins for the game
FreeSpinValue	Decimal?	(Optional) Value per spin for the game
AutoCancelAmount	Decimal?	Amount at which coupon will cancel itself if the balance falls below this level.
WagerRequirement	Int	The wagering multiplier required to complete the coupon
MaxRedemptionIntervalId	Int	Redemption interval set for this Coupon (0 = All Time, 2 = Daily, 3 = Weekly, 4 = Monthly)
FreeConversionMultiplier	Decimal	The maximum amount that will be converted to real money if wagering is met. This is the same field as MaxConversionToRealMultiplier used in the CreateAndApplyBonus()
CouponTypeId	Int	2 Deposit + Percentage 3 NO Deposit: Free Money 4 Blocked Balance - Deposit Percentage 5 NO Deposit: Free Spins 6 Deposit + Free Spins
GameNames List	String	(Optional) List of Game Names applicable to coupon
GameKeyNames List	String	(Optional) List of Game Key Names applicable to coupon
GameName <i>[deprecated]</i>	String	<i>Please use new List<> GameNames</i>
GameKeyName <i>[deprecated]</i>	String	<i>Please use new List<> GameKeyNames</i>

ApplyBonusToPlayerMulti (Transfer and Seamless)

POSTMAN EXAMPLE

Apply a bonus coupon to a player using the coupon code. The coupon may be created in the Backoffice or via the create method previously discussed. If the Username was not found the bonus will be Queued and applied on the next login once the player has been registered

ApplyBonusToPlayerMultiRequest Object:

REQUIRED FIELDS:			
BrandId	String		
APIKey	String		
Code	String	The coupon code	
ReplaceActiveCoupon	Bool	If there is already an active bonus for the player, should this bonus become the active bonus?	
Players	[array]	Array of:	
		Username <string>	Username of player
		CurrencyCode <string>	Currency Code of the player (NOTE: This is required when CreatePlayerIfNotExist is set to TRUE)
QueueUnregisteredPlayers	bool	Controls whether to Queue bonus to unregistered players or skip them. Default = true	
CreatePlayerIfNotExist	bool	Specifies whether to register unregistered players to the system. (Default = false) NOTE: Available for SEAMLESS WALLETS only	

ApplyBonusToPlayerMultiResponse Object:

Players	[array]	Array of:	
		Username <string>	Username of requested player
		Redeemed <bool>	If you specified a username then this indicates if the coupon was successfully applied to the player
		IsQueued <bool>	If the player did not exist, then the coupon will be queued for redemption.
		FailMessage <string>	Information if Created or Redeemed is false
		BonusBalanceId <guid>	If Redeemed = true, then this is the bonus balanceid created for the player. Use this to toggle active state or delete the bonus
		SetToActive<bool>	Indicates if the coupon was toggled to Active
		FreeSpinsGiven <int>	The number of free spins granted
		BonusValue <decimal>	The total value of the entire bonus in player currency
		ValuePerFreeSpin <decimal>	The money value per spin in player currency
		CoinSize	The size of coin in player currency

GetBonusBalancesForPlayer (Transfer and Seamless)

POSTMAN EXAMPLE

Once a bonus has been applied/redeemed for a player, the player may have multiple BonusBalances. This method returns the status of all Bonus Balances a player has. You may use this to indicate bonus progress or build a tool where player can activate/deactivate and delete their bonus balances. Only one Bonus can be active at a time.

BonusGenericPlayerRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	
CurrencyCode	String	Currency code in case username is not yet registered and currencycode was not specified for queued bonus

List<BonusBalancesDTO> Object:

BonusBalanceId	String	The unique id for the bonus balance (use this to activate or delete bonus)
IsQueued	Bool	The player does not yet exist / bonus is queued and will be created when player is first seen
Code	String	The coupon code which is specified or returned in the Create method
Balance	Decimal	The current money balance of bonus
Freespins	Int	Number of free spins remaining
FreepinValue	Decimal	The value of each free spin
DtExpires	String	Expiry date of coupon
CouponId	String	The CouponId which was used to make this Bonus Balance
CouponTypeid	Int	2 Deposit + Percentage 3 NO Deposit: Free Money 4 Blocked Balance - Deposit Percentage 5 NO Deposit: Free Spins 6 Deposit + Free Spins
IsActive	Bool	Indicates if this bonus balance is currently active. The player can have multiple bonus balances but only one can be active at a time.
TotalWagerRequired	Decimal	The total money amount needed to be wager
WagerRemaining	Decimal	The amount of wagering still needed to convert the Bonus to Real
PercentageComplete	Decimal	The percentage completed in wagering
GameNames	String[]	(Optional) List of Game Names applicable to coupon
GameKeyNames	String[]	(Optional) List of Game Key Names applicable to coupon
GameName [deprecated]	String	Please use new List<> GameNames
GameKeyName [deprecated]	String	Please use new List<> GameKeyNames

SetPlayerBonusBalanceActive (Transfer and Seamless)

POSTMAN EXAMPLE

Using the GetBonusBalancesForPlayer() you may find a specific bonus to activate or de-activate

SetBonusBalanceActiveRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	
BonusBalanceId	String	RECOMMENDED. Target a specific bonus balance for the player.
Or		
CouponId		HACK: Toggle all bonuses matching a specific CouponId. This may apply to multiple database records depending on how coupons are created (multi use or not). The response returned is for the last record in the list only.
IsActive <bool>	Bool	Set to True to activate, False to de-activate

ToggleBonusBalanceResponse Object:

BonusBalanceId	String	
IsActive	Bool	Indicates if this bonus balance is currently active. The player can have multiple bonus balances but only one can be active at a time.
Success	Bool	Indicates if the action was successful
Message	String	Reason for not being able to toggle the bonus state is provided in the Message

DeletePlayerBonusBalance (Transfer and Seamless)

[POSTMAN EXAMPLE](#)

Delete a specific bonusbalanceid

DeleteBonusBalanceRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	
BonusBalanceId <guid> Or CouponId <guid>	String	RECOMMENDED. Target a specific bonus balance for the player HACK: Toggle all bonuses matching a specific CouponId. This may apply to multiple database records depending on how coupons are created (multi use or not). The response returned is for the last record in the list only.
ActivateNextBonus <bool>	Bool	If set to true, the next bonus balance the player has will be set as active.

ToggleBonusBalanceResponse Object:

BonusBalanceId <guid>	String	
Success <bool>	Bool	Indicates if the action was successful
Message	String	Reason for not being able to delete the bonus is provided in the Message

ExpireAllBonusBalance (Transfer and Seamless)

[POSTMAN EXAMPLE](#)

Expires all bonus balance in a brand by passing a CouponId or CouponCode

ExpireAllBonusBalanceRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
CouponCode <string> Or CouponId <guid>	String	Can use either CouponId or CouponCode

ExpireBonusBalanceResponse Object:

Count <int>	Int	Number of Player Bonus Balances Expired
Message	String	Action response message

CreateAndApplyBonus *DEPRECATED* use CreateAndApplyBonusMulti

You may create coupons in the Backoffice or use this API method. If you want to apply the created bonus to a player, then specify the username of the player. If a player does not yet exist we will queue the bonus as indicated by the IsQueued response item.

CreateBonusAndApplyRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username (optional)	String	Specify the player username if you also want to redeem the coupon you are creating for this player
CouponCodeToCreate (optional)	String	If you wish to specify a coupon code use this field, otherwise a random code will be generated. Must be unique if set.
ReplaceActiveCoupon	String	If you specify a username, then this field will indicate if you want to set this new coupon as the active bonus balance
CouponTypeId	Int	Use value of 5 for No Deposit: Free Spins coupon.
ExpireAfterDays	Int	Number of days to expire the BonusBalance of the player once the coupon has been applied
MaxRedemptionsPerPlayer	Int	How many times may an individual player redeem this coupon
MaxRedemptionsForBrand	Int	How many times this may be redeemed Brand wide
WagerMultiplierRequirement	Int	The wagering multiplier needed to convert the bonus to real money
MaxConversionToRealMultiplier	Int	The multiplier for max conversion to real. If the bonus is worth \$50 and this field is set to 5 then the player can convert no more than 50x5 into real money
AutoCancelBonusBalanceAt	Decimal	Auto cancel the bonus balance once it reaches a certain minimum amount during "wagering phase". This avoids the player being 'stuck' in a bonus
NumberOfFreeSpins	Int	Number of free spins to award
GameKeyName	String	The Game Keyname to award the free spins on
FreeSpinValuePerSpin	Decimal	The value PER spin to award
CurrencyCode	String	If you do not specify an existing player you must set the Currency for this coupon.
MaxRedemptionIntervalId	Int	Settings for Coupon Redemption (0 = All Time, 2 = Daily, 3 = Weekly, 4 = Monthly)

CreateAndApplyBonusResponse **Object:**

CouponId	String	The internal Habanero guid for the coupon. Also sent in bonusdetails node for Seamless wallet transfers
Created	Bool	Indicates if the coupon was created successfully
Redeemed	Bool	If you specified a username then this indicates if the coupon was successfully applied to the player
IsQueued	Bool	If the player did not exist then the coupon will be queued for redemption.
Message	String	Information if Created or Redeemed is false
CouponCodeCreated	String	The CouponCodeToCreate or the randomly generated code if this was not specified.
BonusBalanceId	String	If Redeemed = true, then this is the bonus balanceid created for the player. Use this to toggle active state or delete the bonus

ApplyBonusToPlayer *DEPRECATED* use ApplyBonusToPlayerMulti

Apply a bonus coupon to a player using the coupon code. The coupon may be created in the Backoffice or via the create method previously discussed. If the Username was not found the bonus will be Queued and applied on the next login once the player has been registered

ApplyBonusToPlayerRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	
Code	String	The coupon code
ReplaceActiveCoupon	Bool	If there is already an active bonus for the player, should this bonus become the active bonus?

CouponResponseMessage **Object:**

Success	Bool	If the coupon was redeemed successfully
IsQueued	Bool	If username does not exist, then IsQueued will be True and all other fields will be null
CouponId	String	The internal couponId
BonusBalanceId	String	The bonus balanceId created for the player. Use this to toggle active state or delete the bonus
Code	String	The coupon code requested
Amount	Decimal	The value of the coupon
FreeSpins	Int	The number of free spins granted
SetToActive	Bool	Indicates if the requested coupon was toggled to Active

DEPRECATED

Reporting Methods

Note! Seamless integrations should not use these reports as we send verbose game information in the Seamless API.

Rate Limiting

Reporting methods are Rate Limited to 25 requests per minute and 1 request per second by ReportType + (BrandId or PlayerUsername) where applicable.

If you are rate limited, you will receive a 429 HTTP error and you should retry the same method again using the same date range to ensure you do not miss data.

Reporting Constraints

Some reports are restricted to 90 days of historical data.

Reporting Delays

Reporting is available in near real-time (< 5 seconds).

If we experience a delay in reporting, you will receive an exception calling the webservice. Please retry the same request periodically until you receive data.

GetBrandCompletedGameResultsV2 (Transfer Wallet) or GetGroupCompletedGameResultsV2 (Group Report)

POSTMAN EXAMPLE

This is the most used report for Transfer wallet allowing you to import all game play data into your own database. You should request this once every minute and take note of the Rate Limit which will throw an exception if you request above the allowed rate.

Get completed game instance results for players where the Completed Date of the game is in the date range with seconds granularity.

This updated method includes necessary BONUS info to the report

ReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerCompletedGamesV2DT0[] Result:

PlayerId	String	Internal Habanero GUID for player
BrandId	String	NOTE: Brandid is populated for GetGroupCompletedGameResultsV2 method
Username	String	Player Username
BrandGameId	String	Game BrandGame Id
GameKeyName	String	Game Identifier
GameTypeId	Int	See gametype addendum
DtStarted	String	Start date of game round
DtCompleted	String	Completed date of game round
FriendlyGameInstanceId	Int64	Unique Game Id as a long integer
GameInstanceId	String	Unique Game Id as a GUID
GameStateId	Int	3 – Completed, 4 – Voided (Insufficient funds), 11 - Expired
Stake	Decimal	Real money stake amount
Payout	Decimal	Real money payout amount
JackpotWin	Decimal	Portion of the Payout which was from a Jackpot Win
JackpotContribution	Decimal	Jackpot contribution amount
CurrencyCode	String	Currency code of Player
ChannelTypeId	Int	See addendum of Channels
BalanceAfter	Decimal	Real balance after game completed
BonusStake	Decimal	Bonus Stake amount (if the game used bonus)
BonusPayout	Decimal	Bonus Payout amount (if the game used bonus)
BonusToReal	Decimal	Converted amount from Bonus to Real Balance (IMPORTANT FOR BONUSING!) Identifies how much money was converted from Free Spin Bonus to Real Balance. It is not shown elsewhere
BonusCoupon	String	Coupon Code used for the Bonus (if the game used bonus)
FeatureCount	Int	Number of feature spins/actions for the game. A non 0 number means a feature was hit.
BuyFeatureId	Int	A non 0 number means a player used the "Buy Feature" option. If you want a mapping of the BuyFeatureId, please refer to Addendum L

GetBrandTransferTransactions (Transfer Wallet)

POSTMAN EXAMPLE

or **GetGroupTransferTransactions** (returns all data for Group)

Get all individual money transfers in and out for a date range. Same result format as **GetPlayerTransferTransactions()**. Seconds granularity.

ReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerTransferTransactionsDTO[] Result:

PlayerId	Username	BrandId	TransactionId	RequestId	DtTx	TransactionTypeId	TransactionTypeName	Amount	BalanceAfter	CurrencyCode
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	..	46bc8e79-1a8c-e511-822c-e4a4bb1f3baf	12111	2015-11-16T04:28:25	195	PlusPlayTransferIn	11.0000	165.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	..	cce68dd6-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:23:51	196	PlusPlayTransferOut	-6.0000	154.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	..	c97157c0-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:23:14	195	PlusPlayTransferIn	6.0000	160.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	..	aad1c362-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:20:37	195	PlusPlayTransferIn	77.0000	154.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	..	a8d1c362-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:20:00	195	PlusPlayTransferIn	77.0000	77.0000	USD
6158963e-088c-e511-822c-e4a4bb1f3baf	timho	..	118b89aa-098c-e511-822c-e4a4bb1f3baf		2015-11-16T02:28:06	196	PlusPlayTransferOut	-100.0000	0.0000	USD
6158963e-088c-e511-822c-e4a4bb1f3baf	timho	..	0f8b89aa-098c-e511-822c-e4a4bb1f3baf		2015-11-16T02:28:05	195	PlusPlayTransferIn	100.0000	100.0000	USD
6158963e-088c-e511-822c-e4a4bb1f3baf	timho	..	394a8d5a-098c-e511-822c-e4a4bb1f3baf		2015-11-16T02:25:51	196	PlusPlayTransferOut	-200.0000	0.0000	USD
6158963e-088c-e511-822c-e4a4bb1f3baf	timho	..	e0e26d4b-098c-e511-822c-e4a4bb1f3baf		2015-11-16T02:25:26	196	PlusPlayTransferOut	-100.0000	200.0000	USD
6158963e-088c-e511-822c-e4a4bb1f3baf	timho	..	e0f75e30-098c-e511-822c-e4a4bb1f3baf		2015-11-16T02:24:20	195	PlusPlayTransferIn	100.0000	300.0000	USD

Note: BrandId is populated for use with GetGroupTransferTransactions

GetPlayerTransferTransactions (Transfer Wallet)

POSTMAN EXAMPLE

Get individual money transfers in and out for a player in a date range. Same result format as GetBrandTransferTransactions(). Seconds granularity.

PlayerReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player's username
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerTransferTransactionsDTO[] Object:

PlayerId	String	
Username	String	
TransactionId	String	
TransactionTypeId	Int	Internal Habanero record Id for transaction
TransactionTypeName	String	The id type for the record
Amount	Decimal	String description of the Typeld
BalanceAfter	Decimal	The real money amount
DtTx	String	The player balance after the transaction completed
RequestId	String	UTC date of the transaction
CurrencyCode	String	Your requestid (if Specified in the Deposit/Withdrawal)

Example Result:

PlayerId	Username	TransactionId	RequestId	DtTx	TransactionTypeId	TransactionTypeName	Amount	BalanceAfter	CurrencyCode
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	46bc8e79-1a8c-e511-822c-e4a4bb1f3baf	12111	2015-11-16T04:28:25	195	PlusPlayTransferIn	11.0000	165.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	cce68dd6-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:23:51	196	PlusPlayTransferOut	-6.0000	154.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	c97157c0-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:23:14	195	PlusPlayTransferIn	6.0000	160.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	aad1c362-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:20:39	195	PlusPlayTransferIn	77.0000	154.0000	USD
2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	timho	a8d1c362-198c-e511-822c-e4a4bb1f3baf		2015-11-16T04:20:37	195	PlusPlayTransferIn	77.0000	77.0000	USD

GetPlayerGameTransactions (Transfer Wallet) **DEPRECATING SOON**

[POSTMAN EXAMPLE](#)

Get individual debit and credit transactions per game for a player in a date range. Seconds granularity.

PlayerReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player's username
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerGameTransactionsDTO[] Object:

PlayerId	String	
Username	String	
BrandGameId	String	
GameKeyName	String	
GameName	String	Name of the game
GameInstanceId	String	The internal Habanero record for the gameinstance
FriendlyGameInstanceId	String	Integer reference to GameInstanced (This is visible to player in game)
TransactionId	String	Internal Habanero record Id for transaction
TransactionTypeId	Int	The id type for the record
TransactionTypeName	String	String description of the Typeld
Amount	Decimal	The real money amount (Negative for debit, Positive for credit)
BalanceAfter	Decimal	The player balance after the transaction completed
PromoAmount	Decimal	The promo / bonus money amount
DtTx	String	UTC date of the transaction

Example Result:

Username	PlayerId	BrandGameId	GameKeyName	GameName	GameInstanceId	FriendlyGameInstanceId	TransactionId	TransactionTypeId
timho	2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	c07552ae-a65c-4d7d-93dd-10add80817be	BlackJack	Blackjack	64c97fdd-1a8c-e511-822c-e4a4bb1f3baf	1306289	67c97fdd-1a8c-e511-822c-e4a4bb1f3baf	301
timho	2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	c07552ae-a65c-4d7d-93dd-10add80817be	BlackJack	Blackjack	64c97fdd-1a8c-e511-822c-e4a4bb1f3baf	1306289	69c97fdd-1a8c-e511-822c-e4a4bb1f3baf	401

columns continue...

TransactionTypeName	Amount	BalanceAfter	PromoAmount	DtTx
BLACKJACK DEAL	-100.0000	65.0000	0.0000	2015-11-16T04:31:12
BLACKJACK PAYOUT	100.0000	165.0000	0.0000	2015-11-16T04:31:14

GetPlayerGameResults (Transfer Wallet)

POSTMAN EXAMPLE

Get individual game instance results for a player in a date range. INCLUDES incomplete games. Seconds granularity.

TIP: This method returns games in **any state** (Complete, InProgress, Void, Expired) so be sure to check the GameStateId

PlayerReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player's username
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerGameResultsDTO[] Object:

BrandGameId	String	
GameName	String	Name of the game
GameKeyName	String	Keyname for the game
GameInstanceId	String	The internal Habanero record for the gameinstance
FriendlyGameInstanceId	Int64	Integer reference to GameInstanced (This is visible to player in game)
Stake	Decimal	Total Staked
Payout	Decimal	Total Payout (inclusive of Jackpot win)
JackpotWin	Decimal	Value of Jackpot win on game (if any)
JackpotContribution	Decimal	Amount of bet contributed to ALL active Jackpots (WAN and LOCAL!) NOTE you MUST use ReportJackpotContribution() or ReportJackpotContributionPerGame() for WAN jackpot billing. Using this will be incorrect! See Jackpot section for more details.
GameStateId	Int	2 = in progress, 3 – Completed, 4 – Voided (Insufficient funds), 11 - Expired
GameStateName	String	Descriptive version of GameStateId
GameTypeId	Int	SEE ADDENDUM D
DtStart	String	UTC date game was started
DtCompleted (NULLABLE)	String	UTC date game was completed (NULL if not completed)
BalanceAfter (NULLABLE)	Decimal	Player Balance on game completion. (NULL if game is not completed)

Example Result:

Username	PlayerId	BrandGameId	GameName	GameKeyName	GameInstanceId	FriendlyGameInstanceId	Stake	Payout
timho	2b5e2e9a-188c-e511-822c-e4a4bb1f3baf	c07552ae-a65c-4d7d-93dd-10add80817be	Blackjack	BlackJack	64c97fdd-1a8c-e511-822c-e4a4bb1f3baf	1306289	100.0000	100.0000
JackpotWin	JackpotContribution	DtStart	DtCompleted	GameStateName	GameStateId	GameTypeId	BalanceAfter	
0	0.152234	2015-11-16T04:31:12	2015-11-16T04:31:14	Completed	3	4	100.50	

GetPlayerStakePayoutSummary (Transfer Wallet)

POSTMAN EXAMPLE

Get a single player's summed Stake, Payout, Jackpot win (portion of the Payout), Jackpot Contributions in 1 row. INCLUDES incomplete games. Hourly granularity

PlayerReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player's username
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerStakePayoutSummaryDTO Object:

Games	Int	Games played
Stake	Decimal	Total Staked
Payout	Decimal	Total Payout (inclusive of Jackpot win)
JackpotWin	Decimal	Value of Jackpot win on game (if any) This is already included in the Payout.
JackpotContribution	Decimal	Amount of bet contributed to ALL active Jackpots (WAN and LOCAL!) NOTE you MUST use ReportJackpotContribution() or ReportJackpotContributionPerGame() for WAN jackpot billing. Using this will be incorrect! See Jackpot section for more details.

Example Result:

```
{
  "Games": 2
  "Stake": 35.0000
  "Payout": 447.7600,
  "JackpotWin": 400.0100
  "JackpotContribution": 0.05000000,
}
```

NOTE!! Of the 447.7600 paid out, 400.01 was for a JACKPOT. Do not add JackpotWin to the Payout – it is already inclusive.

ReportGameOverviewPlayer (Transfer Wallet)

POSTMAN EXAMPLE

Get game overview report for each Game played by Player in date range. Hour granularity.

NOTE: ONLY completed games are reported.

PlayerReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
Username	String	Player's username
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerGameOverviewRecord[] Object:

BrandGameId	String	
GameName	String	Name of the game
GameKeyName	String	Keyname for the game
TotalGames	Int	Completed game count
TotalStake	Decimal	
TotalPayout	Decimal	

Example Output:

BrandGameId	GameKeyName	GameName	GameTypeName	TotalGames	TotalStake	TotalPayout
c07552ae-a65c-4d7d-93dd-10add80817be	BlackJack	Blackjack	Blackjack	1	100.0000	100.0000
895b15bb-3990-458c-bbe9-23987fd95043	TensorBetter5Hand	Tens or Better 5 Hand	Video Poker	6	150.0000	190.0000

ReportPlayerStakePayout (Transfer Wallet)

POSTMAN EXAMPLE

Query a date range and receive a list of usernames with total stake and total payout for the period. Hour granularity.

TIP: Use this report to create a Winners/Losers list showing all players who have played in a date range. You can then drill down on the player and get the players game data using `GetPlayerGameResults()` or `ReportGameOverviewPlayer()`

ReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerStakePayoutDT0[] Object:

Username	String	The player username
PlayerId	String	Internal Habanero playerid
CurrencyCode	String	Player currency code
Games	Int	Number of games completed
Stake	Decimal	Total stake for period requested
Payout	Decimal	Total payout for period requested
Nett	Decimal	Stake minus Payout
JackpotWin	Decimal	Value of Jackpot win on game (if any) This is already included in the Payout.
JackpotContribution	Decimal (20,8)	Amount of bet contributed to ALL active Jackpots (WAN and LOCAL!) NOTE you MUST use ReportJackpotContribution() or ReportJackpotContributionPerGame() for WAN jackpot billing. Using this will be incorrect!

Example Result:

Username	PlayerId	CurrencyCode	Games	Stake	Payout	Nett	JackpotWin	JackpotContribution
mikey	bbff6703-e93c-e511-81c1-74d02b2c397f	CNY	50	100.0000	200.0000	-100.0000	0.0000	1.50000000
doco	113cd3c1-e83c-e511-81c1-74d02b2c397f	USD	121	20.0000	50.0000	-30.0000	0.0000	0.20000000
moomoo	712d0994-e73c-e511-81c1-74d02b2c397f	USD	4	100.0000	100.0000	0.0000	0.0000	1.00000000
aud	af78b417-4505-e511-8195-74d02b2c397f	AUD	153	1670.0000	670.0000	1000.0000	370.0000	16.70000000

ReportJackpotWinner (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Query a date range and receive a list of jackpot winners

ReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

JackpotWinnerRecord[] Object:

BrandId	String	
BrandName	String	
JackpotId	String	Internal GUID of jackpot in Habanero
JackpotName	String	Name given to jackpot as configured in Backoffice
JackpotTypeId	Int	Int – type of jackpot – see jackpot section
TypeName	String	Description of jackpot type
PlayerId	String	Internal GUID of playerid in Habanero
Username	String	Player username
PlayerName	String	Full descriptive name of player
PlayerCurrency	String	Currency Code of player and the amount won in the jackpot
AmountWon	Decimal	Value of jackpot won – in the player currency
DtWon	String	DateTime of the win (UTC)
GameKeyName	String	Keyname of the game where jackpot was won
GameInstancelId	String	<guid> Internal GameInstance which won the jackpot
FriendlyGameInstancelId	Int64	Integer id of the game which won the jackpot
Position	Int	Int – position of winning payout especially related to Jackpot Race type
GameStake	Decimal	Stake for the game round
GameTotalPayout	Decimal	Total amount paid for the game round inclusive of jackpot payout
BrandGameId	String	Internal GUID of Habanero game

Example Result:

BrandId	BrandName	JackpotId	JackpotName	JackpotTypeId	TypeName	PlayerId
6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef	Development - Dev	b423cb13-5526-e711-9be8-240a6407324f	Major	3	Random Major	cce6c429-4426-e711-9be8-240a6407324f

Columns continue...

Username	PlayerName	PlayerCurrency	AmountWon	DtWon	GameKeyName	Username
tonyt	tonyt - Tony Tang	EUR	500.0700	2017-04-21T05:43:03.257	SGTheKoiGate	tonyt

ReportGameOverviewBrand (Transfer Wallet)

POSTMAN EXAMPLE

Get the game overview report for each Game in a Brand. Hour granularity.

NOTE: ONLY completed games are reported.

ReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

GameOverviewRecord[] Result:

BrandGameId	GameKeyName	GameName	GameTypeName	CurrencyCode	TotalPlayers	TotalGames	Stake	GamePayout	JackpotPayout
c07552ae-a65c-4d7d-93dd-10add80817be	BlackJack	Blackjack	Blackjack	USD	1	1	100.0000	100.0000	0.0000
895b15bb-3990-458c-bbe9-23987fd95043	TensorBetter5Hand	Tens or Better 5 Hand	Video Poker	USD	1	6	150.0000	190.0000	0.0000

columns continue...

TotalPayout	GameNett	TotalNett	ExpectedRTP	GameRTP	TotalRTP	AvgWagerPerPlayer	AveragesGamesPerPlayer	AverageBetPerGame
100.0000	0.0000	0.0000	99.59	100.0000	100.0000	100.0000	1	100.0000
190.0000	-40.0000	-40.0000	99.14	126.6600	126.6600	150.0000	6	25.0000

ReportBrandCouponRedemptions (Transfer and Seamless Wallet)

POSTMAN EXAMPLE

Gets all redeemed coupon report for a brand in a specified date

ReportBrandCouponReportRequest **Object:**

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date (yyyyMMddHHmmss format)
DtEndUTC	String	UTC End date (yyyyMMddHHmmss format)
CurrencyCode	String	(OPTIONAL) 3 Letter ISO-Code. Converts the MadeToReal to specified Currency
CouponReportType	int	Flag that will selects what type of reports to show (default is 1): 1 = Converted to Real Money Coupons 2 = Completed Coupons 3 = All Coupons

List<ReportCouponsDTO>

FieldName	Type	Description
PlayerUsername	string	Player Username
CouponCode	string	Coupon Code
CouponName	string	Coupon Name
DtRedeemed	datetime	Coupon Date Redeemed
DtCompleted	datetime?	Date Completed
GameKeyName	string	Habaner Game Keyname
BonusValue	decimal	Bonus value
FreeSpinsAssigned	int	Free Spins Assigned for this coupon
WagerRequirement	decimal	Wagering Requirement set
RealBalanceBefore	decimal	Player's last recorded balance
DepositAmount	decimal	(only applicable for deposit based coupons)
MadeRealAmount	decimal	Amount converted to Real (based on player's currency)
MadeRealAmountConverted	decimal	Amount to Real, and converted to specified Currency Code
TotalWageringRequired	decimal	Total wagering amount needed to finish the coupon
PromoBalanceAtCompletion		Final promo balance won at the end of the coupon
FreeConversionMultiplier	decimal	Maximum amount multiple set for the coupon
CurrencyCode	string	Player's Currency
ConvertedCurrencyCode	string	Specified Currency Code for Conversion
ConversionRate	decimal	Exchange Rate used for the conversion
CouponTypeId	short	(SEE ADDENDUM E for Coupon Types)
CouponStatusId	short	(SEE ADDENDUM I for Coupon Status)

GetBrandCompletedGameResults (Transfer Wallet)

or **GetGroupCompletedGameResults** (return all data for Group)

DEPRECATED use **GetBrandCompletedGameResultsV2** / **GetGroupCompletedGameResultsV2**

This is the most commonly used report for Transfer wallet allowing you to import all game play data into your own database. You should request this once every minute and take note of the Rate Limit which will throw an exception if you request above the allowed rate.

Get completed game instance results for players where the Completed Date of the game is in the date range with seconds granularity.

NOTE: ONLY completed games are reported.

ReportRequest Object:

REQUIRED FIELDS:		
BrandId	String	
APIKey	String	
DtStartUTC	String	UTC Start date for range – yyyyMMddHHmmss. This field is inclusive (>=)
DtEndUTC	String	UTC End date for range – yyyyMMddHHmmss. This field is exclusive (<)

PlayerCompletedGamesDTO[] Results

PlayerId	BrandId	Username	BrandGameId	GameKeyName	GameTypeId	DtStarted	DtCompleted	FriendlyGameInstanceId
80c75680-84cc-407c-a030-3c0ecb1d3d3b	..	timho	895b15bb-3990-458c-bbe9-23987fd95043	TensorBetter5Hand	6	2015-11-16T04:40:50	2015-11-16T04:40:52	1306295
80c75680-84cc-407c-a030-3c0ecb1d3d3b	..	timho	895b15bb-3990-458c-bbe9-23987fd95043	TensorBetter5Hand	6	2015-11-16T04:40:45	2015-11-16T04:40:46	1306294
80c75680-84cc-407c-a030-3c0ecb1d3d3b	..	timho	c07552ae-a65c-4d7d-93dd-10add80817be	BlackJack	4	2015-11-16T04:31:12	2015-11-16T04:31:14	1306289

Columns continue ...

GameInstanceId	Stake	Payout	JackpotWin	JackpotContribution	CurrencyCode	ChannelTypeId	BalanceAfter	
3b0a0e41-245c-461e-a265-02b5f94f16ed	25.0000	40.0000	0.0000		USD	1	105.00	
b3a86924-bc17-4219-8893-8b36ff7de99c	25.0000	15.0000	0.0000		USD	1	90.00	
11fbfc94-0895-43ac-9c7e-8eb84e4d38c4	100.0000	100.0000	0.0000		USD	1	100.00	

Note: BrandId is populated for use with GetGroupCompletedGameResults

PlayerId, BrandId, BrandGameId, GameInstanceId – GUID/uniqueidentifier

Username – nvarchar(150)

GameKeyName – varchar(50)

GameTypeId, ChannelTypeId – smallint

FriendlyGameInstanceId – long/bigint

Stake, Payout, Jackpot, JackpotWin, JackpotContribution, BalanceAfter – money / decimal

CurrencyCode – varchar(5)

Note: JackpotWin is the portion of the Payout which was from a jackpot

-- this page intentionally left blank --

SEAMLESS WALLET API

The following section outlines the API which you will code/implement to accept requests sent from Habanero during game launch and on each game action.

Before you Begin

DO NOT USE A STRICT DATA CONTRACT SERIALIZER

We may add additional fields at any time. If you reject additional or undocumented fields, your implementation *will* break.

We *suggest* using ngrok.com (FREE) for setting up a tunnel to your localhost/development machine. Using ngrok allows you to view all the messages (<http://localhost:4040>) between Habanero and your local API implementation. It also avoids any firewalling / manual port forwarding you might need to do.

Check your API settings in Habanero Backoffice:

In the Global -> Group/Brand, find your Brand and go to +Play Integration.

Make sure you have set your:

- 1) API Version (Habanero v2 Dual D&C version strongly recommended)
- 2) Message type (JSON or XML) XML is not recommended. Please use JSON.
- 3) Passkey – see API Security below
- 4) Endpoints

API Security

All requests contain an **auth** node with a **passkey**

Set the Passkey in the Habanero Backoffice. This will be included in all messages sent to your endpoints. If the passkey sent does not match your provided value, reject the request.

The list of Habanero IP addresses calling your servers is provided in your welcome email.

API Endpoints to Create

Name	URL**	Function
Auth Service	yoursystem.com/api/auth	Validate the token you use in the game launch and return details & balance of player
Transaction Service	yoursystem.com/api/tx	Perform debits and credits against the players wallet
Query Service	yoursystem.com/api/query	Query the status of transactions
Alternate Funds	yoursystem.com/api/altcredit	Accept payouts for tournaments and prize drops


** These URLs are set in the Habanero Backoffice – **you may specify the endpoint however you wish** or use the same endpoint for all and determine the message type by using the "type" field.

API Message Types

"type" and element name	purpose	endpoint
playerdetailrequest	authenticate and get player info and balance	Auth
fundtransferrequest	debit / credit player wallet	Transaction
queryrequest	retrieve status of a fundtransferrequest	Query
altfundsrequest	accept credits for tournaments, prize drops	Alternate Credit
playerendsessionrequest	(OPTIONAL) player end of session notification	End Session
configdetailrequest	(OPTIONAL) custom game/bet configuration	Configuration

Backoffice options:

+ Play Integration

 Audit
Generate Template (after saving configuration)

Common Settings

Limit Multiple Slot RTPs ☒ Enforces limit to only allow 1 active RTP for a slot game. This should be set when using KeyNames instead of BrandGameId as game identifier.
On Bonus Completing

Continue Play

 (what happens when the player completes a bonus)

Seamless Wallet Settings

Seamless API version

Habanero Dual/D&C Transaction API (v2)

[View API Info](#)

☐ Send all 0 credits (For integrations needing an exact transaction log)
☐ Log all API Messages (Errors are always logged)

Internal Logging ☐
Limit UI ☒ If using Seamless API, tick this to hide irrelevant player functionality.
Token Scope ☒ Single Token (Default) ☐ Token per Game (Game Launch Token updates all existing tokens (Single mode), or Token per Game is used)

Timeout

10000

 ms (Recommended 5000 to 8000, longer if there are upstream wallets. Use a timeout ending in 99 to enable 100-Continue)
In-Line Retries

0

 x with

0

 ms wait in between. (Default 0x / 0ms) Includes transient errors (timeouts, net issues), excludes HTTP errors (500, 429 etc). **Warning:** Partner must handle race conditions correctly.
Background Retries

7

 x (Defaults to using schedule as per API document. Set to 0 to disable)
Skip Launch Retries ☐ Not Recommended: Do not send pending transactions when player launches game.
Skip Load Block ☐ Not Recommended: Allow player to launch game even though the game has pending transactions.

API Passkey
Authentication Endpoint [Test Endpoint Access](#)
Transaction Endpoint
QueryRequest Endpoint # blank, then Transaction Endpoint is used.
End Session Endpoint URL for session termination info - **leave blank if not implemented**
Alternate Credit Endpoint URL for alternate credit (promo, tournaments, etc) - **leave blank if not implemented**
Configuration Endpoint URL for Game bet parameters - **leave blank if not implemented**
Bonus Messaging ☐ Send each bonus Debit and Credit (Requires integration Code Changes! Default behaviour is OFF and only 1 credit with total amount won is sent.)

Change Habanero Token ☐ **IMPORTANT!** If API Token changes at launch, also change the internal Habanero Token for already active player. This can break multi-window play. See API document: Session Management
Proxy Server for Requests Warning: Use for debugging only. format: url:port

Error Handling / Retries

Transient errors such as ConnectFailure, ConnectionClosed, ReceiveFailure, SendFailure are retried immediately up to 2 times with a 1 second delay during transmission.

Any server 500 errors or timeouts are queued for retry 60 seconds later as discussed in the section titled "Seamless Wallet Error Resolution".

Deferred Credits

For such instances where clients can only process **1 credit for each round**, the Defer Credit option is available (Discuss with your account manager for further information). With this option enabled, Habanero will collate all round winnings and send it at the end of the round.

Player Detail Request (For Authentication and Balance)

The **playerdetailrequest** is sent to your service to authenticate and retrieve the player's details and balance. It is sent as soon as we receive the game launch request, and during balance refresh events.

The player gets created or updated in Habanero using the returned **playerdetailresponse**.

Request from Habanero:

1. **auth** with attributes
 - a. **username** – always hardcoded as habanero
 - b. **passkey** – the password/secret key you chose and configure in the Backoffice
 - c. **machinename** performing the request. For informational purposes.
 - d. **locale** - the player's locale
 - e. **brandid** - the brandid making the request
2. **basegame** (allows you to scope balance to your own bonusing system etc)
 - a. **brandgameid** and **keyname** – the game which is requesting to launch or refresh balance
3. **playerdetailrequest** with
 - a. **token** – your player token sent to Habanero during game launch
 - b. **gamelaunch** – if true it is the initial authentication request, else it is for balance refresh

```
{
  "type": "playerdetailrequest",
  "dtsent": "2017-07-27T02:55:27.7146136Z",
  "basegame": {
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "keyname": "SGAllForOne"
  },
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "locale": "en",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef"
  },
  "playerdetailrequest": {
    "token": "f4a9b0d1-c129-4407-8106-8916358a3d6a"
    "gamelaunch": true | false
  }
}
```

playerdetailresponse

- a. **status** node (see next page)
 - b. **accountid** - the **UNIQUE Primary Key of the player in your database** (string up to 150 chars) In all Habanero json webservice methods (if used) specify this field where "username" is required.
 - c. **accountname** - usually the screenname/username of your player. If omitted we will set it the same as the accountid (string up to 150 chars)
 - d. **balance** - decimal value with "." decimal separator and no group separator
 - e. **currencycode** – 3 letter ISO code - e.g. EUR – *cannot be changed after player creation*
2. Other **optional** fields: (if you wish to populate these in Habanero Backoffice)
- a. **newtoken** – Change to a different token for subsequent messages (**initial launch/auth request only**)
 - b. **country** - 2 Letter ISO country code - e.g. GB, DE
 - c. **jurisdiction** – 2 letter ISO country code of the Jurisdiction to apply
 - d. **playerrank** – an integer matching the Habanero Player Class rank – use this to specify VIP levels etc
 - e. **fname** - Player First Name
 - f. **lname** - Player Last Name
 - g. **email** - Player Email address
 - h. **tel** - Player contact number
 - i. **segmentkey** - Use this field to specify an operatorid or casino for the player when you only have 1 Habanero Brand but have players from different casinos inside this one brand. Up to 40 chars string.

Successful response returning player details:

```
{
  "playerdetailresponse": {
    "status": {
      "success": true,
      "autherror": false,
      "message": ""
    },
    "accountid": "153",
    "accountname": "lulu",
    "balance": 1500.50,
    "currencycode": "EUR"
  }
}
```

Failed response:

```
{
  "playerdetailresponse": {
    "status": {
      "success": false,
      "message": "Reason for failure description"
    }
  }
}
```

Status node

The Status element for used in all messages (playerdetailresponse and fundtransferresponse) contains the following attributes – usage of each will become clear with examples to follow

```
"status": {
  "success": true,           //used for all responses
  "autherror": false,      //if authentication fails during fundrequest
  "nofunds": false,        //if insufficient funds during fundrequest
  "successdebit": false,   //if using Debit&Credit message
  "successcredit": false,  //if using Debit&Credit message
  "refundstatus": 0        //for refund requests
  "message": "Send a note here" //for logging and displayed to player during failed launch
},
```

Attribute	Type	Required	Notes
success	Bool	Yes	Did the action succeed?
autherror	Bool	No	If the token failed to authenticate (user has logged out) then set autherror=true in addition to success=false
nofunds	Bool	No	If the player has insufficient funds set nofunds=true in addition to success=false. We will show an Insufficient Funds message instead of a Wallet Error message
successdebit	Bool	Depends	When doing a D&C package request and the "debitandcredit=true" you must explicitly set whether the debit was performed successfully
successcredit	Bool	Depends	When doing a D&C package request and the "debitandcredit=true" you must explicitly set whether the credit was performed successfully
refundstatus	Int	Yes, if refund requested	See error resolution section
Message	String	No	This message is shown to a player if the game launch fails. Eg "Token Expired", "You are locked out"

Fund Transfer Request (Game actions)

Habanero will send all financial transactions to your system for every game action performed.

There are 2 formats of the fund transfer request which can be set in the Habanero Backoffice under +Play settings.

1) Habanero Single Transaction API (v1)

- All debits and credits will be sent as separate requests.

2) Habanero Dual/D&C Transaction API (v2) (**RECOMMENDED for reduced latency**)

- A dual transaction consisting of the Debit&Credit (D&C) is sent for:
 - Slot Games (in base game, with no jackpots), Roulette, Baccarat, Sic Bo – these will send the Debit and Credit – in one request.
 - **Note: If Jackpots are enabled Slot games will always send single debits and credits.**
- If you enable the newer API without correctly modifying your code, you will see a message as follows in the game: "D&C Package error – cannot respond success=true without BOTH successcredit and successdebit being true"
- **You must still accept single transaction messages** due to the nature of some games:
 - A Slot game in feature mode will send single credits for the result of each feature spin/pick
 - Video Poker will send a debit when the player Deals and a credit when the player draws.
 - Games such as Blackjack will send multiple debits for actions such as buying Insurance, Doubling or Splitting
 - Jackpot wins and Bonus payouts send a single credit message

fundtransferrequest

1. **auth** element as per player detail request from before
2. **fundtransferrequest** with
 - a. **token**
 - b. **accountid** – you should always lookup the player using the token! We provide the accountid, as received in your playerdetailresponse, in case of token expiration so you may issue a refund or recredit for a player or accept Expired game notifications.
 - c. **gameinstanceid** – a guid Primary Key for 1 instance/round of a game. Use this to represent an entire game if you are ‘cloning’ game play stats in your system.
 - d. **friendlygameinstanceid Int64** – a long integer value representing the game number which is shown on the player’s screen. You **must not** use this as a primary key referencing a Habanero game. Use the Gameinstanceid. This is provided in case of support requests whereby a player sends you a screenshot (this Game Number is visible).
 - e. **customplayertype** - value of 0 indicates regular player. Value of 1 indicates player marked as Tester in Habanero Backoffice. Discuss with your account manager for further information
 - f. **isretry** – true|false – this is set to true in conjunction with either isrefund or isrecredit. You **ONLY** need to check for uniqueness on a transferid IF isretry is set to true.
 - g. **isrefund** – true|false if this is a refund attempt (See seamless wallet error resolution section)
 - h. **isrecredit** – true|false if this is a recredit attempt (See seamless wallet error resolution section)
 - i. **funds** node containing a list with 1 or 2 fundinfo records–
 - i. **debitandcredit** – true|false - indicating if D&C Package
 1. If true, there will be TWO fundinfo records. The first for debit and second for credit.
 2. If false, there will be ONE fundinfo record which might be a debit or a credit
 - ii. **fundinfo** elements
 1. **transferid** – (32-character string). The unique id for this fund transfer element.
 - a. Remember - if **isretry** true - **you must** look up the status of the transferid to prevent duplicate transactions.
 2. **amount** - Decimal value with "." decimal separator. Negative for debits, and >=0 for credits.
 3. **dtevent** – the UTC datetime stamp for the action. By using this value, you can guarantee synchronised stats between your system and Habanero
 4. **currencycode** - will always be the player’s currency code as received your playerdetailresponse
 5. **gamestatemode** – if your system is keeping track of the number of games played and ‘clones’ the state of a game or needs to ‘close a matching bet’ the gamestatemode is of use:
 - a. Value of **1** == Game Round Start
 - b. Value of **2** == Game Round End (last action of the game)
 - c. Value of **3** == Expired (game has been abandoned after X days. This is the last action of a game and a 0 credit is sent)
 - d. Value of **0** == Continuation (game is doing an action which is neither a new game/nor end of game – e.g. paying out a slot feature spin, a jackpot win credit, a blackjack double debit, blackjack insurance debit etc.)
 6. **jpwin** - true|false indicating if this credit "**amount**" is from a jackpot win (Jackpot credits are sent in their own transferrequest)
 7. **jpcont** – the amount contributed towards all active jackpots. Support up to 8 decimal places. Will only have a value for the debit fund node and in Slot games

8. **if jpwin is true:**
 - a. **jpuid** – this field will contain the JackpotId guid referencing which jackpot was won. You may use the jackpot webservice to match this data.
 - b. **jpname** – the name as set in the Backoffice for this jackpot
 - c. **jptypeid** – see Addendum F for jackpot types
 - d. **jpseed** – the seed/start value of the jackpot that was won (in the jackpot base currency)
 - e. **jpwinbase** – Use this for recording jackpot level win information. NOT FOR PLAYER USAGE since it is in the base currency of the jackpot.
 9. **isbonus** – indicates this is a bonus payout from a coupon that has met wagering requirements. There is NO matching debit, only a credit is sent once the bonus is converted to real money. See the bonus example later in this document and the bonusdetail node
 10. **initialdebittransferid** – the transferid for the first debit of the game. Note: This is null for bonus payouts since bonuses have no matching debit.
 11. **accounttransactiontype** – the account transaction type id for this action – **SEE ADDENDUM G**
 12. **gameinfeature** (bool)– indicates if the game (slot) is in feature mode
 - a. when true **featureno** (nullable int16) indicates the number of the feature
 13. **buyfeatureid** (nullable int16) will be populated (not null) IF the player purchased a feature for the game round. See ADDENDUM L for details or accept any non null value as an indicator that feature buy was used – which specific feature is of less importance.
- j. **gamedetails**
- i. **brandgameid** – game identifier
 - ii. **keyname** – KeyName for a game
 - iii. **name** – the English name of the game
 - iv. **gameinstanceid** – copy of the parent gameinstanceid
 - v. **friendlygameinstanceid** Int64 – copy of the parent friendlygameinstanceid
 - vi. **gametypeid** - **SEE ADDENDUM D**
 - vii. **productexternalid** <nullable> - NULL for Habanero games or **SEE ADDENDUM I**
 - viii. **gamesessionid** – the habanero gamesession which is started when opening an instance of a game.
 - ix. **gametypename** – the description of the gametype
 - x. **srij_smdata** - Portugal SRIJ SM Data
 - xi. **channel, browser, device** – **SEE ADDENDUM C**
 - xii. **maxpaylimit** – <nullable decimal> the max payout amount for this round as per the game's bet configuration
- k. **bonusdetails**
- i. **couponid** - guid for coupon created via API/ backoffice (see bonus webservice for more info)
 - ii. **couponcode** – string id used for the coupon
 - iii. **bonusbalanceid** – the instance of the promo for the player (see bonus webservice for more info)
 - iv. **coupontypeid** – **SEE ADDENDUM E**

Note: Credits of 0.00 are always sent if the game is in a completing state (gamestatemode = 2). Intermediate 0.00 credits in a slot feature will not be sent unless the game is completing.

Example 1 – Single Transaction Request (Must be implemented for both API versions)

This Blackjack game is starting a New Game (`gamestatemode=1`) and requesting a debit of -100.00 EUR. Notice there is only 1 fundinfo record in the funds node and `debitandcredit="false"`

```
{
  "type": "fundtransferrequest",
  "dtsent": "2017-07-27T04:41:16.7792962Z",
  "basegame": {
    "brandgameid": "9bc350d1-f968-4e6a-aaa7-5e4d39941e05",
    "keyname": "BlackJack3H"
  },
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "locale": "en",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef"
  },
  "fundtransferrequest": {
    "token": "c157bc6f-b41a-452e-9551-b37284540236",
    "accountid": "153",
    "customplayertype": 0,
    "gameinstanceid": "bf22d5cf-8572-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330635,
    "isretry": false,
    "retrycount": 0,
    "isrefund": false,
    "isrecredit": false,
    "funds": {
      "debitandcredit": false,
      "fundinfo": [
        {
          "gamestatemode": 1,
          "transferid": "bf22d5cf8572e7119c0b74d02b2c397f",
          "currencycode": "EUR",
          "amount": -100.00,
          "jpwin": false,
          "jpcont": 0.0,
          "isbonus": false,
          "dtevent": "2017-07-27T04:41:16.683Z",
          "initialdebittransferid": "bf22d5cf8572e7119c0b74d02b2c397f"
        }
      ]
    }
  },
  "gamedetails": {
    "name": "Blackjack (3 Hand)",
    "keyname": "BlackJack3H",
    "gametypeid": 4,
    "gametypename": "Blackjack",
    "brandgameid": "9bc350d1-f968-4e6a-aaa7-5e4d39941e05",
    "gamesessionid": "be22d5cf-8572-e711-9c0b-74d02b2c397f",
    "gameinstanceid": "bf22d5cf-8572-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330635,
    "channel": 1,
    "device": "Non Mobile",
    "browser": "Non Mobile"
  }
}
```

Example 1 – Success Response

Player had sufficient funds and the debit request was successful. After the debit, his new balance is 900

1. Status node with success="true"
2. Latest **balance** after transaction was committed

```
{
  "fundtransferresponse": {
    "status": {
      "success": true,
    },
    "balance": 2038.1,
    "currencycode": "EUR"
  }
}
```

Example 1 – Insufficient Funds Response

If the player did not have funds, return **nofunds**="true" and the current balance

```
{
  "fundtransferresponse": {
    "status": {
      "success": false,
      "nofunds": true,
    },
    "balance": 15.21,
    "currencycode": "EUR"
  }
}
```

Example 1 – Player Authentication Failed (Session timeout) Response

If the player is no longer logged in, return **autherror**="true". The game client will display "User session Expired"

```
{
  "fundtransferresponse": {
    "status": {
      "success": false,
      "autherror": true,
    }
  }
}
```

Example 2a – D&C Package Request (Debit and Credit in one message)

In this example, Slot game "All For One" is requesting a -12.50 EUR debit and 25 EUR credit in **one message**.

- 1) Note that the `debitandcredit` = "true" and there are 2 `fundinfo` records
 - a. The first `fundinfo` record is the debit with a `gamestatemode` of 1 (new game)
 - b. The second `fundinfo` record is the credit with a `gamestatemode` of 0 – meaning that the game is not complete because the player has entered free spins feature. If the game was finished, the `gamestatemode` would have been "2".

Important: Both `fundinfo` records have unique `transferid` and you must track the status of the each separately.

```
{
  "type": "fundtransferrequest",
  "dtsent": "2017-07-27T03:10:38.6244572Z",
  "basegame": {
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "keyname": "SGAllForOne"
  },
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "locale": "en",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef"
  },
  "fundtransferrequest": {
    "token": "6607ae93-b716-468c-996c-41f8f68b75ca",
    "accountid": "153",
    "customplayertype": 0,
    "gameinstanceid": "00ffcc29-7972-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330623,
    "isretry": false,
    "retrycount": 0,
    "isrefund": false,
    "isrecredit": false,
    "funds": {
      "debitandcredit": true,
      "fundinfo": [
        {
          "gamestatemode": 1,
          "transferid": "00ffcc297972e7119c0b74d02b2c397f",
          "currencycode": "EUR",
          "amount": -12.5,
          "jpwin": false,
          "jpcont": 0.1,
          "isbonus": false,
          "dtevent": "2017-07-27T03:10:38.6Z",
          "initialdebittransferid": "00ffcc297972e7119c0b74d02b2c397f"
        },
        {
          "gamestatemode": 0,
          "transferid": "6b49dd2ade734ba19970207caf7dfb00",
          "currencycode": "EUR",
          "amount": 25.00,
          "jpwin": false,
          "jpcont": 0.0,
          "isbonus": false,
          "dtevent": "2017-07-27T03:10:38.618Z",
          "initialdebittransferid": "00ffcc297972e7119c0b74d02b2c397f"
        }
      ]
    }
  }
}
```

```
    },  
    "gamedetails": {  
      "name": "All For One",  
      "keyname": "SGAllForOne",  
      "gametypeid": 11,  
      "gametypename": "Video Slots",  
      "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",  
      "gamesessionid": "b08b32a6-7872-e711-9c0b-74d02b2c397f",  
      "gameinstanceid": "00ffcc29-7972-e711-9c0b-74d02b2c397f",  
      "friendlygameinstanceid": 1330623,  
      "channel": 1,  
      "device": "Non Mobile",  
      "browser": "Non Mobile"  
    }  
  }  
}
```

Example 2a – Debit and Credit - Success Response

If both the debit and the credit were successful you must set `success`, `successdebit` and `successcredit` to true.

```
{
  "fundtransferresponse": {
    "status": {
      "success": true,
      "successdebit": true,
      "successcredit": true
    },
    "balance": 1500.5,
    "currencycode": "EUR"
  }
}
```

In the event of the `successdebit` = true and `successcredit` = false then we will attempt to resolve the issue.

Example 2b - Game *continuing* in example 2a

In this example, the previous game from 2a is crediting the free spin results as can be seen from the `gamestatemode` = 0. The `gameinstanceid` and the `friendlygameinstanceid` are the same as the previous request since it is still the same game being played. Due to the games current state, `debitandcredit` = false because the game is only sending credits. There is therefore only 1 `fundinfo` element in the array.

```
{
  "type": "fundtransferrequest",
  "dtsent": "2017-07-27T03:14:30.8109652Z",
  "basegame": {
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "keyname": "SGAllForOne"
  },
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "locale": "en",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef"
  },
  "fundtransferrequest": {
    "token": "6607ae93-b716-468c-996c-41f8f68b75ca",
    "accountid": "153",
    "customplayertype": 0,
    "gameinstanceid": "00ffcc29-7972-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330623,
    "isretry": false,
    "retrycount": 0,
    "isrefund": false,
    "isrecredit": false,
    "funds": {
      "debitandcredit": false,
      "fundinfo": [
        {
          "gamestatemode": 0,
          "transferid": "9b2bfa54527148f2ab2202cd7ba553fc",
          "currencycode": "EUR",
          "amount": 200.00,

```

```

        "jpwin": false,
        "jpcont": 0.0,
        "isbonus": false,
        "gameinfeature": true,
        "featureno" : 1,
        "dtevent": "2017-07-27T03:14:30.81Z",
        "initialdebittransferid": "00ffcc297972e7119c0b74d02b2c397f"
    }
}
],
},
"gamedetails": {
    "name": "All For One",
    "keyname": "SGAllForOne",
    "gametypeid": 11,
    "gametypename": "Video Slots",
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "gamesessionid": "b08b32a6-7872-e711-9c0b-74d02b2c397f",
    "gameinstanceid": "00ffcc29-7972-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330623,
    "channel": 1,
    "device": "Non Mobile",
    "browser": "Non Mobile"
}
}
}

```

Example 2c - Game *completing* in example 2a

Each winning free spin will send a credit.

The **final free spin** will always be sent even if the payout for that last spin is 0. This is to ensure you receiving the gamestatemode 2 to complete the game.

```

...

"funds": {
    "debitandcredit": false,
    "fundinfo": [
        {
            "gamestatemode": 2,
            "transferid": "73000169375540c5be99c9d8d3240ec2",
            "currencycode": "EUR",
            "amount": 0.00,
            "jpwin": false,
            "jpcont": 0.0,
            "isbonus": false,
            "dtevent": "2017-07-27T03:15:32.327Z",
            "initialdebittransferid": "00ffcc297972e7119c0b74d02b2c397f"
        }
    ]
}
...

```

Example 3 – Jackpot payout

NOTE: If Jackpots are Enabled for a game, Habanero will always send **single debits and credits**

When jackpots are enabled D&C is not used as we cannot fund a jackpot without knowing the status of the debit. Therefore you would first receive the debit, then the jackpot win credit and finally the game credit.

The request will have the same gamedetails as 2a representing the gameinstance which won the jackpot. The additional information is:

- **jpuid** – this field will contain the JackpotId guid referencing which jackpot was won. You may use the jackpot webservice to match this data.
- **jpname** – the name as set in the Backoffice for this jackpot
- **jptypeid** – see addendum F for jackpot types
- **jpseed** – the seed/start value of the jackpot that was won (in the jackpot base currency)
- **jpwinbase** – Use this for recording jackpot level win information. NOT FOR PLAYER USAGE since it is in the base currency of the jackpot. **gamestatemode** – will always be 0

```
...  
"funds": {  
  "debitandcredit": false,  
  "fundinfo": [  
    {  
      "gamestatemode": 0,  
      "transferid": "73000169375540c5be99c9d8d3240ec2",  
      "currencycode": "EUR",  
      "amount": 1340.99,  
      "jpwin": true,  
      "jpuid": "762c4031-d6f3-427c-bf1e-6d95ad49d542",  
      "jpname": "My Test Jackpot",  
      "jptypeid": 4,  
      "jpseed": 1000.00,  
      "jpwinbase": 1340.99,  
      "jpcont": 0.0,  
      "dtevent": "2017-07-27T03:15:12.114Z",  
      "initialdebittransferid": "00ffcc297972e7119c0b74d02b2c397f"  
    }  
  ]  
}  
...
```

Alternative Fund Credits (Tournament and Prize Drops)

An Alternative Credit Endpoint can be configured via backoffice to receive alternate (non game) credits for tournament prizes and prize drops. Implementing this reduces manual crediting which will otherwise be required and greatly improves the player experience.

This request is only sent if your PlusPlay endpoint is configured.

Alternative Fund Request from Habanero (POST)

Format:

```
{
  "type": "altfundsrequest",
  "dtsent": " 2020-03-11T06:54:36.6215235Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef"
  },
  "altfundsrequest": {
    "accountid": "153",
    "altcreditttype": 1,
    "amount": 10.54,
    "currencycode": "EUR",
    "transferid": "524f180f6463ea1182e3cc2f711c5024",
    "dtevent": "2020-03-11T06:54:36.6175302Z",
    "description": "Tournament 'Superwin', Winner Position 2 (Money Prize) ",
    "tournamentdetails": {
      "score": 1,
      "rank": 1,
      "tournamenteventid": 43,
      "tournamenteventtype": 2,
      "tournamentprizetype": 1
    }
  }
}
```

The fields that will be sent in the request are:

Field	Description
accountid	Registered Third Party Player ID
altcreditttype	(int) 1 – Tournament
amount	Amount (greater or equal to 0)
currencycode	Player's currency code
transferid	Unique Transaction Id
dtevent	Date of Event
description	Payout Description

For Tournament Payout (**altcreditttype = 1**), the following info will also be sent in the request:

Field	Description
tournamentdetails	Tournament Details Object
partnerprizetext	PartnerPrize Custom Text (ONLY SHOWN FOR tournamentprizetype = 4)
partnerprizecode	PartnerPrize Custom Code (ONLY SHOWN FOR tournamentprizetype = 4)

Tournament Details Object:

Field	Description
score	(decimal) Score of Player in Tournament
rank	(int) Rank of Winner in Tournament
tournamenteventid	(int) Event ID of Tournament
tournamenteventtype	(int) Refer to Addendum J
tournamentprizetype	(int) Refer to Addendum K

RESPONSE:

```
{
  "altfundsresponse": {
    "status": {
      "success": true
    },
    "balance": 223.81,
    "currencycode": "EUR"
  }
}
```

NOTE:

Payouts using this endpoint will be processed a few minutes after the Tournament event and will be retried automatically up to 7 times.

Changes are backwards compatible for existing integrations that respond with “fundtransferresponse”

There will be no altfuundsrequest for Prize Type = 2 (Free Spins). This will be applied to the players directly.

Example 1 – Tournament Prize Type 1 (Money Prize)

The following example is a payout for a Money Prize Tournament.

```
{
  "type": "altfundsrequest",
  "dtsent": "2022-03-07T02:52:48.2327381Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef"
  },
  "altfundsrequest": {
    "accountid": "153",
    "altcreditttype": 1,
    "amount": 10,
    "currencycode": "EUR",
    "transferid": "8422dd97c2ef4b258f7f0946318aeb0e",
    "dtevent": "2022-03-07T02:52:48.2307367Z",
    "description": "Tournament 'PZ 1', Winner Position 1 (Money Prize)",
    "tournamentdetails": {
      "score": 0,
      "rank": 1,
      "tournamenteventid": 1192,
      "iscentral": true,
      "tournamenteventtype": 1,
      "tournamentprizetype": 1
    }
  }
}
```

Example 2– Tournament Prize Type 3 (Bet Multiplier)

The following example is a payout for a Bet Multiplier Prize Tournament.

```
{
  "type": "altfundsrequest",
  "dtsent": "2022-03-07T02:52:48.2327381Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef"
  },
  "altfundsrequest": {
    "accountid": "153",
    "altcreditttype": 1,
    "amount": 3.6,
    "currencycode": "EUR",
    "transferid": "064ac617812e4128bd7434cb8f8e9cc4",
    "dtevent": "2022-03-07T02:52:48.2307367Z",
    "description": "Tournament 'PZ 3', Winner Position 2 (Bet Multiplier)",
    "tournamentdetails": {
      "score": 0,
      "rank": 2,
      "tournamenteventid": 1194,
      "iscentral": true,
      "tournamenteventtype": 1,
      "tournamentprizetype": 3
    }
  }
}
```

Example 3 – Tournament Prize Type 4 (Partner Prize)

The following example is a payout for a Partner Prize Tournament.

Note: Amount can be "0" if there is no value configured for the prize type (Amount/Value for tournamentprizetype 4 is optional)

```
{
  "type": "altfundsrequest",
  "dtsent": "2022-03-07T02:52:48.2327381Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef"
  },
  "altfundsrequest": {
    "accountid": "153",
    "altcreditttype": 1,
    "amount": 0,
    "currencycode": "EUR",
    "transferid": "90518228223a414cb211c91f3fd16f52",
    "dtevent": "2022-03-07T02:52:48.2307367Z",
    "description": "Tournament 'PZ 4', Winner Position 1 (Partner Prize)",
    "partnerprizetext": "BMW 5 Series Sedan!",
    "partnerprizecode": "CAR",
    "tournamentdetails": {
      "score": 0,
      "rank": 1,
      "tournamenteventid": 1195,
      "iscentral": true,
      "tournamenteventtype": 1,
      "tournamentprizetype": 4
    }
  }
}
```

Note about Session/Token Management

The Backoffice contains a +Play Integration option called "Change Habanero Token":

When a new game is launched for a Seamless wallet player, we authenticate the token you send (example "111") and create an internal Habanero session for the player - example token "AAA".

Games communicate to Habanero with the internal token "AAA" and all Seamless wallet API actions use token "111".

If a new game is launched for the same player who has an active session in Habanero BUT your Seamless token for the player has been changed from "111" to "222" then:

Habanero starts sending the new Seamless wallet API token "222" for all actions for the player.

IF the setting '**Change Habanero Token**' is **enabled**:

- the Habanero internal token changes to a new token "BBB"
- any open games using the old token "AAA" will stop working because the old "AAA" token has expired.

IF '**Change Habanero Token**' is **disabled**:

- the internal Habanero token remains "AAA"
- any open games using token "AAA" will continue to work

Recommended

Enable the 'Change Habanero Token' setting and use the same Seamless token for all games launched during the player's session. By doing so, a player can play multiple games at the same time, but should they log in into your casino from a different location, your changed token will invalidate any previous game windows.

Not Recommended:

If you generate a new seamless Token on each game launch and this setting is enabled, a player will not be able to play multiple games at the same time. If you want to enable multiple games at a time disable this setting but be aware of the security risk.

If DISABLED, the following security issue can arise:

- Player opens a game in internet cafe Computer A, leaves the window open, and goes home.
- She then logs in and starts playing from home on Computer B.
- Computer A will remain active since the internal Habanero token is still valid (it did not change on new game launch) and would now be associated with Computer B's Seamless token.

Bonus (Regular)

Regular bonus requests will be sent at the end of the bonus rounds (credit without debit).

If you use Free Spin bonusing on Habanero, after the free spins have been completed and any wagering required has been completed, we will send you a credit message with the total payout from the bonus, **including 0 if no payout was made.**

Take note of:

IsBonus = true

Bonusdetails - containing the couponid, bonusbalanceid and coupontypeid, fsvalue, fscount, as created in Habanero via the backoffice or webservice API.

GameStateMode = will always be set to **2 (completing)** for requests sent at the end of the bonus round

fsvalue and fscount – the value per spin and total spins given when coupontypeid = 5

There is no **initialdebittransferid** – because there was no debit.

```
...
  "fundtransferrequest": {
    "token": "6607ae93-b716-468c-996c-41f8f68b75ca",
    "accountid": "153",
    "customplayertype": 0,
    "gameinstanceid": "35cedf60-7d72-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330634,
    "isretry": false,
    "retrycount": 0,
    "isrefund": false,
    "isrecredit": false,
    "funds": {
      "debitandcredit": false,
      "fundinfo": [
        {
          "gamestatemode": 2,
          "transferid": "fc08d90450164c38b55e42180179cc9b",
          "currencycode": "EUR",
          "amount": 342.6,
          "jpwin": false,
          "jpcont": 0.0,
          "isbonus": true,
          "dtevent": "2017-07-27T03:40:55.799Z"
        }
      ]
    },
    "gamedetails": {
      ...
    },
    "bonusdetails": {
      "couponid": "1606f337-7d72-e711-9c0b-74d02b2c397f",
      "couponcode": "MYCODE123",
      "bonusbalanceid": "96909B0e-3725-4152-93a1-d0387B0e6cbd",
      "coupontypeid": 5,
      "fsvalue": 0.15,
      "fscount": 10
    }
  }
}
```

Bonus Send-Per-Spin (Bonus transaction)

NOTE!! You most likely do not need this feature which must be enabled in Plus Play settings. See previous 'Bonus (Regular)' for the default/standard bonus credit message.

If your brand/account is required to obtain each debit and credit of Bonus/Coupon rounds, then this section should be integrated properly to ensure that information will be accurate.

By default, Habanero sends the Bonus/Coupon winnings at the **end of the round** for all coupon types. With this option enabled in the Backoffice of your brand, Habanero will then send (via fundtransferrequest) each round's Debit and Credit to your specified Transaction Endpoint.

NOTE: Only the following Coupon Types and configuration are supported as of now: CouponTypeid = 5 (NO Deposit: Free Spins), with NO MAX and NO WAGERING REQUIREMENTS. Otherwise it will revert to the default behaviour which will send the coupon's winnings at the end of the bonus round.

To Identify if the transaction is a bonus transaction, refer to the flags found in the transaction node:

1. **isbonus** - specifies that the transaction is a bonus transaction
2. **lastbonusaction <nullable>** - specifies if the transaction is the last transaction for the whole bonus/coupon and is either null (false) or true
3. **bonusamount** - specifies the Bonus Bet or the Bonus Win amount
(NOTE: This is the field to use to identify the amount of the bet/win of the transaction)
4. **amount** –specifies amount considered as real money
5. **bonusdetails** – this object will be present for all bonus transaction

For reference, the following logic should be applied to identify the transactions:

- BONUS DEBIT - If (isbonus and bonusamount < 0)
- BONUS CREDIT – if (isbonus and bonusamount >= 0 | | amount >= 0)
- FINAL BONUS TRANSACTION – if (isbonus and lastbonusaction)

Examples on the next page

Example 1 – Bonus Debit Transaction (First action)

The following example identifies if the transaction is a Bonus Debit Transaction:

IsBonus = true

lastbonusaction = null / false (indicates that the bonus round is ongoing)

bonusamount < 0

```
...
  "fundtransferrequest": {
    ...
    "funds": {
      "debitandcredit": false,
      "fundinfo": [
        {
          "gamestatemode": 1,
          "transferid": "5be85c23450b40f6b6502f500ce54805",
          "currencycode": "EUR",
          "amount": 0,
          "bonusamount": -1.8,
          "jpwin": false,
          "jpcont": 0,
          "isbonus": true,
          "dtevent": "2019-11-05T05:25:59.56Z",
          "initialdebittransferid": "5be85c23450b40f6b6502f500ce54805",
          "accounttransactiontype": 712,
          "gameinfeature": false,
          "lastbonusaction": false-NULL
        }
      ]
    },
    "gamedetails": {
      "name": "12 Zodiacs",
      "keyname": "SG12Zodiacs",
      "gametypeid": 11,
      "gametypename": "Video Slots",
      "brandgameid": "7ae03dd9-b0cf-44fb-aa4f-9196ee7ca468",
      "gamesessionid": "300ccf74-8cff-e911-849d-f44d30070f6d",
      "gameinstanceid": "5be85c23-450b-40f6-b650-2f500ce54805",
      "friendlygameinstanceid": 1315652,
      "channel": 1,
      "device": "Non Mobile",
      "browser": "Non Mobile"
    },
    "bonusdetails": {
      "bonusbalanceid": "05a33f5a-8cff-e911-849d-f44d30070f6d",
      "couponid": "68f888a4-ffd9-e911-8496-f44d30070f6d",
      "coupontypeid": 5,
      "couponcode": "3S8VX2PD",
      "fsvalue": 0.15,
      "fscount": 10
    }
  }
}
```

Example 2 – Bonus Credit Transaction (1st or Nth credit action)

The following example identifies if the transaction is a Bonus Credit Transaction:

IsBonus = true

bonusamount >= 0 OR amount >= 0

```
...
  "fundtransferrequest": {
    ...
    "funds": {
      "debitandcredit": false,
      "fundinfo": [
        {
          "gamestatemode": 1,
          "transferid": "5be85c23450b40f6b6502f500ce54805",
          "currencycode": "EUR",
          "amount": 1.6,
          "bonusamount": 1.6,
          "jpwin": false,
          "jpcont": 0,
          "isbonus": true,
          "dtevent": "2019-11-05T05:25:59.56Z",
          "initialdebittransferid": "5be85c23450b40f6b6502f500ce54805",
          "accounttransactiontype": 712,
          "gameinfeature": false
        }
      ]
    },
    "gamedetails": {
      "name": "12 Zodiacs",
      "keyname": "SG12Zodiacs",
      "gametypeid": 11,
      "gametypename": "Video Slots",
      "brandgameid": "7ae03dd9-b0cf-44fb-aa4f-9196ee7ca468",
      "gamesessionid": "300ccf74-8cff-e911-849d-f44d30070f6d",
      "gameinstanceid": "5be85c23-450b-40f6-b650-2f500ce54805",
      "friendlygameinstanceid": 1315652,
      "channel": 1,
      "device": "Non Mobile",
      "browser": "Non Mobile"
    },
    "bonusdetails": {
      "bonusbalanceid": "05a33f5a-8cff-e911-849d-f44d30070f6d",
      "couponid": "68f888a4-ffd9-e911-8496-f44d30070f6d",
      "coupontypeid": 5,
      "couponcode": "3S8VX2PD",
      "fsvalue": 0.15,
      "fscount": 10
    }
  }
}
```


Example 3 – Final Bonus Transaction

The following example identifies if the transaction is a Final Bonus Transaction:

isbonus = true

lastbonusaction = true

bonusamount >= 0 OR amount >= 0

```
...
  "fundtransferrequest": {
    ...
    "funds": {
      "debitandcredit": false,
      "fundinfo": [
        {
          "gamestatemode": 1,
          "transferid": "5be85c23450b40f6b6502f500ce54805",
          "currencycode": "EUR",
          "amount": 0.3,
          "bonusamount": 0.3,
          "jpwin": false,
          "jpcont": 0,
          "isbonus": true,
          "dtevent": "2019-11-05T05:25:59.56Z",
          "initialdebittransferid": "5be85c23450b40f6b6502f500ce54805",
          "accounttransactiontype": 712,
          "gameinfeature": false,
          "lastbonusaction": true
        }
      ]
    },
    "gamedetails": {
      "name": "12 Zodiacs",
      "keyname": "SG12Zodiacs",
      "gametypeid": 11,
      "gametypename": "Video Slots",
      "brandgameid": "7ae03dd9-b0cf-44fb-aa4f-9196ee7ca468",
      "gamesessionid": "300ccf74-8cff-e911-849d-f44d30070f6d",
      "gameinstanceid": "5be85c23-450b-40f6-b650-2f500ce54805",
      "friendlygameinstanceid": 1315652,
      "channel": 1,
      "device": "Non Mobile",
      "browser": "Non Mobile"
    },
    "bonusdetails": {
      "bonusbalanceid": "05a33f5a-8cff-e911-849d-f44d30070f6d",
      "couponid": "68f888a4-ffd9-e911-8496-f44d30070f6d",
      "coupontypeid": 5,
      "couponcode": "3S8VX2PD",
      "fsvalue": 0.15,
      "fscount": 10
    }
  }
...

```

Seamless Wallet Error Resolution

Errors are always resolved in a way which credits the player account – we will never debit a player's account without their knowledge:

- When a refund is requested - refund the original debit if you performed it successfully
- When a re-credit is requested, credit the funds if **not** originally performed
- If we receive a failure during a D&C Package request the following happens. We query if the debit was performed. If the debit was not performed, we void the game. If the debit was performed, we then query the status of the credit. If the credit was not done, we request a re-credit. In this case we do not perform refunds - we roll forward the outcome of the credit.

Depending on your token system, you should use the AccountId field to authenticate/check the player since the token may be expired on your side, especially with Expired games.

When are retries sent?

During Integration:

Login to the BackOffice at <https://bo-test.insvr.com> and go to Operations -> +Play Error Log.

This grid will show any API errors that occur and allow you to manually resolve them.

Note: If you wish - request Support to Enable the Batch Job to retry as per Live environment.

Live:

Retries will automatically be sent within 60 seconds after the failed request and retried up to 7 times in incrementing period of 1, 2, 3, 5, 15, 30, 90 minutes. The Error log used during integration is also available and you can manually resend transactions at any time.

How to test errors?

Trigger messages (only available on test environment)

In a slot game **press Shift D** to display the testing/debug panel. You will see an Expert Trigger section where text can be entered. Make sure you are logged in and not in fun play.

Habanero triggers (works out the box)

`{"jptriggers":"RandomMajor"}` – this will trigger a jackpot win (assuming that a RandomMajor jackpot is setup for the game). Valid options are RandomGrand, RandomMajor, RandomMinor, RandomMini, CoinSize, Race.

`{"txerror":"fail"}` - this will emulate failing to receive a response message from seamless api. Test this with both D&C messages and single credit and debit messages.

Passthrough triggers (optional for you to implement in your code)

All triggers of type "txerror" are passed from the game client to the third party in an element called "tt" (TestTrigger)

Example:

The tester has entered {"txerror":"dc-debiterror-before"} into the trigger field. The value "dc-debiterror-before" is therefore sent to the third party api in the "tt" field

```
"type": "fundtransferrequest",  
"tt": "dc-debiterror-before",  
"auth": {  
  ...
```

You may therefore build in test scenarios in your API!

Recommended Passthrough triggers to implement:

Authentication error:

[{"txerror":"auth"}] – fail authentication. Add a custom dialogmessageresponse and see how you can add your own custom message.

Debit&Credit flow:

{"txerror":"dc-debiterror-before"} - throw an exception before doing the debit.

{"txerror":"dc-creditorerror-before"} - throw an exception before the credit

{"txerror":"dc-creditorerror-after"} - throw an exception after the credit.

Single message flow:

{"txerror":"single-debiterror-before"} – throw before doing the debit

{"txerror":"single-creditorerror-before"} – throw before the credit

{"txerror":"single-creditorerror-after"} – throw after the credit

Refund Request

If the outcome of a single debit is unknown to Habanero (which will happen if we receive an exception or timeout) then we will request a refund.

In the example below, a debit was requested but Habanero received an exception/timeout. The thirdparty wallet did manage to complete the debit request. Therefore, the debit must be refunded and this information returned to Habanero.

Refund Request:

The refund request is the like a regular fundtransferrequest with the following exceptions:

1. `isretry` – set to true
2. `isrefund` – set to true
3. The `funds` node contains a single fundinfo object named `refund`
4. The `refund` node contains the `originaltransferid` **which is the transferid for the original debit which we want to refund/cancel**. The `transferid` **is the transferid for this refund request** which allows you to track the refund correctly.
5. The `originaltransferid` is the transferid of the transaction which failed.
6. `gamestatemode` will be set to 2 (Completed) if the refund is for an initial bet. If its for an additional bet, `gamestatemode` will be 0 (Continuing)

In this example, we tried to debit a customer 100.00 EUR but we did not receive a response. A Refund request is sent:

```
{
  "type": "fundtransferrequest",
  "dtsent": "2017-07-27T05:27:44.1831845Z",
  "basegame": {
    "brandgameid": "9bc350d1-f968-4e6a-aaa7-5e4d39941e05",
    "keyname": "BlackJack3H"
  },
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "locale": "en",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef"
  },
  "fundtransferrequest": {
    "token": "c157bc6f-b41a-452e-9551-b37284540236",
    "accountid": "153",
    "customplayertype": 0,
    "gameinstanceid": "8890a049-8c72-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330636,
    "isretry": true,
    "retrycount": 1,
    "isrefund": true,
    "isrecredit": false,
    "funds": {
      "debitandcredit": false,
      "refund": {
        "gamestatemode": 2,
        "originaltransferid": "8890a0498c72e7119c0b74d02b2c397f",
        "transferid": "b77bbebd796e480a9e08157adf12a644",
        "currencycode": "EUR",
        "amount": 100.00,
        "jpcont": 0.0,
        "dtevent": "2017-07-27T05:27:32.433Z",
        "initialdebittransferid": "8890a0498c72e7119c0b74d02b2c397f"
      }
    }
  }
}
```

```

    },
  },
  "gamedetails": {
    "name": "Blackjack (3 Hand)",
    "keyname": "BlackJack3H",
    "gametypeid": 4,
    "gametypename": "Blackjack",
    "brandgameid": "9bc350d1-f968-4e6a-aaa7-5e4d39941e05",
    "gamesessionid": "be22d5cf-8572-e711-9c0b-74d02b2c397f",
    "gameinstanceid": "8890a049-8c72-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330636,
    "channel": 1,
    "device": "Non Mobile",
    "browser": "Non Mobile"
  }
}
}

```

Refund Response:

On receipt of the refund request you must:

1. Depending on your logic - check if this [transferid](#), has been done (in other words has this refund been processed before otherwise use the [originaltransferid](#) value to check the outcome of the original debit request
 - a. If the original debit was done, you must **refund it** and return the [refundstatus](#) of 1
 - b. If the original debit was never performed, do nothing and return [refundstatus](#) of 2

refundstatus	Description
1	Refunded – you performed a refund for the request because the original debit was done.
2	Refund Not Required – no refund was required because the original request was not completed

2. If for any reason you could not refund, return success="false". These games will require manual resolving.

```

{
  "fundtransferresponse": {
    "status": {
      "success": true,
      "refundstatus": 1
    },
    "balance": 2238.1,
    "currencycode": "EUR"
  }
}

```

Habanero game continuation:

When we receive a [refundstatus](#) of "1" (Refunded), then we will write a debit record for the original debit (because you did perform it) followed by a credit for the refund that was performed. This allows our records to match.

When we receive a [refundstatus](#) of "2" (Refund not Required) – then we do not need to write any records because the original debit was never performed.

If the **gamestatemode** of the original debit was New Game (1) – then the gameinstance is set to "Refunded" if the original debit was done it is set to "Void – Debit Not Done". If the gamestatemode was 0 (continuing) then the game will be set back to "In Progress" and resume the next time the player opens the game.

Re-Credit Requests

If the outcome of a credit is unknown to Habanero, we will reissue the credit request. If the original credit was not successful you must redo/commit the credit. If the original credit was successful return the same data as if the original credit was done

Re-credit Request:

The re-credit request is the same as a regular fundtransferrequest with the following exceptions:

1. `isretry` – set to true
2. `isrecredit` – set to true.

Note: The `transferid` and `originaltransferid` will match the original credit request.

Recredit Response:

If the credit has been done previously or you have now successfully committed the credit you return `success = "true"`. Habanero will then resolve the game status accordingly and write any transaction records required to match your records.

```
{
  "fundtransferresponse": {
    "status": {
      "success": true
    },
    "balance": 2238.1,
    "currencycode": "EUR"
  }
}
```

QueryRequest (used for Dual D&C failure resolution)

If the outcome of a D&C Package request is unknown then we need to determine where the error occurred since it could have debited ok but failed on the credit. The problem is resolved in the following manner:

1. Query the status of the Debit's TransferId
 - a. If the debit was **not** performed, we void the game
2. If the Debit was performed, we query the status of the Credit's TransferId. If the credit was successful we complete/resume the game, otherwise we perform the re-credit before completing/resuming the game.

Process Flow:

```
if (DebitSuccess == false)
{
    //Habanero voids the game
}
else
{
    if (CreditSuccess == false)
    {
        //Re-Credit the transaction
    }

    //Habanero resumes/completes the game
}
```

To query the status we send a Query Request.

IMPORTANT Note: The queryrequest includes both the token and accountid *IF you need* to use them, but you should not be doing any authentication on the token.

Example on next page...

Here is the Original Debit and Credit request which **did not receive a response**:

```
{
  "type": "fundtransferrequest",
  "dtsent": "2017-07-27T05:36:01.4299356Z",
  "basegame": {
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "keyname": "SGAllForOne"
  },
  "auth": {... },
  "fundtransferrequest": {
    "token": "8f6a28c9-5c7f-456a-a9cf-f33e4ccf4a16",
    "accountid": "153",
    "customplayertype": 0,
    "gameinstanceid": "e39d1773-8d72-e711-9c0b-74d02b2c397f",
    "friendlygameinstanceid": 1330637,
    "isretry": false,
    "retrycount": 0,
    "isrefund": false,
    "isrecredit": false,
    "funds": {
      "debitandcredit": true,
      "fundinfo": [
        {
          "gamestatemode": 1,
          "transferid": "e39d17738d72e7119c0b74d02b2c397f",
          "currencycode": "EUR",
          "amount": -125.00,
          "jpwin": false,
          "jpcont": 1.0,
          "isbonus": false,
          "dtevent": "2017-07-27T05:36:01.4Z",
          "initialdebittransferid": "e39d17738d72e7119c0b74d02b2c397f"
        },
        {
          "gamestatemode": 2,
          "transferid": "a04e00021b88415eafbcaa5d89ea66fa",
          "currencycode": "EUR",
          "amount": 85.00,
          "jpwin": false,
          "jpcont": 0.0,
          "isbonus": false,
          "dtevent": "2017-07-27T05:36:01.426Z",
          "initialdebittransferid": "e39d17738d72e7119c0b74d02b2c397f"
        }
      ]
    },
    "gamedetails": {
      "name": "All For One",
      "keyname": "SGAllForOne",
      "gametypeid": 11,
      "gametypename": "Video Slots",
      "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
      "gamesessionid": "e29d1773-8d72-e711-9c0b-74d02b2c397f",
      "gameinstanceid": "e39d1773-8d72-e711-9c0b-74d02b2c397f",
      "friendlygameinstanceid": 1330637,
      "channel": 1,
      "device": "Non Mobile",
      "browser": "Non Mobile"
    }
  }
}
```


Step 1: Perform a QueryRequest for the Debit transfer:

```
{
  "type": "queryrequest",
  "dtsent": "2017-07-27T05:37:50.6913212Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "locale": "en",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef"
  },
  "basegame": {
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "keyname": "SGAllForOne"
  },
  "queryrequest": {
    "transferid": "e39d17738d72e7119c0b74d02b2c397f",
    "accountid": "153",
    "gameinstanceid": "e39d1773-8d72-e711-9c0b-74d02b2c397f",
    "initialdebittransferid": "e39d17738d72e7119c0b74d02b2c397f",
    "friendlygameinstanceid": 1330637,
    "token": "8f6a28c9-5c7f-456a-a9cf-f33e4ccf4a16", //do NOT authenticate on this token.
    //included for certain customer needs
    "queryamount": -125 //value of the original transfer being queried
  }
}
```

Response:

Return a fundtransferresponse with a status node. We only use the success attribute.

```
{
  "fundtransferresponse": {
    "status": {
      "success": true,
    }
  }
}
```

When replying to the status of the Debit transfer:

- **success** = false means that the original debit was never done. Habanero will therefore void the game and no further actions will be requested on this game instance.
- **success** = true means that the original debit was done. Habanero will therefore perform another query on the credit transfer status.

Step 2: Perform a QueryRequest for the Credit transfer (because the Debit was done)

```
{
  "type": "queryrequest",
  "dtsent": "2017-07-27T05:37:50.6913212Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "locale": "en",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcfef6ef"
  },
  "basegame": {
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "keyname": "SGAllForOne"
  },
  "queryrequest": {
    "transferid": "a04e00021b88415eafbcaa5d89ea66fa",
    "accountid": "153",
    "gameinstanceid": "e39d1773-8d72-e711-9c0b-74d02b2c397f",
    "initialdebittransferid": "e39d17738d72e7119c0b74d02b2c397f",
    "friendlygameinstanceid": 1330637,
    "token": "8f6a28c9-5c7f-456a-a9cf-f33e4ccf4a16",
    "queryamount": 85
  }
}
```

When replying to the status of the Credit transfer:

- **success = true** means that the original credit was also done, and we will resolve the status of the game.
- **success = false** means that the original credit **was not done**, and we will then **send a Re-Credit** message as per 11.2

```
{
  "fundtransferresponse": {
    "status": {
      "success": true,
    }
  }
}
```

Custom Game Dialog Messages (Reality Checks, Maintenance and more)

To return some information to a player for Reality Checks or to show a maintenance message return the DialogMessageResponse node with the dialog type:

Dialog Types:

0	Pressing OK button will dismiss message box and the player can continue.
1	Pressing OK reloads the current game.
2	Pressing OK returns the player to the lobby if you provided a Lobby URL during game launch
3	Non dismissable - The player will see the message and cannot dismiss the dialog
5	Reality Check Option A: Displays: <ul style="list-style-type: none">- Close button : which will redirect to the LobbyUrl (set during launch) or close window- Continue button : will dismiss the popup and do a blind GET request to specified "continueurl"- History button : will open the HistoryUrl specified during launch or what is specified in "historyurl"
6	Shows the message along with a Lobby or Cashier button.

Example:

A player is requesting a debit but you wish to deny the debit and show a message:

You must return success=false and either nofunds=true or autherror=true. Failing to send nofunds="true" or autherror="true" will result in a transaction which will be retried.

```
{
  "fundtransferresponse":{
    "status":{
      "success":false,
      "autherror":true,
    },
    "balance":150.0,
    "currencycode":"EUR"
  },
  "dialogmessageresponse":{
    "message":"You have reached your limit. Return tomorrow.",
    "type":2
  }
}
```

UKGC Reality Check Example:

Use type: "5"

```
{
  "fundtransferresponse":{
    "status":{
      "success":false,
      "autherror":true
    },
    "balance":150.0,
    "currencycode":"EUR",
  },
  "dialogmessageresponse":{
    "message":"You have been playing for X minutes",
    "type":5,
    "continueurl" : "https://yourserver.com/clearcheck/{playerid}"
    "historyurl" : "https://yourserver.com/history"
  }
}
```

Game Bet/Config Parameters (Optional) [Seamless Wallet only]

NOTE: This is NOT Recommended – please discuss with your Habanero account manager if considering.

Game parameters such as min/max and increments can be configured in the Habanero backend or alternately, you may configure settings on your system which will be queried by Habanero during the start-up of a game.

This allows you to control the parameters as well as to control/run the jackpot.

This method is only called if your PlusPlay settings in the Habanero Backoffice include a configured endpoint url.

When a game loads, this request will be sent to you. In the event of a failed response we will fall back to the default settings. If you do not want the default settings, ensure that all Habanero Backoffice Game configs are disabled.

This request is also sent once our internal config cache expires - You should always be prepared to respond.

Config Detail Request from Habanero

The message will include:

1. **token** – the player token as usual
2. **accountid** – alternative to using the token which is acceptable in this scenario
3. **brandgameid** – this is the Habanero BrandGameId requesting configuration details
4. **keyname** – the game Keyname if using this instead of BrandGameId
5. **gametypeid** – the type of game (See Addendum D)

If this is a slot game then the following will be populated with values, otherwise will be 0 or false

6. **slot_maxcoinsplayed** - the number of coins played in the game when the top number of lines/ways are selected
7. **slot_isfixedlinesorways** - true|false – whether the player can change number of paylines or ways
8. **slot_numberlinesorways** - the number of ways or payline this game has. But this is **not** of much importance.

```
{
  "type": "configdetailrequest",
  "dtsent": "2017-08-04T03:19:33.4666559Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "A-DESKTOP",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcf6ef"
  },
  "configdetailrequest": {
    "token": "9fb7baf1-6df7-4a7b-adca-323938fbd7c",
    "accountid": "153",
    "customplayertype": 0,
    "brandgameid": "26876366-799b-417a-89d9-63cfab6475fe",
    "keyname": "SGAllForOne",
    "gametypeid": 4,
    "slot_maxcoinsplayed": 30,
    "slot_numberlinesorways": 243,
    "slot_isfixedlinesorways": false
  }
}
```

Response:

```
{
  "configdetailresponse": {
    "status": {
      "success": true,
      "message": "Config Found",
    },
    "minstake": 1.0,
    "maxstake": 5.0,
    "stakeincrement": "0.10|0.50|1|5|10",
    "levelincrement": "1|5|10",
    "defaultstake": 0.50,
    "maxpaylimit": 0.0,
    "mininsidestake": 0.0,
    "maxinsidestake": 0.0,
    "minoutsidestake": 0.0,
    "maxoutsidestake": 0.0
  },
}
```

The response message varies according to the **gametype**.

1. **status** – "true" or "false" (lower case) and descriptive message
 - j. – If success = false then we will fall back to Habanero defaults
2. **configdetailresponse** node
 - a. **minstake** - the minimum stake amount (not applicable to slots)
 - b. **maxstake** – maximum stake amount (not applicable to slots)
 - c. **stakeincrement** (PIPE delimited decimals) – the stake/coin values
 - d. **levelincrement** (PIPE delimited integers – bet level increment for slot games),
 - e. **defaultstake** – the default selected chip which must be in the stakeincrement field
 - f. **maxpaylimit** – maximum payout per spin (for slot games)
 - g. **mininsidestake** - roulette only
 - h. **maxinsidestake** - roulette only
 - i. **minoutsidestake** - roulette only
 - j. **maxoutsidestake** - roulette only

Calculating the default bet for a slot game: (you use the slot_maxcoinsplayed received in the request)

Default Bet = DefaultStake * lowest LevelIncrement * slot_maxcoinsplayed

Max Bet = highest StakeIncrement * highest LevelIncrement * slot_maxcoinsplayed

Player Game End Session (Optional) [Seamless Wallet only]

You can receive a request when the player closes the game window or gets logged out of Habanero.

NOTE:

- This method is only called if your PlusPlay settings in the BO include an End Session Endpoint

When the player has moved out of the game (game closes, refreshes, redirected to LobbyURL), this request will be sent to you. When the player has been logged out of Habanero (forced or triggered), this request will also be sent to your endpoint.

EndSession Detail Request from Habanero (POST)

The message will include:

1. **token** – the player token as usual
2. **accountid** – alternative to using the token which is acceptable in this scenario
3. **command** – this can be “logout” or “gameclose”. This will specify where the call has originated

```
{
  "type": "playerendsession",
  "dtsent": "2019-05-09T10:22:35.7422097Z",
  "auth": {
    "username": "habanero",
    "passkey": "85CDVPBzrVzpVK02raSCK0g71bLBU",
    "machinename": "HB-DEV01",
    "brandid": "6cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef"
  },
  "basegame": {
    "brandgameid": "045a80d0-5b2a-e311-80bb-74d02b2c397f",
    "keyname": "SGAllForOne"
  },
  "playerendsessionrequest": {
    "token": "9fb7baf1-6df7-4a7b-adca-323938fdbc7c",
    "accountid": "153",
    "command": "logout"
  }
}
```

RESPONSE: Any HTTP 200 response

NOTE: We can send this request more than once, depending on the gamesession found in the system

Seamless Wallet Checklist / Signoff

Scenario	
Additional Credits and Debits	In slot games test feature mode by pressing Shift-D in the game and Free Spins button. Ensure additional credits where gamestatemode=0 are correctly credited. In Blackjack game double to test additional debits.
Insufficient funds	Check that insufficient funds scenario is correctly handled. E.g. Have a balance of \$5 and bet \$10. The game client must show "Insufficient Funds". If you see a different error message you have not implemented it correctly - ensure you are sending "nofunds" = true in the status message and the latest balance.
Jackpot win	Trigger a jackpot and ensure that you correctly log the win/credit
Bonus/Coupon credit	After a bonus is completed we send a fundtransferrequest with all the information of the game for which the bonus has been won. The isbonus=true field is set in the fundinfo as well as bonusdetails node is populated. Ensure you accept this credit which will have no matching debit.
Expired Session	Expire your token and check that the Habanero game shows the correct message of Session Expired. This request should not show in the +Play Error Log If you are seeing an error, make sure you are sending "autherror" = true in the status message.
Game Debit Error	Use the triggers to fail a debit. Habanero will request a refund of the game. Ensure that the refund is only done once.
Refund response Error	Return an error to Habanero after completing a Refund. Resend the refund from Habanero and make sure you do not refund twice. Also check that you are using the refundstatus type correctly.
Game Credit Error	Use the triggers to fail a credit. Habanero will request a recredit. Ensure you only re-credit if not previously done/successful.
Dual Debit/Credit (D&C) Error	<ol style="list-style-type: none"> Throw an error before the debit. The game will be voided. Throw an error before the credit. <ul style="list-style-type: none"> Habanero queries if the original debit was done. If it was not done we void the game. If the debit was done, we then continue to query the Credit. If the credit was done the game is completed/resumed If the credit was not done, we send the re-credit and then complete/resume the game.
Expired Games	Make sure you accept gamestatemode=3 and the 0 credit which is sent when an incomplete game is expired after X days. Expire a game by using the Backoffice game detail view from the playersheet.
Alternative Fund Credits	Implement this method to accept auto payouts for tournaments and prize drops to avoid manually having to credit player accounts.
Send a custom message (Optional)	Use the Game Dialog Message to send a custom message to the game for Reality Checks, Daily safety limits hit etc.

-- this page intentionally left blank --

Graphical Game Result for Customer Support / Backoffice

Note 1: Players can see their Game history directly from the Habanero game by clicking on the Menu and going to Game History. This option can be disabled in the Backoffice if you wish.

Note 2: If you want to import game result data or for reporting use the JSON/SOAP Web service. This functionality is only for creating a user-friendly view.

View a **Game** result

You may display the graphical game result HTML markup in an iframe on your own system for CS support.

Perform an HTTP **GET** to "https://app.____.com/games/history/?"
and output the content to an iframe.

Use the following variables:

Variable	Description
brandid	brandid
Gameinstanceid or FriendlyId	Gameinstanceid guid Friendly Game id (Int64)
hash	SHA256 Hash of (gameinstanceid or friendlyid). tolower() + brandid. tolower() + apikey. tolower()
viewtype	Set to " game "
locale	Language code



Example:

<habanero-history-url>/games/history/?brandid={**brandid**&gameinstanceid={**gameinstanceid**}
&hash={**hash**&locale={**locale**&viewtype=**game**

<habanero-history-url>/games/history/?brandid={**brandid**&friendlyid={**friendlyid**}
&hash={**hash**&locale={**locale**&viewtype=**game**

Output:

12 Zedexes	n° 137888829
Status	Completed
Currency	IDR
Lines	18
Coin Denomination	IDR 50
Coins	18
Bet Level	1
Stake (Final)	IDR 900
Payout (Final)	IDR 400
Start Date	March 12, 2021, 13:24:26 GMT+8
End Date	March 12, 2021, 13:24:26 GMT+8
Jackpot Contribution	IDR 18.0000
Balance	IDR 8.588.580

Event # 1	Main	IDR 400		
Type: Spin	FINAL REELS			
Payout: IDR 400				
Type	Multiplier	Symbol	Payout	Payline
Payline	1		IDR 400	11

View **Player** history

You can open the players Habanero game history view (the same view the player can see by clicking History within a game) and include optional date range and limit to a specific game keyname. A grid will be displayed with the game data. Clicking on a line will show the graphical result.

Perform an HTTP GET to "https://app.____.com/games/history/?"
and display it in an iframe

Use the following variables:

Variable	Description
brandid	brandid
username	Player username
OR playerid	OR instead of username, the Habanero playerid <guid>
hash	SHA256 Hash of (username OR playerid). tolower() + brandid. tolower() + apikey. tolower()
viewtype	Set to " player "
locale	Language code
dtStartUTC	OPTIONAL – yyyyMMddHHmmss UTC date (If used must also set dtEndUTC)
dtEndUTC	OPTIONAL – yyyyMMddHHmmss UTC date (If used must also set dtStartUTC)
KeyName	The game keyname you wish to filter by

Example:

Redirect to:

View last completed games with no other filter

```
<habanero-history-url>/games/history/?brandid={brandid}&username={username}  
&hash={hash}&locale={locale}&viewtype=player
```

View specific game keyname in a date range

```
<habanero-history-url>/games/history/?brandid={brandid}&username={username}&hash={hash}  
&dtStartUTC={dtstartUTC}&dtEndUTC={dtEndUTC}&keyname={keyname}  
&locale={locale}&viewtype=player
```

View a Game Event result

You can also display the graphical game result events of a game. This view type will omit the base game details and only show the game event information.

Perform an HTTP **GET** to "https://app.____.com/games/history/?" and output the content to an iframe.

Use the following variables:

Variable	Description
brandid	brandid
Gameinstanceid or Friendlyid	Gameinstanceid guid Friendly Game id (Int64)
hash	SHA256 Hash of (gameinstanceid or friendlyid). tolower() + brandid. tolower() + apikey. tolower()
viewtype	Set to " gameevents "
locale	Language code
eventindex	(Optional) display only a specific event. Used in conjunction with the "featureno" provided in Seamless API message.

Example:

```
<habanero-history-url>/games/history/?brandid={brandid}&gameinstanceid={gameinstanceid}
&hash={hash}&locale={locale}&viewtype=gameevents
```

```
<habanero-history-url>/games/history/?brandid={brandid}&friendlyid={friendlyid}
&hash={hash}&locale={locale}&viewtype=gameevents
```

Output:



Show game Paytable

You can also show the Paytable in the game history (only applicable for specific Slot games). If the game has no payable information, the image will not be shown.

Perform an HTTP **GET** to "https://app.____.com/games/history/?" and output the content to an iframe.

Use the following variables:

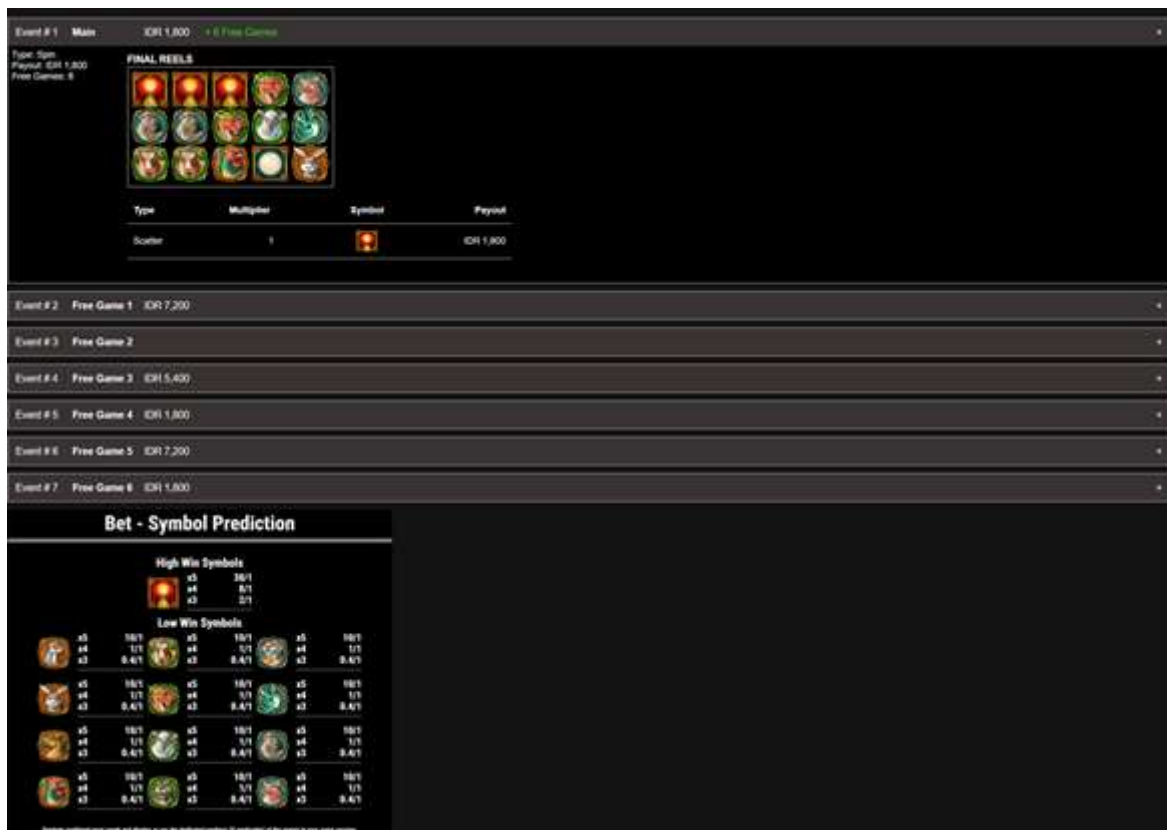
Variable	Description
brandid	brandid
Gameinstanceid or FriendlyId	Gameinstanceid guid Friendly Game id (Int64)
hash	SHA256 Hash of (gameinstanceid or friendlyid). tolower() + brandid. tolower() + apikey. tolower()
viewtype	Set to " game "
locale	Language code
showpaytable	1 or 0. Shows the game's payable information at the bottom of the page

Example:

```
<habanero-history-url>/games/history/?brandid={brandid}&gameinstanceid={gameinstanceid}
&hash={hash}&locale={locale}&viewtype=game&showpaytable=1
```

```
<habanero-history-url>/games/history/?brandid={brandid}&friendlyid={friendlyid}
&hash={hash}&locale={locale}&viewtype=game&showpaytable=1
```

Output:



The screenshot displays a slot game interface. At the top, it shows 'Event #1: Main' with a 'JCR 1,000' and '6 Free Games' bonus. Below this, a 3x3 grid of symbols is shown, including various fruit and animal icons. A 'FINAL REELS' section is visible. Below the grid, a table lists the symbols and their payouts:

Type	Multiples	Symbol	Payout
Scatter	1	[Symbol]	JCR 1,000

Below the table, a list of events is shown:

- Event #2: Free Game 1: JCR 7,200
- Event #3: Free Game 2
- Event #4: Free Game 3: JCR 5,400
- Event #5: Free Game 4: JCR 1,000
- Event #6: Free Game 5: JCR 7,200
- Event #7: Free Game 6: JCR 1,000

At the bottom, a 'Bet - Symbol Prediction' section is visible, showing a grid of symbols and their corresponding payouts. The grid is organized into 'High Win Symbols' and 'Low Win Symbols' sections.

Client-side Jackpot Ticker

Get active jackpots with various querystring parameters from the client-side browser.

Perform an HTTP **GET** to "https://app-[environment].insvr.com/service/jackpot/api/getjackpots?"

Querystring Variable	Description	Example	Notes
brandid	Your brandid	6cf6f2f8-0ecd-4829-9bb7-e78abcffe6ef	Get all active jackpots if leave keyname and jackpotid blank
Optional parameters:			
keyname	string	SG12Zodiacs	use with empty jackpotid
jackpotid	guid	4f989b0e-86ff-e911-b647-7085c2c29947	Ignores keyname if supplied
ig	int	1	shows the list of keynames associated with each jackpot group

Response:

Variable	Description	Example
JackpotGroup	string	JP Group
JackpotId	guid	4f989b0e-86ff-e911-b647-7085c2c29947
Name	string	JP Grand Jackpot
CurrencyCode	string	Eur, CNY
CurrencySymbol	string	€, ¥
JackpotAmount	decimal	1000.15
JackpotTypeId	int	3 (See Addendum F)
Increment	decimal (adjustable setting from the Backoffice)	0.01
Offset	decimal (adjustable setting from the Backoffice)	0.99
Interval	int (adjustable setting from the Backoffice)	10
Games	string[]	Array of games associated with the Jackpot (only shows if ig=1)

Example:

Parameter **Increment, Offset, Interval**:

Ticker start from (**{JackpotAmount}** x **{Offset}**) and increase by **{Increment}** for every **{Interval}** millisecond

JackpotAmount=1000, Increment=0.01, Offset=0.99, Interval=10

The ticker would start from 990 and increase by 0.01 every 10 milliseconds.

Get all active jackpots for the **brandid**

https://app-test.insvr.com/service/jackpot/api/getjackpots?brandid={brandid}

Get jackpots linked to game by **brandid** and game **keyname**

https://app-test.insvr.com/service/jackpot/api/getjackpots?brandid={brandid}&keyname={keyname}

Get specific jackpot using **brandid** and **jackpotid** (JackpotId may be viewed in Backoffice or supplied by CS)

https://app-test.insvr.com/service/jackpot/api/getjackpots?brandid={brandid}&jackpotid={jackpotid}

Successful response returning jackpots:

```
[
  {
    "JackpotGroup": "JP Group",
    "JackpotId": "4f989b0e-86ff-e911-b647-7085c2c29947",
    "Name": "JP Grand Jackpot",
    "CurrencyCode": "EUR",
    "CurrencySymbol": "€",
    "JackpotAmount": 1000.15,
    "JackpotTypeId": 3,
    "Increment": 0.01,
    "Offset": 0.99,
    "Interval": 10
  },
  {
    "JackpotGroup": "CNY Group",
    "JackpotId": "5a389b0e-12bf-c123-e248-5791a4b34891",
    "Name": "CNY Minor Jackpot",
    "CurrencyCode": "CNY",
    "CurrencySymbol": "¥",
    "JackpotAmount": 1999.89,
    "JackpotTypeId": 4,
    "Increment": 0.01,
    "Offset": 0.99,
    "Interval": 10
  }
]
```

Failed response:

```
"Error: Invalid BrandId"
```

Test/Demo:

Demo tickers implement the function in <https://app-test.insvr.com/service/jackpot/test.aspx>. You can copy the example html, javascript, and css to your page then change the url with your brandId, keyname, and jackpotId.

-- this page intentionally left blank --

WebHooks

A WebHook is a simple event-notification system which does an HTTP callback when certain things happen.

Multiple endpoints can be configured per Brand in the Global/Brand, find brand, Go to Webhooks.

- Each endpoint has its own queue to which messages are added.
- The JSON message will be sent to the endpoint in near real time
- If the endpoint does not accept/respond with an HTTP 200 message within 5 seconds, then the message is added to the Defer queue
- The Defer queue is processed every 2 minutes and each message will be attempted a maximum of 2 times
- If both deferred attempts fail, the message is added to the Fail queue and will not be sent again

Message Types

There are currently two categories of messages which can be subscribed to by configuring settings in the Backoffice:

- Bonus events
- Backoffice events

Wrapper Message Structure

All messages have a common wrapper in the format of:

Id <string>	Unique message Id which can be used to ensure you do not process a message twice in case of failure to return HTTP 200 on receipt of message
Type <string>	The type of message
Secret <string>	Secret key / Password which you can authenticate
RetryCount <int>	Indicates if this is a retry. First attempt will be 0, second attempt will be 1 etc.
DtQueued <datetime>	The Date and Time the message was queued. Note! This is not the date the event happened. It is for informational purposes. Format is: ISO 8601 formatted string: "2009-02-15T00:00:00Z"
Message <string>	An embedded JSON message of the type indicated by the Type field.

Example JSON message showing a user login to the Backoffice

```
{
  "Id": "e1acba0d-f4e2-44a6-ba68-06306614c121",
  "Type": "Brand_UserLogin",
  "Secret": "our-complex-password-here",
  "DtQueued": "2015-04-14T10:05:54.5929922Z",
  "RetryCount": 0,
  "Message": "{\\"IPAddress\\":\\"127.0.0.1\\",\\"GeoCity\\":\\"Local
Host\\",\\"GeoCountry\\":\\"Local\\",\\"BrandId\\":\\"6cf6f2f8-0ecd-4829-9bb7-
e78abcffe6ef\\",\\"Username\\":\\"tony\\",\\"UserId\\":\\"996178d5-3d1b-4c36-a42e-
e65557ac6a46\\",\\"DtOfEvent\\":\\"2015-04-14T10:05:54.5929922Z\\"}"
}
```

C# example to receive a message and parse it:

```
string rawMessage = System.Text.Encoding.UTF8.GetString(Request.BinaryRead(Request.TotalBytes));

//using Json.Net and Dynamic type
dynamic wrapper = JObject.Parse(rawMessage);
dynamic msg = JObject.Parse(wrapper.Message.ToString());
if (wrapper.Type == "Brand_UserLogin")
{
    var test = ("Username {0} logged in from IP {1} at {2} to BrandId {3}", msg.Username,
msg.IPAddress, msg.DtOfEvent, msg.BrandId);
}
```

Base Message structure:

All web hook messages contain:

BrandId <guid>	The BrandId owning this event
DtOfEvent	The Date and Time of the action which triggered the event. Format is: ISO 8601 formatted string: "2009-02-15T00:00:00Z"

Bonus event Messages:

All Bonus_ messages contain:

BonusBalanceId <guid>	Instance of this bonus and balance for the player
PlayerId <guid>	The internal habanero <guid> for playerid
Username <string>	The username (transfer wallet) or the AccountId (PK) (seamless wallet)
CouponId <guid>	Internal guid of the coupon
CouponCode <string>	The coupon code
CouponTypeId <int>	The coupontypeid
GameKeyName <string>	The keyname identifier of the game if this bonus is for a specific game

Bonus_Redeemed

This event fires when a bonus is redeemed/applied to a player. Note: it does NOT send if the bonus was redeemed using the Json/Soap webservice. It DOES send for queued bonuses, scheduled bonuses, batch upload bonuses, manual backoffice redemption etc.

TotalRedeemed <int>	Total number of coupons redeemed (useful if you generate multi-use coupons)
BonusValue <decimal?>	The total money value of the bonus
FreeSpinsGiven <int?>	Indicated how many free spins assigned (IF coupontypeid is for freespins)
ValuePerFreeSpin <decimal?>	The money value of each free spin (IF coupontypeid is for freespins)
CoinSize <decimal?>	The coin denomination for free spin (IF coupontypeid is for freespins)

Bonus_Completed

This event fires when a bonus is completed.

CouponStatusId <int>	The reason why the bonus is completed: 3 Cancelled by Withdrawal 4 Expired 7 Completed - Wager Met 8 Completed - Out of Funds 9 Cancelled by Deposit 10 Deleted by Player 11 Deleted by Admin
PromoBalance <decimal>	Money value of the bonus balance at completion
ConvertedAmount <decimal>	Money value converted to real balance (Max Conversion may have been applied)

Bonus_GameAction

This event fires when a bonus is used in a game. e.g., after each spin

GameInstanceId <guid>	Game round identifier
FriendlyGameInstanceId <Int64>	Integer game round identifier
FreeSpinsBalance <int>	Number of free spins remaining
BonusBalance <decimal>	Money value of the bonus balance
WageringProgressPercent <decimal>	% Complete – applicable if the coupon had a Wagering requirement
WagerRemaining <decimal>	The amount of wagering still needed to convert the Bonus to Real
TotalWagerRequired <decimal>	The total money amount needed to be wager
DtExpires <string>	Expiry date of bonus

Backoffice event Messages:

Brand_UserLogin

This event fires when a Backoffice user selects a Brand from the drop-down list after they have logged in.

Username	Backoffice user's username
UserId	Backoffice users internal userid
IPAddress	The users IP Address
GeoCity	The city data from a Geo IP Lookup of the IP Address
GeoCountry	The Country data from the Geo IP Lookup of the IP Address

Brand_UserLogOff

This event fires when a Backoffice user presses Logout. This event is not guaranteed to be sent since the Backoffice does not have a process of logging off users when they timeout. The POSSessionStart and POSSessionEnd are most likely of more use.

Username	Backoffice user's username
UserId	Backoffice users internal userid

-- this page intentionally left blank --

Certification Files

Habanero offers an endpoint/URL for viewing certified files for the games. It includes the hash details, version, and other related information. You can select from the different response formats below:

HTML Table: <https://gi-<server>.insvr.com/hashtable>

JSON: <https://gi-<server>.insvr.com/hashlist>

The links will render the list of games with their respective information. It can also be filtered to your specific brand by passing additional parameter values.

Name	Description
brandId	Filter the files based on configured games in the specified BrandId
gamesonly	This will remove the Jackpot and RNG information from the response. 1 shows games only (no RNG and Jackpot). Omit to show all info. gamesonly =1 – Show Games only
format	Valid values: (for hashlist only) Xml – Return list in XML format Json – return list in JSON format
sd	sd=1 Show Dependencies. Only works if brandId is passed and forces gamesonly to true. Shows other dlls for each game and adds “dependencies” in hashlist api
cd	Cd=1 Custom Dependency. Only works if sd=1. Shows Jackpot DLLs according to the settings from the backoffice.

-- this page intentionally left blank --

Addendum

Addendum A - Languages

Supported Languages:

Language	Locale
Bulgarian	bg
Burmese	my
Chinese (Simplified)	zh-CN
Chinese (Traditional)	zh-TW
Croatian	hr
Danish	da
Dutch	nl
English	en
Estonian	et
Finnish / Suomi	fi
French	fr
German	de
Greek	el
Hebrew	he
Hungarian	hu
Indonesian	id
Italian	it
Japanese	ja
Korean	ko
Latvian	lv
Lithuanian	lt
Malay	ms
Norwegian	nn
Portuguese	pt
Romanian	ro
Russian	ru
Serbian	sr
Spanish	es
Swahili	sw
Swedish	sv
Thai	th
Turkish	tr
Vietnamese	vi
Social Gaming English	en-soc

Addendum B - Currencies

Please notify us if you require a currency that is not in this list and not in the Backoffice

Currency	Code
Albania Lek	ALL
Algerian Dinar	DZD
Angolan Kwanza	AOA
Argentine Peso	ARS
Armenian Dram	AMD
Australian Dollars	AUD
Azerbaijani Manat	AZN
Bahamian Dollar	BSD
Bahraini Dinar	BHD
Bangladeshi Taka	BDT
Belarusian Ruble (NEW)	BYN
Belarusian Ruble (OLD)	BYR
Bolivian Boliviano	BOB
Bosnia-Herzegovina Mark	BAM
Botswanan Pula	BWP
Brazilian Real	BRL
British Pounds	GBP
Brunei Dollar	BND
Bulgarian Lev	BGN
Burmese Kyat	MMK
Burundian Franc	BIF
Cambodian Riel	KHR
Canada Dollars	CAD
Cape Verdean Escudo	CVE
Central African CFA Franc	XAF
Chilean Peso	CLP
China Yuan Renminbi	CNY
Colombian Peso	COP
Comorian Franc	KMF
Congolese franc	CDF
Costa Rican colón	CRC
Croatian Kuna	HRK
Cuban peso	CUP
Czech Koruna	CZK
Danish Krone	DKK
Djiboutian Franc	DJF
Dominican Peso	DOP
Egyptian Pound	EGP
Eritrean Nakfa	ERN
Ethiopian Birr	ETB
Euros	EUR
Gambian Dalasi	GMD
Georgian Lari	GEL
Ghana Cedi	GHS
Guatemalan quetzal	GTQ
Guinean franc	GNF
Haitian Gourde	HTG
Honduran Lempira	HNL

Hong Kong Dollars	HKD
Hungarian Forint	HUF
Icelandic Króna	ISK
Indian rupee	INR
Indonesia Rupiahs (1 = Rp1)	IDR
Iranian Rial	IRR
Iranian Toman (1 = IRR10)	IRT
Iraqi Dinar	IQD
Israeli New Sheqel	ILS
Japan Yen	JPY
Jordanian Dinar	JOD
Kazakhstani Tenge	KZT
Kenyan Shilling	KES
Korea (South) Won	KRW
Kuwaiti dinar	KWD
Kyrgystani Som	KGS
Laotian Kip (1=K1)	LAK
Lesotho Loti	LSL
Liberian Dollar	LRD
Libyan Dinar	LYD
Macau Patacas	MOP
Macedonian Denar	MKD
Malagasy Ariary	MGA
Malawian Kwacha	MWK
Malaysian Ringgit	MYR
Mauritanian Ouguiya	MRU
Mauritian Rupee	MUR
Mexican Pesos	MXN
Moldovan Leu	MDL
Mongolian Tugrik	MNT
Moroccan Dirham	MAD
Mozambican Metical	MZN
Namibian Dollar	NAD
Nepalese Rupee	NPR
New Zealand Dollars	NZD
Nicaraguan Córdoba	NIO
Nigerian naira	NGN
Norwegian krone	NOK
Omani Rial	OMR
Pakistani Rupee	PKR
Panamanian balboa	PAB
Paraguayan Guarani	PYG
Peruvian Nuevo Sol	PEN
Philippines Pesos	PHP
Polish złoty	PLN
Qatari Rial	QAR
Renminbi	RMB
Romanian Leu	RON
Russian Ruble	RUB
Rwandan Franc	RWF
Salvadoran Colón	SVC
Saudi Arabia Riyal	SAR
Serbian Dinar	RSD

Seychelles Rupee	SCR
Sierra Leonean Leone	SLL
Somali Shilling	SOS
South African Rands	ZAR
South Sudanese Pound	SSP
Special - HKP (1/10 HKD)	HKP
Sri Lankan Rupee	LKR
Sudan Pound	SDG
Swaziland Lilangeni	SZL
Swedish krona	SEK
Swiss Franc	CHF
Tajikistani Somoni	TJS
Tanzanian Shilling	TZS
Thailand Baht	THB
Trinidad & Tobago Dollar	TTD
Tunisian Dinar	TND
Turkish Lira	TRY
Turkmenistani manat	TMT
UAE Dirham	AED
Ugandan Shilling	UGX
Ukrainian Hryvnia	UAH
Uruguayan Peso	UYU
US Dollars	USD
Uzbekistani Som	UZS
Venezuelan Bolívar	VEF
Venezuelan Bolívar Soberano	VES
Vietnam Dong (1 = ₫1)	VND
West African CFA Franc	XOF
Yemeni rial	YER
Zambian Kwacha	ZMW

Crypto / Non-Standard	Code
Basic Attention Token	BAT
Binance Coin	BNB
Bitcoin Cash	BCH
BitShares	BTS
BitTorrent	BTT
ByteCoin	BCN
ChainLink	LINK
CoinPoker Token	CHP
Digibyte	DGB
Dogecoin	DOGE
EOS	EOS
Ethereum	ETH
GameCredits	GAME
Litecoin	LTC
milli-Binance Coin (1 = 0.001 BNB)	MBNB
micro-Bitcoin (1 = 0.000001BTC)	UBTC
micro-Bitcoin Cash (1 = 0.000001BCH)	uBCH
milli-Bitcoin (1 = 0.001BTC)	MBTC

milli-Bitcoin Cash (1 = 0.001BCH)	mBCH
milli-BitCoin Gold	MBTG
milli-Ethereum (1 = 0.001 ETH)	METH
milli-Ethereum Classic	METC
milli-Litecoin (1 = 0.001LTC)	MLTC
milli-Monero	MXMR
milli-Zcash	MZEC
NEM	XEM
OmiseGo	OMG
Paxos Standard	PAX
QTUM	QTUM
Ripple	XRP
Tether	USDT
TON Token	TON
Tron	TRX
True USD	TUSD
Usd Coin	USDC
Verge	XVG
XPC Coins	XPC
Gold Coins	GCC
LP Coins	SOC
Moon	MOON
Social USD	SUSD
Special - Gold Bean (1/1000CNY)	S01
Sweepstakes Coin	SSC
Burmese Kyat (1 = MMK 1000)	MMK2
Cambodian Riel (1=1000)	KHR2
Indonesia Rupiahs (1 = Rp1000)	IDR2
Laotian Kip (1=₭1000)	LAK2
Vietnam Dong (1 = ₫1000)	VND2

Addendum C – ChannelTypeld

In some parts of the API we provide a Channel / ChannelTypeld identifying how the player has connected.

Name	Value	UserAgent notes
1	Desktop Browser	
4	Mobile Browser	
5	App – Android	Append to useragent: ";AppName=Android-YOURAPPNAME"
6	App – iOS	Append to useragent: ";AppName=iOS-YOURAPPNAME"

For Native apps **ensure you keep the existing UserAgent** and only **append** the additional information.

If the Channel is 4,5 or 6 (Mobile) we will provide you with the Device and Browser in the fundtransfer.gamedetails (Seamless wallet API only).

If the type is 5 or 6, the browser field will contain the AppName you specified.

For Desktop channels the device and browser will be empty.

Addendum D – GameTypeld

GameTypeld (int16)	Name	
2	Baccarat	
4	Blackjack	
5	Roulette	
6	Video Poker	
7	Gamble	Do not display in menu
8	Casino Poker	
11	Video Slots	
15	Sic Bo	
16	War	
17	Dragon Tiger	
18	Other Table Games	
20	Shooting Games	

IMPORTANT NOTE: GameTypeld 7 / Gamble cannot be launched from outside a game. It is provided in the GetGames() list so that you may associate data to the game. Do not show the Game as an option for a player to Launch. We also do not provide the game logos gamble games.

Addendum E – CouponTypeId

CouponTypeId	Name
1	Variable No Deposit Bonus (Admin Only)
2	Deposit + Percentage
3	NO Deposit: Free Money
5	NO Deposit: Free Spins
6	Deposit + Free Spins

IMPORTANT NOTE: only type 3,5 is currently supported in this integration and the couponTypeId changes to 3 once the free spins are completed and the player is in wagering phase.

Addendum F – JackpotTypeId

JackpotTypeId	Name
1	Random Mini
2	Random Minor
3	Random Major
4	Random Grand
8	Race
9	Random Multi Brand Progressive (Currently Deprecated)
15	Coin Size Jackpot

Addendum G – AccountTransactionType

For fundtransferrequest, we provide an AccountTransactionType field, which identifies the transaction type of a debit/credit.

AccountTransactionTypeId	Description
1	GAME EXPIRED CREDIT
301	BLACKJACK DEAL
302	BLACKJACK DOUBLE
303	BLACKJACK SPLIT
304	BLACKJACK INSURANCE
401	BLACKJACK PAYOUT
420	ROULETTE GAME BET
421	ROULETTE GAME RESULT
425	BACCARAT GAME BET
426	BACCARAT GAME RESULT
427	SICBO GAME BET
428	SICBO GAME BET RESULT
440	VIDEO POKER BET
441	VIDEO POKER RESULT
450	EVENT GAME BET
451	EVENT GAME RESULT
501	SLOT GAME SPIN REQUEST
601	SLOT GAME SPIN RESULT
701	GAMBLE STAKE
702	GAMBLE PAYOUT
710	CLASSIC SLOT BET
711	CLASSIC SLOT PAYOUT
712	VIDEO SLOT BET
713	VIDEO SLOT PAYOUT
715	DEBIT REFUNDED
800	JACKPOT WIN
810	EXTERNAL PRODUCT DEBIT
811	EXTERNAL PRODUCT CREDIT
900	CASINO HOLDEM DEAL
901	CASINO HOLDEM CALL
902	CASINO HOLDEM WIN
903	CASINO HOLDEM PUSH
904	CARIBBEAN STUD POKER DEAL
905	CARIBBEAN STUD POKER CALL
906	CARIBBEAN STUD POKER WIN
907	CARIBBEAN STUD POKER PUSH
908	SCRATCH CARD BET
909	SCRATCH CARD RESULT
961	THREE CARD POKER ANTE
962	THREE CARD PAIR PLUS BET
963	THREE CARD PAIR PLUS PAYOUT
964	THREE CARD RAISE BET

965	THREE CARD ANTE PAYOUT
966	WAR GAME BET
967	WAR TIE BET
968	WAR TIE BET PAYOUT
969	WAR GO TO WAR BET
970	WAR GAME BET PAYOUT
971	DRAGON TIGER GAME BET
972	DRAGON TIGER GAME BET PAYOUT
973	RAISE IT UP ANTE
974	RAISE IT UP BET 1
975	RAISE IT UP BET 2
976	RAISE IT UP PAYOUT
980	WAR COMBINED STAKE
981	WAR COMBINED PAYOUT
982	NIU NIU BET
983	NIU NIU PAYOUT
984	ANDAR BAHAR BET
985	ANDAR BAHAR PAYOUT

Addendum H – CouponStatusId

For fundtransferrequest, we provide an AccountTransactionType field, which identifies the transaction type of a debit/credit.

CouponStatusId	Description
1	Redeemed
4	Expired
7	Completed - Wager Met
8	Completed - Out of Funds
10	Deleted by Player
11	Deleted by Admin

Addendum I – ProductExternalId

A guid representing the external product (non Habanero games)

Habanero	the field will be null/empty
GMW	0A8E78FB-9BF6-4562-A6D6-788452910B2D
KA Gaming	8C71545B-2822-4EAE-AA6A-B96BB8C95477

Addendum J – TournamentEventType

Tournament Event Type ID passed on requests for Alternative Fund Credits and reports

1	Prize Drops
2	Tournament - Total Wagering
3	Tournament - Total Rounds
4	Tournament - Total Win/Bet Ratio
5	Tournament - Single Highest Win/Bet Ratio
6	Tournament - Single Highest Payout

Addendum K – TournamentPrizeType

Prize type ID sent on requests for Alternative Fund Credits and reports

1	Money Prize
2	Free Spins
3	Bet Multiplier
4	Partner Prize

Addendum L – BuyFeatureId

Buy Feature ID mapping used for some reports such as GetBrandCompletedGameResultsV2, GetBrandIncompleteGames, etc

It is advised that you simply record true/false on whether a buy feature was used rather than which specific buy feature is used to avoid having to maintain this list on your end.

BuyFeatureId	Feature Name	Keyname
1	CGR - Nudge Feature	SGChristmasGiftRush
2	ORBS - 3 to 8 Scatters and 10-500 FG	SGOrbsOfAtlantis
3	ORBS - 4 Scatters and 25 FG	SGOrbsOfAtlantis
4	ORBS - 5 Scatters and 50 FG	SGOrbsOfAtlantis
5	RTTF - 3 Scatters and 10 FG	SGReturnToTheFeature
6	RTTF - 4/5 Scatters and 20/100 FG	SGReturnToTheFeature
7	RTTF - Money Re-Spin Feature	SGReturnToTheFeature
8	GUD - 3-5 Palace and 10/25 or 100 FG	SGGoldenUnicornDeluxe
9	GUD - Chest Feature	SGGoldenUnicornDeluxe
10	BR - Buy Bombs	SGBombRunner
11	LB - 3 Scatters and 8 FG	SGLaughingBuddha
12	TTT - 3 to 5 Scatters and 1-3 Wilds	SGTukTukThailand
13	TTT - 4 Scatters and 2 Wilds on re-trigger	SGTukTukThailand
14	TTT - 5 Scatters and 3 Wilds on re-trigger	SGTukTukThailand
15	RM - 3 Scatters and 13 FG	SGRainbowmania
16	RM - 3 - 5 Scatters and 13/33/133 FG	SGRainbowmania
17	RM - 4/5 Scatters and 33/133 FG	SGRainbowmania