1X2 NETWORK

Est.2002

1X2NETWORK
PANORAMA HUB
API - Integration Manual

Version: 2.0
Created By: Jack Rogers
Last Updated By: Jacob Hewitt
Last Updated: 29th March 2019

Change Log

Date	Version	Changed by	Change Description
2018-03-13	1.0	Jack Rogers	1st Draft
2018-03-14	1.1	Jacob Hewitt	Section 4 Launching games.
2018-04-03	1.2	Jacob Hewitt	Updated example JSON Strings.
2018-04-10	1.3	Jacob Hewitt	Added section 4.7 regarding the dynamic game list.
2018-04-16	1.4	Jacob Hewitt	Require that JSON parameter "bet_id" must be numeric.
2018-06-08	1.5	Jacob Hewitt	Added additional error codes for the Reconciliation Mechanism. Added **GetBalance** API Method information.
2018-07-24	1.6	Jacob Hewitt	Fixed parameter name 'stake' to 'amount' in /CancelBet request.
2019-03-29	2.0	Jacob Hewitt	Updated Communication Workflow Diagram. Removed 'QueryBet' API method (depreciated). Added section 2.8:Free-Round support & 2.9 CMA Messages. Re-ordering of some pages and re-wording of some parts.

Contents

1. Overview

1X2 Network's HUB API integration manual provides instructions for operators to integrate this gaming platform onto their existing software host system.

The Hub platform offeres 300+ games from 8 different game providers all in a single integration.

1.1 Contact details

When setting up an integration, you will be assigned a dedicated liaison from the 1X2 team that will be on hand to answer any questions you may have. You should be given these contacts by your sales representative, however please see below for additional contacts.

Telephone:	+44 (0)330 330 9021
Integration support:	support@1X2gaming.com
Commercial:	clients@1X2gaming.com

1.2 Stages of the integration

Below are the stages of the integration:

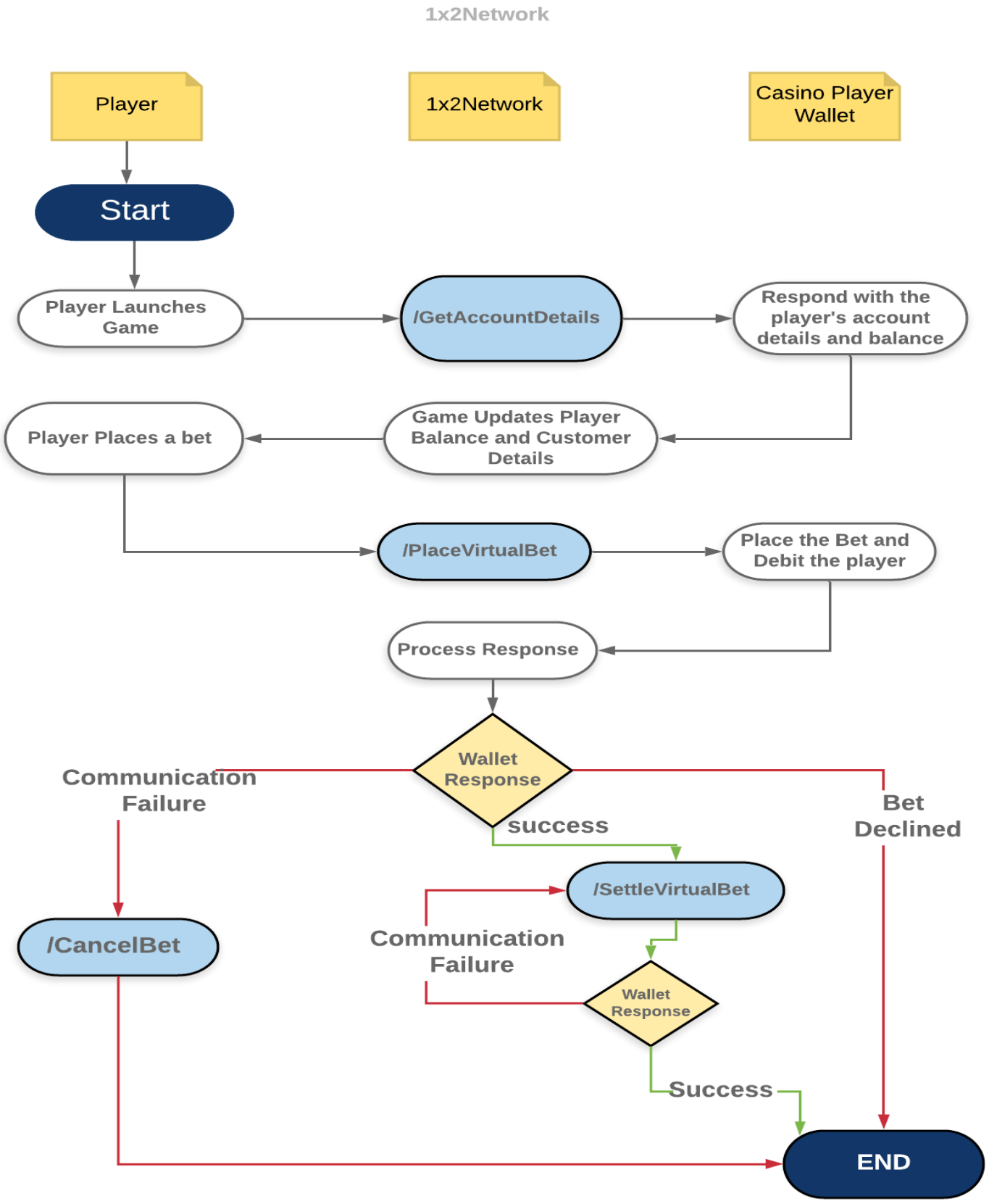
1. Getting play for fun to run on the test server.
2. Getting play for real to run on the test server.
3. Testing your API methods against our automated tests.
4. Testing the games on your staging/dev site on an exact mirror of the live hardware and software architecture.
5. Manual bet reconciliation to compare both side's figures after testing.
6. Live deployment

2. Integration/Communication

Based on our experience of previous integrations there are 4 core points at which we need to communicate with the host system. These are:

1) GetAccountDetails
2) PlaceVirtualBet
3) SettleVirtualBet
4) CancelBet

2.0.1 Communication Workflow



The system works by using web services that are JSON Post data based. The majority of the communication is achieved by 1X2 NETWORK JSON Post data to the host system, which then responds in a JSON format which will be parsed and passed back to the game.

We're able to support multiple wallet endpoint URLs and also multiple sites.

Each site that you configure with us will need to be added by ourselves in order to send you the "siteID" which will be used in communication.

For each wallet Endpoint URL you will need to send us the full URL path for each of the below requests:

- GetAccountInfo
- PlaceVirtualBet
- SettleVirtualBet
- CancelBet
- GetBalance

2.0.2 Authentication

All communications are authenticated using a parameter name "key" which is generated by creating an MD5 hash using a salt and the session ID that are sent as a parameter.

During development the salt is set as 'secretKey'.

The 'key' parameter is calculated and sent as below:

salt = secretKey
sess_id = e8978a4658
key = MD5(secretKeye8978a4658)
key = 119cd830b30f1b3aab27525ac9d10eb9

2.0.3 Idempotency

It is very important that the player wallet that integrates with 1X2 NETWORK is idempotent. In other words, we expect that the player wallet cannot settle or place a bet more than once. It must be not be possible to credit/debit the same bet more than once. This is important for the **PlaceVirtualBet**, **SettleVirtualBet** and **CancelBet** methods to ensure that players are not debited, credited or refunded bets more than once.

2.0.4 Balance and Transaction amounts

All values representing account balances and transaction amounts must be sent as a String with 2 decimal places for example:

```
"balance":"150.00"
"amount":"0.25"
```

2.0.7 Reality Check Messages

These parameters are used in the game launch link to configure the reality check messages shown to the player.

Parameter	Description
realitycheck_uk_elapsed	This parameter takes the time the player has already been playing since the limit was set (in seconds).
realitycheck_uk_limit	This parameter takes the time limit the player has set (in seconds).
realitycheck_uk_proceed	The URL to hit if the player clicks continue to reset the time limit.
realitycheck_uk_exit	The URL to hit when the player exits the game after reality message.
realitycheck_uk_history	The URL to hit for player bet history (if none provided we will use in game history).

2.1 GetAccountDetails

This will be used to load the player's account information from their session token. It assumes that player is already logged into the casino. That is if they are not logged in they cannot click on the play for real version of the game. The user may still be able to play for fun in which case there is no need to communicate with the host systems customer accounts records.

Example request:

http://playerwalleturl/GetAccountDetails

JSON Post Data:

```
{
"sess_id":"ce539034a5",
"site_id":"12",
"key":"a8cd2a99978196a1055ed2a5f26fe100",
"game_id":"2006"
}
```

JSON Object Values	Description
sess_id	This will hold the player's session token
site_id	Unique identifier of the site/skin
game_id	Unique identifier of the game
key	Security key used to validate the request. key = MD5(salt+sess_id)

JSON Response:

```
{
"error_code":0,
"error_desc":"success",
"account_id":"1840400",
"iso_code" : "EUR",
"balance":"149.11"
}
```


JSON Object Values	Description
error_code	Used to denote whether the request has been successful (error_code=0) or whether the request has failed.
error_desc	Used to describe the outcome of the request.
account_id	This is the unique customer identifier.
iso_code	Currency the player uses.
balance	Current real money balance for the player.
aams_session_id	Required for .IT jurisdictions.
aams_ticket_id	Required for .IT jurisdictions.
messages	Optional JSON Array, please see section: **2.0.5 CMA Messages** for details.

The most important parameter is the account_id which is the persistent unique identifier for the player which will be used to identify this player on your system when placing and settling bets.

2.1.1 GetBalance

This request can be used to refresh the displayed player's balance in our games from your player wallet.

http://playerwalleturl/GetBalance

JSON Post Data:

```
{
"sess_id": "a5952f97ad",
"account_id": "1234",
"site_id":"1",
"game_id":"2006",
"key":"878e7be5fa45f52100ae8d0aa482239e"
}
```

Example response:

```
{
"balance":"15.50",
"error_code":0,
"error_desc":"success"
}
```

2.1.2 Refresh Balance(HTML5 Desktop)

If you need some functionality to refresh the in game balance then this can be achieved by using the Post Message API https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage if the game is launched from within an iframe.

Please use the syntax below where otherWindow denotes the window which the iframe resides. The targetOrigin is the domain of the window where the iframe is located.

```
Window.postMessage("refresh1x2GameBalance",targetOrigin)
```

2.2 PlaceVirtualBet

This is for bet placement communication where we send you the information about this bet so that you can perform any necessary checks before the bet is accepted. E.g. the player has enough funds to cover the requested bet. Please note that it is important that your system accepts f1x2IDs that are of the 12 digits in size for example 200094934341

It is also required that you can accept bets with '0.00' amount. This is due to free-spins on our third-party games.

The minimum checks the target system must make before executing the place bet transaction are as follows:

1. Check that the player is logged in and their session has not expired.
2. Check the player has sufficient credit to place the bet.
3. Check there is no other bet from any player with the same f1x2ID.

URL:

http://playerwalleturl/PlaceVirtualBet

JSON Post Data:

```
{
"amount":"1.00",
"account_id":"4353138",
"sess_id":"a5952f97ad",
"f1x2ID":"9706228",
"site_id":"1",
"description":"xxxx",
"key":"878e7be5fa45f52100ae8d0aa482239e",
"game_id":"2005",
"freeplay":"false",
}
```

JSON Object Values	Description
sess_id	This will hold the players session token
account_id	This is the unique customer identifier
f1x2ID	The unique 1X2 NETWORK betID
amount	Stake(debit) amount. (possible to be '0.00')

game_id	Unique identifier of the game the bet was placed on
site_id	Unique identifier of the skin/site
description	Bet Description
key	Security key used to validate the request. key = MD5(salt+sess_id)
freeplay	please see section: **2.0.5 Free-Round support** for details.

In the response we will expect an error code, an error description and a numeric betID which will be the host systems unique identifier for this bet.

```
JSON Response:

{
"error_code":0,
"error_desc":"success",
"bet_id":"983468100",
"balance":"9.7"
}
```

JSON Object Values	Description
error_code	Used to denote whether the request has been successful (error_code=0) or whether the request has failed.
error_desc	Used to describe the outcome of the request
bet_id	The unique bet identifier from the player wallet system. (This must be numeric.)
balance	The balance after the transaction has been applied
messages	Optional JSON Array, please see section: **2.0.6 CMA Messages** for details.

2.3 SettleVirtualBet

This is for bet settlement communication where we send the host system the winnings due on an individual bet so that they can be credited to the player's account. This method is used by all the "play on demand" games to settle bets. This request is required to be accepted outside of player's sessions or when they are logged out.

The target system must as a minimum, check the corresponding bet has been placed and that this bet has not been previously settled. If the bet has been previously settled, then the target system must respond with error code: 302.

URL:

http://playerwalleturl/SettleVirtualBet

JSON Post Data:

```
{
"amount":"2.00"
"account_id":"4353138",
"sess_id":"a5952f97ad",
```

```
"betID":"983468100",
"f1x2ID":"9706228",
"site_id":"1",
"game_id":"2005",
"key":"f9a118563acca0a8608790f5a7a8b0e6",
"freeplay":"false",
}
```


JSON Object Values	Description
sess_id	This will hold the players session token
account_id	This is the unique customer identifier
betID	The unique bet identifier from the player wallet system. (This must be numeric.)
amount	Return(credit) amount
game_id	Unique game identifier
site_id	Unique identifier of the skin/site
f1x2ID	Unique bet identifier from 1X2 NETWORK
key	Security key used to validate the request. key = MD5(salt+sess_id)
freeplay	please see section: **2.0.5 Free-Round support** for details.

In the response we will expect an error code, an error description and a numeric betID which will be the host systems unique identifier for this bet.

JSON Response:


```
{
"error_code":0,
"error_desc":"success",
"bet_id":"43036",
"balance":"10026.67"
}
```


JSON Object Values	Description
error_code	Used to denote whether the request has been successful (error_code=0) or whether the request has failed.
error_desc	Used to describe the outcome of the request
bet_id	The unique bet identifier from the player wallet system
balance	The balance after the transaction has been applied
messages	Optional JSON Array, please see section: **2.0.5 CMA Messages** for details.

2.4 CancelBet

This method allows for the refunding of player stakes which is needed in the event of communication failures on '/PlaceVirtualBet'.

If the bet being cancelled was not debited from the players account during place bet then the error code: 204 needs to be returned so the HUB API can mark this bet as failed.

URL:

http://playerwalleturl/CancelBet

JSON Post Data:

```
{
"amount":"0.15"
"account_id":"4561095",
"sess_id":"a5952f97ad",
"f1x2ID":"1283464",
"game_id":"2006",
"site_id":"1",
"key":"878e7be5fa45f52100ae8d0aa482239e"
}
```

JSON Object Values	Description
sess_id	Session token
account_id	Unique player identifier
f1x2ID	Unique bet identifier from 1X2 NETWORK system
amount	Stake amount to refund
game_id	Unique game identifier
site_id	Brand identifier
key	Security key used to validate the request. key = MD5(salt+sess_id)

JSON Response:

```
{
"error_code":0,
"error_desc":"success",
"f1x2ID:"1283464",
"bet_id":"9726139",
"balance":"15.50"
}
```

2.5 CloseSession

This section refers to the closing of AAMS sessions on game exit for .it jurisdictions. This is called when the player finishes their game session by closing the game. When this event is triggered the game will make this API call to signal that the session needs to be closed on the host system.

Example request:

http://playerwalleturl/CloseSession

JSON Post Data:

```
{
"ticketAAMS": "N42F821DDC22EABL"
}
```

JSON Object Values	Description
ticketAAMS	The players AAMS session token

Example response:

```
{
"error_code":0,
"error_desc":"success"
}
```

2.6 Error Codes

When integrating with our system. You will need to return the below error codes in the case of failure inside of the JSON Object.

```
SUCCESS=0

Loading Account Info
INVALID_PLAYER_ID=101
ACCOUNT_LOCKED=102
ERROR_RETRIEVING_DATA=103
GENERIC_ERROR=104
ACCOUNT_INACTIVE=105
ACCOUNT_EXPIRED=106

Placing bets
INSUFFICIENT_MONEY=201
DUPLICATE_BETID=202

Settling bets
BET ALREADY SETTLED=302
BET_NOT_FOUND=204

Cancelling Bets
BET_NOT_FOUND=204

General Errors
INVALID_GAME_ID=401
INVALID_SITE_ID=402
INVALID_GAME_SESSION_ID=403
INVALID_BET_ID=404
SECURITY_KEY_MISMATCH=-7
```

2.7 Failed Bet Reconciliation and Retries

This section explains how the HUB API deals with communication failures in order to prevent incomplete bets.

2.7.1 PlaceVirtualBet Communication Failure Protocol

- If we receive a communication failure during **PlaceVirtualBet** we automatically attempt to cancel the bet through the **CancelBet** method.

- If this initial request is unsuccessful we add it to our **Reconciliation Mechanism** to ensure the player is refunded their debited stake.

2.7.2 SettleVirtualBet Communication Failure Protocol

- If we receive a communication failure during the **SettleVirtualBet** we automatically retry to settle this bet until we receive a relevant error code. (0 = successful, 302 = bet already settled).
- The times that the requests are sent are dependent on the **Reconciliation mechanism** below.

2.7.3 CloseSession Communication Failure Protocol

- The **CloseSession** call is retried until a successful response is received by the player wallet API confirming the session has been closed.

2.7.4 Reconciliation Mechanism Details

The reconciliation mechanism ensures the two systems (HUB API and Player Wallet System) are kept in sync.

HUB API has an automated process (cron job) that reads a Recon table from the database and runs Recon requests within the queue.

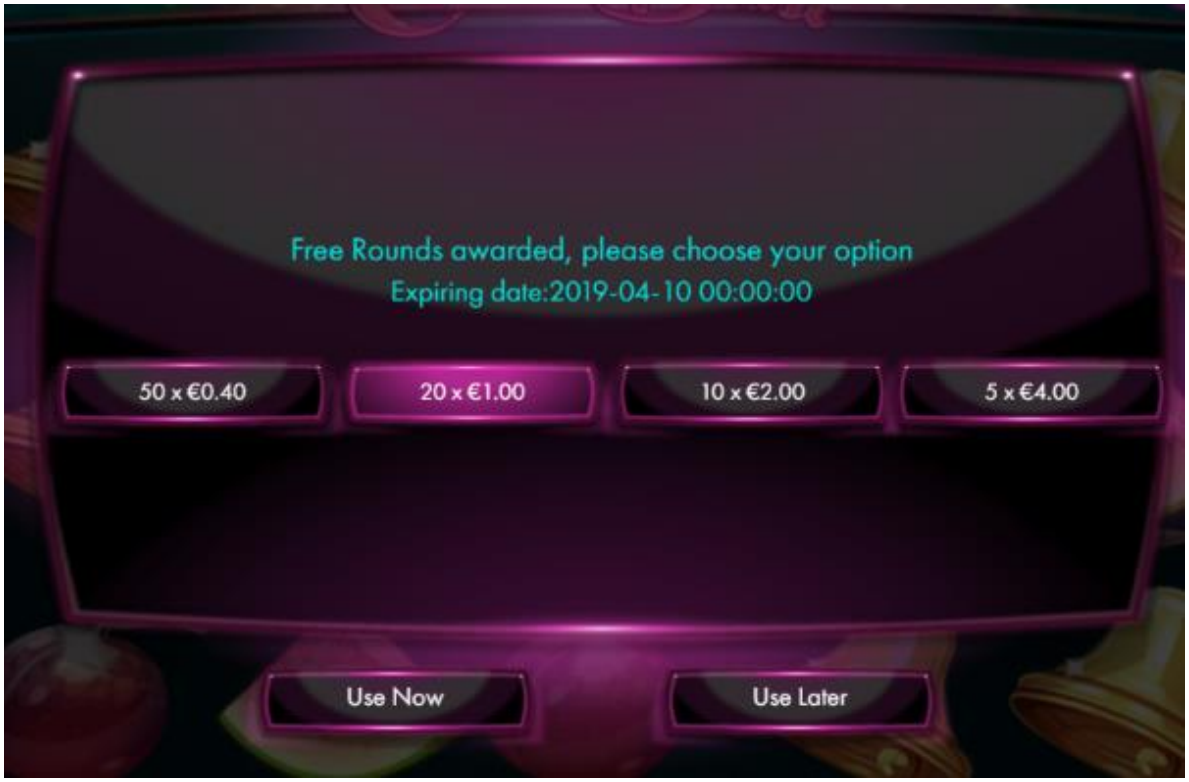
For any new entry, a Recon request is sent as follows:

- Every minute for the first 10 minutes
- Every 10 minutes for the first 1 hour
- Every hour for the first 24 hours

After 24 hours, the requests that are still unsuccessful should be marked for manual reconciliation.

2.8 Free Round Support

The Hub platform has the functionality to offer free-rounds. You're also able to give the player up to 4 options to choose from when they enter the game:



This is done by creating a 'Campaign' through our Admin Site and assigning players to it.

When creating a campaign, you will need to specify the value of the campaign in EUR. This value is how much the 'campaign' is worth in EUR for each individual player. For the above image the campaign value is 20.00 EUR.

For currencies other than EUR we will need you to explicitly add them. We will then automatically calculate the equivalent stake amount and options in accordance with the 'campaign value' mentioned above.

The Free-Rounds are different to normal bets, as the players should not be credited their winnings until the last Free-Round is completed. The reason that we still send them individually, is so they can remain in your player's transaction history.

To denote a free-round, we have the JSON parameter named "freeplay". This parameter is sent both on '/PlaceVirtualBet' and '/SettleVirtualBet'.
To denote the end of the player's free-rounds we have the JSON String parameter "status". This parameter is sent in the '/SettleVirtualBet' request. This parameter will be set as "pending" while a player's free-rounds are ongoing and "completed" when the free-rounds are over.

The player should receive the winnings for all the Free-Rounds only when the last '/SettleVirtualBet' request with the status "completed" is sent.

We have two additional parameters named “total_win_so_far” and “plays_remaining”. You may credit the player’s full winnings from the “total_win_so_far” parameter on the last Free-Round, or you may also calculate the winnings yourselves from the standard “amount” parameter sent in the previous “pending” rounds.

We also have two parameters named “promoCampaignID” and “promoActivationID”.

The “promoCampaignID” is a numeric unique identifier for the campaign that you have created. This is automatically generated on our-side when you create a tournament. The value is viewable after you have created a new tournament if you wish to use it.

The “promoActivationID” is a numeric unique identifier for each individual player’s set of free-rounds. It is also generated internally on our-side when you add a player to a specific tournament. This is also viewable/retrievable from our admin site if you wish to use it.

Example /PlaceVirtualBet JSON:


```
{
"amount":"1.00",
"account_id":"4353138",
"sess_id":"a5952f97ad",
"f1x2ID":"9706228",
"site_id":"1",
"description":"xxxx",
"key":"878e7be5fa45f52100ae8d0aa482239e",
"game_id":"2005",
"freeplay":"true",
"promoCampaignID":"1",
"promoActivationID":"1"
}
```


Example /SettleVirtualBet JSON:


```
{
"amount":"2.00"
"account_id":"4353138",
"sess_id":"a5952f97ad",
"betID":"983468100",
"f1x2ID":"9706228",
"site_id":"1",
"game_id":"2005",
"key":"f9a118563acca0a8608790f5a7a8b0e6",
"freeplay":"false",
"promoCampaignID":"1",
"promoActivationID":"1",
"total_win_so_far":"2.00",
"plays_remaining":9,
"status":"pending"
}
```

2.9 CMA Messages

We have the optional functionality for you to display custom messages to the players inside our games. The messages in-game are composed of a **Code**, **Title**, **Message** and a **Button**.

The **Code** parameter should be left empty: "", unless you have specific functionality required. This will need to be discussed during the development stage of the integration.

The **Title** parameter takes plain text and will be the title of the message popup.

The **Message** parameter takes plain text and will be displayed inside the popup. You can use the characters \n to create a new line.

The **Button** can have different functionality depending on what action you specify:

Button Action	Button Action Behaviour
continue	Close the message box and resume game play. Send an asynchronous GET to the optional URL indicated in the "url" attribute, if no url is specified no is call is made.
exit	Send an asynchronous GET to the URL indicated in the "url" attribute, if no url is specified no call is made. Then redirect to the "lobbyUrl" passed into the game on game-launch.
history	Redirect browser window to the "url" attribute specified. If the "url" attribute is null then we will attempt to use the optional "realitycheck_uk_history" parameter passed in on game-launch.
choice	Close the message box and resume game play. Also send an asynchronous GET to the URL indicated in the "url" attribute to convey the decision to the wallet.
close	Close the message box and redirect to the URL indicated in the "url" attribute or if that is null the "lobbyUrl" parameter passed into the game on game-launch.

The messages are specified and triggered through your response to the below API calls:

GetAccountDetails: If you return a message in your response to this method call the message will be displayed as soon as the player enters the game.
PlaceVirtualBet: The message will be displayed after the bet completes.
SettleVirtualBet: The message will be displayed after the bet completes.

It is the responsibility of the operator to provide all the text in the language in which they want the message displayed to the player.
It Is also the responsibility of the operator to format all text appropriately.

JSON Message Format

Please see the page below for an example of the JSON response.

This JSON Array can be included in your JSON response for any of the 3 API calls specified above.

Example *messages* JSON


```
"messages": [
 {
  "message": {
   "code": "UK_RC",
   "title": "Title for the popup message",
   "text": "message. Including \r\n or just \n for enter",
   "buttons": [
    {
     "text": "Quit Game",
     "action": "exit",
     "url": "some_url"
    },
    {
     "text": "Continue Game",
     "action": "continue",
     "url": "some_url2"
    },
    {
     "text": "Account History",
     "action": "history",
     "url": "some_url3"
    }
   ]
  }
 },
 {
  "message": {
   "code": "",
   "title": "Title for the popup message",
   "text": "message. Including \r\n or just \n for enter",
   "buttons": [
    {
     "text": "Accept",
     "action": "choice",
     "url": "some_url"
    },
    {
     "text": "Decline",
     "action": "choice",
     "url": "some_url2"
    }
   ]
  }
 }
```

```
 ]
}
```

3. Internationalization

3.1 Languages

1X2 NETWORK provides a broad range of translations for the games, spanning around 24 dialects. This can all be seen Appendix 2 with the games list. If you require other languages please contact your designated 1X2 representative to discuss how this can be achieved.

3.2 Base currency & Exchange rates

The HUB platform retrieves and updates its exchange rates every ½ hour from https://currencylayer.com/ . EUR is our base currency. We strive to accept all currencies possible. However, some of our older games have limitations and don't allow us to accept currencies with very high or low exchange-rates. Please let us know what currencies you need supporting before the integration process begins so the necessary configurations can be made.

4. Launching games

Please refer to Appendix 2 (an excel sheet), for the list of games available for the integration.

To launch games you will need to call our "/loadGame.jsp" page. We have a single parameter called "platform" to differentiate between desktop games, mobile games and mini games.

4.1 Desktop Games

```
https://domain/f1x2games/loadGame.jsp?sessionID=1234&gameID=1001&playMode=real&lang=en&siteID=1&platform=desktop
```

Parameter	Description
sessionID	This parameter requires the player's session token to be passed
gameID	This parameter lets the game client server know which game to launch.
	This parameter determines whether the player is playing for real money or fun.
lang	The language the game is to be launched in(two letter country code)
siteID	This is a unique identifier for each of your sites. This will be assigned to each site you support from our-side.
platform	This parameter's value should be "desktop" when loading desktop game versions.
lobbyUrl	This parameter is optional on desktop. It is bound to the back button in-game.

4.2 Mobile games

```
https://domain/f1x2games/loadGame.jsp?sessionID=1234&gameID=1001&playMode=real&lang=en&siteID=1&platform=mobile&lobbyUrl=http://www.yourdomain.com/lobby
```

Parameter	Description
sessionID	This parameter requires the player's session token to be passed
gameID	This parameter lets the game client server know which game to launch.
playMode	This parameter determines whether the player is playing for real money or fun.
lang	The language the game is to be launched in(two letter country code)
siteID	This is a unique identifier for each of your sites. This will be assigned to each site you support from our-side.
platform	This parameter's value should be "mobile" when loading mobile game versions.
lobbyUrl	This parameter takes a URL to bind to the back button in-game.

4.2.1 Mobile Operating System and Browser Support

1X2 NETWORK's HTML5 mobile ready suite currently supports Android, iOS, Blackberry and Windows. For the latest versions and browsers please reference our document "Mobile Operating System and Browser Support".

4.2.2 Branding and features

Branding

- Some mobile branding is more flexible and other options are available to suit each client's specific needs. Please enquire about this at the time of integration.

Features

- We encourage the user to add all games to the homescreen when on ios devices resulting in a native app like experience.
- Where possible we encourage the user to enter fullscreen using the fullscreen api.
- We keep account of the device each user accesses the games on using their unique account id. The device, operating system and browser are stored in a sql table alongside the unique account id.
- We recommend that users do not launch mobile games in an iframe as this hinders our tracking to make the most of the devices size, nuances and features.

4.3 Mini Games

Our minigames have a very compact front-end which makes them perfect for sidebars.
Please see the information below on how to launch our minigames.

4.3.1 Launching a mini game

```
https://domain/f1x2games/loadGame.jsp?sessionID=1234&gameID=1048&playMode=real&lang=en&siteID=1&platform=minigame
```

Parameter	Description
sessionID	This parameter requires the player's session token to be passed
gameID	This parameter lets the game client server know which game to launch.
playMode	This parameter determines whether the player is playing for real money or fun.
lang	The language the game is to be launched in(two letter country code)
siteID	This is a unique identifier for each of your sites. This will be assigned to each site you support from our-side.
platform	This parameter's value should be "minigame" when loading minigame game versions.

4.3.2 Dimensions

- The mini games are designed to fit responsively into most small containers however you will find that the optimum size is 270px x 350px.
- For virtual sports there is now room to reduce the height of the container.

4.4 Graphical Virtual Sports

4.4.1 Launching in an Iframe

Simply create an iframe using a minimum width of 1024px (no max).

src = As described in section 1.

Use the id "`lobbyFrameContainer`".

```
<iframe id="lobbyFrameContainer" scrolling="no" src=""></iframe>
```

1X2 NETWORK will provide a script that you need to include on the page that houses the iframe. This script uses the postMessage api to send messages between the parent page and the page contained within the iframe. It will dynamically resize the iframe to the correct height and ensure the bet slip scrolls down the page following the window scroll position.

4.4.2 Launching in a pop up

The game can also be launched in a pop up. Here you simply follow section1 to launch the pop up.

Ensure it the pop up window is:

1. Minimum width 1024px
2. Is of height 100%
3. Allows scrolling

4.5 Jurisdiction

These parameters are used in the game launch link to specify which jurisdiction rules to apply in game. This includes functionality such as reality check messages if they are not separately specified/sent.

Parameter	Description
jurisdiction	Denotes the jurisdiction rules to apply

4.7 Dynamic game list

There is a page located at:

https://domain/f1x2games/GameList?siteID=1&platform=desktop

This page returns a JSON string returning the gameID used to launch, the game's name, the game supplier, the game's category and whether game supports Mobile or not.

The 'siteID' parameter is the same one we will assign to you and the same one that you pass in during game launch.
The 'platform' parameter will return you the available games for that specific platform.

5. System testing on a mirror version of the deployment architecture

In order to fully test and approve the new system it should be setup from scratch on an exact mirror of the architecture that will be used on the deployed version. Exactly what test facilities are made available will vary, but as close a match to the deployment architecture should be used as possible.

6. Deploying the tested configurations of the game to the live site

This should be a case of moving the tested configuration of the game from the test servers to the live. However, there are a few points to be aware of before doing so.

1. If the test configuration does not exactly match the live configuration then there may be issues to overcome when deploying to the live site.
2. Any licensing issues.

7. Manual Bet Reconciliation

Once the integration testing is complete, we will ask you to send all the bets/transactions placed on your Dev/Staging environments. With this data we will manually compare both sides figures to confirm that they are matching and correct. This is the last step to identify so they can be addressed before the go-live.