CasinoEngine Direct Wallet Integration Guide

Version 1.11

Last edited 23 January 2019

Change Control

Version	Date	Name	Role	Comment
0.0.1	07.03.2017	Tim Walls	OM CTO	Initial draft.
0.0.2	08.03.2017	Tim Walls	OM CTO	Feedback from SH/BM/MC.
1.0.0	20.03.2017	Andrij Kovalchuk, Svitlana Fitsak	CE CTO Technical Writer	Changed the `GetAccount`, `GetBalance` and `Result` methods. The `GetTransactionStatus` method has been added.
1.1.0	26.07.2017	Natalia Kostetska Svitlana Fitsak	Software Engineer Technical Writer	Added the parameter `RoundStatus` to the `Wager` method request.
1.2.0	29.08.2017	Svitlana Fitsak	Technical Writer	Redrew the interaction diagram on p.6.
1.3.0	7.11.2017	Svitlana Fitsak	Technical Writer	Detailed the information received via JSON feeds in *Game and Lobby Management*.
1.4	25.12.2017	Svitlana Fitsak	Technical Writer	Updated the example in Embedding a GameLoader Game Example.
1.5	14.02.2018	Svitlana Fitsak	Technical Writer	Added the requirement to Security.
1.6	01.03.2018	Evgen Kovalenko Svitlana Fitsak	Software Engineer Technical Writer	Added `KeepAliveURL` and `KeepAliveInterval` to Optional Parameters.
1.7	15.06.2018	Evgen Kovalenko Mariia Ishchuk	Software Engineer Technical Writer	Changed _sid64 into _sid in GameLoader URL example.
1.8	12.07.2018	Evgen Kovalenko Mariia Ishchuk	Software Engineer Technical Writer	Added `RCPeriod` variable to `GetAccount` method response.
1.9	04.10.2018	Evgen Kovalenko Iryna Halych	Software Engineer Technical Writer	Added the description of retry to `Result` and `Rollback` methods. Renamed Double transaction error to `TransactionNotFound` error, changed the error's description.
1.10	09.11.2018	Evgen Kovalenko Iryna Halych	Software Engineer Technical Writer	Substituted `ExternalUserId` and `InternalUserId` fields with `AccountId` in `GetBalance` method. Removed `ExternalUserId` and

				`InternalUserId` fields from `Wager method`.
1.11	23.01.2019	Evgen Kovalenko Mariia Matskiv	Software Engineer Technical Writer	Replaced `ExternalAccountId` with `AccountId` response variable and removed `InternalAccountId` response variable from `GetAccount method`.

Sign-off / Approval

Version	Date	Name	Role	Comment

Filename, Location, Owner, and Classification

File Name	CE Direct Wallet Integration Guide V.1.8.pdf
File Location	
Owner	CE CTO
Classification	Public

Document Distribution

Role	Name	Organization
Integration Management		Everymatrix
Sales		Everymatrix

References

Reference	Version	Section/Control/Requirement
N/A		

Table of Contents

1 Introduction

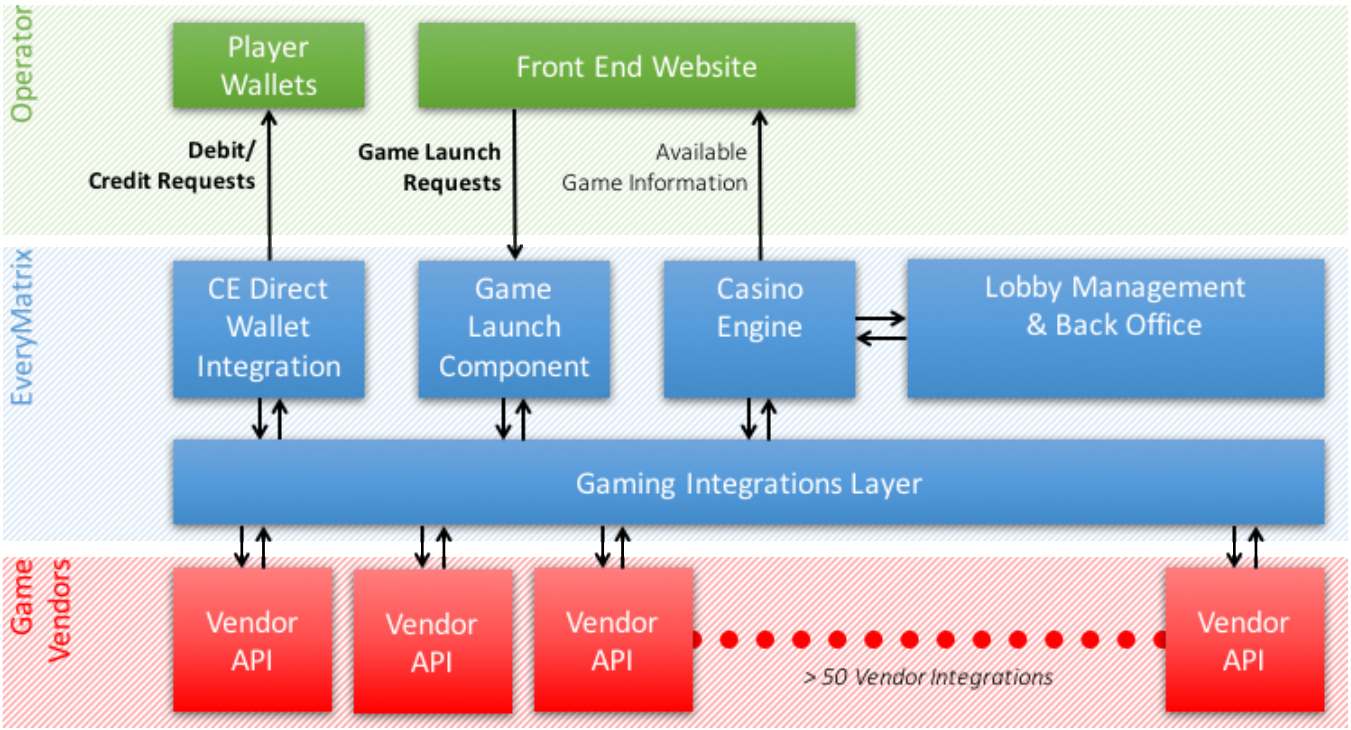
CasinoEngine is the foremost casino gaming aggregator in the industry – with a single integration, operators can gain access to a library of thousands of games provided by tens of vendors, as a complete turn-key solution when combined with EveryMatrix's own GamMatrix gaming platform.

GamMatrix is not the only option, though – with CasinoEngine Direct (CE Direct), operators can integrate CasinoEngine's casino gaming directly into their own web portal/wallet. This document describes the technical requirements for such an integration.

1.1 Overview

CasinoEngine allows an operator to, with a single integration, gain access to a library of over 4000 games from more than 50 gaming vendors, a library which is continuously growing as EveryMatrix adds more vendors and games to its portfolio. As well as removing the technical headache of developing (and maintaining) integrations with multiple vendor APIs, EveryMatrix also takes away the business overhead of negotiating separate contracts with every provider, providing a true one-stop-shop for an unbeatable online and mobile games library.

With CE Direct, an operator can take advantage of CasinoEngine while maintaining the complete freedom to develop and host their own website/portal and player wallets, as illustrated below:



This document is concerned with the two key integration points shown in bold on the diagram:

- **Game Launch** – a standardized mechanism for the operator's website to launch any vendor's game.
- **Wallet Integration** – integration of bet stake (debit) and winnings (credit) with the operator's player account management system (wallet).

1.2 CasinoEngine JSON Feeds

CasinoEngine offers a full-featured back office allowing operators to configure the vendors and games available to their players, as well as allowing customization of presentation details such as game names and imagery.

This information is exposed through a rich array of JSON feeds, which allow the operator's platform to determine the games available using static (e.g. games by vendor) or dynamic (e.g. currently popular games) criteria, and using CasinoEngine's unique Game Recommendation Engine to even obtain personalized recommendations for an individual player.

Use the following API methods to get the necessary information from the JSON feeds:

HTTP Method	Method Name	URL
GET	Get Mix Data	/jsonFeeds/mix/{operatorKey}?types=types&topWinnerChannel={topWinnerChannel}&recentWinnerChannel={recentWinnerChannel}, where an `operatorKey` is provided per operator.
GET	Get Game Info	/jsonFeeds/casinoGameInfo/{OperatorKey}?gameID={gameID}&language={language}

The following table details the information obtained:

Parameter	Type	Description
domainID	*Integer*	The domain identifier.
type	*String*	A type of the updated object. Possible values: **vendor**, ***contentprovider, game, table, jackpot, topwinner, recentwinner, tableseat, datasource***
action	*String*	The parameter value indicating one of the possible actions: **initialize_start** signifies that the initialize has started; **initialize_complete** means that the initialize has completed; **update** means that certain data has been updated.
data	*Object*	The parameter value containing comprehensive information about this JSON object.
id	*Integer*	The EveryMatrix game identifier.
slug	*String*	The parameter value indicating the user-friendly URL.
vendor	*String*	The vendor's name.
vendorID	*Integer*	The vendor's identifier.
contentProvider	*String*	The sub-vendor's name.
gameCode	*String*	The game code.
gameID	*String*	The game identifier.
categories	*String*	The parameter value denoting the game category. Possible values: ***3DSLOTS***, ***CLASSICSLOTS***, ***JACKPOTGAMES***, ***LIVEDEALER***, ***LOTTERY***, ***MINIGAMES***, ***OTHERGAMES***, ***SCRATCHCARDS***, ***TABLEGAMES***, ***VIDEOPOKERS, VIDEOSLOTS, VIRTUALSPORTS*** .
tags	*String*	The parameter used for marking games.
enabled	*Boolean*	The parameter value indicating if the game is enabled in the operator system. If it is set to ***true***, the game is activated. If it is set to ***false***, the game is inactive.
operatorVisible	*Boolean*	The parameter value indicating if the game is visible for players. If it is set to ***true***, the player can see the game; if it is set to ***false***, the player is unaware of the game.
theoretical Payout	*Decimal*	The parameter value denoting the maximum amount the player can win in the game.
thirdPartyFee	*Decimal*	The parameter value indicating the fees of the third party.

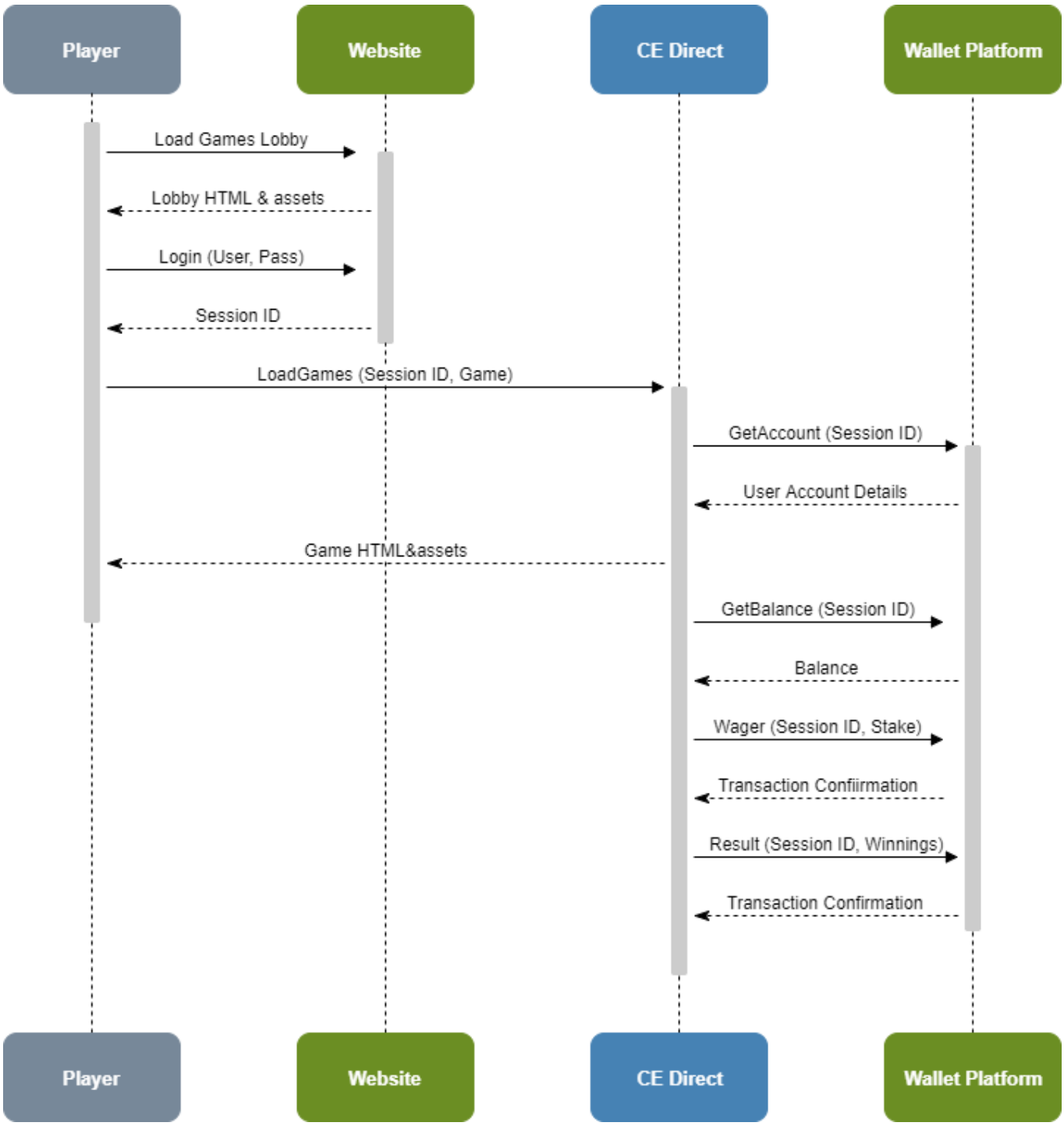
fpp	*Decimal*	A frequent player point. The parameter value indicating the points generated by wagering real money on a game.
restrictedTerritories	*String*	The parameter value indicating the countries where the game is disabled.
languages	*String*	The parameter value specifying the languages of the game.
currencies	*String*	The parameter value indicating the currency set for the game.
url	*String*	The game loader URL comprising the CE application (always **Loader**), the CE command (always **Start**), `domain ID, slug, _sid, language` and `funMode`.
creation	*Object*	The JSON object detailing when the game was created.
time	*String*	The parameter value indicating the time when the game was first created. A valid format for `time` is ***YYYY-MMDDTHH:MM:SS.FFFFFFF***.
newGameExpiryTime	*String*	The parameter value indicating when the game expires. A valid format for `newGameExpiryTime` is ***YYYY-MMDDTHH:MM:SS.FFFFFFF***.
lastModified	*String*	The parameter value indicating the time when the game was last modified.
property	*Object*	The JSON object comprising game properties.
width	*Integer*	The width of the game window.
height	*Integer*	The height of the game window.
license	*String*	The parameter value specifying under which license the vendor operates.
terminal	*Array*	The parameter value indicating the device the game can played on. Possible values: **PC**, **iPad**, **iPhone**, **Android**.
freeSpin	*Object*	The JSON object containing the information about free spins.
support	*Boolean*	The parameter value denoting if the free spins are available in the game when it is set to ***true***.
hitFrequency	*Object*	The JSON object containing range of the hit frequency.
min	*Integer*	The minimum number of hits needed for a win.
max	*Integer*	The maximum number of hits needed for a win.
launchGame InHtml5	*Boolean*	The parameter value indicating if the game in HTML 5 when it is set to ***true***.
ageLimit	*Boolean*	The parameter value indicating age restrictions. If it is set to ***true***, there is such a constraint. If it is set to ***false***, the game is available for players of all ages.
report	*Object*	The JSON object containing the information about reporting.
category	*String*	The parameter value denoting game categories. Possible values: ***3DSLOTS, CLASSICSLOTS, JACKPOTGAMES, LIVEDEALER, LOTTERY, MINIGAMES, OTHERGAMES, SCRATCHCARDS, TABLEGAMES, VIDEOPOKERS, VIDEOSLOTS, VIRTUALSPORTS***.
invoicingGroup	*String*	The parameter value indicating the pricing policy. Possible values: ***STANDARDPRICING, PREMIUMSLOTS, TRADEMARK, PLATINUM, BRANDEDGAMES, PREMIUMTABLE, OTHERS, LIVECASINO***.
popularity	*Object*	The JSON object containing the information about how popular the game is.
coefficient	*Decimal*	The parameter value indicating the popularity coefficient set in CasinoEngine.
ranking	*Decimal*	The parameter value determining the popularity of a game – the higher the index is, the more popular the game is.
playMode	*Object*	The JSON object containing the information about modes of the game.
fun	*Boolean*	The parameter value indicating whether the game is available in play for fun mode. If it is set to ***true***, it is possible; if it is set to ***false***, it is impossible.
anonymity	*Boolean*	The parameter value indicating whether the player needs to be authorized to play the game in Play for Fun mode. If it is set to ***true***, it is possible; if it is set to ***false***, it is impossible.
realMoney	*Boolean*	The parameter value indicating whether the game can be played in Real Money mode. If it is set to ***true***, it is possible; if it is set to ***false***, it is impossible.
bonus	*Object*	The JSON object containing the information about the bonus contribution.
contribution	*Decimal*	The parameter value indicating how much real money the player should bet to be awarded a bonus, provided by the vendor.
presentation	*Object*	The JSON object containing the information about how the game will be presented.
gameName	*Object*	The name of the game.
shortName	*Object*	The shortened name of the game.
description	*Object*	The description of the game.
thumbnail	*Object*	A small picture of the game.

`logo`	*Object*	The logo of the game.
`backgroundImage`	*Object*	The image shown in the background of the game.
`iconFormat`	*Object*	The format of the icons.
`icons`	*Object*	The parameter value incorporating all icons used in the game.

2 Integration Overview

2.1 Platform Interactions & Responsibilities

The interaction diagram below shows a typical gameplay flow for a player using a CE Direct integrated casino gaming site, from login to gameplay:



2.1.1 Operator Website

The operator is free to build their website using any technology they choose. The website uses the JSON feeds provided by CasinoEngine – see *Game and Lobby Management* – to generate the casino lobby and game selection pages which allow the player to choose a game to play.

When a user plays a game, the operator's website provides the user's browser with an <iFrame>, window or link which directs the user to a URL hosted by CE Direct which will load and return the relevant game assets to the user's browser. In addition, CE Direct will transparently perform any game vendor-specific initialization required. The necessary URLs are described in the *Game Loader* section of this document.

Note that the operator website is also responsible for performing any player session management (login/logout) required. When a game is loaded, the website will pass a `Session ID` to the CE Direct platform; this token is remembered by the CE Direct platform, and passed with any subsequent wallet transaction calls to allow the operator platform to validate the requests.

2.1.2 Operator Wallet Platform

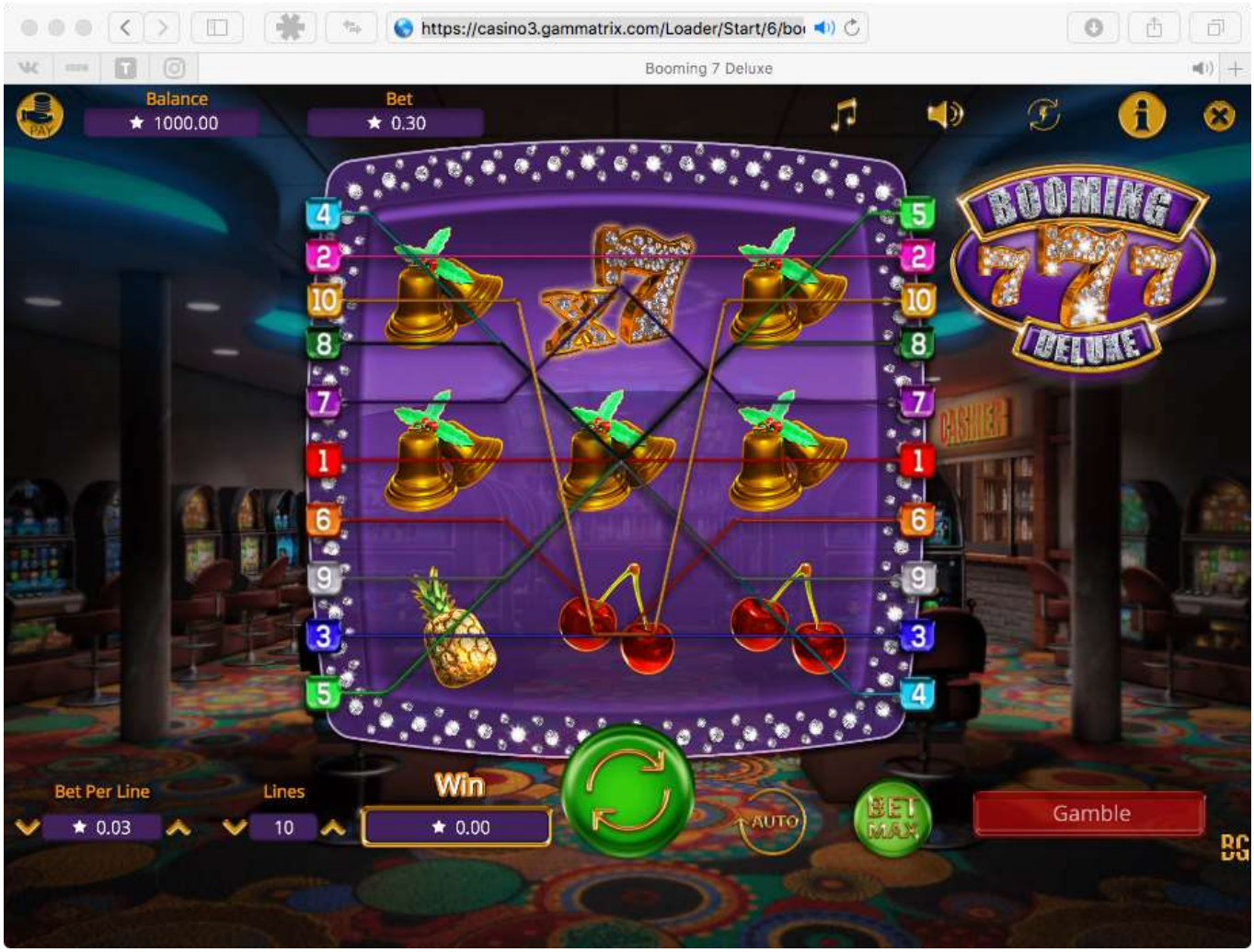
The wallet platform holds the user's balance of funds for gameplay (real money and/or bonus money.) As a user interacts with a game (e.g. by spinning the reels of an online slots game), CE Direct will debit stakes from the user's wallet by means of API calls to the operator platform – and will, of course, similarly credit winnings.

The *Wallet API* section of this document describes the API which the operator wallet must expose to CE to take place.

3 Game Loader

3.1 Overview

The GameLoader function of CE Direct provides a standardized means for an operator's website to launch any game, regardless of vendor. Typically, the website will construct the URL (based on the game information received from the CasinoEngine JSON feeds, and the player's choice of game to play) and then direct the user's browser to load from that URL via an <iFrame>, popup or similar. The image below shows an example of loading a GameLoader URL directly in a browser window:



3.2 Constructing a GameLoader URL

The link to the **Stage environment**: http://gamelaunch-stage.everymatrix.com/

The link to the **Production environment**: http://gamelaunch.everymatrix.com/

The following is an example of a typical GameLoader URL:

```
https://gamelaunch.everymatrix.com/Loader/Start/1256/zeus-
1000?language=en&funMode=False&_sid=2eZKHRZFKKXzTpT9ypvQnDEndCXt8VV5kB8cfJkUyEwLxQXz_4l
6wg--?
  ?language=en
  funMode=False
  &_sid=2eZKHRZFKKXzTpT9ypvQnDEndCXt8VV5kB8cfJkUyEwLxQXz_4l6wg--?
```

The URL comprises three key components:

- The CE Direct Endpoint
  This will be provided by your EveryMatrix Integration Manager. Different endpoints will be provided for stage/integration and production environments.
- Command, Operator & Game Identification
  Identifies the game to be loaded.
- Optional Parameters
  A number of optional parameters can be passed to adjust the gameplay experience – for example, game language. The example shown is not exhaustive – possible parameters are described below.

3.2.1 Command, Operator & Game Identification

The path of the constructed URL comprises four components, each of which is mandatory for a well-formed URL. Using the example path `/Loader/Start/6/booming-7-deluxe/`, those components are as follows:

Component	Example	Description
CE Application	`Loader`	Always **`Loader`**.
CE Command	`Start`	Always **`Start.`**
Operator ID	`1256`	Numeric Operator ID uniquely identifying the operator; this will be provided by your EveryMatrix Integration Manager.
Game ID	`zeus-1000`	Alphanumeric string uniquely identifying a game, obtained from the CasinoEngine JSON data feeds.

3.2.2 Optional Parameters

The path may optionally be followed by a number of parameters in URL *query string* format – i.e. is used to delimit the first key/value pair from the game path, and subsequent parameters are delimited by *&*. All parameters should be URL encoded if appropriate.

While all the following parameters are in strict terms optional, one or more will in practice be essential for a properly functioning gaming site.

Parameter	Type	Description
`_sid`	*String*	Session Token. Note that the format of this token is opaque to CasinoEngine – it is passed unmodified to the operator's wallet with every transaction request resulting from gameplay. **If this parameter is not passed, the game will load in Play for Fun mode.**
`language`	*String(2)*	Two-letter (ISO 639-1) language code. If available, the game interface will be loaded in this language. If the relevant language is not available, or this parameter is not passed, the default will be English.

funMode	*Boolean*	If the value of this parameter is `true`, the game will be launched in Play for Fun mode (regardless of the presence or otherwise of the `sid` session token parameter.)
casinolobbyurl	*URL*	(Appropriately encoded) URL of the operator site's casino lobby. Determines the URL that the in-game 'home' or 'lobby' button will navigate to.
cashierurl	*URL*	(Appropriately encoded) URL of the operator site's cashier page. Determines the URL that the in-game 'deposit' or 'buy chips' button will navigate to.
tableId	*Integer*	For live-dealer casino games, the game path is not sufficient to uniquely identify both the game and the specific dealer/table being loaded. For such games, it is required to also pass in the `tableId` parameter, as obtained from the CasinoEngine JSON data feeds.
KeepAliveURL	*String*	A ping signal feature sent to the operator's webpage at a specified `KeepAliveInterval`. The game makes a GET request to the page pointed out by the `KeepAliveURL` parameter. This feature is required to prevent the player's session at the operator's website from expiring during a game play. Note this feature does not contain a session cookie. Therefore, an extra parameter should be manually added to `KeepAliveURL` to make it work.
KeepAliveInternal	*Integer*	The parameter value indicating the frequency of the ping signal.

3.2.3 Embedding a GameLoader Game Example

The following HTML fragment is a complete example of how to embed a game into a site using an <iframe> and the appropriate GameLoader URL. In this example, the game is *Down the PUB* by *Playson*, loaded in Play for Fun mode.

```
<html>
    <body>
        <div class="GameHolder">
            <iframe id="iframeGame_1" width="1400" height="788"
                    src="https://gamelaunch.everymatrix.com/Loader/Start/6/down-the-
pub-
desktop/?funMode=True&language=en&casinolobbyurl=http%3a%2f%2foperator.com%2fCasino%2f&
cashierurl=http%3a%2f%2foperator.com%2fdeposit"
                    scrolling="no" frameborder="0">
            </iframe>
        </div>
    </body>
</html>
```

4 Wallet API

4.1 Overview

The Wallet API is the API exposed by the operator's wallet platform to CE Direct. This section of the integration guide should therefore be interpreted as a specification of the API which the operator wallet must provide, rather than as a guide to API calls provided by CE Direct.

The Wallet API provides CE Direct with the ability to, based on an operator's own Session Token (provided to CE Direct when a game is launched), determine the basic details of a player's account, and to debit wagers from/credit returns to the player's wallet.

4.2 Implementation Requirements

Typically, the CE Direct platform and Operator wallet platform will be geographically separated, with communication taking place over public networks or VPN. Consequently, the integration must be robust to unpredictable latency and potentially unreliable communications (lost or delayed messages) without leaving the user's wallet in an unpredictable state and with minimal effect on the gameplay experience.

For this reason, in order to guarantee good player experience, EveryMatrix requires that the operator implements the wallet API in such a way that transactions can be safely retried and if necessary rolled back regardless of platform or network failures. These, and other requirements intended to ensure player security and API resilience are detailed below. EveryMatrix cannot guarantee the performance or correctness of any integration in which the operator platform does not meet these minimum requirements.

Idempotency

All transaction methods (`Wager`, `Result` and `Rollback`) **must** be implemented as idempotent methods – i.e. if the operator wallet receives the same API call more than once with the same transaction ID, the relevant transaction will only be recorded once in the wallet platform.

This permits CE Direct to safely retry a transaction in the event of network error or lack of response from the operator wallet, without the risk of duplicate transactions being applied to the player's wallet.

The operator wallet **must not** make any assumptions about when an operation might be retried; a retried `Wager`, `Result` and `Rollback` may take place several hours or days after the original call.

The operator wallet **must** return an identical response to any retried transaction as would have been returned by the first transaction call – i.e. if an operation has been successful once, it must always be accepted when retried even if the session ID or game round has expired.

Transaction Ordering

The operator wallet **must not** make any assumptions about the ordering of `Wager` and `Rollback` methods. I.e. the operator platform must treat the net effect on a user's wallet of receiving a `Rollback` *before* the corresponding `Wager` as being identical to receiving the `Wager` before the `Rollback`.

This allows CE Direct to immediately `Rollback` a transaction even if, owing to network error or operator wallet failure, a response has not been received to the initial transaction request. The transaction and `Rollback` requests may safely be retried by CE Direct until the network/platform has stabilized and a response is received to both.

Implementation of this requirement in the operator wallet can be achieved by maintaining a log of transaction `Rollback` requests; if a Rollback is received without a corresponding transaction, the details of the

`Rollback` will be retained by the operator wallet and subsequently checked (and rollback applied) when the corresponding transaction is applied.

Security

The operator **must** provide an SSL/TLS (https:) endpoint for the Wallet API, if an alternative means of securing communications (VPN or dedicated link) has not been agreed with EveryMatrix.

The operator platform **is responsible** for managing player login/sessions, and for validating that the Session Token passed by CE Direct is valid before applying any transaction to a user's wallet.

The operator **must** protect the Wallet API with a username and a password.

Performance

The performance of the Wallet API is critical and has a direct effect on player gaming experience (poor wallet response times lead to long reel-spin times and potential glitching in games, and in time-sensitive gaming experiences such as Live Dealer may lead to player wagers being rejected and rolled back.) The operator **should** ensure that total response time *including total network latency between EM's CE Direct platform and the Wallet API* to all transactions is below 200ms, and **must** be below 500ms at all times, regardless of transaction volumes.

Backward-Compatibility

EveryMatrix is continually improving its products and introducing new features. In order to achieve this without introducing breaking changes, the operator Wallet API **must** ignore without failure the presence of new or additional parameters to requests not included in this documentation or available at the time of integration.

The ordering of parameters within request/response objects is not fixed, i.e. the wallet API **must** parse parameters in any order.

4.3 Implementation Approach

All methods are implemented as HTTP **POST** methods to the operator's wallet endpoint. The payload of the POST is a JSON encoded object containing the request parameters (as specified in this document.)

The response from the operator Wallet API must be a JSON encoded object containing the required response parameters.

The HTTP response status is used to indicate that the API request was successfully received and decoded by the operator wallet, not to indicate the success or failure of the transaction. I.e. a transaction which violates a rule of business logic (e.g. an attempt to debit an account without the necessary funds available) must nevertheless return an HTTP 200 'Success' status code, with the transaction status ('Insufficient Funds') returned in the relevant fields of the JSON payload (see *Error Handling*.)

4.4 Wallet API Methods

CE Direct requires the operator API to implement just six methods to complete an integration:

Method Name	Description
GetAccount	Provides details of the logged in user and his account to the CE Direct platform when a user launches a game.
GetBalance	Returns the current user's account balance to the CE Direct platform.
Wager	Debits funds from a user's account in response to a gaming transaction (reel spin.)
Result	Finalises (marks as closed) a game round, and also credits any winnings accrued.
Rollback	Cancels a wager transaction (e.g. because the user disconnected from the gameplay server before the gaming transaction was completed, or because a transaction could not be completed by the operator's wallet in time.)
GetTransactionStatus	Returns the status of an incomplete transaction. Even though this method is optional, CEDirect uses it to get the problem automatically fixed when the transaction has failed.

Common Request Parameters

The following parameters are sent by CE Direct to the operator's Wallet API with every request. They are used by the wallet platform to verify that the request was sent by the EveryMatrix platform, and to determine the request type:

Parameter (**Bold**=mandatory)	Example	Type	Description
ApiVersion	1.0	*String(32)*	The API version. Currently version 1.0.
Request	GetAccount	*String(32)*	The method name identifying the request.
OperatorId	6	*Long*	A unique ID identifying the operator within the EveryMatrix platform.
LoginName	EveryMatrix	*String(250)*	A username agreed between Operator and EveryMatrix identifying the CE Direct *application* (not an individual player) to the operator's Wallet API.
Password	jHbyoBoDMcvVxL	*String(250)*	A shared secret (password) agreed between Operator and EveryMatrix used by the operator to verify that the request is legitimately from the CE Direct platform.

Common Response Variables

The operator's Wallet API must return the following response variables with every processed request, in addition to the request-specific responses detailed with each method:

Variable (**Bold**=mandatory)	Example	Type	Description
ApiVersion	1.0	*String(32)*	The API version. Currently version 1.0.
Request	GetAccount	*String(32)*	The method name identifying the request.
ReturnCode	0	*Integer*	A value of 0 indicates Success; see the section *Error Handling* for details of other possible values.
Message	Success	*String(250)*	A human-readable indication of the transaction status.

The remainder of this section considers each API method in detail.

4.4.1 GetAccount Method

The `GetAccount` method is called to get the information about the player's location, currency and personal details. It is called by CE Direct when a player launches a game.

Certain account information is optional, depending on the type of integration and the license. For an operator operating under an EveryMatrix gaming license, it may be mandatory to provide certain personal information (required for compliance with *Know Your Customer* regulations) to EveryMatrix; this data may not be required for an operator with its own license. Your EveryMatrix Integration Manager will discuss the required information with you during the integration process.

Request Parameters and Example Request

Parameter (**Bold**=mandatory)	Example	Type	Description
SessionId	CFHA5347RUPD89	*String(250)*	The Session identifier/token passed to CE Direct when the player game was launched (via the `sid` parameter to GameLoader.)

Example `GetAccount` request:

```
{
  "ApiVersion":  "1.0",
  "LoginName":   "EveryMatrix",
  "Password":    "EveryMatrixpassword",
  "Request":     "GetAccount",
  "OperatorId":  6,
  "SessionId":   "CF125F-1624-FDGC21-102562"
}
```

Response Variables

Variable (**Bold**=mandatory)	Example	Type	Description
AccountId	112345	*String(250)*	The identifier of the player's account from the operator's wallet platform. This will be used in subsequent transaction requests from CE Direct.
City	Oslo	*String(85)*	The player's city of residence.
Country	NOR	*String(3)*	The player's country of residence (ISO3166-1 Alpha3 code.)
Currency	NOK	*String(3)*	Player's account currency (ISO4217 Alpha3 code.) If the gaming provider does not support the currency, CE Direct will perform appropriate conversions to the default operator system currency configured in the CE Direct back office.
SessionId	CFHA5347RUPD89	*String(250)*	The operator's Session ID. Should match the request.
UserName	BuoyantHero	*String(250)*	The user's username in the operator platform.
FirstName	Peter	*String(250)*	The user's first (given) name.
LastName	Heinen	*String(250)*	The user's last/family name.
Alias	Hase	*String(250)*	A user alias intended for display in-game. This may be visible to other users (e.g. at a live dealer table,) and should be solicited from the user on that basis. If this is not provided, the `AccountID` will be substituted where necessary (the `UserName` is not substituted by default because it may contain personal information not intended for public view.)
Birthdate	1991-10-17	*String(10)*	User's date of birth, in ISO8601 format.
RCPeriod	15	*String(2)*	Reality check period duration in minutes.

Example `GetAccount` response:


```
{
  "ApiVersion": "1.0",
  "Request":    "GetAccount",
  "ReturnCode": 0,
  "Message":    "Success",
  "AccountId":  "112345",
  "City":       "Oslo",
  "Country":    "NOR",
  "Currency":   "NOK",
  "SessionId":  "CF125F-1624-FDGC21-102562",
  "UserName":   "BuoyantHero",
  "FirstName":  "Peter",
  "LastName":   "Heinen",
  "Alias":      "Hase",
  "Birthdate":  "1991-10-17",
 "RCPeriod": "15"
}
```

4.4.2 GetBalance Method

The `GetBalance` method is called to obtain the current user wallet balance. This is used by vendors to show current game balance within the game screen (it is not used to enforce business logic – i.e. it is the responsibility of the operator wallet platform to validate availability of funds on wager transaction requests.)

Request Parameters and Example Request

Parameter (**Bold**=mandatory)	Example	Type	Description
SessionId	CFHA5347RUPD89	*String(250)*	The Session identifier/token passed to CE Direct when the player game was launched (via the `sid` parameter to GameLoader.)
AccountId	112345	*String(250)*	The ID of the player's account.

Example `GetBalance` request:

```
 {
"ApiVersion": "1.0",
"LoginName": "EveryMatrix",
"Password": "EveryMatrixpassword",
"Request": "GetBalance",
"OperatorId": 6,
"SessionId": "CF125F-1624-FDGC21-102562",
"AccountId": "112345",
}
```

Response Variables

Variable (**Bold**=mandatory)	Example	Type	Description
Balance	350.25	*Decimal(10,2)*	The balance of the player's wallet.
BonusMoney	125.25	*Decimal(10,2)*	The balance of the player's bonus wallet
RealMoney	200	*Decimal(10,2)*	The balance of the player's real money wallet
Currency	NOK	*String(3)*	Player's account currency (ISO4217 Alpha3 code.)
SessionId	CFHA5347RUPD89	*String(250)*	The operator's Session ID. Should match the request.

Example `GetBalance` response:

```
{
    "ApiVersion": "1.0",
    "Request": "GetBalance",
    "ReturnCode": 0,
    "Message": "Success",
    "Balance": 350.25,
    "BonusMoney": 125.25,
    "RealMoney": 200,
    "Currency": "NOK",
    "SessionId": "CF125F-1624-FDGC21-102562"
}
```

4.4.3 Wager Method

The `Wager` method debits funds from a user's wallet. Wager transactions within a game are associated with *game rounds* – there may be multiple wagers associated with a single game round. Each individual wager will be sent separately and will have a unique `TransactionId`, but may have the same `RoundId`.

Request Parameters and Example Request

Parameter (**Bold**=mandatory)	Example	Type	Description
SessionId	CFHA5347RUPD89	*String(250)*	The Session identifier/token passed to CE Direct when the player game was launched (via the `sid` parameter to GameLoader.)
AccountId	112345	*String(250)*	The ID of the player's account (as determined by the GetAccount method.)
Amount	25.25	*Decimal(10,2)*	The amount of the wager, in the player's account currency (as determined by the `GetAccount` method.)
GameType	tablegames	*String(32)*	The type of a game. *Not in use now.*
GPGameId	Livegames	*String(128)*	The game identifier in the vendor's system.
GPId	789	*Integer*	The vendor's identifier.
EMGameId	space-lights-pc	*String(50)*	The EveryMatrix game identifier.
Product	casino	*String(32)*	The domain identifier.
RoundId	213e4567-12fbde	*String(36)*	The `Round ID` identifying this gaming round, as described above.
Device	desktop	*String(32)*	The device being used by the player, currently ***desktop*** or ***mobile***.
TransactionId	1234567	*Long*	The unique transaction identifier sent by EveryMatrix. This `TransactionId` may be used in subsequent `Rollback` requests, and multiple `Wager` requests with the same `TransactionId` must result in *at most one* corresponding adjustment to the user's wallet balance.
RoundStatus	closed	*String(32)*	Indicates if the gaming round is closed (`RoundStatus` = ***closed***), or if further results may be received for this gaming round (`RoundStatus` = ***open***).

Example `Wager` request:

```
{
  "ApiVersion": "1.0",
  "LoginName": "EveryMatrix",
  "Password": "EveryMatrixpassword",
  "Request": "Wager",
  "Amount": 25.25,
  "AccountId": "112345",
  "Device": "desktop",
  "SessionId": "CF125F-1624-FDGC21-102562",
  "GameType": "tablegames",
  "GPGameId": "livegames",
  "EMGameId": "space-lights-pc",
  "GPId": 789,
  "Product": "casino",
  "RoundId": "123e4567-e89b-12d3-a456-426655332222",
  "TransactionId": 12345,
  "RoundStatus": "closed"
}
```

Response Variables

Variable (**Bold**=mandatory)	Example	Type	Description
AccountTransactionId	9812345	*String(250)*	Unique transaction ID allocated by the wallet platform.
Currency	NOK	*String(3)*	Player's account currency (ISO4217 Alpha3 code.)
Balance	325.00	*Decimal(10,2)*	The player's account balance after the application of the wager transaction.
SessionId	CFHA5347RUPD89	*String(250)*	The operator's Session ID. Should match the request.
BonusMoneyBet	4.75	*Decimal(10,2)*	The amount of Bonus Money used to fulfil the wager.
RealMoneyBet	20.50	*Decimal(10,2)*	The amount of Real Money used to fulfil the wager.

Example `Wager` response:

```
{
  "ApiVersion":           "1.0",
  "Request":              "Wager",
  "ReturnCode":           0,
  "Message":              "Success",
  "BonusMoneyBet":        "4.75",
  "RealMoneyBet":         "20.50",
  "AccountTransactionId": "98765",
  "Currency":             "NOK",
  "Balance":              "325",
  "SessionId":            "CF125F-1624-FDGC21-102562"
}
```

4.4.4 Result Method

The `Result` method credits winnings to the user's wallet. Depending on the game type, there may be multiple intermediate Results in a single game round. The `RoundStatus` parameter is used to indicate if the round is also closed.

If EM receive any error from Wallet, `Result` request will be retried. In case of retry wallet must act in idempotent way, that means:

- if original `Result` request was processed with success, then return success also to retries;
- if original `Result` request was failed on wallet, then wallet must process it again.

EM does not accept any errors on that result request and it will be retried.

Request Parameters and Example Request

Parameter (**Bold**=mandatory)	Example	Type	Description
SessionId	CFHA5347RUPD89	*String(250)*	The Session identifier/token passed to CE Direct when the player game was launched (via the *sid* parameter to GameLoader.)
AccountId	112345	*String(250)*	The ID of the player's account (as determined by the GetAccount method.)
Amount	25.25	*Decimal(10,2)*	The amount of winnings (if any) to credit to the player wallet, in the player's account currency (as determined by the GetAccount method.)
GameType	tablegames	*String(32)*	The type of a game. *Not in use now.*
GPGameId	Livegames	*String(128)*	The game identifier in the vendor's system.
GPId	789	*Integer*	The vendor's identifier.
EMGameId	space-lights-pc	*String(50)*	The EveryMatrix game identifier.
Product	Casino	*String(32)*	The domain identifier.
Device	desktop	*String(32)*	The device being used by the player, currently **`desktop`** or **`mobile`**.
RoundId	213e4567-12fbde	*String(36)*	The Round ID identifying this gaming round.
RoundStatus	closed	*String(32)*	Indicates if the gaming round is closed (`RoundStatus` = ***closed***), or if further results may be received for this gaming round (`RoundStatus` = ***open***).
TransactionId	1234567	*Long*	The unique Transaction ID sent by EveryMatrix. Multiple `Wager` requests with the same `TransactionId` must result in at most one corresponding adjustment to the user's wallet balance.
VendorData		*String*	Vendor-specific gaming data. This is optional, and may only be sent for certain types of games (e.g. details of winning hand in a poker game.) Data format is game/vendor specific
BetPayload	25.25	*Decimal (10,2)*	The amount of the wager, which the wallet platform can use for internal purposes.

Example `Result` request:

```
{
    "ApiVersion": "1.0",
    "LoginName": "EveryMatrix",
    "Password": "EveryMatrixpassword",
    "Request": "Result",
    "OperatorId": 6,
    "Amount": "25.25",
    "AccountId": "112345",
```

```
    "SessionId": "CF125F-1624-FDGC21-102562",
    "GameType": "tablegames",
    "GPGameId": "livegames",
    "EMGameId": "space-lights-pc",
    "GPId": 789,
    "Product": "casino",
    "Device": "desktop",
    "RoundId": "123e4567-e89b-12d3-a456-426655332222",
    "RoundStatus": "closed",
    "TransactionId": 12345,
    "VendorData": "FiveCards",
    "BetPayload": 25.5
}
```

Response Variables

Variable (**Bold**=mandatory)	Example	Type	Description
AccountTransactionId	9812345	*String(250)*	Unique transaction ID allocated by the wallet platform.
Currency	NOK	*String(3)*	Player's account currency (ISO4217 Alpha3 code.)
Balance	325.00	*Decimal(10,2)*	The player's account balance after the application of the result credit transaction.
SessionId	CFHA5347RUPD89	*String(250)*	The operator's Session ID. Should match the request.

Example `Result` response:

```
{
  "ApiVersion":           "1.0",
  "Request":              "Result",
  "ReturnCode":           0,
  "Message":              "Success",
  "AccountTransactionId": "98765",
  "Balance":              "375.50",
  "SessionId":            "CF125F-1624-FDGC21-102562",
  "Currency":             "NOK"
}
```

4.4.5 Rollback Method

The `Rollback` method is used to cancel a `Wager` transaction. This may occur, for example, the player's game client disconnects from the gaming server mid-game. Another possible scenario requiring a rollback would be if a valid response was not received from the Wallet API to a Wager transaction within an acceptable time limit (e.g. a transaction on a Live Dealer game or Live Sports bet was not processed before betting was suspended.)

If EM receive communication error from Wallet API or General error, `Rollback` request will be retried. Before processing rollback, wallet must check status of original bet. If original bet was not found in the wallet system or was processed with error, than wallet can return `Success` or `TransactionNotFoundError`, which means that there is nothing to rollback.

In case of retry, wallet must act in idempotent way, that means:

- if original `Rollback` request was processed with success, then return success also to retries.
- if original `Rollback` request was failed on wallet - then wallet must try to process it again.

If wallet returns `TransactionNotFoundError`, no further retries will be sent to wallet.

Request Parameters and Example Request

Parameter (**Bold**=mandatory)	Example	Type	Description
SessionId	CFHA5347RUPD89	*String(250)*	The Session identifier/token passed to CE Direct when the player game was launched (via the `sid` parameter to GameLoader.)
AccountId	112345	*String(250)*	The ID of the player's account (as determined by the `GetAccount` method.)
Amount	25.25	*Decimal(10,2)*	The amount of the transaction to roll back.
TransactionId	1234567	*Long*	The transaction ID sent by EveryMatrix with the Wager transaction to be rolled back.
GameType	tablegames	*String(32)*	The type of a game. *Not in use now.*
GPGameId	Livegames	*String(128)*	The game identifier in the vendor's system.
GPId	789	*Integer*	The vendor's identifier.
EMGameId	space-lights-pc	*String(50)*	The EveryMatrix game identifier.
Product	Casino	*String(32)*	The domain identifier.
Device	desktop	*String(32)*	The device being used by the player, currently ***desktop*** or ***mobile***.
RoundId	213e4567-12fbde	*String(36)*	The Round ID identifying this gaming round.

Example `Rollback` request:

```
{
  "ApiVersion":    "1.0",
  "LoginName":     "EveryMatrix",
  "Password":      "EveryMatrixpassword",
  "Request":       "Rollback",
  "OperatorId":    6,
  "Amount":        "25.25",
  "AccountId":     "112345",
  "SessionId":     "CF125F-1624-FDGC21-102562",
  "GameType":      "tablegames",
  "GPGameId":      "livegames",
  "EMGameId":      "space-lights-pc",
  "GPId":          789,
  "Product":       "casino",
  "Device":        "mobile",
```

```
  "RoundId":       "123e4567-e89b-12d3-a456-426655332222",
  "TransactionId": 12345
}
```

Response Variables

Variable (**Bold**=mandatory)	Example	Type	Description
AccountTransactionId	9812345	*String(250)*	Unique transaction ID allocated by the wallet platform.
Currency	NOK	*String(3)*	Player's account currency (ISO4217 Alpha3 code.)
Balance	325.00	*Decimal(10,2)*	The player's account balance after the application of the rollback.
SessionId	CFHA5347RUPD89	*String(250)*	The operator's Session ID. Should match the request.

Example `Rollback` response:

```
{
  "ApiVersion":           "1.0",
  "Request":              "Rollback",
  "ReturnCode":           0,
  "Message":              "Success",
  "AccountTransactionId": "98765",
  "Balance":              "375.50",
  "SessionId":            "CF125F-1624-FDGC21-102562",
  "Currency":             "NOK"
}
```

4.4.6 GetTransactionStatus Method

The `GetTransactionStatus` API method returns the status of an incomplete transaction. Even though this method is optional, CEDirect uses it to get the problem automatically fixed when the transaction has failed.

Request Parameters and Example Request

Parameter (**Bold**=mandatory)	Example	Type	Description
`TransactionId`	12345	*Long*	The transaction ID sent by EveryMatrix.

Example `GetTransactionStatus` request:

```
{
    "ApiVersion": "1.0",
    "LoginName": "EveryMatrix",
    "Password": "EveryMatrixpassword",
    "Request": "GetTransactionStatus",
    "OperatorId": 6,
    "TransactionId": 12345
}
```

Response Variables

Variable (**Bold**=mandatory)	Example	Type	Description
`TransactionId`	12345	Long	The transaction ID sent by EveryMatrix
`TransactionStatus`	Processed	String (9)	Transaction status with possible values : ***Processed***, ***Notexists***, ***Failed***.

Example `GetTransactionStatus` response:

```
{
    "ApiVersion": "1.0",
    "LoginName": "EveryMatrix",
    "Password": "EveryMatrixpassword",
    "Request": "GetTransactionStatus",
    "TransactionId": 12345,
    "TransactionStatus": "Processed"
}
```

4.5 Error Handling

As noted, an HTTP response status of 200 is used to indicate that an API request was successfully received and decoded by the operator wallet, even if the transaction itself was not applied because of a business logic restriction (e.g. if funds are not available for an attempted wager.)

In the event of such a transaction failure, the response should return an appropriate `ReturnCode` value indicating the error condition – for example:

```
{
  "ApiVersion": "1.0",
  "Request":    "Wager",
  "ReturnCode": 104,
  "Message":    "Insufficient funds"
}
```

The following table lists the valid `ReturnCode` responses the Wallet API may return:

ReturnCode	Title	Description	Applicable Methods
0	**Success**	The request has been processed successfully	All methods
101	**Unknown error**	Unknown error	All methods
102	**User is blocked**	The user has been blocked and cannot wager	`GetAccount, GetBalance, Wager, Result, Rollback`
103	**User not found**	The user has not been found	`GetAccount, GetBalance, Wager, Result, Rollback`
104	**Insufficient funds**	The user does not have enough funds to wager	`Wager`
105	**IP is not allowed**	The request has been made from a prohibited IP address	All methods
106	**Currency not supported**	The currency in the request is different from the one sent in the Authenticate method	`Wager, Result`
108	**TransactionNotFound**	Wallet system has nothing to rollback	`Rollback`
109	**CasinoLimitExceededLoss**	The user has exceeded the loss limit set per period	`Wager`
110	**CasinoLimitExceededStake**	The wager has exceeded the expense limit	`Wager`
111	**CasinoLimitExceededSession**	The user has reached the time limit of the current session	`Wager`
112	**MaxStakeLimitExceeded**	The bet has exceeded the maximum limit of the expense	`Wager`
113	**UserSelfExcluded**	The user has excluded himself/herself on the website	`GetAccount, GetBalance, Wager, Result, Rollback`
114	**UserNotActive**	The user has been inactive	`GetAccount, GetBalance, Wager, Result, Rollback`