Manual

ENGAGE API

Play’n GO

PLAY'N GO ENGAGE API

Table of Contents

1 Introduction

To enable external code to communicate with a Play’n GO game client, Play’n GO provides the Engage API.

The Engage API is a middle layer between the Game and the Operator. It allows the Operator to listen on specific game events and trigger functionality in the game.

1.1 Supported technologies

- Desktop
- Mobile

2 Engage API

The Engage API is a JavaScript module which simplifies the communication between the Operator client side and code and the game.

The API is based on **requests** from an external implementation to the game and **events** from the game to the external implementation.

2.1.1 Mobile and Desktop /ContainerLauncher/

If the ContainerLauncher (launching the game embedded in an Iframe) is used, window.postMessage will be used for communication between the Operator implementation and the Engage API.

2.1.2 Desktop /JS/ launcher

The API is globally accessible from the hosting page via the Engage JavaScript object.

2.1.3 Mobile /PlayMobile/ launcher

Not supported. To use Engage for Mobile games, please see section 2.1.1

2.2 Functions

2.2.1 Desktop /JS/ launcher

In the Desktop implementation the Engage API is directly accessible within the global scope.

Example of request call: `Engage.request({ req: "gameDisable"});`

Example of addEventListener call: `Engage.addEventListener("gameDisabled", onGameDisabled);`

Function	Description	Content

request	Used for requesting call to game functionality e.g. “gameDisable”, “gameEnable”, “gameEnd”
	
	Format:
	Engage.request(
	<{req:string, data:object}>
	)
		object:{
		req: (string)
		data: (object)(optional)
		}

addEventListener	Adds event listener to act upon **Format:** Engage.addEventListener (<event:string>,<callback:function>)	Event:String Callback: Function
removeEventListener	Removes an event listener **Format:** Engage.removeEventListener (<event:string>,<callback:function>)	
enableDebug	Enables debug logging of events and requests. **Format:** Engage.enableDebug();	

2.2.2 Mobile and desktop /ContainerLauncher/

The ContainerLauncher implementation of Engage is accessible only via a window.postMessage. Therefore, no direct function calls will be available. Instead the requested function is sent in the message call.

Example of request call: *.postMessage({ messageType: "request", request: "spin" }, targetOrigin)

Example of addEventListener call: *.postMessage({ messageType: "addEventListener", eventType: "roundStarted" }, targetOrigin)

Functions	Description	Content

request	postMessage{ messageType: "request", request: "spin" }, targetOrigin);	object:{
		messageType: (string)
		request: (string)
		data: (object)(optional)
		}

addEventListener	postMessage ({ messageType: "addEventListener", eventType: "roundStarted" }, targetOrigin)	object:{
		messageType: (string)
		eventType : (string)
		}

2.3 Requests

2.3.1 Generic requests

Request	Request data	Callback event	Description
gameDisable		gameDisabled	Disables all user interactions and stops Autoplay (if any). Will take effect next time game is in an idle state.
gameEnable		gameEnabled	Enables user interactions.

gameEnd	`(object)`
	`data:{`
	`redirectUrl:`**`encoded-url`**
	`}`
		logout	Ends the game and redirect the user to defined url.

logout		logout	Ends game.
refreshBalance		balanceUpdate	Updates the in-game balance.

inGameMessage	`(object)`
	`data:{`
	`id:”`**`id_of_message`**`”`
	`}`
		Button response events:
		Continue:
		externalMessageOk
		Action:
		externalMessageAction
		Exit:
		externalMessageExit
			Creates an in-game message which will be displayed once game is idle. User interaction will be limited to message while it is showing.
			
			Example:
			Engage.request(
			{ req: "inGameMessage", data: {id:"", title: "", message: "", okBtn: "", exitBtn: "", actionBtn: ""} }
			);
			Possible button combinations:
			okBtn-exitBtn-actionBtn
			okBtn-exitBtn
			okBtn

soundOn			Turns sound on in game
soundOff			Turns sound off in game

getBalance		balance
		(object) data: {
		BalanceInMoney (string)
		Currency(string)
		}
			Returns an object with the current balance in money as a currency formatted string and the currency as a string.

getBet		bet
		(object) data: {
		BetInMoney (string)
		Currency(string)
		}
			Returns an object with the current bet in money as a currency formatted string and the currency as a string.

getWin		win
		(object)data: {
		WinInMoney (string)
		Currency(string)
		}
			Returns an object with the last win in money as a currency formatted string and the currency as a string.

2.3.2 VideoSlot requests

Request	Request data	Callback event	Description
stopAutoplay		autoplayEnded	Stops the autoplay after the active round.
getSelectedCoin		selectedCoin Data:Number	Returns the currently selected coin as a number.
getSelectedCoinValue		selectedCoinValue Data:String	Returns the currently selected

			coin value as a currency formatted string.
getSelectedLines		selectedCoin Data:Number	Returns number of selected lines as a number.
getAvailableCoins		availableCoins Data:Number	Returns the available number of coins as a number.
getBalance		balance (object)data: { BalanceInCoins(number) BalanceInMoney (string) Currency(string) }	Returns an object with the current balance in coins as a number, the balance in money as a currency formatted string and the currency as a string.
getBet		bet (object) data: { BetInCoins(number) BetInMoney (string) Currency(string) }	Returns an object with the current bet in coins as a number, the bet in money as a currency formatted string and the currency as a string.

getWin		win
		
		(object) data:{
		WinInCoins(number)
		WinInMoney (string)
		Currency(string)
		}	

2.3.3 VideoBingo requests

Request	Request data	Callback event	Description
getActiveCards		activeCards Data:Number	Returns number of selected Cards as a number.

2.4 Events

2.4.1 Generic events

Event	Callback data	Description
gameReady		Event is sent when game is ready for user interaction.

gameError	(object) data{
	title:string,
	message:string
	}
		Event is sent when game is showing an message.

running		Event is sent when the round is started (same as roundStarted).
roundStarted		Event is sent when the round is started (same as running).
roundEnded		Event is sent when the round is ended.
gameEnabled		Event is sent when the game is enabled.
gameDisabled		Event is sent when the game is disabled.
gameIdle		Event is sent when game is idle.

logout		Event is sent when game is logged out
backToLobby		Event is sent when game is closed using the lobby button
balanceUpdate	(number) rawBalance (string) currency	Event is sent when balance changes.
roundWin	(number) winAmount	Event is sent when a win occurs.
externalMessageOk	(string) id	Event is sent when user clicks Ok/Continue button of an external message window. Passes back id used when requested external message.
externalMessageExit	(string) id	Event is sent when user clicks Exit/Cancel button of an external message window. Passes back id used when requested external message.
externalMessageAction	(string) id	Event is sent when user clicks Action button of an external message window. Passes back id used when requested external message.
playForReal		Event is sent when user clicks the play for real button.
reloadGame		Event is sent when the game requires a reload.

bet	(Object)data:{
	BetInMoney (string)
	Currency(string)
	}
		Event is sent when game receives the request getBet

balance	(Object)data:{
	BalanceInCoins(number)
	BalanceInMoney (string)
	Currency(string)
		Event is sent when game receives the request getBalance

	}	

win	(Object)data:{
	WinInCoins(number)
	WinInMoney (string)
	Currency(string)
	}
		Event is sent when game receives the request getWin

realityCheckEvent	(Object)data:{
	(string) type
	(number) bet
	(number) minutes
	(number) win
	}
		Event is sent when the game is showing realitycheck message

2.4.2 VideoSlot events

Event	Callback data	Description
spinStarted		Event is sent when a spin starts
spinEnded		Event is sent when all reels have stopped.
paylineWin		Even is sent when win presentation is started.
autoplayStarted	(number) numAutoplay	Event is sent when Autoplay is started.
autoplayNextRound	(number) numAutoplayLeft	Event is sent for each new Autoplay spin.
autoplayEnded		Event is sent when Autoplay is stopped.
freespinStarted		Event is sent when FreeSpins mode is started.
freespinEnded		Event is sent when FreeSpins mode is ended.
bonusGameStarted		Event is sent when Bonus mode is started. See list of supported games below.
bonusGameEnded		Event is sent when Bonus mode is ended. See list of supported games below.
gambleStarted		Event is sent when Gamble mode is started.

gambleEnded		Event is sent when Gamble mode is ended.

bet	(Object)data:{
	BetInMoney (string)
	BetInCoins(number)
	Currency(string)
	}
		Event is sent when game receives the request getBet

getAvailableCoins	(Number)data	
getSelectedLines	(Number) data	
getSelectedCoin	(Number) data	