Amazon GameLift Streams Web SDK 1.0.0

Table of contents

Variables

VERSION

• Const **VERSION**: "1.0.0"

Functions

setLogLevel

▸ **setLogLevel**(level): void

Controls the amount of console log output generated by the Amazon GameLift Streams Web SDK

Parameters

Name	Type	Description
level	"none" | "debug"	the desired console logging level (“none” or “debug”)

Returns

void

Class: GameLiftStreams

Manages the connection between an existing HTML video element and a single Amazon GameLift Streams stream

Table of contents

Constructors

- constructor

Accessors

- id

Methods

- addGamepad
- attachInput
- close
- detachInput
- enableMicrophone
- generateSignalRequest
- getAudioRTCStats
- getVideoRTCStats
- processGamepads
- processKeyboardEvent
- processMouseEvent
- processSignalResponse
- removeGamepad
- sendApplicationMessage

Constructors

constructor

• **new GameLiftStreams**(args)

Parameters

Name	Type	Description
args	GameLiftStreamsArgs	GameLiftStreamsArgs configuration object

Accessors

id

• get **id**(): string

Unique 128-bit identifier (as a 16-character hex string)

Returns

string

Methods

addGamepad

▸ **addGamepad**(gamepad): boolean

Adds a gamepad to the set of tracked gamepads. This can be a gamepad from the browser's navigator.getGamepads() API, or a fake gamepad-like object. This is not necessary if the autoGamepad input configuration property was set.

If adding a gamepad-like object, you should set the .index property of the object to be a high number (100+) to avoid conflicting with the .index property of any real gamepads.

See

https://developer.mozilla.org/en-US/docs/Web/API/Gamepad

Parameters

Name	Type	Description
gamepad	Gamepad	the W3C Gamepad or gamepad-like object to add

Returns

boolean

true if the gamepad was added to the set of tracked gamepads

---

attachInput

▸ **attachInput**(): void

Enables mouse/keyboard/gamepad input event transmission and enables any automatic event listeners that were specified in the input configuration passed to the Amazon GameLift Streams constructor. Only valid after the stream connection is established and before calling close().

Returns

void

---

close

▸ **close**(): void

Closes the stream and destroys internal state of this Amazon GameLift Streams object.

Returns

void

---

detachInput

▸ **detachInput**(): void

Disables mouse/keyboard/gamepad input event transmission and disables any automatic input listeners that were specified in the input configuration passed to the Amazon GameLift Streams constructor. Only valid after the stream connection is established and before calling close().

Returns

void

---

enableMicrophone

▸ **enableMicrophone**(constraints?): Promise<void>

Attempts to attach microphone input to this Amazon GameLift Streams session. This may trigger UI in the web browser which asks the user if they wish to allow microphone access.

Note that this operation must be completed/resolved before generateSignalRequest().

Throws

DOMException if the web browser denies microphone access

Parameters

Name	Type	Description
`constraints`	`MediaTrackConstraints`	(Optional) device selection criteria, such as deviceId (see https://developer.mozilla.org/en-US/docs/Web/API/MediaTrackConstraints)

Returns

Promise<void>

Promise that resolves when the web browser has allowed microphone access

---

generateSignalRequest

▸ **generateSignalRequest**(): Promise<string>

Prepares to connect to an Amazon GameLift Streams session by internally generating a WebRTC offer and converting that offer into a SignalRequest string value. The resulting string should be passed to your backend server, which will forward the string as the SignalRequest parameter to an Amazon GameLift Streams StartStreamSession API call.

See

https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity

Returns

Promise<string>

Promise that resolves into a SignalRequest string value

---

getAudioRTCStats

▸ **getAudioRTCStats**(): Promise<RTCStatsReport>

Retrieves the WebRTC RTCStatsReport objects with browser-specific stream metrics for the audio track.

Returns

Promise<RTCStatsReport>

Promise which resolves to a WebRTC RTCStatsReport object

---

getVideoRTCStats

▸ **getVideoRTCStats**(): Promise<RTCStatsReport>

Retrieves the WebRTC RTCStatsReport objects with browser-specific stream metrics for the video track.

Returns

Promise<RTCStatsReport>

Promise which resolves to a WebRTC RTCStatsReport object

---

processGamepads

▸ **processGamepads**(): boolean

Scan all tracked gamepads for any input changes forward the updated inputs to the remote side of the stream. This is not necessary if the autoGamepad input configuration property was set.

Returns

boolean

true if the stream is connected and there were no internal errors

---

processKeyboardEvent

▸ **processKeyboardEvent**(e): boolean

Transmits a KeyboardEvent (keydown or keyup) to the remote side of the stream. This is not necessary if the autoKeyboard input configuration property was set.

Parameters

Name	Type	Description
e	KeyboardEvent	the event to pass

Returns

boolean

true if the event was transmitted to the remote side of the stream

---

processMouseEvent

▸ **processMouseEvent**(e): boolean

Transmits a MouseEvent (mousedown, mouseup, mousemove, or mousewheel) to the remote side of the stream. This is not necessary if the autoMouse input configuration property was set.

Parameters

Name	Type	Description
e	MouseEvent	the event to pass

Returns

boolean

true if the event was transmitted to the remote side of the stream

---

processSignalResponse

▸ **processSignalResponse**(signalResponse): Promise<void>

Completes the connection to a Amazon GameLift Streams stream by parsing the SignalResponse string retrieved by your backend server, internally transforming it into a WebRTC answer, and then completing the WebRTC stream connection from browser to Amazon GameLift Streams.

See

https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity

Parameters

Name	Type	Description
signalResponse	string	SignalResponse value retrieved by your backend server

Returns

Promise<void>

Promise that resolves when the stream has finished connecting

---

removeGamepad

▸ **removeGamepad**(gamepad): boolean

Remove a gamepad from the set of tracked gamepads. This is not necessary if the autoGamepad input configuration property was set.

Parameters

Name	Type	Description
gamepad	Gamepad	the W3C Gamepad or gamepad-like object to remove

Returns

boolean

true if the gamepad was removed from the set of tracked gamepads

---

sendApplicationMessage

▸ **sendApplicationMessage** (message): boolean

Send a message (as byte array) to the game application.

Parameters

Name	Type	Description
`message`	`Uint8Array`	the message to send as byte array

Returns

boolean

true if the message was sent to the application

Interface: GameLiftStreamsArgs

Amazon GameLift Streams constructor configuration object

Table of contents

Properties

Properties

audioElement

• **audioElement:** HTMLAudioElement

HTML audio element which will be used to play the stream audio

---

clientConnection

• Optional **clientConnection**: IClientConnection

Connection callbacks

---

inputConfiguration

• Optional **inputConfiguration**: InputConfiguration

Mouse/keyboard/gamepad configuration properties

---

streamConfiguration

• Optional **streamConfiguration**: GameLiftStreamsStreamConfiguration

Stream configuration properties

---

videoElement

• **videoElement**: HTMLVideoElement

HTML video element which will be used to display the stream

Interface: GameLiftStreamsStreamConfiguration

Table of contents

Properties

Properties

enableAudio

- Optional **enableAudio**: boolean

Enable audio stream

Default Value

true

---

iceServers

- Optional **iceServers()**: Promise<GameLiftStreamsIceServerConfiguration[]>

(Experimental) Client-supplied ICE server information to include in WebRTC connection negotiation. This setting is experimental and may be removed in the future. It is not recommended outside of specific enterprise scenarios where clients are known to be behind a restrictive firewall.

Default Value

undefined

---

maximumKbps

- Optional **maximumKbps**: number

Maximum client download bitrate, as kilobits-per-second

Default Value

undefined

Interface: GamepadFilterResult

Table of contents

Properties

Properties

suppressButtonIndexes

• **suppressButtonIndexes**: readonly number[ ]

An array of button indices that should be suppressed when forwarding gamepad input to the stream. Each array element should be in the [0, 16] range according to the Standard gamepad mapping, as defined in the W3C Gamepad specification.

See

https://w3c.github.io/gamepad/#remapping

Interface: IClientConnection

Table of contents

Properties

Properties

applicationMessage

• Optional **applicationMessage**: (message: Uint8Array) => void

The applicationMessage callback is triggered whenever a message is received from the application through the data channel.

Parameters

Name	Type	Description
`message`	`Uint8Array`	Message received from application (as byte array)

Returns

void

---

channelError

• Optional **channelError**: (e: any) => void

The channelError callback is triggered whenever there is an internal WebRTC data channel error. Any such errors are usually unrecoverable and your code should terminate the streaming session.

Parameters

Name	Type	Description
e	any	error object reported by browser WebRTC implementation

Returns

void

---

connectionState

• Optional **connectionState**: (state: string) => void

The connectionState callback is triggered whenever the RTCPeerConnection connectionState value changes.

See

https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection

Parameters

Name	Type	Description
state	string	“connecting” | “connected” | “disconnected” | “failed” | “closed”

Returns

void

---

serverDisconnect

• Optional **serverDisconnect**: (reasoncode: string) => void

The serverDisconnect callback is triggered when the streaming server initiates a disconnection.

Parameters

Name	Type	Description
reasoncode	string	“idle” | “terminated” | (other strings reserved for future use)

Returns

void

Interface: InputConfiguration

Table of contents

Properties

- trackWindowFocus

Properties

autoGamepad

• Optional **autoGamepad**: boolean

Automatically look for connected gamepads and forward gamepad input every frame.

Default Value

true

---

autoKeyboard

• Optional **autoKeyboard**: boolean

Automatically listen for and forward keyboard events when input is attached.

Default Value

true

---

autoMouse

• Optional **autoMouse**: boolean

Automatically listen for and forward mouse events.

Default Value

true

---

autoPointerLock

• Optional **autoPointerLock**: boolean | "fullscreen"

Automatically capture the mouse (hiding the cursor and locking the mouse to the current window) whenever the remote cursor has been made invisible on the stream host.

Options:

- true - the mouse is always captured

- false - the mouse is never captured
- **'fullscreen'** - the mouse is only captured if the video element is fullscreen

This is often helpful for "first-person" games which use the mouse for aiming.

This has no effect in the Safari browser or on the iOS platform due to platform limitations.

Default Value

'fullscreen'

---

gamepadFilter

• Optional **gamepadFilter**: null | (event: Gamepad) => boolean | GamepadFilterResult

If set, the gamepadFilter callback function is used to filter gamepad inputs. The callback should return 'false' to completely suppress the gamepad, or a GamepadFilterResult object to suppress only individual buttons.

Default Value

null

---

hapticFeedback

• Optional **hapticFeedback**: boolean

Allow haptic feedback events (eg, controller vibration) to be triggered by the stream.

Default Value

true

---

keyboardFilter

• Optional **keyboardFilter**: null | (event: KeyboardEvent) => boolean

If set, the keyboardFilter callback function is used to filter every keyboard event. The callback should return 'false' to prevent the event from being forwarded to the stream.

Default Value

null

---

mouseFilter

• Optional **mouseFilter**: null | (event: MouseEvent) => boolean

If set, the mouseFilter callback function is used to filter every mouse event. The callback should return 'false' to prevent the event from being forwarded to the stream.

Default Value

null

---

resetOnDetach

• Optional **resetOnDetach**: boolean

Reset all input device state whenever detachInput is called. If false, keyboard and mouse buttons will be frozen in their current state (down vs up) when detachInput() is called.

Default Value

true

---

setCursor

• Optional **setCursor**: boolean | "visibility" | (cursorStyle: string) => void

Allow mouse cursor shape to be controlled by the stream.

Options:

- true - the local cursor replicates the remote cursor shape on the stream host
- false - the local cursor is never modified or hidden
- **'visibility'** - the local cursor is never modified, but it can be hidden
- function - the local cursor is never modified, but the function is invoked with a CSS style string representing the remote cursor whenever the remote cursor changes

Default Value

'visibility'

trackWindowFocus

• Optional **trackWindowFocus**: boolean

Detaches input devices automatically when the window loses focus, and restores input devices when the window regains focus. If false, Amazon GameLift Streams ignores window focus and you must call detachInput() manually to stop tracking user input devices and attachInput() to resume. This setting combines with resetOnDetach to automatically release any held keys or buttons when window focus is lost.

Default Value

true

Type Aliases

GameLiftStreamsIceServerConfiguration

• **GameLiftStreamsIceServerConfiguration**: RTCIceServer | { credential?: string ; url: string ; username?: string }

Amazon GameLift Streams STUN/TURN server configuration, allowing either 'urls' or 'url'

Parameters

Name	Type	Description
`url | urls`	`string | string[]`	One or more URLs representing this server, starting with 'turn:' or 'turns:' or 'stun:' or 'stuns:'
`username`	`string`	(Optional) Username for TURN server authentication
`credential`	`string`	(Optional) Credential (password) for TURN server authentication