

---

## obs-browser

obs-browser introduces a cross-platform Browser Source, powered by CEF (Chromium Embedded Framework), to OBS Studio. A Browser Source allows the user to integrate web-based overlays into their scenes, with complete access to modern web APIs.

Additionally, obs-browser enables Service Integration (linking third party services) and Browser Docks (webpages loaded into the interface itself) on all supported platforms, except for Wayland (Linux).

**This plugin is included by default** on official packages on Windows, macOS, the Ubuntu PPA and the official Flatpak (most Linux distributions).

### JS Bindings

obs-browser provides a global object that allows access to some OBS-specific functionality from JavaScript. This can be used to create an overlay that adapts dynamically to changes in OBS.

### TypeScript Type Definitions

If you're using TypeScript, type definitions for the obs-browser bindings are available through npm and yarn.

```
1 # npm
2 npm install --save-dev @types/obs-studio
3
4 # yarn
5 yarn add --dev @types/obs-studio
```

### Get Browser Plugin Version

```
1 /**
2  * @returns {string} OBS Browser plugin version
3  */
4 window.obsstudio.pluginVersion
5 // => 2.17.0
```

### Register for event callbacks

```
1 /**
2  * @callback EventListener
3  * @param {CustomEvent} event
```

---

```
4  */
5
6  /**
7   * @param {string} type
8   * @param {EventListener} listener
9   */
10 window.addEventListener('obsSceneChanged', function(event) {
11     var t = document.createTextNode(event.detail.name)
12     document.body.appendChild(t)
13 })
```

**Available events** Descriptions for these events can be found [here](#).

- obsSceneChanged
- obsSceneListChanged
- obsTransitionChanged
- obsTransitionListChanged
- obsSourceVisibleChanged
- obsSourceActiveChanged
- obsStreamingStarting
- obsStreamingStarted
- obsStreamingStopping
- obsStreamingStopped
- obsRecordingStarting
- obsRecordingStarted
- obsRecordingPaused
- obsRecordingUnpaused
- obsRecordingStopping
- obsRecordingStopped
- obsReplaybufferStarting
- obsReplaybufferStarted
- obsReplaybufferSaved
- obsReplaybufferStopping
- obsReplaybufferStopped
- obsVirtualcamStarted
- obsVirtualcamStopped
- obsExit
- [Any custom event emitted via obs-websocket vendor requests]

---

## Control OBS

### Get webpage control permissions    Permissions required: NONE

```
1  /**
2   * @typedef {number} Level - The level of permissions. 0 for NONE, 1
   *   for READ_OBS (OBS data), 2 for READ_USER (User data), 3 for BASIC,
   *   4 for ADVANCED and 5 for ALL
3   */
4
5  /**
6   * @callback LevelCallback
7   * @param {Level} level
8   */
9
10 /**
11  * @param {LevelCallback} cb - The callback that receives the current
   *   control level.
12  */
13 window.obsstudio.getControlLevel(function (level) {
14     console.log(level)
15 })
```

### Get OBS output status    Permissions required: READ\_OBS

```
1  /**
2   * @typedef {Object} Status
3   * @property {boolean} recording - not affected by pause state
4   * @property {boolean} recordingPaused
5   * @property {boolean} streaming
6   * @property {boolean} replaybuffer
7   * @property {boolean} virtualcam
8   */
9
10 /**
11  * @callback StatusCallback
12  * @param {Status} status
13  */
14
15 /**
16  * @param {StatusCallback} cb - The callback that receives the current
   *   output status of OBS.
17  */
18 window.obsstudio.getStatus(function (status) {
19     console.log(status)
20 })
```

### Get the current scene    Permissions required: READ\_USER

---

```
1 /**
2  * @typedef {Object} Scene
3  * @property {string} name - name of the scene
4  * @property {number} width - width of the scene
5  * @property {number} height - height of the scene
6  */
7
8 /**
9  * @callback SceneCallback
10 * @param {Scene} scene
11 */
12
13 /**
14 * @param {SceneCallback} cb - The callback that receives the current
    scene in OBS.
15 */
16 window.obsstudio.getCurrentScene(function(scene) {
17     console.log(scene)
18 })
```

**Get scenes** Permissions required: READ\_USER

```
1 /**
2  * @callback ScenesCallback
3  * @param {string[]} scenes
4  */
5
6 /**
7  * @param {ScenesCallback} cb - The callback that receives the scenes.
8  */
9 window.obsstudio.getScenes(function (scenes) {
10     console.log(scenes)
11 })
```

**Get transitions** Permissions required: READ\_USER

```
1 /**
2  * @callback TransitionsCallback
3  * @param {string[]} transitions
4  */
5
6 /**
7  * @param {TransitionsCallback} cb - The callback that receives the
    transitions.
8  */
9 window.obsstudio.getTransitions(function (transitions) {
10     console.log(transitions)
```

---

```
11 })
```

### **Get current transition** Permissions required: READ\_USER

```
1 /**
2  * @callback TransitionCallback
3  * @param {string} transition
4  */
5
6 /**
7  * @param {TransitionCallback} cb - The callback that receives the
8   * transition currently set.
9  */
10 window.obsstudio.getCurrentTransition(function (transition) {
11   console.log(transition)
12 })
```

### **Save the Replay Buffer** Permissions required: BASIC

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.saveReplayBuffer()
```

### **Start the Replay Buffer** Permissions required: ADVANCED

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.startReplayBuffer()
```

### **Stop the Replay Buffer** Permissions required: ADVANCED

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.stopReplayBuffer()
```

### **Change scene** Permissions required: ADVANCED

```
1 /**
2  * @param {string} name - Name of the scene
3  */
4 window.obsstudio.setCurrentScene(name)
```

---

**Set the current transition** Permissions required: ADVANCED

```
1 /**
2  * @param {string} name - Name of the transition
3  */
4 window.obsstudio.setCurrentTransition(name)
```

**Start streaming** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.startStreaming()
```

**Stop streaming** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.stopStreaming()
```

**Start recording** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.startRecording()
```

**Stop recording** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.stopRecording()
```

**Pause recording** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.pauseRecording()
```

---

**Unpause recording** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.unpauseRecording()
```

**Start the Virtual Camera** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.startVirtualcam()
```

**Stop the Virtual Camera** Permissions required: ALL

```
1 /**
2  * Does not accept any parameters and does not return anything
3  */
4 window.obsstudio.stopVirtualcam()
```

**Register for visibility callbacks**

**This method is legacy. Register an event listener instead.**

```
1 /**
2  * onVisibilityChange gets callbacks when the visibility of the browser
   source changes in OBS
3  *
4  * @deprecated
5  * @see obsSourceVisibleChanged
6  * @param {boolean} visibility - True -> visible, False -> hidden
7  */
8 window.obsstudio.onVisibilityChange = function(visibility) {
9
10 };
```

**Register for active/inactive callbacks**

**This method is legacy. Register an event listener instead.**

```
1 /**
2  * onActiveChange gets callbacks when the active/inactive state of the
   browser source changes in OBS
```

---

```
3  *
4  * @deprecated
5  * @see obsSourceActiveChanged
6  * @param {bool} True -> active, False -> inactive
7  */
8  window.obsstudio.onActiveChange = function(active) {
9
10 };
```

## obs-websocket Vendor

obs-browser includes integration with obs-websocket's Vendor requests. The vendor name to use is `obs-browser`, and available requests are:

- `emit_event` - Takes `event_name` and `?event_data` parameters. Emits a custom event to all browser sources. To subscribe to events, see here
  - See #340 for example usage.

There are no available vendor events at this time.

## Building

OBS Browser cannot be built standalone. It is built as part of OBS Studio.

By following the instructions, this will enable Browser Source & Custom Browser Docks on all three platforms. Both `BUILD_BROWSER` and `CEF_ROOT_DIR` are required.

### On Windows

Follow the build instructions and be sure to download the **CEF Wrapper** and set `CEF_ROOT_DIR` in CMake to point to the extracted wrapper.

### On macOS

Use the macOS Full Build Script. This will automatically download & enable OBS Browser.

### On Linux

Follow the build instructions and choose the “If building with browser source” option. This includes steps to download/extract the CEF Wrapper, and set the required CMake variables.