WEBSOCKET

View online: https://www.construct.net/en/make-games/manuals/construct-3/plugin-reference/websocket

The **WebSocket** plugin is a simple wrapper around the standardised WebSocket protocol. It allows for low-overhead bi-directional communication in real-time. Since WebSockets are standards-based, it should be compatible with any standards-compliant WebSocket server.

Using the WebSocket plugin requires a WebSocket server. Construct does not provide a server nor can the WebSocket plugin be used to make a server. If you don't already have a WebSocket server set up, you will need to create one yourself using a technology like node.js with WebSocket support. This can be a significant undertaking and require server-side programming knowledge.

Scripting

This object has no script interface, because when using JavaScript or TypeScript coding you can use the browser built-in WebSocket API.

Secure WebSockets and preview mode

Construct runs and previews your projects on a secure (HTTPS) site. Some browsers block connections from secure to insecure sites. This means if you use an insecure WebSocket (`ws:` rather than `wss:`) the connection may be blocked in Construct's preview mode.

On today's web, it is best practice to use secure connections for all services. Your site should also host your project on HTTPS, and any WebSockets you use should use secure connections as well. HTTPS hosting is actually required for many features to work, secure WebSockets are more likely to connect successfully, and obviously it makes your content much more secure, so this is a good idea anyway. In some browsers you may be able to temporarily unblock the use of insecure WebSockets by clicking a button in the browser or changing settings, but it is a much better idea to ensure all your services are secure. In future, browsers may block the use of insecure sites and connections entirely.

WebSockets and multiplayer games

It may be tempting to use WebSockets to design real-time multiplayer games. Despite the fact they communicate in real-time, WebSockets are not a suitable choice for latency-sensitive real-time games. The underlying transport uses reliable transmission, meaning a single dropped packet can hold up all transmission until the packet is retransmitted successfully. For games with demanding real-time requirements, this can cause unplayable levels of latency. It is usually difficult to design around this without changing the transmission mode, which WebSockets do not support. Consider using the specially-designed Multiplayer plugin Paid plans only instead.

On the other hand, WebSockets should be suitable for games without such a demanding real-time requirement, like turn-based games. It should also be useful for application services, like chat rooms. Note this will still require you to create your own WebSocket server.

WebSocket properties

The WebSocket object has no properties.

WebSocket conditions

Is connecting

True if currently in the process of establishing a connection to a server. The connection is not yet successfully established; there may still be an error.

Is open

True if a connection has been successfully established and the communication channel is currently open.

Is supported

Use before attempting any connections to verify the current browser or platform supports WebSockets.

On closed

Triggered when the connection is closed, either deliberately or due to an error. The *CloseCode* and *CloseReason* expressions can indicate why the connection was closed.

On error

Triggered when an error occurs in the WebSocket connection. Use the *ErrorMsg* expression to get the error message text.

On opened

Triggered when the connection is successfully established and the communication channel is now open.

On binary message

Triggered when a binary message arrives from the server over an open connection. The specified Binary Data object has the contents of the message written to it when the trigger fires, allowing access to the message content.

On text message

Triggered when a text message arrives from the server over an open connection. Use the *MessageText* expression to retrieve the content of the message.

WebSocket actions

Close

Close any active connection. No more messages can be sent or will be received after closing.

Connect

Connect to a WebSocket server. WebSocket server addresses typically start with **ws://** for non-secure transmission and **wss://** for secure transmission. Note some network configurations may require secure transmission in order to function correctly.
The *Protocol* parameter may be optionally set to a required sub-protocol (sent with the *Sec-WebSocket-Protocol* header in the WebSocket handshake). If the server does not indicate it supports the chosen sub-protocol, the connection will fail to be established. This can be used to prevent the client connecting to WebSocket servers that do not understand your application's specific messages.

Send binary

Send the contents of a Binary Data object as a binary message to the server. This is ignored if the connection is not currently open.

Send text

Send a text string to the server. This is ignored if the connection is not currently open.

WebSocket expressions

CloseCode

In the *On closed* trigger, returns the numeric code of the close reason. This can be one of the standard-specified return values, or a user-defined value.

CloseReason

In the *On closed* trigger, returns a string describing the reason the connection was closed. This is optional and may be empty.

ErrorMsg

In *On error*, the error message text.

MessageText

In *On text message*, the text content of the message just received from the server.