NAME

libcurl-ws - WebSocket interface overview

DESCRIPTION

The WebSocket interface provides functions for receiving and sending WebSocket data.

INCLUDE

You still only include <curl/curl.h> in your code.

SETUP

WebSocket is also often known as *WebSockets*, in plural. It is done by upgrading a regular HTTP(S) GET request to a WebSocket connection.

WebSocket is a TCP-like message-based communication protocol done over HTTP, specified in RFC 6455.

To initiate a WebSocket session with libcurl, setup an easy handle to use a URL with a "WS://" or "WSS://" scheme. "WS" is for cleartext communication over HTTP and "WSS" is for doing WebSocket securely over HTTPS.

A WebSocket request is done as an HTTP/1 GET request with an "Upgrade WebSocket" request header field. When the upgrade is accepted by the server, it responds with a 101 Switching and then the client can speak WebSocket with the server. The communication can happen in both directions at the same time.

MESSAGES

WebSocket communication is message based. That means that both ends send and receive entire messages, not streams like TCP. A WebSocket message is sent over the wire in one or more frames. Each frame in a message can have a size up to 2^63 bytes.

libcurl delivers WebSocket data as frame fragments. It might send a whole frame, but it might also deliver them in pieces depending on size and network patterns. It makes sure to provide the API user about the exact specifics about the fragment: type, offset, size and how much data there is pending to arrive for the same frame.

A message has an unknown size until the last frame header for the message has been received since only frames have set sizes.

Raw mode

libcurl can be told to speak WebSocket in "raw mode" by setting the **CURLWS_RAW_MODE** bit to

the *CURLOPT_WS_OPTIONS(3)* option.

Raw WebSocket means that libcurl passes on the data from the network without parsing it leaving that entirely to the application. This mode assumes that the user of this knows WebSocket and can parse and figure out the data all by itself.

This mode is intended for applications that already have a WebSocket parser/engine that want to switch over to use libcurl for enabling WebSocket, but keep parts of the existing software architecture.

PING

WebSocket is designed to allow long-lived sessions and in order to keep the connections alive, both ends can send PING messages for the other end to respond with a PONG.

libcurl automatically responds to server PING messages with a PONG. It does not send any PING messages automatically.

MODELS

Because of the many different ways WebSocket can be used, which is much more flexible than limited to plain downloads or uploads, libcurl offers two different API models to use it:

1. Using a write callback with *CURLOPT_WRITEFUNCTION(3)* much like other downloads for when the traffic is download oriented.

2. Using *CURLOPT_CONNECT_ONLY(3)* and use the WebSocket recv/send functions.

Callback model

When a write callback is set and a WebSocket transfer is performed, the callback is called to deliver all WebSocket data that arrives.

The callback can then call *curl_ws_meta(3)* to learn about the details of the incoming data fragment.

CONNECT_ONLY model

By setting *CURLOPT_CONNECT_ONLY(3)* to **2L**, the transfer only establishes and setups the WebSocket communication and then returns control back to the application.

Once such a setup has been successfully performed, the application can proceed and use *curl_ws_recv(3)* and *curl_ws_send(3)* freely to exchange WebSocket messages with the server.

AVAILABILITY

The WebSocket API was introduced as experimental in 7.86.0 and is still experimental today.

It is only built-in if explicitly opted in at build time. We discourage use of the WebSocket API in production because of its experimental state. We might change API, ABI and behavior before this "goes live".

SEE ALSO

curl_easy_init(3), **curl_ws_meta**(3), **curl_ws_recv**(3), **curl_ws_send**(3), **CURLOPT_CONNECT_ONLY**(3), **CURLOPT_WRITEFUNCTION**(3), **CURLOPT_WS_OPTIONS**(3)