NAME

curl_ws_recv − receive WebSocket data

SYNOPSIS

```
#include <curl/curl.h>

CURLcode curl_ws_recv(CURL *curl, void *buffer, size_t buflen,
                      size_t *recv, const struct curl_ws_frame **meta);
```

DESCRIPTION

Retrieves as much as possible of a received WebSocket frame into the *buffer*, but not more than *buflen* bytes. *recv* is set to the number of bytes actually stored.

If the function call is successful, the *meta* pointer gets set to point to a *const struct curl_ws_frame* that contains information about the received data. That struct must not be freed and its contents must not be relied upon anymore once another WebSocket function is called. See *curl_ws_meta(3)* for more details on that struct.

The application must check *meta−>bytesleft* to determine whether the complete frame has been received. If more payload is pending, the application must call this function again with an updated *buffer* and *buflen* to resume receiving. This may for example happen when the data does not fit into the provided buffer or when not all frame data has been delivered over the network yet.

If the application wants to read the metadata without consuming any payload, it may call this function with a *buflen* of zero. Setting *buffer* to a NULL pointer is permitted in this case. Note that frames without payload are consumed by this action.

If the received message consists of multiple fragments, the *CURLWS_CONT* bit is set in all frames except the final one. The appropriate *CURLWS_TEXT* or *CURLWS_BINARY* flag is set in every frame, regardless whether it is the first fragment, an intermediate fragment or the final fragment. The application is responsible for reassembling fragmented messages. Special care must be taken to correctly handle control frames (i.e. CLOSE, PING and PONG) arriving in between consecutive fragments of a fragmented TEXT or BINARY message. See *curl_ws_meta(3)* for more details on *CURLWS_CONT*.

The WebSocket protocol consists of *messages* that can be delivered over the wire as one or more *frames* − but since a frame can be too large to buffer in memory, libcurl may need to deliver partial frames to the application. Fragments, or chunks, of frames.

PROTOCOLS

This functionality affects ws only

EXAMPLE

```
int main(void)
{
  char buffer[256];
  size_t offset = 0;
  CURLcode result = CURLE_OK;
  CURL *curl = curl_easy_init();

  curl_easy_setopt(curl, CURLOPT_URL, "wss://example.com/");
  curl_easy_setopt(curl, CURLOPT_CONNECT_ONLY, 2L);
  /* start HTTPS connection and upgrade to WSS, then return control */
  curl_easy_perform(curl);

  /* Note: This example neglects fragmented messages.
     (CURLWS_CONT bit) A real application must handle them
     appropriately. */
```

```
 while(!result) {
  size_t recv;
  const struct curl_ws_frame *meta;
  result = curl_ws_recv(curl, buffer + offset,
                        sizeof(buffer) - offset, &recv,
                        &meta);
  offset += recv;

  if(result == CURLE_OK) {
   if(meta->bytesleft == 0)
    break; /* finished receiving */
   if(meta->bytesleft > (curl_off_t)(sizeof(buffer) - offset))
    result = CURLE_TOO_LARGE;
  }

  if(result == CURLE_AGAIN)
   /* in real application: wait for socket here, e.g. using select() */
   result = CURLE_OK;
 }

 curl_easy_cleanup(curl);
 return (int)result;
}
```

AVAILABILITY

Added in curl 7.86.0

RETURN VALUE

This function returns a CURLcode indicating success or error.

CURLE_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*. If *CURLOPT_ERRORBUFFER(3)* was set with *curl_easy_setopt(3)* there can be an error message stored in the error buffer when non−zero is returned.

Returns **CURLE_GOT_NOTHING** if the associated connection is closed.

Instead of blocking, the function returns **CURLE_AGAIN**. The correct behavior is then to wait for the socket to signal readability before calling this function again.

Any other non−zero return value indicates an error. See the *libcurl−errors(3)* man page for the full list with descriptions.

Returns **CURLE_GOT_NOTHING** if the associated connection is closed.

SEE ALSO

curl_easy_getinfo(3), **curl_easy_perform**(3), **curl_easy_setopt**(3), **curl_ws_send**(3), **libcurl-ws**(3)