NAME

CURLOPT_HAPPY_EYEBALLS_TIMEOUT_MS − timing of connect attempts

SYNOPSIS

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

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_HAPPY_EYEBALLS_TIMEOUT_MS,
                          long timeout);
```

DESCRIPTION

Happy eyeballs is an algorithm that controls connecting to a host that resolves to more than one IP address. A common setup is to expose an IPv4 and IPv6 address (dual−stack). Other host offer a range of addresses for one or both stacks.

IP Addresses

When curl is built with IPv6 support, it attempts to connect to IPv6 first, when available. When that fails, another connect attempt for the first IPv4 address (again, if available) is started. Should that fail, the next IPv6 address is used, then the next IPv4, etc. If there are only addresses for one stack, those are tried one after the other.

When there is neither a positive nor negative response to an attempt, another attempt is started after *timeout* has passed. Then another, after *timeout* has passed again. As long as there are addresses available.

When all addresses have been tried and failed, the transfer fails. All attempts are aborted after *CURLOPT_CONNECTTIMEOUT_MS(3)* has passed, counted from the first attempt onward.

The range of suggested useful values for *timeout* is limited. Happy Eyeballs RFC 6555 says "It is RECOMMENDED that connection attempts be paced 150−250 ms apart to balance human factors against network load." libcurl currently defaults to 200 ms. Firefox and Chrome currently default to 300 ms.

As an example, for a host that resolves to 'a1_v4, a2_v4, a3_v6, a4_v6' curl opens a socket to 'a3_v6' first. When that does not report back, it opens another socket to 'a1_v4' after 200ms. The first socket is left open and might still succeed. When 200ms have gone by again, a socket for 'a4_v6' is opened. 200ms later, 'a2_v4' is tried.

At this point, there are 4 sockets open (unless the network has reported anything back). That took 3 times the happy eyeballs timeout, so 600ms in the default setting. When any of those four report a success, that socket is used for the transfer and the other three are closed.

There is a limit on the number of sockets opened for connect attempts. When that limit is reached and more addresses are available, the oldest attempt is discarded. This limit is currently 6. With the default happy eyeballs timeout of 200ms, this closes attempts after 1.2 seconds *as long as there are more addresses to try*.

There are situations where connect attempts fail, but the failure is considered being inconclusive. The QUIC protocol may encounter this. When a QUIC server restarts, it may send replies indicating that it is not accepting new connections right now, but maybe later.

Such "inclusive" connect attempt failures cause a restart of the attempt, with the same address on a new socket, closing the previous one. Repeatedly until *CURLOPT_CONNECTTIMEOUT_MS(3)* strikes.

HTTPS When connection with the HTTPS protocol to a host that may talk HTTP/3, HTTP/2 or HTTP/1.1, curl applies a similar happy eyeballs strategy when attempting these versions.

When HTTPS only involves a TCP connection, the versions are negotiated via ALPN, the TLS extension, in a single connect. Since HTTP/3 runs on QUIC (which runs on UDP), it requires a separate connect attempt.

The HTTP/3 attempt is started first and, after *timeout* expires, the HTTP/2 (or 1.1) attempt is started in parallel.

DEFAULT

CURL_HET_DEFAULT (currently defined as 200L)

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
int main(void)
{
  CURL *curl = curl_easy_init();
  if(curl) {
    CURLcode result;
    curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");

    /* Set the timeout to 300 milliseconds */
    curl_easy_setopt(curl, CURLOPT_HAPPY_EYEBALLS_TIMEOUT_MS, 300L);

    result = curl_easy_perform(curl);

    /* always cleanup */
    curl_easy_cleanup(curl);
  }
}
```

AVAILABILITY

Added in curl 7.59.0

RETURN VALUE

curl_easy_setopt(3) returns a CURLcode indicating success or error.

CURLE_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*.

SEE ALSO

CURLOPT_CONNECTTIMEOUT_MS(3), **CURLOPT_LOW_SPEED_LIMIT**(3), **CURLOPT_TIMEOUT**(3)