

## NAME

`curl_easy_recv` - receives raw data on an "easy" connection

## SYNOPSIS

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

```
CURLcode curl_easy_recv( CURL *curl, void *buffer, size_t buflen, size_t *n);
```

## DESCRIPTION

This function receives raw data from the established connection. You may use it together with `curl_easy_send(3)` to implement custom protocols using libcurl. This functionality can be particularly useful if you use proxies and/or SSL encryption: libcurl will take care of proxy negotiation and connection set-up.

**buffer** is a pointer to your buffer that will get the received data. **buflen** is the maximum amount of data you can get in that buffer. The variable **n** points to will receive the number of received bytes.

To establish the connection, set **CURLOPT\_CONNECT\_ONLY** option before calling `curl_easy_perform(3)`. Note that `curl_easy_recv(3)` does not work on connections that were created without this option.

You must ensure that the socket has data to read before calling `curl_easy_recv(3)`, otherwise the call will return **CURLE\_AGAIN** - the socket is used in non-blocking mode internally. Use `curl_easy_getinfo(3)` with **CURLINFO\_LASTSOCKET** to obtain the socket; use your operating system facilities like `select(2)` to check if it has any data you can read.

## AVAILABILITY

Added in 7.18.2.

## RETURN VALUE

On success, returns **CURLE\_OK**, stores the received data into **buffer**, and the number of bytes it actually read into **\*n**.

On failure, returns the appropriate error code.

If there is no data to read, the function returns **CURLE\_AGAIN**. Use your operating system facilities to wait until the data is ready, and retry.

## EXAMPLE

See `sendrecv.c` in `docs/examples` directory for usage example.

## SEE ALSO

`curl_easy_setopt(3)`, `curl_easy_perform(3)`, `curl_easy_getinfo(3)`, `curl_easy_send(3)`