NAME

CURLMOPT_PUSHFUNCTION − callback that approves or denies server pushes

SYNOPSIS

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

int curl_push_callback(CURL *parent,
                       CURL *easy,
                       size_t num_headers,
                       struct curl_pushheaders *headers,
                       void *clientp);

CURLMcode curl_multi_setopt(CURLM *handle, CURLMOPT_PUSHFUNCTION,
                            curl_push_callback func);
```

DESCRIPTION

This callback gets called when a new HTTP/2 stream is being pushed by the server (using the PUSH_PROMISE frame). If no push callback is set, all offered pushes are denied automatically.

CALLBACK DESCRIPTION

The callback gets its arguments like this:

parent is the handle of the stream on which this push arrives. The new handle has been duplicated from the parent, meaning that it has gotten all its options inherited. It is then up to the application to alter any options if desired.

easy is a newly created handle that represents this upcoming transfer.

num_headers is the number of name+value pairs that was received and can be accessed

headers is a handle used to access push headers using the accessor functions described below. This only accesses and provides the PUSH_PROMISE headers, the normal response headers are provided in the header callback as usual.

clientp is the pointer set with *CURLMOPT_PUSHDATA(3)*

If the callback returns CURL_PUSH_OK, the new easy handle is added to the multi handle, the callback must not do that by itself.

The callback can access PUSH_PROMISE headers with two accessor functions. These functions can only be used from within this callback and they can only access the PUSH_PROMISE headers: *curl_pushheader_byname(3)* and *curl_pushheader_bynum(3)*. The normal response headers are passed to the header callback for pushed streams like for normal streams.

The header fields can also be accessed with *curl_easy_header(3)*, introduced in later libcurl versions.

CALLBACK RETURN VALUE

CURL_PUSH_OK (0)
: The application has accepted the stream and it can now start receiving data, the ownership of the curl handle has been taken over by the application.

CURL_PUSH_DENY (1)
: The callback denies the stream and no data reaches the application, the easy handle is destroyed by libcurl.

CURL_PUSH_ERROROUT (2)
: Returning this code rejects the pushed stream and returns an error back on the parent stream making it get closed with an error. (Added in 7.72.0)

* All other return codes are reserved for future use.

DEFAULT

NULL, no callback

PROTOCOLS

This functionality affects http only

EXAMPLE

```
#include <string.h>

/* only allow pushes for filenames starting with "push-" */
int push_callback(CURL *parent,
                  CURL *easy,
                  size_t num_headers,
                  struct curl_pushheaders *headers,
                  void *clientp)
{
 char *headp;
 int *transfers = (int *)clientp;
 FILE *out;
 headp = curl_pushheader_byname(headers, ":path");
 if(headp && !strncmp(headp, "/push-", 6)) {
  fprintf(stderr, "The PATH is %s\n", headp);

  /* save the push here */
  out = fopen("pushed-stream", "wb");

  /* write to this file */
  curl_easy_setopt(easy, CURLOPT_WRITEDATA, out);

  (*transfers)++; /* one more */

  return CURL_PUSH_OK;
 }
 return CURL_PUSH_DENY;
}

int main(void)
{
 int counter;
 CURLM *multi = curl_multi_init();
 curl_multi_setopt(multi, CURLMOPT_PUSHFUNCTION, push_callback);
 curl_multi_setopt(multi, CURLMOPT_PUSHDATA, &counter);
}
```

AVAILABILITY

Added in curl 7.44.0

RETURN VALUE

curl_multi_setopt(3) returns a CURLMcode indicating success or error.

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

SEE ALSO

CURLMOPT_PIPELINING(3), **CURLMOPT_PUSHDATA**(3), **CURLOPT_PIPEWAIT**(3), **RFC**7540