NAME

CURLMOPT_PUSHFUNCTION − callback that approves or denies server pushes

SYNOPSIS

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

char *curl_pushheader_bynum(struct curl_pushheaders *h, size_t num);
char *curl_pushheader_byname(struct curl_pushheaders *h, const char *name);

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

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 will be 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 duphandle()d 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 will be provided in the header callback as usual.

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

If the callback returns CURL_PUSH_OK, the ’easy’ handle will be 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. The normal response headers will be passed to the header callback for pushed streams just as for normal streams.

curl_pushheader_bynum
: Returns the header at index ’num’ (or NULL). The returned pointer points to a "name:value" string that will be freed when this callback returns.

curl_pushheader_byname
: Returns the value for the given header name (or NULL). This is a shortcut so that the application doesn’t have to loop through all headers to find the one it is interested in. The data pointed will be freed when this callback returns.

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 for this will reach the application, the easy handle will be destroyed by libcurl.

* All other return codes are reserved for future use.

DEFAULT

NULL, no callback

PROTOCOLS

HTTP(S) (HTTP/2 only)

EXAMPLE

```
/* only allow pushes for file names starting with "push-" */
int push_callback(CURL *parent,
                  CURL *easy,
                  size_t num_headers,
                  struct curl_pushheaders *headers,
                  void *userp)
{
 char *headp;
 int *transfers = (int *)userp;
 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;
}

curl_multi_setopt(multi, CURLMOPT_PUSHFUNCTION, push_callback);
curl_multi_setopt(multi, CURLMOPT_PUSHDATA, &counter);
```

AVAILABILITY

Added in 7.44.0

RETURN VALUE

Returns CURLM_OK if the option is supported, and CURLM_UNKNOWN_OPTION if not.

SEE ALSO

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