

**NAME**

curl\_url\_set - set a URL part

**SYNOPSIS**

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

```
CURLUcode curl_url_set(CURLU *url,  
                        CURLUPart part,  
                        const char *content,  
                        unsigned int flags);
```

**DESCRIPTION**

The *url* handle to work on, passed in as the first argument, must be a handle previously created by *curl\_url(3)* or *curl\_url\_dup(3)*.

This function sets or updates individual URL components, or parts, held by the URL object the handle identifies.

The *part* argument should identify the particular URL part (see list below) to set or change, with *content* pointing to a null-terminated string with the new contents for that URL part. The contents should be in the form and encoding they would use in a URL: URL encoded.

When setting a part in the URL object that was previously already set, it replaces the data that was previously stored for that part with the new *content*.

The caller does not have to keep *content* around after a successful call as this function copies the content.

Setting a part to a NULL pointer removes that part's contents from the *CURLU* handle.

This function has an 8 MB maximum length limit for all provided input strings. In the real world, excessively long fields in URLs cause problems even if this function accepts them.

When setting or updating contents of individual URL parts, *curl\_url\_set(3)* might accept data that would not be otherwise possible to set in the string when it gets populated as a result of a full URL parse. Beware. If done so, extracting a full URL later on from such components might render an invalid URL.

The *flags* argument is a bitmask with independent features.

## PARTS

### CURLUPART\_URL

Allows the full URL of the handle to be replaced. If the handle already is populated with a URL, the new URL can be relative to the previous.

When successfully setting a new URL, relative or absolute, the handle contents is replaced with the components of the newly set URL.

Pass a pointer to a null-terminated string to the *url* parameter. The string must point to a correctly formatted "RFC 3986+" URL or be a NULL pointer. The URL parser only understands and parses the subset of URLs that are "hierarchical" and therefore contain a *://* separator - not the ones that are normally specified with only a colon separator.

By default this API only parses URLs using schemes for protocols that are supported built-in. To make libcurl parse URLs generically even for schemes it does not know about, the **CURLU\_NON\_SUPPORT\_SCHEME** flags bit must be set. Otherwise, this function returns *CURLUE\_UNSUPPORTED\_SCHEME* for URL schemes it does not recognize.

Unless *CURLU\_NO\_AUTHORITY* is set, a blank hostname is not allowed in the URL.

### CURLUPART\_SCHEME

Scheme cannot be URL decoded on set. libcurl only accepts setting schemes up to 40 bytes long.

### CURLUPART\_USER

If only the user part is set and not the password, the URL is represented with a blank password.

### CURLUPART\_PASSWORD

If only the password part is set and not the user, the URL is represented with a blank user.

### CURLUPART\_OPTIONS

The options field is an optional field that might follow the password in the userinfo part. It is only recognized/used when parsing URLs for the following schemes: pop3, smtp and imap. This function however allows users to independently set this field.

### CURLUPART\_HOST

The hostname. If it is International Domain Name (IDN) the string must then be encoded as your locale says or UTF-8 (when WinIDN is used). If it is a bracketed IPv6 numeric address it may contain a zone id (or you can use *CURLUPART\_ZONEID*).

Note that if you set an IPv6 address, it gets ruined and causes an error if you also set the

`CURLU_URLENCODE` flag.

Unless `CURLU_NO_AUTHORITY` is set, a blank hostname is not allowed to set.

#### `CURLUPART_ZONEID`

If the hostname is a numeric IPv6 address, this field can also be set.

#### `CURLUPART_PORT`

The port number cannot be URL encoded on set. The given port number is provided as a string and the decimal number in it must be between 0 and 65535. Anything else returns an error.

#### `CURLUPART_PATH`

If a path is set in the URL without a leading slash, a slash is prepended automatically.

#### `CURLUPART_QUERY`

The query part gets spaces converted to pluses when asked to URL encode on set with the `CURLU_URLENCODE` bit.

If used together with the `CURLU_APPENDQUERY` bit, the provided part is appended on the end of the existing query.

The question mark in the URL is not part of the actual query contents.

#### `CURLUPART_FRAGMENT`

The hash sign in the URL is not part of the actual fragment contents.

### **FLAGS**

The flags argument is zero, one or more bits set in a bitmask.

#### `CURLU_APPENDQUERY`

Can be used when setting the `CURLUPART_QUERY` component. The provided new part is then appended at the end of the existing query - and if the previous part did not end with an ampersand (&), an ampersand gets inserted before the new appended part.

When `CURLU_APPENDQUERY` is used together with `CURLU_URLENCODE`, the first '=' symbol is not URL encoded.

#### `CURLU_NON_SUPPORT_SCHEME`

If set, allows `curl_url_set(3)` to set a non-supported scheme. It then of course cannot know if the provided scheme is a valid one or not.

**CURLU\_URL\_ENCODE**

When set, *curl\_url\_set(3)* URL encodes the part on entry, except for **scheme**, **port** and **URL**.

When setting the path component with URL encoding enabled, the slash character is skipped.

The query part gets space-to-plus converted before the URL conversion is applied.

This URL encoding is charset unaware and converts the input in a byte-by-byte manner.

**CURLU\_DEFAULT\_SCHEME**

If set, allows the URL to be set without a scheme and then sets that to the default scheme: HTTPS.

Overrides the *CURLU\_GUESS\_SCHEME* option if both are set.

**CURLU\_GUESS\_SCHEME**

If set, allows the URL to be set without a scheme and it instead "guesses" which scheme that was intended based on the hostname. If the outermost subdomain name matches DICT, FTP, IMAP, LDAP, POP3 or SMTP then that scheme is used, otherwise it picks HTTP. Conflicts with the *CURLU\_DEFAULT\_SCHEME* option which takes precedence if both are set.

If guessing is not allowed and there is no default scheme set, trying to parse a URL without a scheme returns error.

If the scheme ends up set as a result of guessing, i.e. it is not actually present in the parsed URL, it can later be figured out by using the **CURLU\_NO\_GUESS\_SCHEME** flag when subsequently getting the URL or the scheme with *curl\_url\_get(3)*.

**CURLU\_NO\_AUTHORITY**

If set, skips authority checks. The RFC allows individual schemes to omit the host part (normally the only mandatory part of the authority), but libcurl cannot know whether this is permitted for custom schemes. Specifying the flag permits empty authority sections, similar to how file scheme is handled.

**CURLU\_PATH\_AS\_IS**

When set for **CURLUPART\_URL**, this skips the normalization of the path. That is the procedure where libcurl otherwise removes sequences of dot-slash and dot-dot etc. The same option used for transfers is called *CURLOPT\_PATH\_AS\_IS(3)*.

**CURLU\_ALLOW\_SPACE**

If set, the URL parser allows space (ASCII 32) where possible. The URL syntax does normally not allow spaces anywhere, but they should be encoded as %20 or '+'. When spaces are allowed, they

are still not allowed in the scheme. When space is used and allowed in a URL, it is stored as-is unless *CURLU\_URL\_ENCODE* is also set, which then makes libcurl URL encode the space before stored. This affects how the URL is constructed when *curl\_url\_get(3)* is subsequently used to extract the full URL or individual parts. (Added in 7.78.0)

#### CURLU\_DISALLOW\_USER

If set, the URL parser does not accept embedded credentials for the **CURLUPART\_URL**, and instead returns **CURLUE\_USER\_NOT\_ALLOWED** for such URLs.

### PROTOCOLS

This functionality affects all supported protocols

### EXAMPLE

```
int main(void)
{
    CURLUcode rc;
    CURLU *url = curl_url();
    rc = curl_url_set(url, CURLUPART_URL, "https://example.com", 0);
    if(!rc) {
        /* change it to an FTP URL */
        rc = curl_url_set(url, CURLUPART_SCHEME, "ftp", 0);
    }
    curl_url_cleanup(url);
}
```

### AVAILABILITY

Added in curl 7.78.0

### RETURN VALUE

Returns a *CURLUcode* error value, which is **CURLUE\_OK** (0) if everything went fine. See the *libcurl-errors(3)* man page for the full list with descriptions.

The input string passed to *curl\_url\_set(3)* must be shorter than eight million bytes. Otherwise this function returns **CURLUE\_MALFORMED\_INPUT**.

If this function returns an error, no URL part is set.

### SEE ALSO

**CURLOPT\_CURLU(3)**, **curl\_url(3)**, **curl\_url\_cleanup(3)**, **curl\_url\_dup(3)**, **curl\_url\_get(3)**, **curl\_url\_strerror(3)**