

My Project

Generated by Doxygen 1.9.8

1 Curling: Modern C++ libcurl Wrapper	1
1.1 Features	1
1.2 Example	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 curling Namespace Reference	11
6.1.1 Detailed Description	12
6.2 curling::detail Namespace Reference	12
6.2.1 Detailed Description	12
7 Class Documentation	13
7.1 curling::CurlHandleDeleter Struct Reference	13
7.1.1 Detailed Description	13
7.2 curling::CurlingException Class Reference	13
7.2.1 Detailed Description	14
7.3 curling::CurlMimeDeleter Struct Reference	14
7.4 curling::CurlSlistDeleter Struct Reference	14
7.5 curling::HeaderException Class Reference	15
7.5.1 Detailed Description	16
7.6 curling::InitializationException Class Reference	16
7.6.1 Detailed Description	17
7.7 curling::LogicException Class Reference	17
7.7.1 Detailed Description	18
7.8 curling::MimeException Class Reference	18
7.8.1 Detailed Description	19
7.9 curling::Request Class Reference	20
7.9.1 Detailed Description	21
7.9.2 Member Enumeration Documentation	22
7.9.2.1 HttpVersion	22
7.9.2.2 Method	22
7.9.3 Constructor & Destructor Documentation	22
7.9.3.1 Request()	22

7.9.4 Member Function Documentation	23
7.9.4.1 addArg()	23
7.9.4.2 addFormField()	23
7.9.4.3 addFormFile()	24
7.9.4.4 addHeader()	24
7.9.4.5 downloadToFile()	24
7.9.4.6 enableVerbose()	25
7.9.4.7 send()	25
7.9.4.8 setAuthToken()	25
7.9.4.9 setBody()	26
7.9.4.10 setConnectTimeout()	26
7.9.4.11 setCookiePath()	26
7.9.4.12 setFollowRedirects()	27
7.9.4.13 setHttpAuth()	27
7.9.4.14 setHttpAuthMethod()	27
7.9.4.15 setMethod()	28
7.9.4.16 setProgressCallback()	28
7.9.4.17 setProxy()	28
7.9.4.18 setProxyAuth()	29
7.9.4.19 setProxyAuthMethod()	29
7.9.4.20 setRawOption()	29
7.9.4.21 setTimeout()	30
7.9.4.22 setURL()	30
7.9.4.23 setUserAgent()	30
7.10 curling::RequestException Class Reference	31
7.10.1 Detailed Description	32
7.11 curling::Response Struct Reference	32
7.11.1 Detailed Description	32
8 File Documentation	33
8.1 curling.hpp	33
Index	37

Chapter 1

Curling: Modern C++ libcurl Wrapper

Curling is a lightweight, header-only C++17 wrapper around libcurl for making HTTP requests with a modern design and safe resource handling.

1.1 Features

- RAII and smart-pointer-based resource management
- MIME support for file uploads
- Fluent API for intuitive chaining
- Proxy and authentication support
- Persistent cookie management

1.2 Example

```
curling::Request req;  
req.setMethod(curling::Request::Method::POST)  
  .setURL("https://example.com")  
  .addHeader("Content-Type: application/json")  
  .setBody(R"({"key": "value"})");  
curling::Response res = req.send();  
std::cout << res.toString();
```


Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

curling	11
curling::detail	
Util/helper section	12

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

curling::CurlHandleDeleter	13
curling::CurlMimeDeleter	14
curling::CurlSlistDeleter	14
curling::Request	20
curling::Response	32
std::runtime_error	
curling::CurlingException	13
curling::HeaderException	15
curling::InitializationException	16
curling::LogicException	17
curling::MimeException	18
curling::RequestException	31

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

curling::CurlHandleDeleter	13
curling::CurlingException	
Base exception class for Curling errors	13
curling::CurlMimeDeleter	14
curling::CurlSlistDeleter	14
curling::HeaderException	
Thrown when header operations fail	15
curling::InitializationException	
Thrown when curl initialization fails	16
curling::LogicException	
Thrown when library logic prohibits an operation	17
curling::MimeException	
Thrown when MIME operations fail	18
curling::Request	
Provides a fluent wrapper for HTTP requests via libcurl	20
curling::RequestException	
Thrown when a request operation fails	31
curling::Response	
Represents an HTTP response	32

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

include/ curling.hpp	33
--	----

Chapter 6

Namespace Documentation

6.1 curling Namespace Reference

Namespaces

- namespace [detail](#)
Util/helper section.

Classes

- struct [CurlHandleDeleter](#)
- class [CurlingException](#)
Base exception class for Curling errors.
- struct [CurlMimeDeleter](#)
- struct [CurlSlistDeleter](#)
- class [HeaderException](#)
Thrown when header operations fail.
- class [InitializationException](#)
Thrown when curl initialization fails.
- class [LogicException](#)
Thrown when library logic prohibits an operation.
- class [MimeException](#)
Thrown when MIME operations fail.
- class [Request](#)
Provides a fluent wrapper for HTTP requests via libcurl.
- class [RequestException](#)
Thrown when a request operation fails.
- struct [Response](#)
Represents an HTTP response.

Typedefs

- using **CurlIPtr** = std::unique_ptr< CURL, [CurlHandleDeleter](#) >
- using **CurlSlistPtr** = std::unique_ptr< curl_slist, [CurlSlistDeleter](#) >
- using **CurlMimePtr** = std::unique_ptr< curl_mime, [CurlMimeDeleter](#) >

Functions

- `std::string version ()`

Variables

- `constexpr int version_major = 1`
- `constexpr int version_minor = 2`
- `constexpr int version_patch = 0`

6.1.1 Detailed Description

Note

If you use `Method::MIME` (multipart POST), you must reset the [Request](#) before switching to another method. Attempting to change it afterward throws `logic_error`.

Curling internally manages `curl_global_init()` and `curl_global_cleanup()` one cleanup per init.

Header keys in [Response::headers](#) are stored in lowercase to support case-insensitive lookup.

Calling `enableVerbose(true)` enables libcurl's verbose output to `stderr`, which is useful for debugging.

Warning

This class is not thread-safe. Do not share a [Request](#) instance across threads. Each thread should use its own [Request](#) object.

6.2 curling::detail Namespace Reference

Util/helper section.

Functions

- `void ensureCurlGlobalInit ()`
- `void maybeCleanupGlobalCurl () noexcept`
- `void trim (std::string &s)`
- `void toLowerCase (std::string &s)`
- `size_t WriteCallback (void *contents, size_t size, size_t nmemb, void *userp)`
- `size_t HeaderCallback (char *buffer, size_t size, size_t nitems, void *userdata)`
- `int ProgressCallbackBridge (void *clientp, curl_off_t dltotal, curl_off_t dlnow, curl_off_t ultotal, curl_off_t ulnow)`

Variables

- `std::once_flag curlGlobalInitFlag`
- `std::mutex curlGlobalMutex`
- `int instanceCount = 0`

6.2.1 Detailed Description

Util/helper section.

Note

meant for internal library use only

Chapter 7

Class Documentation

7.1 curling::CurlHandleDeleter Struct Reference

```
#include <curling.hpp>
```

Public Member Functions

- void **operator()** (CURL *h) const noexcept

7.1.1 Detailed Description

SAFETY: RAII deleters for CURL handles

The documentation for this struct was generated from the following file:

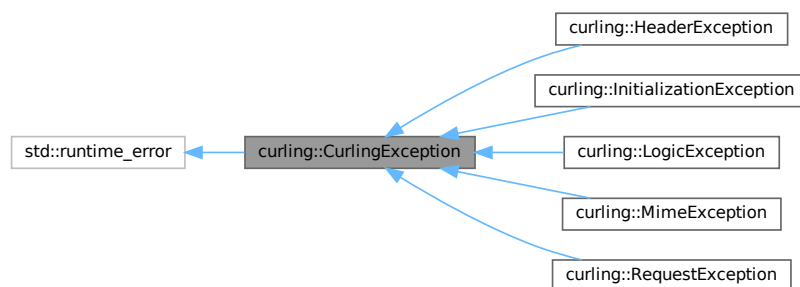
- include/curling.hpp

7.2 curling::CurlingException Class Reference

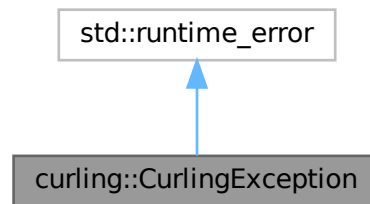
Base exception class for Curling errors.

```
#include <curling.hpp>
```

Inheritance diagram for curling::CurlingException:



Collaboration diagram for `curling::CurlingException`:



Public Member Functions

- **CurlingException** (const std::string &msg)

7.2.1 Detailed Description

Base exception class for Curling errors.

The documentation for this class was generated from the following file:

- include/curling.hpp

7.3 curling::CurlMimeDeleter Struct Reference

Public Member Functions

- void **operator()** (curl_mime *m) const noexcept

The documentation for this struct was generated from the following file:

- include/curling.hpp

7.4 curling::CurlSlistDeleter Struct Reference

Public Member Functions

- void **operator()** (curl_slist *l) const noexcept

The documentation for this struct was generated from the following file:

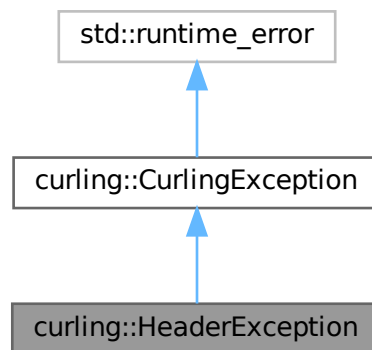
- include/curling.hpp

7.5 `curling::HeaderException` Class Reference

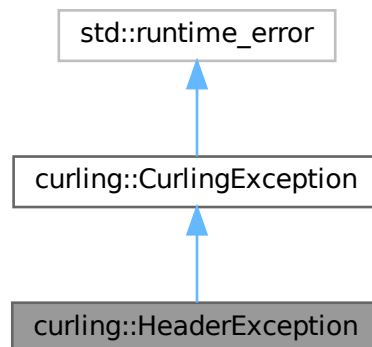
Thrown when header operations fail.

```
#include <curling.hpp>
```

Inheritance diagram for `curling::HeaderException`:



Collaboration diagram for `curling::HeaderException`:



Public Member Functions

- **`HeaderException`** (const `std::string` &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **`CurlingException`** (const `std::string` &msg)

7.5.1 Detailed Description

Thrown when header operations fail.

The documentation for this class was generated from the following file:

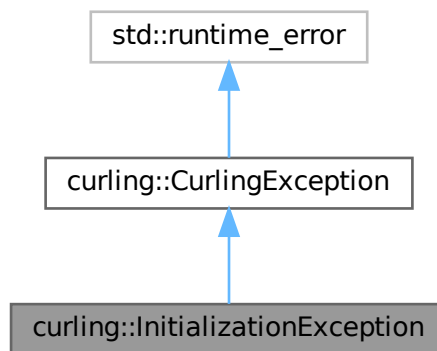
- include/curling.hpp

7.6 curling::InitializationException Class Reference

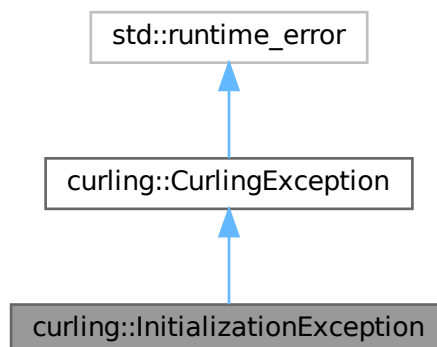
Thrown when curl initialization fails.

```
#include <curling.hpp>
```

Inheritance diagram for curling::InitializationException:



Collaboration diagram for curling::InitializationException:



Public Member Functions

- **InitializationException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.6.1 Detailed Description

Thrown when curl initialization fails.

The documentation for this class was generated from the following file:

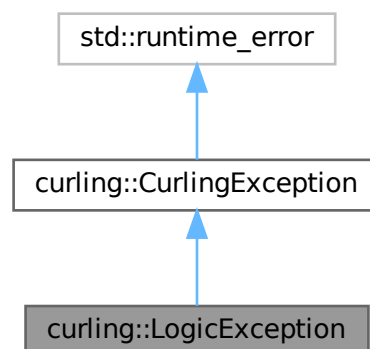
- include/curling.hpp

7.7 curling::LogicException Class Reference

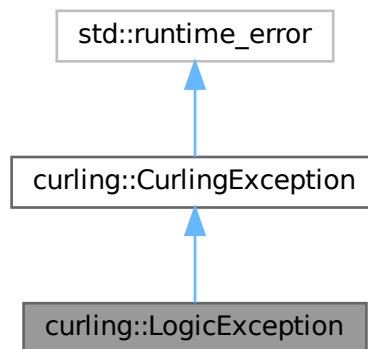
Thrown when library logic prohibits an operation.

```
#include <curling.hpp>
```

Inheritance diagram for curling::LogicException:



Collaboration diagram for `curling::LogicException`:



Public Member Functions

- **LogicException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.7.1 Detailed Description

Thrown when library logic prohibits an operation.

The documentation for this class was generated from the following file:

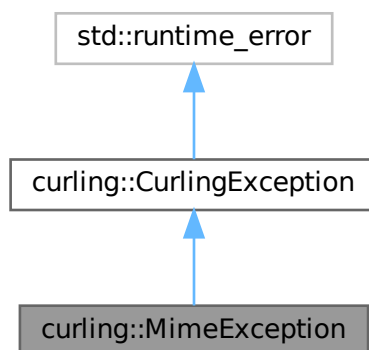
- `include/curling.hpp`

7.8 `curling::MimeException` Class Reference

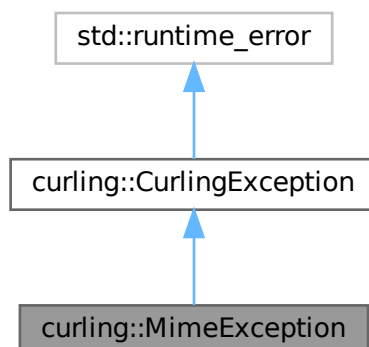
Thrown when MIME operations fail.

```
#include <curling.hpp>
```

Inheritance diagram for curling::MimeException:



Collaboration diagram for curling::MimeException:



Public Member Functions

- **MimeException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.8.1 Detailed Description

Thrown when MIME operations fail.

The documentation for this class was generated from the following file:

- include/curling.hpp

7.9 curling::Request Class Reference

Provides a fluent wrapper for HTTP requests via libcurl.

```
#include <curling.hpp>
```

Public Types

- enum class [Method](#) {
[GET](#) , [POST](#) , [PUT](#) , [DEL](#) ,
[PATCH](#) , [HEAD](#) , [MIME](#) }
Supported HTTP methods.
- enum class [AuthMethod](#) { **BASIC** = CURLAUTH_BASIC , **NTLM** = CURLAUTH_NTLM , **DIGEST** = CURLAUTH_DIGEST }
HTTP authentication schemes.
- enum class [HttpVersion](#) { [DEFAULT](#) , [HTTP_1_1](#) , [HTTP_2](#) , [HTTP_3](#) }
Specifies the HTTP version to be used for the request.
- using **ProgressCallback** = std::function< bool(curl_off_t dltotal, curl_off_t dlnow, curl_off_t ultotal, curl_off_t ulnow)>

Public Member Functions

- [Request](#) ()
Constructor initializes curl global state.
- [~Request](#) () noexcept
Destructor cleans up curl state if last instance.
- **Request** ([Request](#) &&) noexcept
- [Request](#) & **operator=** ([Request](#) &&) noexcept
- **Request** (const [Request](#) &)=delete
- [Request](#) & **operator=** (const [Request](#) &)=delete
- [Request](#) & **setProgressCallback** (ProgressCallback cb)
Sets the progress callback function.
- [Request](#) & **setMethod** ([Method](#) m)
Sets the HTTP method for the request.
- [Request](#) & **setURL** (const std::string &url)
Sets the request URL.
- [Request](#) & **setProxy** (const std::string &url)
Enables proxy usage.
- [Request](#) & **setProxyAuth** (const std::string &username, const std::string &password)
Sets credentials for proxy authentication.
- [Request](#) & **setProxyAuthMethod** ([AuthMethod](#) method)
Sets proxy authentication scheme.
- [Request](#) & **setHttpAuth** (const std::string &username, const std::string &password)
Sets credentials for HTTP auth (Basic/Digest/NTLM).
- [Request](#) & **setHttpAuthMethod** ([AuthMethod](#) method)
Sets HTTP authentication scheme.
- [Request](#) & **addArg** (const std::string &key, const std::string &value)
Adds a query parameter to the URL.
- [Request](#) & **addHeader** (const std::string &header)
Adds a custom HTTP header.

- [Request & setBody](#) (const std::string &body)
Sets the body of the request (for POST/PUT/PATCH).
- [Request & downloadToFile](#) (const std::string &path)
Enables download streaming to a file.
- [Request & setTimeout](#) (long seconds)
Sets a timeout for the request (in seconds).
- [Request & setConnectTimeout](#) (long seconds)
Sets connection timeout (in seconds).
- [Request & setFollowRedirects](#) (bool follow)
Enables or disables automatic redirect-following.
- [Request & setAuthToken](#) (const std::string &token)
Adds a Bearer token for Authorization header.
- [Request & setCookiePath](#) (const std::string &path)
Overrides default cookie file for persistence.
- [Request & setUserAgent](#) (const std::string &userAgent)
Sets the User-Agent header.
- [Request & addFormField](#) (const std::string &fieldName, const std::string &value)
Adds a field to multipart/form-data.
- [Request & addFormFile](#) (const std::string &fieldName, const std::string &filePath)
Adds a file to multipart upload.
- [Request & enableVerbose](#) (bool enabled=true)
Enables or disables libcurl verbose output.
- [Response send](#) ()
Executes the HTTP request.
- void **reset** ()
Resets internal state to allow reuse.
- [Request & setHttpVersion](#) ([HttpVersion](#) version)
Set the HTTP protocol version (http1.1, 2 or 3)
- template<typename T >
[Request & setRawOption](#) (CURLOption opt, T value)
Low level access to define curl options.

Friends

- int **detail::ProgressCallbackBridge** (void *clientp, curl_off_t dltotal, curl_off_t dlnow, curl_off_t ultotal, curl_off_t ulnow)

7.9.1 Detailed Description

Provides a fluent wrapper for HTTP requests via libcurl.

7.9.2 Member Enumeration Documentation

7.9.2.1 HttpVersion

```
enum class curling::Request::HttpVersion [strong]
```

Specifies the HTTP version to be used for the request.

Allows explicit control over the HTTP version that libcurl should use when performing a request. If the selected version is not supported by the libcurl build or the server, fallback behavior may occur.

Note

If you request a version not supported by libcurl at runtime, Curling will throw a [LogicException](#) during configuration. If the server doesn't support the version (e.g., HTTP/2), libcurl may fall back to an older version without error.

Enumerator

DEFAULT	Let libcurl automatically negotiate the best supported HTTP version.
HTTP_1↔ _1	Force HTTP/1.1 for all requests.
HTTP_2	Force HTTP/2 (requires libcurl built with nghttp2 support).
HTTP_3	Force HTTP/3 (requires libcurl built with HTTP/3 support, e.g. with quiche or ngtcp2).

7.9.2.2 Method

```
enum class curling::Request::Method [strong]
```

Supported HTTP methods.

Enumerator

GET	Standard GET.
POST	Standard POST.
PUT	PUT.
DEL	DELETE (named DEL to avoid macro clash)
PATCH	PATCH.
HEAD	HEAD request just headers.
MIME	Multipart/form-data POST.

7.9.3 Constructor & Destructor Documentation

7.9.3.1 Request()

```
curling::Request::Request ( )
```

Constructor initializes curl global state.

Exceptions

<i>InitializationException</i>	if initialization fails.
--	--------------------------

7.9.4 Member Function Documentation

7.9.4.1 addArg()

```
Request & curling::Request::addArg (
    const std::string & key,
    const std::string & value )
```

Adds a query parameter to the URL.

Parameters

<i>key</i>	Parameter name.
<i>value</i>	Parameter value.

Returns

*this

7.9.4.2 addFormField()

```
Request & curling::Request::addFormField (
    const std::string & fieldName,
    const std::string & value )
```

Adds a field to multipart/form-data.

Parameters

<i>fieldName</i>	Field name.
<i>value</i>	Field value.

Returns

*this

Exceptions

<i>MimeException</i>	on internal curl errors.
--------------------------------------	--------------------------

7.9.4.3 addFormFile()

```
Request & curling::Request::addFormFile (
    const std::string & fieldName,
    const std::string & filePath )
```

Adds a file to multipart upload.

Parameters

<i>fieldName</i>	Field name.
<i>filePath</i>	Path to file on disk.

Returns

*this

Exceptions

<i>MimeException</i>	on internal curl errors.
--------------------------------------	--------------------------

7.9.4.4 addHeader()

```
Request & curling::Request::addHeader (
    const std::string & header )
```

Adds a custom HTTP header.

Parameters

<i>header</i>	A full header line, e.g. "Accept: application/json".
---------------	--

Returns

*this

7.9.4.5 downloadToFile()

```
Request & curling::Request::downloadToFile (
    const std::string & path )
```

Enables download streaming to a file.

Parameters

<i>path</i>	Local file path for saving response.
-------------	--------------------------------------

Returns

*this

7.9.4.6 enableVerbose()

```
Request & curling::Request::enableVerbose (
    bool enabled = true )
```

Enables or disables libcurl verbose output.

Parameters

<i>enabled</i>	True to enable verbose mode.
----------------	------------------------------

Returns

*this

7.9.4.7 send()

```
Response curling::Request::send ( )
```

Executes the HTTP request.

Returns

[Response](#) object with status, body, headers.

Exceptions

RequestException	on failure.
----------------------------------	-------------

7.9.4.8 setAuthToken()

```
Request & curling::Request::setAuthToken (
    const std::string & token )
```

Adds a Bearer token for Authorization header.

Parameters

<i>token</i>	Bearer token string.
--------------	----------------------

Returns

*this

7.9.4.9 setBody()

```
Request & curling::Request::setBody (
    const std::string & body )
```

Sets the body of the request (for POST/PUT/PATCH).

Parameters

<i>body</i>	Request body content.
-------------	-----------------------

Returns

*this

7.9.4.10 setConnectTimeout()

```
Request & curling::Request::setConnectTimeout (
    long seconds )
```

Sets connection timeout (in seconds).

Parameters

<i>seconds</i>	Timeout in seconds.
----------------	---------------------

Returns

*this

7.9.4.11 setCookiePath()

```
Request & curling::Request::setCookiePath (
    const std::string & path )
```

Overrides default cookie file for persistence.

Parameters

<i>path</i>	File path for storing cookies.
-------------	--------------------------------

Returns

*this

7.9.4.12 setFollowRedirects()

```
Request & curling::Request::setFollowRedirects (
    bool follow )
```

Enables or disables automatic redirect-following.

Parameters

<i>follow</i>	True to follow redirects.
---------------	---------------------------

Returns

*this

7.9.4.13 setHttpAuth()

```
Request & curling::Request::setHttpAuth (
    const std::string & username,
    const std::string & password )
```

Sets credentials for HTTP auth (Basic/Digest/NTLM).

Parameters

<i>username</i>	Username.
<i>password</i>	Password.

Returns

*this

7.9.4.14 setHttpAuthMethod()

```
Request & curling::Request::setHttpAuthMethod (
    AuthMethod method )
```

Sets HTTP authentication scheme.

Parameters

<i>method</i>	Authentication method.
---------------	------------------------

Returns

*this

7.9.4.15 setMethod()

```
Request & curling::Request::setMethod (
    Method m )
```

Sets the HTTP method for the request.

Parameters

<i>m</i>	Enum value for HTTP method.
----------	-----------------------------

Returns

*this

7.9.4.16 setProgressCallback()

```
Request & curling::Request::setProgressCallback (
    ProgressCallback cb )
```

Sets the progress callback function.

Parameters

<i>cb</i>	Callback receiving download/upload progress. Return true to abort.
-----------	--

Returns

*this

7.9.4.17 setProxy()

```
Request & curling::Request::setProxy (
    const std::string & url )
```

Enables proxy usage.

Parameters

<i>url</i>	Proxy URL.
------------	------------

Returns

*this

7.9.4.18 setProxyAuth()

```
Request & curling::Request::setProxyAuth (
    const std::string & username,
    const std::string & password )
```

Sets credentials for proxy authentication.

Parameters

<i>username</i>	Proxy username.
<i>password</i>	Proxy password.

Returns

*this

7.9.4.19 setProxyAuthMethod()

```
Request & curling::Request::setProxyAuthMethod (
    AuthMethod method )
```

Sets proxy authentication scheme.

Parameters

<i>method</i>	Authentication method.
---------------	------------------------

Returns

*this

7.9.4.20 setRawOption()

```
template<typename T >
Request & curling::Request::setRawOption (
    CURLOPToption opt,
    T value ) [inline]
```

Low level access to define curl options.

Warning

Should be used with caution. Meant for the libcurl advanced users.

7.9.4.21 setTimeout()

```
Request & curling::Request::setTimeout (
    long seconds )
```

Sets a timeout for the request (in seconds).

Parameters

<i>seconds</i>	Timeout in seconds.
----------------	---------------------

Returns

*this

7.9.4.22 setURL()

```
Request & curling::Request::setURL (
    const std::string & url )
```

Sets the request URL.

Parameters

<i>url</i>	URL to fetch.
------------	---------------

Returns

*this

7.9.4.23 setUserAgent()

```
Request & curling::Request::setUserAgent (
    const std::string & userAgent )
```

Sets the User-Agent header.

Parameters

<i>userAgent</i>	Agent string.
------------------	---------------

Returns

*this

The documentation for this class was generated from the following file:

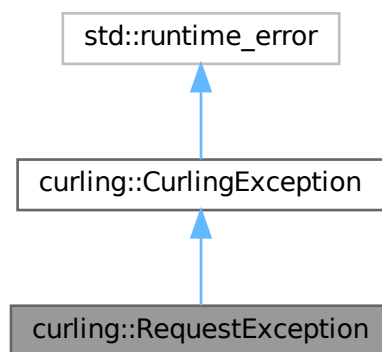
- include/curling.hpp

7.10 curling::RequestException Class Reference

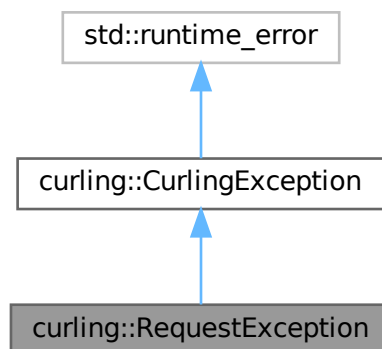
Thrown when a request operation fails.

```
#include <curling.hpp>
```

Inheritance diagram for curling::RequestException:



Collaboration diagram for curling::RequestException:



Public Member Functions

- **RequestException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.10.1 Detailed Description

Thrown when a request operation fails.

The documentation for this class was generated from the following file:

- include/curling.hpp

7.11 curling::Response Struct Reference

Represents an HTTP response.

```
#include <curling.hpp>
```

Public Member Functions

- std::string **toString** () const
- std::vector< std::string > **getHeader** (const std::string &key) const

Public Attributes

- long **httpCode**
HTTP status code.
- std::string **body**
[Response](#) body.
- std::map< std::string, std::vector< std::string > > **headers**
Header map (key: lowercase).

7.11.1 Detailed Description

Represents an HTTP response.

Contains the HTTP status code, body, and headers.

The documentation for this struct was generated from the following file:

- include/curling.hpp

Chapter 8

File Documentation

8.1 curling.hpp

```
00001 /*
00002  * Copyright (c) 2025 Paul Caron
00003  *
00004  * This file is part of Curling - a modern C++ wrapper for libcurl.
00005  *
00006  * Licensed under the MIT License. You may obtain a copy of the license at
00007  * https://opensource.org/licenses/MIT
00008  */
00009
00010
00062 #pragma once
00063 #include <string>
00064 #include <map>
00065 #include <vector>
00066 #include <sstream>
00067 #include <mutex>
00068 #include <stdexcept>
00069 #include <algorithm>
00070 #include <cctype>
00071 #include <memory>
00072 #include <functional>
00073 #include <iostream>
00074 #include <curl/curl.h>
00075
00076 namespace curling {
00077
00082 class CurlingException : public std::runtime_error {
00083 public:
00084     explicit CurlingException(const std::string& msg) : std::runtime_error(msg) {}
00085 };
00086
00090 class InitializationException : public CurlingException {
00091 public:
00092     explicit InitializationException(const std::string& msg) : CurlingException(msg) {}
00093 };
00094
00098 class RequestException : public CurlingException {
00099 public:
00100     explicit RequestException(const std::string& msg) : CurlingException(msg) {}
00101 };
00102
00106 class HeaderException : public CurlingException {
00107 public:
00108     explicit HeaderException(const std::string& msg) : CurlingException(msg) {}
00109 };
00110
00114 class MimeException : public CurlingException {
00115 public:
00116     explicit MimeException(const std::string& msg) : CurlingException(msg) {}
00117 };
00118
00122 class LogicException : public CurlingException {
00123 public:
00124     explicit LogicException(const std::string& msg) : CurlingException(msg) {}
00125 };
00126
00127 inline constexpr int version_major = 1;
00128 inline constexpr int version_minor = 2;
```

```

00129 inline constexpr int version_patch = 0;
00130
00131 inline std::string version() {
00132     std::ostringstream oss;
00133     oss << version_major << '.' << version_minor << '.' << version_patch;
00134     return oss.str();
00135 }
00136
00141 namespace detail {
00142
00143 inline std::once_flag curlGlobalInitFlag;
00144 inline std::mutex curlGlobalMutex;
00145
00146 inline int instanceCount = 0;
00147
00148 inline void ensureCurlGlobalInit() {
00149     std::lock_guard<std::mutex> lock(curlGlobalMutex);
00150     if (instanceCount++ == 0) {
00151         if (curl_global_init(CURL_GLOBAL_DEFAULT) != CURLE_OK) {
00152             throw InitializationException("Failed to initialize libcurl globally");
00153         }
00154     }
00155 }
00156
00157 inline void maybeCleanupGlobalCurl() noexcept {
00158     std::lock_guard<std::mutex> lock(curlGlobalMutex);
00159     if (--instanceCount == 0) {
00160         curl_global_cleanup();
00161     }
00162 }
00163
00164 inline void trim(std::string& s) {
00165     s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](unsigned char ch){
00166         return !std::isspace(ch);
00167     }));
00168     s.erase(std::find_if(s.rbegin(), s.rend(), [](unsigned char ch){
00169         return !std::isspace(ch);
00170     }).base(), s.end());
00171 }
00172
00173 inline void toLowerCase(std::string& s) {
00174     std::transform(s.begin(), s.end(), s.begin(),
00175         [](unsigned char c) { return std::tolower(c); });
00176 }
00177
00178
00179 inline size_t WriteCallback(void* contents, size_t size, size_t nmemb, void* userp) {
00180     auto responseStream = static_cast<std::ostream*>(userp);
00181     responseStream->write(static_cast<char*>(contents), size * nmemb);
00182     return size * nmemb;
00183 }
00184
00185 inline size_t HeaderCallback(char* buffer, size_t size, size_t nitems, void* userdata) {
00186     auto* headerMap = static_cast<std::map<std::string, std::vector<std::string>>*>(userdata);
00187     std::string headerLine(buffer, size * nitems);
00188
00189     if (headerLine.empty()) return 0; // skip the separation line
00190
00191     auto colonPos = headerLine.find(":");
00192     if (colonPos != std::string::npos) {
00193         std::string key = headerLine.substr(0, colonPos);
00194         std::string value = headerLine.substr(colonPos + 1);
00195         detail::trim(key);
00196         detail::trim(value);
00197         detail::toLowerCase(key);
00198         (*headerMap)[key].push_back(value);
00199     }
00200
00201     return size * nitems;
00202 }
00203
00204 inline int ProgressCallbackBridge(void* clientp, curl_off_t dltotal, curl_off_t dlnow,
00205     curl_off_t ultotal, curl_off_t ulnow);
00206
00207
00208 ///detail end
00209
00210
00211
00213 struct CurlHandleDeleter { void operator()(CURL* h) const noexcept { if (h) curl_easy_cleanup(h); }};
00214 struct CurlSlistDeleter { void operator()(curl_slist* l) const noexcept { if (l)
    curl_slist_free_all(l); }};
00215 struct CurlMimeDeleter { void operator()(curl_mime* m) const noexcept { if (m) curl_mime_free(m); }};
00216
00217 using CurlPtr = std::unique_ptr<CURL, CurlHandleDeleter>;
00218 using CurlSlistPtr = std::unique_ptr<curl_slist, CurlSlistDeleter>;
00219 using CurlMimePtr = std::unique_ptr<curl_mime, CurlMimeDeleter>;

```

```

00220
00227 struct Response {
00228     long httpCode;
00229     std::string body;
00230     std::map<std::string, std::vector<std::string>> headers;
00231
00232     std::string toString() const {
00233         std::ostringstream oss;
00234         oss << "status: " << httpCode << "\nbody:\n" << body << "\nheaders:\n";
00235         for (auto const& h : headers) {
00236             oss << h.first << ": ";
00237             for (auto const& v : h.second) oss << v << " ";
00238             oss << "\n";
00239         }
00240         return oss.str();
00241     }
00242     std::vector<std::string> getHeader(const std::string& key) const {
00243         std::string lowered = key;
00244         detail::toLowerCase(lowered);
00245         auto it = headers.find(lowered);
00246         return (it != headers.end()) ? it->second : std::vector<std::string>{};
00247     }
00248 };
00249
00254 class Request {
00255 public:
00256     using ProgressCallback = std::function<bool(curl_off_t dltotal, curl_off_t dlnow,
00257         curl_off_t ultotal, curl_off_t ulnow)>;
00258
00263     enum class Method {
00264         GET,
00265         POST,
00266         PUT,
00267         DEL,
00268         PATCH,
00269         HEAD,
00270         MIME
00271     };
00272
00277     enum class AuthMethod {
00278         BASIC = CURLAUTH_BASIC,
00279         NTLM = CURLAUTH_NTLM,
00280         DIGEST = CURLAUTH_DIGEST
00281     };
00282
00296     enum class HttpVersion {
00297         DEFAULT,
00298         HTTP_1_1,
00299         HTTP_2,
00300         HTTP_3
00301     };
00302
00307     Request();
00308
00312     ~Request() noexcept;
00313
00314     Request(Request&&) noexcept;
00315     Request& operator=(Request&&) noexcept;
00316
00317     Request(const Request&) = delete;
00318     Request& operator=(const Request&) = delete;
00319
00325     Request& setProgressCallback(ProgressCallback cb);
00326
00332     Request& setMethod(Method m);
00333
00339     Request& setURL(const std::string& url);
00340
00346     Request& setProxy(const std::string& url);
00347
00354     Request& setProxyAuth(const std::string& username, const std::string& password);
00355
00361     Request& setProxyAuthMethod(AuthMethod method);
00362
00369     Request& setHttpAuth(const std::string& username, const std::string& password);
00370
00376     Request& setHttpAuthMethod(AuthMethod method);
00377
00384     Request& addArg(const std::string& key, const std::string& value);
00385
00391     Request& addHeader(const std::string& header);
00392
00398     Request& setBody(const std::string& body);
00399
00405     Request& downloadToFile(const std::string& path);
00406
00412     Request& setTimeout(long seconds);

```

```

00413
00419     Request& setConnectTimeout(long seconds);
00420
00426     Request& setFollowRedirects(bool follow);
00427
00433     Request& setAuthToken(const std::string& token);
00434
00440     Request& setCookiePath(const std::string& path);
00441
00447     Request& setUserAgent(const std::string& userAgent);
00448
00456     Request& addFormField(const std::string& fieldName, const std::string& value);
00457
00465     Request& addFormFile(const std::string& fieldName, const std::string& filePath);
00466
00472     Request& enableVerbose(bool enabled = true);
00473
00479     Response send();
00480
00484     void reset();
00485
00489     Request& setHttpVersion(HttpVersion version);
00490
00496     template<typename T>
00497     Request& setRawOption(CURLOption opt, T value) {
00498         static_assert(std::is_pointer<T>::value || std::is_arithmetic<T>::value,
00499             "setRawOption only supports pointer or arithmetic types");
00500         curl_easy_setopt(curlHandle.get(), opt, value);
00501         return *this;
00502     }
00503
00504     friend int detail::ProgressCallbackBridge(void* clientp, curl_off_t dltotal, curl_off_t dlnow,
00505         curl_off_t ultotal, curl_off_t ulnow);
00506
00507 private:
00508     Method method;
00509     CurlPtr curlHandle;
00510     CurlSlistPtr list; //headers;
00511     std::string url, args, body, cookieFile, cookieJar;
00512     CurlMimePtr mime;
00513     std::string downloadFilePath;
00514     ProgressCallback progressCallback;
00515     HttpVersion httpVersion = HttpVersion::DEFAULT;
00516
00517     void clean() noexcept;
00518     void updateURL();
00519 };
00520
00521 static_assert(!std::is_copy_constructible_v<Request> && !std::is_copy_assignable_v<Request>,
00522     "curling::Request is not copyable: it is thread-unsafe and must not be shared between
00523     threads. One instance per thread.");
00524
00525 namespace detail{
00526 inline int ProgressCallbackBridge(void* clientp, curl_off_t dltotal, curl_off_t dlnow,
00527     curl_off_t ultotal, curl_off_t ulnow) {
00528     auto* req = static_cast<Request*>(clientp);
00529     if (req->progressCallback) {
00530         bool shouldCancel = req->progressCallback(dltotal, dlnow, ultotal, ulnow);
00531         return shouldCancel ? 1 : 0; // Returning non-zero aborts transfer
00532     }
00533     return 0;
00534 }
00535 } // namespace detail
00536
00537 } // namespace curling
00538

```


Index

- addArg
 - curling::Request, [23](#)
- addFormField
 - curling::Request, [23](#)
- addFormFile
 - curling::Request, [23](#)
- addHeader
 - curling::Request, [24](#)
- curling, [11](#)
- Curling: Modern C++ libcurl Wrapper, [1](#)
- curling::CurlHandleDeleter, [13](#)
- curling::CurlingException, [13](#)
- curling::CurlMimeDeleter, [14](#)
- curling::CurlSlistDeleter, [14](#)
- curling::detail, [12](#)
- curling::HeaderException, [15](#)
- curling::InitializationException, [16](#)
- curling::LogicException, [17](#)
- curling::MimeException, [18](#)
- curling::Request, [20](#)
 - addArg, [23](#)
 - addFormField, [23](#)
 - addFormFile, [23](#)
 - addHeader, [24](#)
 - DEFAULT, [22](#)
 - DEL, [22](#)
 - downloadToFile, [24](#)
 - enableVerbose, [25](#)
 - GET, [22](#)
 - HEAD, [22](#)
 - HTTP_1_1, [22](#)
 - HTTP_2, [22](#)
 - HTTP_3, [22](#)
 - HttpVersion, [22](#)
 - Method, [22](#)
 - MIME, [22](#)
 - PATCH, [22](#)
 - POST, [22](#)
 - PUT, [22](#)
 - Request, [22](#)
 - send, [25](#)
 - setAuthToken, [25](#)
 - setBody, [26](#)
 - setConnectTimeout, [26](#)
 - setCookiePath, [26](#)
 - setFollowRedirects, [27](#)
 - setHttpAuth, [27](#)
 - setHttpAuthMethod, [27](#)
 - setMethod, [28](#)
 - setProgressCallback, [28](#)
 - setProxy, [28](#)
 - setProxyAuth, [29](#)
 - setProxyAuthMethod, [29](#)
 - setRawOption, [29](#)
 - setTimeout, [29](#)
 - setURL, [30](#)
 - setUserAgent, [30](#)
- curling::RequestException, [31](#)
- curling::Response, [32](#)
- DEFAULT
 - curling::Request, [22](#)
- DEL
 - curling::Request, [22](#)
- downloadToFile
 - curling::Request, [24](#)
- enableVerbose
 - curling::Request, [25](#)
- GET
 - curling::Request, [22](#)
- HEAD
 - curling::Request, [22](#)
- HTTP_1_1
 - curling::Request, [22](#)
- HTTP_2
 - curling::Request, [22](#)
- HTTP_3
 - curling::Request, [22](#)
- HttpVersion
 - curling::Request, [22](#)
- include/curling.hpp, [33](#)
- Method
 - curling::Request, [22](#)
- MIME
 - curling::Request, [22](#)
- PATCH
 - curling::Request, [22](#)
- POST
 - curling::Request, [22](#)
- PUT
 - curling::Request, [22](#)
- Request
 - curling::Request, [22](#)

- send
 - curling::Request, [25](#)
- setAuthToken
 - curling::Request, [25](#)
- setBody
 - curling::Request, [26](#)
- setConnectTimeout
 - curling::Request, [26](#)
- setCookiePath
 - curling::Request, [26](#)
- setFollowRedirects
 - curling::Request, [27](#)
- setHttpAuth
 - curling::Request, [27](#)
- setHttpAuthMethod
 - curling::Request, [27](#)
- setMethod
 - curling::Request, [28](#)
- setProgressCallback
 - curling::Request, [28](#)
- setProxy
 - curling::Request, [28](#)
- setProxyAuth
 - curling::Request, [29](#)
- setProxyAuthMethod
 - curling::Request, [29](#)
- setRawOption
 - curling::Request, [29](#)
- setTimeout
 - curling::Request, [29](#)
- setURL
 - curling::Request, [30](#)
- setUserAgent
 - curling::Request, [30](#)