



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

***Rocky Enterprise Linux 9.2 Manual Pages on command 'LWP::Protocol.3pm'***

***\$ man LWP::Protocol.3pm***

LWP::Protocol(3pm)      User Contributed Perl Documentation      LWP::Protocol(3pm)

**NAME**

LWP::Protocol - Base class for LWP protocols

**SYNOPSIS**

```
package LWP::Protocol::foo;

use parent qw(LWP::Protocol);
```

**DESCRIPTION**

This class is used as the base class for all protocol implementations supported by the LWP library.

When creating an instance of this class using "LWP::Protocol::create(\$url)", and you get an initialized subclass appropriate for that access method. In other words, the "create" in LWP::Protocol function calls the constructor for one of its subclasses.

All derived "LWP::Protocol" classes need to override the "request()" method which is used to service a request. The overridden method can make use of the "collect()" method to collect together chunks of data as it is received.

**METHODS**

The following methods and functions are provided:

new

```
my $prot = LWP::Protocol->new();
```

The LWP::Protocol constructor is inherited by subclasses. As this is a virtual base class this method should not be called directly.

create

```
my $prot = LWP::Protocol::create($scheme)
```

Create an object of the class implementing the protocol to handle the given scheme. This is a function, not a method. It is more an object factory than a constructor. This is the function user agents should use to access protocols.

implementor

```
my $class = LWP::Protocol::implementor($scheme, [$class])
```

Get and/or set implementor class for a scheme. Returns "" if the specified scheme is not supported.

request

```
$response = $protocol->request($request, $proxy, undef);
```

```
$response = $protocol->request($request, $proxy, '/tmp/sss');
```

```
$response = $protocol->request($request, $proxy, \&callback, 1024);
```

Dispatches a request over the protocol, and returns a response object. This method needs to be overridden in subclasses. Refer to LWP::UserAgent for description of the arguments.

collect

```
my $res = $prot->collect(undef, $response, $collector); # stored in $response
```

```
my $res = $prot->collect($filename, $response, $collector);
```

```
my $res = $prot->collect(sub { ... }, $response, $collector);
```

Collect the content of a request, and process it appropriately into a scalar, file, or by calling a callback. If the first parameter is undefined, then the content is stored within the `$response`. If it's a simple scalar, then it's interpreted as a file name and the content is written to this file. If it's a code reference, then content is passed to this routine.

The collector is a routine that will be called and which is responsible for returning pieces (as ref to scalar) of the content to process. The `$collector` signals "EOF" by returning a reference to an empty string.

The return value is the `HTTP::Response` object reference.

Note: We will only use the callback or file argument if "`$response->is_success()`". This avoids sending content data for redirects and authentication responses to the callback which would be confusing.

`collect_once`

```
$prot->collect_once($arg, $response, $content)
```

Can be called when the whole response content is available as content. This will invoke "collect" in `LWP::Protocol` with a collector callback that returns a reference to `$content` the first time and an empty string the next.

## SEE ALSO

Inspect the `LWP/Protocol/file.pm` and `LWP/Protocol/http.pm` files for examples of usage.

## COPYRIGHT

Copyright 1995-2001 Gisle Aas.

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.