

1 Apache2::Response - Perl API for Apache HTTP request response methods

1.1 Synopsis

```
use Apache2::Response ();

$r->custom_response(Apache2::Const::FORBIDDEN, "No Entry today");

$etag = $r->make_etag($force_weak);
$r->set_etag();
$status = $r->meets_conditions();

$mtime_rat = $r->rationalize_mtime($mtime);
$r->set_last_modified($mtime);
$r->update_mtime($mtime);

$r->send_cgi_header($buffer);

$r->set_content_length($length);

$ret = $r->set_keepalive();
```

1.2 Description

`Apache2::Response` provides the Apache request object utilities API for dealing with HTTP response generation process.

1.3 API

`Apache2::Response` provides the following functions and/or methods:

1.3.1 *custom_response*

Install a custom response handler for a given status

```
$r->custom_response($status, $string);
```

- **obj: \$r (Apache2::RequestRec object)**

The current request

- **arg1: \$status (Apache2::Const constant)**

The status for which the custom response should be used (e.g. `Apache2::Const::AUTH_REQUIRED`)

- **arg2: \$string (string)**

The custom response to use. This can be a static string, or a URL, full or just the uri path (*/foo/bar.txt*).

- **ret: no return value**
- **since: 2.0.00**

`custom_response()` doesn't alter the response code, but is used to replace the standard response body. For example, here is how to change the response body for the access handler failure:

```
package MyApache2::MyShop;
use Apache2::Response ();
use Apache2::Const -compile => qw(FORBIDDEN OK);
sub access {
    my $r = shift;

    if (MyApache2::MyShop::tired_squirrels()) {
        $r->custom_response(Apache2::Const::FORBIDDEN,
            "It's siesta time, please try later");
        return Apache2::Const::FORBIDDEN;
    }

    return Apache2::Const::OK;
}
...

# httpd.conf
PerlModule MyApache2::MyShop
<Location /TestAPI__custom_response>
    AuthName dummy
    AuthType none
    PerlAccessHandler MyApache2::MyShop::access
    PerlResponseHandler MyApache2::MyShop::response
</Location>
```

When squirrels can't run any more, the handler will return 403, with the custom message:

```
It's siesta time, please try later
```

1.3.2 *make_etag*

Construct an entity tag from the resource information. If it's a real file, build in some of the file characteristics.

```
$etag = $r->make_etag($force_weak);
```

- **obj: \$r (Apache2::RequestRec object)**

The current request

- **arg1: \$force_weak (number)**

Force the entity tag to be weak - it could be modified again in as short an interval.

- **ret: \$etag (string)**

The entity tag

- **since: 2.0.00**

1.3.3 meets_conditions

Implements condition GET rules for HTTP/1.1 specification. This function inspects the client headers and determines if the response fulfills the specified requirements.

```
$status = $r->meets_conditions();
```

- **obj: \$r (Apache2::RequestRec object)**

The current request

- **ret: \$status (Apache2::Const status constant)**

Apache2::Const::OK if the response fulfills the condition GET rules. Otherwise some other status code (which should be returned to Apache).

- **since: 2.0.00**

Refer to the Generating Correct HTTP Headers document for an indepth discussion of this method.

1.3.4 rationalize_mtime

Return the latest rational time from a request/mtime pair.

```
$mtime_rat = $r->rationalize_mtime($mtime);
```

- **obj: \$r (Apache2::RequestRec object)**

The current request

- **arg1: \$mtime (time in seconds)**

The last modified time

- **ret: \$mtime_rat (time in seconds)**

the latest rational time from a request/mtime pair. Mtime is returned unless it's in the future, in which case we return the current time.

- **since: 2.0.00**

1.3.5 *send_cgi_header*

Parse the header

```
$r->send_cgi_header($buffer);
```

- **obj:** `$r` (`Apache2::RequestRec` object)
- **arg1:** `$buffer` (string)

headers and optionally a response body

- **ret:** no return value
- **since:** 2.0.00

This method is really for back-compatibility with `mod_perl` 1.0. It's very inefficient to send headers this way, because of the parsing overhead.

If there is a response body following the headers it'll be handled too (as if it was sent via `print()`).

Notice that if only HTTP headers are included they won't be sent until some body is sent (again the "send" part is retained from the `mod_perl` 1.0 method).

1.3.6 *set_content_length*

Set the content length for this request.

```
$r->set_content_length($length);
```

- **obj:** `$r` (`Apache2::RequestRec` object)

The current request

- **arg1:** `$length` (integer)

The new content length

- **ret:** no return value
- **since:** 2.0.00

1.3.7 *set_etag*

Set the E-tag outgoing header

```
$r->set_etag();
```

- **obj:** `$r` (`Apache2::RequestRec` object)
- **ret:** no return value
- **since:** 2.0.00

1.3.8 *set_keepalive*

Set the keepalive status for this request

```
$ret = $r->set_keepalive();
```

- **obj: \$r (Apache2::RequestRec object)**

The current request

- **ret: \$ret (boolean)**

true if keepalive can be set, false otherwise

- **since: 2.0.00**

It's called by `ap_http_header_filter()`. For the complete complicated logic implemented by this method see `httpd-2.0/server/http_protocol.c`.

1.3.9 *set_last_modified*

sets the Last-Modified response header field to the value of the `mtime` field in the request structure -- rationalized to keep it from being in the future.

```
$r->set_last_modified($mtime);
```

- **obj: \$r (Apache2::RequestRec object)**
- **opt arg1: \$mtime (time in seconds)**

if the `$mtime` argument is passed, `$r->update_mtime` will be first run with that argument.

- **ret: no return value**
- **since: 2.0.00**

1.3.10 *update_mtime*

Set the `$r->mtime` field to the specified value if it's later than what's already there.

```
$r->update_mtime($mtime);
```

- **obj: \$r (Apache2::RequestRec object)**

The current request

- **arg1: \$mtime (time in seconds)**
- **ret: no return value**
- **since: 2.0.00**

See also: `$r->set_last_modified`.

1.4 Unsupported API

`Apache2::Response` also provides auto-generated Perl interface for a few other methods which aren't tested at the moment and therefore their API is a subject to change. These methods will be finalized later as a need arises. If you want to rely on any of the following methods please contact the `mod_perl` development mailing list so we can help each other take the steps necessary to shift the method to an officially supported API.

1.4.1 *send_error_response*

Send an "error" response back to client. It is used for any response that can be generated by the server from the request record. This includes all 204 (no content), 3xx (redirect), 4xx (client error), and 5xx (server error) messages that have not been redirected to another handler via the `ErrorDocument` feature.

```
$r->send_error_response($recursive_error);
```

- **obj:** `$r` (`Apache2::RequestRec` object)

The current request

- **arg1:** `$recursive_error` (boolean)

the error status in case we get an error in the process of trying to deal with an `ErrorDocument` to handle some other error. In that case, we print the default report for the first thing that went wrong, and more briefly report on the problem with the `ErrorDocument`.

- **ret:** no return value
- **since:** 2.0.00

META: it's really an internal Apache method, I'm not quite sure how can it be used externally.

1.4.2 *send_mmap*

META: Autogenerated - needs to be reviewed/completed

Send an MMAP'ed file to the client

```
$ret = $r->send_mmap($mm, $offset, $length);
```

- **obj:** `$r` (`Apache2::RequestRec` object)

The current request

- **arg1:** `$mm` (`APR::Mmap`)

1.5 See Also

The MMAP'ed file to send

- **arg2: \$offset (number)**

The offset into the MMAP to start sending

- **arg3: \$length (integer)**

The amount of data to send

- **ret: \$ret (integer)**

The number of bytes sent

- **since: 2.0.00**

META: requires a working APR::Mmap, which is not supported at the moment.

1.5 See Also

mod_perl 2.0 documentation.

1.6 Copyright

mod_perl 2.0 and its core modules are copyrighted under The Apache Software License, Version 2.0.

1.7 Authors

The mod_perl development team and numerous contributors.

Table of Contents:

1	Apache2::Response - Perl API for Apache HTTP request response methods	1
1.1	Synopsis	2
1.2	Description	2
1.3	API	2
1.3.1	custom_response	2
1.3.2	make_etag	3
1.3.3	meets_conditions	4
1.3.4	rationalize_mtime	4
1.3.5	send_cgi_header	5
1.3.6	set_content_length	5
1.3.7	set_etag	5
1.3.8	set_keepalive	6
1.3.9	set_last_modified	6
1.3.10	update_mtime	6
1.4	Unsupported API	7
1.4.1	send_error_response	7
1.4.2	send_mmap	7
1.5	See Also	8
1.6	Copyright	8
1.7	Authors	8