aio-libs / **aiohttp** Public

Code Issues 185 Pull requests 91 Discussions Actions Security an

Commit `0c2e9da`

Dreamsorcerer authored on Jan 12 · 34 / 38 · Verified

```
Add max_headers parameter (#11955) (#11959) (#11960)

(cherry picked from commit ed6440c)
(cherry picked from commit 30ec25f)
```

`master` (#11960) · v3.13.5 v3.13.4

1 parent 016e652 commit 0c2e9da

12 files changed +277 -130 lines changed

Top

Filter files…

- CHANGES
  - 11955.feature.rst
- aiohttp
  - _http_parser.pyx
  - client.py
  - client_proto.py
  - http_exceptions.py
  - http_parser.py
  - web_protocol.py
- docs
  - client_reference.rst
  - web_reference.rst
- tests
  - test_client_functional.py
  - test_http_exceptions.py

test_http_parser.py

12 files changed +277 -130 lines changed

Search within code

CHANGES/11955.feature.rst

```
@@ -0,0 +1 @@
+ Added ``max_headers`` parameter to limit the number of headers that should be
  read from a response -- by :user:`Dreamsorcerer`.
```

aiohttp/_http_parser.pyx

```
@@ -279,6 +279,7 @@ cdef class HttpParser:
279 279           object _name
280 280           bytes _raw_value
281 281           bint      _has_value
    282 +         int _header_name_size
282 283
283 284           object _protocol
284 285           object _loop
@@ -329,7 +330,7 @@ cdef class HttpParser:
329 330           self, cparser.llhttp_type mode,
330 331           object protocol, object loop, int limit,
331 332           object timer=None,
332     -         size_t max_line_size=8190, size_t max_headers=32768,
    333 +         size_t max_line_size=8190, size_t max_headers=128,
333 334           size_t max_field_size=8190, payload_exception=None,
334 335           bint response_with_body=True, bint read_until_eof=False,
335 336           bint auto_decompress=True,
@@ -352,6 +353,7 @@ cdef class HttpParser:
352 353           self._raw_name = EMPTY_BYTES
353 354           self._raw_value = EMPTY_BYTES
354 355           self._has_value = False
    356 +         self._header_name_size = 0
355 357
356 358           self._max_line_size = max_line_size
357 359           self._max_headers = max_headers
@@ -383,11 +385,14 @@ cdef class HttpParser:
```

```
383  385                 value = self._raw_value.decode('utf-8', 'surrogateescape')
384  386
385  387                 self._headers.append((name, value))
     388 +               if len(self._headers) > self._max_headers:
     389 +                   raise BadHttpMessage("Too many headers received")
386  390
387  391                 if name is CONTENT_ENCODING:
388  392                     self._content_encoding = value
389  393
390  394                 self._has_value = False
     395 +               self._header_name_size = 0
391  396                 self._raw_headers.append((self._raw_name, self._raw_value))
392  397                 self._raw_name = EMPTY_BYTES
393  398                 self._raw_value = EMPTY_BYTES
          @@ -574,7 +579,7 @@ cdef class HttpRequestParser(HttpParser):
574  579
575  580         def __init__(
576  581             self, protocol, loop, int limit, timer=None,
577       -         size_t max_line_size=8190, size_t max_headers=32768,
     582 +         size_t max_line_size=8190, size_t max_headers=128,
578  583             size_t max_field_size=8190, payload_exception=None,
579  584             bint response_with_body=True, bint read_until_eof=False,
580  585             bint auto_decompress=True,
          @@ -638,7 +643,7 @@ cdef class HttpResponseParser(HttpParser):
638  643
639  644         def __init__(
640  645             self, protocol, loop, int limit, timer=None,
641       -             size_t max_line_size=8190, size_t max_headers=32768,
     646 +             size_t max_line_size=8190, size_t max_headers=128,
642  647                 size_t max_field_size=8190, payload_exception=None,
643  648                 bint response_with_body=True, bint read_until_eof=False,
644  649                 bint auto_decompress=True
          @@ -677,8 +682,8 @@ cdef int cb_on_url(cparser.llhttp_t* parser,
677  682         cdef HttpParser pyparser = <HttpParser>parser.data
678  683         try:
679  684             if length > pyparser._max_line_size:
680       -             raise LineTooLong(
```

```
681       -                'Status line is too long', pyparser._max_line_size, length)
      685 +                status = pyparser._buf + at[:length]
      686 +                raise LineTooLong(status[:100] + b"...", pyparser._max_line_size)
682   687              extend(pyparser._buf, at, length)
683   688          except BaseException as ex:
684   689              pyparser._last_error = ex
          @@ -690,11 +695,10 @@ cdef int cb_on_url(cparser.llhttp_t* parser,
690   695      cdef int cb_on_status(cparser.llhttp_t* parser,
691   696                            const char *at, size_t length) except -1:
692   697          cdef HttpParser pyparser = <HttpParser>parser.data
693       -      cdef str reason
694   698          try:
695   699              if length > pyparser._max_line_size:
696       -                raise LineTooLong(
697       -                    'Status line is too long', pyparser._max_line_size, length)
      700 +                reason = pyparser._buf + at[:length]
      701 +                raise LineTooLong(reason[:100] + b"...", pyparser._max_line_size)
698   702              extend(pyparser._buf, at, length)
699   703          except BaseException as ex:
700   704              pyparser._last_error = ex
          @@ -711,8 +715,9 @@ cdef int cb_on_header_field(cparser.llhttp_t* parser,
711   715              pyparser._on_status_complete()
712   716              size = len(pyparser._raw_name) + length
713   717              if size > pyparser._max_field_size:
714       -                raise LineTooLong(
715       -                    'Header name is too long', pyparser._max_field_size, size)
      718 +                name = pyparser._raw_name + at[:length]
      719 +                raise LineTooLong(name[:100] + b"...", pyparser._max_field_size)
      720 +            pyparser._header_name_size = size
716   721              pyparser._on_header_field(at, length)
717   722          except BaseException as ex:
718   723              pyparser._last_error = ex
          @@ -727,9 +732,9 @@ cdef int cb_on_header_value(cparser.llhttp_t* parser,
727   732          cdef Py_ssize_t size
728   733          try:
729   734              size = len(pyparser._raw_value) + length
730       -            if size > pyparser._max_field_size:
731       -                raise LineTooLong(
732       -                    'Header value is too long', pyparser._max_field_size, size)
      735 +            if pyparser._header_name_size + size > pyparser._max_field_size:
```

```
      736 +                value = pyparser._raw_value + at[:length]
      737 +                raise LineTooLong(value[:100] + b"...", pyparser._max_field_size)
733   738              pyparser._on_header_value(at, length)
734   739          except BaseException as ex:
735   740              pyparser._last_error = ex
```

aiohttp/client.py

```
@@ -195,6 +195,7 @@ class _RequestOptions(TypedDict, total=False):
195   195      auto_decompress: Union[bool, None]
196   196      max_line_size: Union[int, None]
197   197      max_field_size: Union[int, None]
      198 +    max_headers: Union[int, None]
198   199      middlewares: Optional[Sequence[ClientMiddlewareType]]
199   200
200   201
@@ -259,6 +260,7 @@ class ClientSession:
259   260              "_read_bufsize",
260   261              "_max_line_size",
261   262              "_max_field_size",
      263 +            "_max_headers",
262   264              "_resolve_charset",
263   265              "_default_proxy",
264   266              "_default_proxy_auth",
@@ -303,6 +305,7 @@ def __init__(
303   305          read_bufsize: int = 2**16,
304   306          max_line_size: int = 8190,
305   307          max_field_size: int = 8190,
      308 +        max_headers: int = 128,
306   309          fallback_charset_resolver: _CharsetResolver = lambda r, b: "utf-8",
307   310          middlewares: Sequence[ClientMiddlewareType] = (),
308   311          ssl_shutdown_timeout: Union[_SENTINEL, None, float] = sentinel,
@@ -402,6 +405,7 @@ def __init__(
402   405          self._read_bufsize = read_bufsize
403   406          self._max_line_size = max_line_size
404   407          self._max_field_size = max_field_size
      408 +        self._max_headers = max_headers
```

```
405  409
406  410             # Convert to list of tuples
407  411             if headers:
     @@ -518,6 +522,7 @@ async def _request(
518  522             auto_decompress: Optional[bool] = None,
519  523             max_line_size: Optional[int] = None,
520  524             max_field_size: Optional[int] = None,
     525 +           max_headers: Optional[int] = None,
521  526             middlewares: Optional[Sequence[ClientMiddlewareType]] = None,
522  527         ) -> ClientResponse:
523  528
     @@ -607,6 +612,9 @@ async def _request(
607  612             if max_field_size is None:
608  613                 max_field_size = self._max_field_size
609  614
     615 +           if max_headers is None:
     616 +               max_headers = self._max_headers
     617 +
610  618             traces = [
611  619                 Trace(
612  620                     self,
     @@ -750,6 +758,7 @@ async def _connect_and_send_request(
750  758
                                timeout_ceil_threshold=self._connector._timeout_ceil_threshold,
751  759                                max_line_size=max_line_size,
752  760                                max_field_size=max_field_size,
     761 +                          max_headers=max_headers,
753  762                            )
754  763                            try:
755  764                                resp = await req.send(conn)
```

aiohttp/client_proto.py

```
     @@ -230,6 +230,7 @@ def set_response_params(
230  230             timeout_ceil_threshold: float = 5,
231  231             max_line_size: int = 8190,
232  232             max_field_size: int = 8190,
```

```
      233 +          max_headers: int = 128,
233   234          ) -> None:
234   235              self._skip_payload = skip_payload
235   236
          @@ -248,6 +249,7 @@ def set_response_params(
248   249                  auto_decompress=auto_decompress,
249   250                  max_line_size=max_line_size,
250   251                  max_field_size=max_field_size,
      252 +                max_headers=max_headers,
251   253              )
252   254
253   255              if self._tail:
```

aiohttp/http_exceptions.py

```
          @@ -80,11 +80,12 @@ class DecompressSizeError(PayloadEncodingError):
80    80
81    81      class LineTooLong(BadHttpMessage):
82    82          def __init__(
83        -             self, line: str, limit: str = "Unknown", actual_size: str = "Unknown"
      83 +             self,
      84 +             line: Union[str, bytes],
      85 +             limit: Union[str, int] = "Unknown",
      86 +             actual_size: str = "Unknown",
84    87          ) -> None:
85        -             super().__init__(
86        -                 f"Got more than {limit} bytes ({actual_size}) when reading {line}."
87        -             )
      88 +             super().__init__(f"Got more than {limit} bytes when reading: {line!r}.")
88    89              self.args = (line, limit, actual_size)
89    90
90    91
```

Comments 0

Please sign in to comment.