**java.net**

# Class HttpURLConnection

java.lang.Object
    java.net.URLConnection
        java.net.HttpURLConnection

**Direct Known Subclasses:**
    HttpsURLConnection

---

```
public abstract class HttpURLConnection
extends URLConnection
```

A URLConnection with support for HTTP-specific features. See the spec for details.

Each HttpURLConnection instance is used to make a single request but the underlying network connection to the HTTP server may be transparently shared by other instances. Calling the close() methods on the InputStream or OutputStream of an HttpURLConnection after a request may free network resources associated with this instance but has no effect on any shared persistent connection. Calling the disconnect() method may close the underlying socket if a persistent connection is otherwise idle at that time.

The HTTP protocol handler has a few settings that can be accessed through System Properties. This covers Proxy settings as well as various other settings.

**Since:**
    JDK1.1

**See Also:**
    disconnect()

## Field Summary

| Modifier and Type | Field and Description |
|---|---|
| protected int | **chunkLength**<br>The chunk-length when using chunked encoding streaming mode for output. |
| protected int | **fixedContentLength**<br>The fixed content-length when using fixed-length streaming mode. |
| protected | **fixedContentLengthLong** |

| | |
|---|---|
| long | The fixed content-length when using fixed-length streaming mode. |
| static int | **HTTP_ACCEPTED**<br>HTTP Status-Code 202: Accepted. |
| static int | **HTTP_BAD_GATEWAY**<br>HTTP Status-Code 502: Bad Gateway. |
| static int | **HTTP_BAD_METHOD**<br>HTTP Status-Code 405: Method Not Allowed. |
| static int | **HTTP_BAD_REQUEST**<br>HTTP Status-Code 400: Bad Request. |
| static int | **HTTP_CLIENT_TIMEOUT**<br>HTTP Status-Code 408: Request Time-Out. |
| static int | **HTTP_CONFLICT**<br>HTTP Status-Code 409: Conflict. |
| static int | **HTTP_CREATED**<br>HTTP Status-Code 201: Created. |
| static int | **HTTP_ENTITY_TOO_LARGE**<br>HTTP Status-Code 413: Request Entity Too Large. |
| static int | **HTTP_FORBIDDEN**<br>HTTP Status-Code 403: Forbidden. |
| static int | **HTTP_GATEWAY_TIMEOUT**<br>HTTP Status-Code 504: Gateway Timeout. |
| static int | **HTTP_GONE**<br>HTTP Status-Code 410: Gone. |
| static int | **HTTP_INTERNAL_ERROR**<br>HTTP Status-Code 500: Internal Server Error. |
| static int | **HTTP_LENGTH_REQUIRED**<br>HTTP Status-Code 411: Length Required. |
| static int | **HTTP_MOVED_PERM**<br>HTTP Status-Code 301: Moved Permanently. |
| static int | **HTTP_MOVED_TEMP**<br>HTTP Status-Code 302: Temporary Redirect. |
| static int | **HTTP_MULT_CHOICE**<br>HTTP Status-Code 300: Multiple Choices. |
| static int | **HTTP_NO_CONTENT**<br>HTTP Status-Code 204: No Content. |
| static int | **HTTP_NOT_ACCEPTABLE**<br>HTTP Status-Code 406: Not Acceptable. |
| static int | **HTTP_NOT_AUTHORITATIVE**<br>HTTP Status-Code 203: Non-Authoritative Information. |
| static int | **HTTP_NOT_FOUND**<br>HTTP Status-Code 404: Not Found. |
| static int | **HTTP_NOT_IMPLEMENTED**<br>HTTP Status-Code 501: Not Implemented. |
| static int | **HTTP_NOT_MODIFIED**<br>HTTP Status-Code 304: Not Modified. |
| static int | **HTTP_OK**<br>HTTP Status-Code 200: OK. |

| | |
|---|---|
| static int | **HTTP_PARTIAL**<br>HTTP Status-Code 206: Partial Content. |
| static int | **HTTP_PAYMENT_REQUIRED**<br>HTTP Status-Code 402: Payment Required. |
| static int | **HTTP_PRECON_FAILED**<br>HTTP Status-Code 412: Precondition Failed. |
| static int | **HTTP_PROXY_AUTH**<br>HTTP Status-Code 407: Proxy Authentication Required. |
| static int | **HTTP_REQ_TOO_LONG**<br>HTTP Status-Code 414: Request-URI Too Large. |
| static int | **HTTP_RESET**<br>HTTP Status-Code 205: Reset Content. |
| static int | **HTTP_SEE_OTHER**<br>HTTP Status-Code 303: See Other. |
| static int | **HTTP_SERVER_ERROR**<br>**Deprecated.**<br>*it is misplaced and shouldn't have existed.* |
| static int | **HTTP_UNAUTHORIZED**<br>HTTP Status-Code 401: Unauthorized. |
| static int | **HTTP_UNAVAILABLE**<br>HTTP Status-Code 503: Service Unavailable. |
| static int | **HTTP_UNSUPPORTED_TYPE**<br>HTTP Status-Code 415: Unsupported Media Type. |
| static int | **HTTP_USE_PROXY**<br>HTTP Status-Code 305: Use Proxy. |
| static int | **HTTP_VERSION**<br>HTTP Status-Code 505: HTTP Version Not Supported. |
| protected boolean | **instanceFollowRedirects**<br>If `true`, the protocol will automatically follow redirects. |
| protected String | **method**<br>The HTTP method (GET,POST,PUT,etc.). |
| protected int | **responseCode**<br>An `int` representing the three digit HTTP Status-Code. |
| protected String | **responseMessage**<br>The HTTP response message. |

## Fields inherited from class java.net.URLConnection

allowUserInteraction, connected, doInput, doOutput, ifModifiedSince, url, useCaches

## Constructor Summary

| Modifier | Constructor and Description |
|---|---|
| protected | **HttpURLConnection**(URL u)<br>Constructor for the HttpURLConnection. |

## Method Summary

| Modifier | Method and Description |
|---|---|

| and Type | |
|---|---|
| abstract void | **disconnect**()<br>Indicates that other requests to the server are unlikely in the near future. |
| InputStream | **getErrorStream**()<br>Returns the error stream if the connection failed but the server sent useful data nonetheless. |
| static boolean | **getFollowRedirects**()<br>Returns a `boolean` indicating whether or not HTTP redirects (3xx) should be automatically followed. |
| String | **getHeaderField**(int n)<br>Returns the value for the $n^{th}$ header field. |
| long | **getHeaderFieldDate**(String name, long Default)<br>Returns the value of the named field parsed as date. |
| String | **getHeaderFieldKey**(int n)<br>Returns the key for the $n^{th}$ header field. |
| boolean | **getInstanceFollowRedirects**()<br>Returns the value of this `HttpURLConnection`'s `instanceFollowRedirects` field. |
| Permission | **getPermission**()<br>Returns a `SocketPermission` object representing the permission necessary to connect to the destination host and port. |
| String | **getRequestMethod**()<br>Get the request method. |
| int | **getResponseCode**()<br>Gets the status code from an HTTP response message. |
| String | **getResponseMessage**()<br>Gets the HTTP response message, if any, returned along with the response code from a server. |
| void | **setChunkedStreamingMode**(int chunklen)<br>This method is used to enable streaming of a HTTP request body without internal buffering, when the content length is **not** known in advance. |
| void | **setFixedLengthStreamingMode**(int contentLength)<br>This method is used to enable streaming of a HTTP request body without internal buffering, when the content length is known in advance. |
| void | **setFixedLengthStreamingMode**(long contentLength)<br>This method is used to enable streaming of a HTTP request body without internal buffering, when the content length is known in advance. |
| static void | **setFollowRedirects**(boolean set)<br>Sets whether HTTP redirects (requests with response code 3xx) should be automatically followed by this class. |
| void | **setInstanceFollowRedirects**(boolean followRedirects)<br>Sets whether HTTP redirects (requests with response code 3xx) should be automatically followed by this `HttpURLConnection` instance. |
| void | **setRequestMethod**(String method)<br>Set the method for the URL request, one of: GET POST HEAD OPTIONS PUT DELETE TRACE are legal, subject to protocol restrictions. |

| | |
|---|---|
| abstract boolean | **usingProxy**()<br>Indicates if the connection is going through a proxy. |

## Methods inherited from class java.net.URLConnection

addRequestProperty, connect, getAllowUserInteraction, getConnectTimeout, getContent, getContent, getContentEncoding, getContentLength, getContentLengthLong, getContentType, getDate, getDefaultAllowUserInteraction, getDefaultRequestProperty, getDefaultUseCaches, getDoInput, getDoOutput, getExpiration, getFileNameMap, getHeaderField, getHeaderFieldInt, getHeaderFieldLong, getHeaderFields, getIfModifiedSince, getInputStream, getLastModified, getOutputStream, getReadTimeout, getRequestProperties, getRequestProperty, getURL, getUseCaches, guessContentTypeFromName, guessContentTypeFromStream, setAllowUserInteraction, setConnectTimeout, setContentHandlerFactory, setDefaultAllowUserInteraction, setDefaultRequestProperty, setDefaultUseCaches, setDoInput, setDoOutput, setFileNameMap, setIfModifiedSince, setReadTimeout, setRequestProperty, setUseCaches, toString

## Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

## Field Detail

### method

```
protected String method
```

The HTTP method (GET,POST,PUT,etc.).

---

### chunkLength

```
protected int chunkLength
```

The chunk-length when using chunked encoding streaming mode for output. A value of -1 means chunked encoding is disabled for output.

**Since:**
1.5

---

### fixedContentLength

```
protected int fixedContentLength
```

The fixed content-length when using fixed-length streaming mode. A value of -1 means fixed-length streaming mode is disabled for output.

**NOTE:** fixedContentLengthLong is recommended instead of this field, as it allows larger content lengths to be set.

**Since:**

1.5

---

## fixedContentLengthLong

```
protected long fixedContentLengthLong
```

The fixed content-length when using fixed-length streaming mode. A value of `-1` means fixed-length streaming mode is disabled for output.

**Since:**
1.7

---

## responseCode

```
protected int responseCode
```

An `int` representing the three digit HTTP Status-Code.

- 1xx: Informational
- 2xx: Success
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

---

## responseMessage

```
protected String responseMessage
```

The HTTP response message.

---

## instanceFollowRedirects

```
protected boolean instanceFollowRedirects
```

If `true`, the protocol will automatically follow redirects. If `false`, the protocol will not automatically follow redirects.

This field is set by the `setInstanceFollowRedirects` method. Its value is returned by the `getInstanceFollowRedirects` method.

Its default value is based on the value of the static followRedirects at HttpURLConnection construction time.

**See Also:**

setInstanceFollowRedirects(boolean), getInstanceFollowRedirects(), setFollowRedirects(boolean)

---

## HTTP_OK

```
public static final int HTTP_OK
```

HTTP Status-Code 200: OK.

**See Also:**
Constant Field Values

---

## HTTP_CREATED

```
public static final int HTTP_CREATED
```

HTTP Status-Code 201: Created.

**See Also:**
Constant Field Values

---

## HTTP_ACCEPTED

```
public static final int HTTP_ACCEPTED
```

HTTP Status-Code 202: Accepted.

**See Also:**
Constant Field Values

---

## HTTP_NOT_AUTHORITATIVE

```
public static final int HTTP_NOT_AUTHORITATIVE
```

HTTP Status-Code 203: Non-Authoritative Information.

**See Also:**
Constant Field Values

---

## HTTP_NO_CONTENT

```
public static final int HTTP_NO_CONTENT
```

HTTP Status-Code 204: No Content.

**See Also:**
Constant Field Values

---

## HTTP_RESET

```
public static final int HTTP_RESET
```

HTTP Status-Code 205: Reset Content.

**See Also:**
Constant Field Values

---

## HTTP_PARTIAL

```
public static final int HTTP_PARTIAL
```

HTTP Status-Code 206: Partial Content.

**See Also:**
Constant Field Values

---

## HTTP_MULT_CHOICE

```
public static final int HTTP_MULT_CHOICE
```

HTTP Status-Code 300: Multiple Choices.

**See Also:**
Constant Field Values

---

## HTTP_MOVED_PERM

```
public static final int HTTP_MOVED_PERM
```

HTTP Status-Code 301: Moved Permanently.

**See Also:**
Constant Field Values

---

## HTTP_MOVED_TEMP

```
public static final int HTTP_MOVED_TEMP
```

HTTP Status-Code 302: Temporary Redirect.

**See Also:**
Constant Field Values

---

## HTTP_SEE_OTHER

```
public static final int HTTP_SEE_OTHER
```

HTTP Status-Code 303: See Other.

**See Also:**
Constant Field Values

---

## HTTP_NOT_MODIFIED

```
public static final int HTTP_NOT_MODIFIED
```

HTTP Status-Code 304: Not Modified.

**See Also:**
Constant Field Values

---

## HTTP_USE_PROXY

```
public static final int HTTP_USE_PROXY
```

HTTP Status-Code 305: Use Proxy.

**See Also:**
Constant Field Values

---

## HTTP_BAD_REQUEST

```
public static final int HTTP_BAD_REQUEST
```

HTTP Status-Code 400: Bad Request.

**See Also:**
Constant Field Values

---

## HTTP_UNAUTHORIZED

```
public static final int HTTP_UNAUTHORIZED
```

HTTP Status-Code 401: Unauthorized.

**See Also:**
Constant Field Values

---

## HTTP_PAYMENT_REQUIRED

```
public static final int HTTP_PAYMENT_REQUIRED
```

HTTP Status-Code 402: Payment Required.

**See Also:**
Constant Field Values

---

## HTTP_FORBIDDEN

```
public static final int HTTP_FORBIDDEN
```

HTTP Status-Code 403: Forbidden.

**See Also:**
Constant Field Values

---

## HTTP_NOT_FOUND

```
public static final int HTTP_NOT_FOUND
```

HTTP Status-Code 404: Not Found.

**See Also:**
Constant Field Values

---

## HTTP_BAD_METHOD

```
public static final int HTTP_BAD_METHOD
```

HTTP Status-Code 405: Method Not Allowed.

**See Also:**
Constant Field Values

---

## HTTP_NOT_ACCEPTABLE

```
public static final int HTTP_NOT_ACCEPTABLE
```

HTTP Status-Code 406: Not Acceptable.

**See Also:**
Constant Field Values

---

## HTTP_PROXY_AUTH

```
public static final int HTTP_PROXY_AUTH
```

HTTP Status-Code 407: Proxy Authentication Required.

**See Also:**
Constant Field Values

---

## HTTP_CLIENT_TIMEOUT

```
public static final int HTTP_CLIENT_TIMEOUT
```

HTTP Status-Code 408: Request Time-Out.

**See Also:**
Constant Field Values

---

## HTTP_CONFLICT

```
public static final int HTTP_CONFLICT
```

HTTP Status-Code 409: Conflict.

**See Also:**
Constant Field Values

---

## HTTP_GONE

```
public static final int HTTP_GONE
```

HTTP Status-Code 410: Gone.

**See Also:**
Constant Field Values

---

## HTTP_LENGTH_REQUIRED

```
public static final int HTTP_LENGTH_REQUIRED
```

HTTP Status-Code 411: Length Required.

**See Also:**
Constant Field Values

---

## HTTP_PRECON_FAILED

```
public static final int HTTP_PRECON_FAILED
```

HTTP Status-Code 412: Precondition Failed.

**See Also:**
Constant Field Values

---

## HTTP_ENTITY_TOO_LARGE

```
public static final int HTTP_ENTITY_TOO_LARGE
```

HTTP Status-Code 413: Request Entity Too Large.

**See Also:**
Constant Field Values

---

## HTTP_REQ_TOO_LONG

```
public static final int HTTP_REQ_TOO_LONG
```

HTTP Status-Code 414: Request-URI Too Large.

**See Also:**
Constant Field Values

---

## HTTP_UNSUPPORTED_TYPE

```
public static final int HTTP_UNSUPPORTED_TYPE
```

HTTP Status-Code 415: Unsupported Media Type.

**See Also:**
Constant Field Values

---

## HTTP_SERVER_ERROR

```
@Deprecated
public static final int HTTP_SERVER_ERROR
```

**Deprecated.** *it is misplaced and shouldn't have existed.*

HTTP Status-Code 500: Internal Server Error.

**See Also:**
Constant Field Values

---

## HTTP_INTERNAL_ERROR

```
public static final int HTTP_INTERNAL_ERROR
```

HTTP Status-Code 500: Internal Server Error.

**See Also:**
Constant Field Values

---

## HTTP_NOT_IMPLEMENTED

```
public static final int HTTP_NOT_IMPLEMENTED
```

HTTP Status-Code 501: Not Implemented.

**See Also:**
Constant Field Values

---

## HTTP_BAD_GATEWAY

```
public static final int HTTP_BAD_GATEWAY
```

HTTP Status-Code 502: Bad Gateway.

**See Also:**

Constant Field Values

---

### HTTP_UNAVAILABLE

```
public static final int HTTP_UNAVAILABLE
```

HTTP Status-Code 503: Service Unavailable.

**See Also:**
Constant Field Values

---

### HTTP_GATEWAY_TIMEOUT

```
public static final int HTTP_GATEWAY_TIMEOUT
```

HTTP Status-Code 504: Gateway Timeout.

**See Also:**
Constant Field Values

---

### HTTP_VERSION

```
public static final int HTTP_VERSION
```

HTTP Status-Code 505: HTTP Version Not Supported.

**See Also:**
Constant Field Values

## Constructor Detail

### HttpURLConnection

```
protected HttpURLConnection(URL u)
```

Constructor for the HttpURLConnection.

**Parameters:**
`u` - the URL

## Method Detail

### getHeaderFieldKey

```
public String getHeaderFieldKey(int n)
```

Returns the key for the $n^{th}$ header field. Some implementations may treat the $0^{th}$ header field as special, i.e. as the status line returned by the HTTP server. In this case, `getHeaderField(0)` returns the status line, but `getHeaderFieldKey(0)` returns null.

**Overrides:**

`getHeaderFieldKey` in class `URLConnection`

**Parameters:**

`n` - an index, where n >=0.

**Returns:**

the key for the $n^{th}$ header field, or `null` if the key does not exist.

---

## setFixedLengthStreamingMode

```
public void setFixedLengthStreamingMode(int contentLength)
```

This method is used to enable streaming of a HTTP request body without internal buffering, when the content length is known in advance.

An exception will be thrown if the application attempts to write more data than the indicated content-length, or if the application closes the OutputStream before writing the indicated amount.

When output streaming is enabled, authentication and redirection cannot be handled automatically. A HttpRetryException will be thrown when reading the response if authentication or redirection are required. This exception can be queried for the details of the error.

This method must be called before the URLConnection is connected.

**NOTE:** `setFixedLengthStreamingMode(long)` is recommended instead of this method as it allows larger content lengths to be set.

**Parameters:**

`contentLength` - The number of bytes which will be written to the OutputStream.

**Throws:**

`IllegalStateException` - if URLConnection is already connected or if a different streaming mode is already enabled.
`IllegalArgumentException` - if a content length less than zero is specified.

**Since:**

1.5

**See Also:**

`setChunkedStreamingMode(int)`

---

## setFixedLengthStreamingMode

```
public void setFixedLengthStreamingMode(long contentLength)
```

This method is used to enable streaming of a HTTP request body without internal buffering, when the content length is known in advance.

An exception will be thrown if the application attempts to write more data than the indicated content-length, or if the application closes the OutputStream before writing the indicated amount.

When output streaming is enabled, authentication and redirection cannot be handled automatically. A HttpRetryException will be thrown when reading the response if authentication or redirection are required. This exception can be queried for the details of the error.

This method must be called before the URLConnection is connected.

The content length set by invoking this method takes precedence over any value set by `setFixedLengthStreamingMode(int)`.

**Parameters:**

`contentLength` - The number of bytes which will be written to the OutputStream.

**Throws:**

`IllegalStateException` - if URLConnection is already connected or if a different streaming mode is already enabled.
`IllegalArgumentException` - if a content length less than zero is specified.

**Since:**

1.7

---

## setChunkedStreamingMode

```
public void setChunkedStreamingMode(int chunklen)
```

This method is used to enable streaming of a HTTP request body without internal buffering, when the content length is **not** known in advance. In this mode, chunked transfer encoding is used to send the request body. Note, not all HTTP servers support this mode.

When output streaming is enabled, authentication and redirection cannot be handled automatically. A HttpRetryException will be thrown when reading the response if authentication or redirection are required. This exception can be queried for the details of the error.

This method must be called before the URLConnection is connected.

**Parameters:**

`chunklen` - The number of bytes to write in each chunk. If chunklen is less than or equal to zero, a default value will be used.

**Throws:**

`IllegalStateException` - if URLConnection is already connected or if a different streaming mode is already enabled.

**Since:**

1.5

**See Also:**

`setFixedLengthStreamingMode(int)`

---

## getHeaderField

```
public String getHeaderField(int n)
```

Returns the value for the $n^{th}$ header field. Some implementations may treat the $0^{th}$ header field as special, i.e. as the status line returned by the HTTP server.

This method can be used in conjunction with the `getHeaderFieldKey` method to iterate through all the headers in the message.

**Overrides:**

`getHeaderField` in class `URLConnection`

**Parameters:**

`n` - an index, where n>=0.

**Returns:**

the value of the $n^{th}$ header field, or `null` if the value does not exist.

**See Also:**

`getHeaderFieldKey(int)`

---

## setFollowRedirects

```
public static void setFollowRedirects(boolean set)
```

Sets whether HTTP redirects (requests with response code 3xx) should be automatically followed by this class. True by default. Applets cannot change this variable.

If there is a security manager, this method first calls the security manager's `checkSetFactory` method to ensure the operation is allowed. This could result in a SecurityException.

**Parameters:**

`set` - a `boolean` indicating whether or not to follow HTTP redirects.

**Throws:**

`SecurityException` - if a security manager exists and its `checkSetFactory` method doesn't allow the operation.

**See Also:**

`SecurityManager.checkSetFactory()`, `getFollowRedirects()`

---

## getFollowRedirects

```
public static boolean getFollowRedirects()
```

Returns a `boolean` indicating whether or not HTTP redirects (3xx) should be automatically followed.

**Returns:**

`true` if HTTP redirects should be automatically followed, `false` if not.

**See Also:**

`setFollowRedirects(boolean)`

---

## setInstanceFollowRedirects

```
public void setInstanceFollowRedirects(boolean followRedirects)
```

Sets whether HTTP redirects (requests with response code 3xx) should be automatically followed by this `HttpURLConnection` instance.

The default value comes from followRedirects, which defaults to true.

**Parameters:**

`followRedirects` - a `boolean` indicating whether or not to follow HTTP redirects.

**Since:**

1.3

**See Also:**

`instanceFollowRedirects`, `getInstanceFollowRedirects()`

---

## getInstanceFollowRedirects

```
public boolean getInstanceFollowRedirects()
```

Returns the value of this `HttpURLConnection`'s `instanceFollowRedirects` field.

**Returns:**

the value of this `HttpURLConnection`'s `instanceFollowRedirects` field.

**Since:**

1.3

**See Also:**

instanceFollowRedirects, setInstanceFollowRedirects(boolean)

---

## setRequestMethod

```
public void setRequestMethod(String method)
                      throws ProtocolException
```

Set the method for the URL request, one of:

- GET
- POST
- HEAD
- OPTIONS
- PUT
- DELETE
- TRACE

are legal, subject to protocol restrictions. The default method is GET.

**Parameters:**

`method` - the HTTP method

**Throws:**

ProtocolException - if the method cannot be reset or if the requested method isn't valid for HTTP.

SecurityException - if a security manager is set and the "allowHttpTrace" NetPermission is not granted.

**See Also:**

getRequestMethod()

---

## getRequestMethod

```
public String getRequestMethod()
```

Get the request method.

**Returns:**

the HTTP request method

**See Also:**

setRequestMethod(java.lang.String)

---

## getResponseCode

```
public int getResponseCode()
                    throws IOException
```

Gets the status code from an HTTP response message. For example, in the case of the following status lines:

```
HTTP/1.0 200 OK
HTTP/1.0 401 Unauthorized
```

It will return 200 and 401 respectively. Returns -1 if no code can be discerned from the response (i.e., the response is not valid HTTP).

**Returns:**
the HTTP Status-Code, or -1

**Throws:**
IOException - if an error occurred connecting to the server.

---

## getResponseMessage

```
public String getResponseMessage()
                          throws IOException
```

Gets the HTTP response message, if any, returned along with the response code from a server. From responses like:

```
HTTP/1.0 200 OK
HTTP/1.0 404 Not Found
```

Extracts the Strings "OK" and "Not Found" respectively. Returns null if none could be discerned from the responses (the result was not valid HTTP).

**Returns:**
the HTTP response message, or `null`

**Throws:**
IOException - if an error occurred connecting to the server.

---

## getHeaderFieldDate

```
public long getHeaderFieldDate(String name,
                       long Default)
```

**Description copied from class: URLConnection**

Returns the value of the named field parsed as date. The result is the number of milliseconds since January 1, 1970 GMT represented by the named field.

This form of `getHeaderField` exists because some connection types (e.g., `http-ng`) have pre-parsed headers. Classes for that connection type can override this method and short-circuit the parsing.

**Overrides:**
getHeaderFieldDate in class URLConnection

**Parameters:**
`name` - the name of the header field.
`Default` - a default value.

**Returns:**
the value of the field, parsed as a date. The value of the `Default` argument is returned if the field is missing or malformed.

---

## disconnect

```
public abstract void disconnect()
```

Indicates that other requests to the server are unlikely in the near future. Calling disconnect() should not imply that this HttpURLConnection instance can be reused for other requests.

---

## usingProxy

```
public abstract boolean usingProxy()
```

Indicates if the connection is going through a proxy.

**Returns:**
a boolean indicating if the connection is using a proxy.

---

## getPermission

```
public Permission getPermission()
                        throws IOException
```

Returns a SocketPermission object representing the permission necessary to connect to the destination host and port.

**Overrides:**
getPermission in class URLConnection

**Returns:**
a `SocketPermission` object representing the permission necessary to connect to the destination host and port.

**Throws:**
IOException - if an error occurs while computing the permission.

---

## getErrorStream

```
public InputStream getErrorStream()
```

Returns the error stream if the connection failed but the server sent useful data nonetheless. The typical example is when an HTTP server responds with a 404, which will cause a FileNotFoundException to be thrown in connect, but the server sent an HTML help page with suggestions as to what to do.

This method will not cause a connection to be initiated. If the connection was not connected, or if the server did not have an error while connecting or if the server had an error but no error data was sent, this method will return null. This is the default.

**Returns:**

an error stream if any, null if there have been no errors, the connection is not connected or the server sent no useful data.