HTTP（HyperTextTransferProtocol）是超文本传输协议的缩写，它用于传送 WWW 方式的数据，关于 HTTP 协议的详细内容请参考 RFC2616。HTTP 协议采用了请求/响应模型。客户端向服务器发送一个请求，请求头包含请求的方法、URI、协议版本、以及包含请求修饰符、客户信息和内容的类似于 MIME 的消息结构。服务器以一个状态行作为响应，相应的内容包括消息协议的版本，成功或者错误编码加上包含服务器信息、实体元信息以及可能的实体内容。

通常 HTTP 消息包括客户机向服务器的请求消息和服务器向客户机的响应消息。这两种类型的消息由一个起始行，一个或者多个头域，一个只是头域结束的空行和可选的消息体组成。HTTP 的头域包括通用头，请求头，响应头和实体头四个部分。每个头域由一个域名，冒号（:）和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。

通用头域

通用头域包含请求和响应消息都支持的头域，通用头域包含 Cache-Control、 Connection、Date、Pragma、Transfer-Encoding、Upgrade、Via。对通用头域的扩展要求通讯双方都支持此扩展，如果存在不支持的通用头域，一般将会作为实体头域处理。下面简单介绍几个在 UPnP 消息中使用的通用头域。

Cache-Control 头域

Cache-Control 指定请求和响应遵循的缓存机制。在请求消息或响应消息中设置 Cache-Control 并不会修改另一个消息处理过程中的缓存处理过程。请求时的缓存指令包括 no-cache、no-store、max-age、 max-stale、min-fresh、only-if-cached，响应消息中的指令包括 public、private、no-cache、no- store、no-transform、must-revalidate、proxy-revalidate、max-age。各个消息中的指令含义如下：

Public 指示响应可被任何缓存区缓存。

Private 指示对于单个用户的整个或部分响应消息，不能被共享缓存处理。这允许服务器仅仅描述当用户的部分响应消息，此响应消息对于其他用户的请求无效。

no-cache 指示请求或响应消息不能缓存

no-store 用于防止重要的信息被无意的发布。在请求消息中发送将使得请求和响应消息都不使用缓存。

max-age 指示客户机可以接收生存期不大于指定时间（以秒为单位）的响应。

min-fresh 指示客户机可以接收响应时间小于当前时间加上指定时间的响应。

max-stale 指示客户机可以接收超出超时期间的响应消息。如果指定 max-stale 消息的值，那么客户机可以接收超出超时期指定值之内的响应消息。

Date 头域

Date 头域表示消息发送的时间，时间的描述格式由 rfc822 定义。例如，Date:Mon,31Dec200104:25:57GMT。Date 描述的时间表示世界标准时，换算成本地时间，需要知道用户所在的时区。

Pragma 头域

Pragma 头域用来包含实现特定的指令，最常用的是 Pragma:no-cache。在 HTTP/1.1 协议中，它的含义和 Cache- Control:no-cache 相同。

请求消息

请求消息的第一行为下面的格式：

MethodSPRequest-URISPHTTP-VersionCRLFMethod 表示对于 Request-URI 完成的方法，这个字段是大小写敏感的，包括 OPTIONS、GET、HEAD、POST、PUT、DELETE、 TRACE。方法 GET 和 HEAD 应该被所有的通用 WEB 服务器支持，其他所有方法的实现是可选的。GET 方法取回由 Request-URI 标识的信息。 HEAD 方法也是取回由 Request-URI 标识的信息，只是可以在响应时，不返回消息体。POST 方法可以请求服务器接收包含在请求中的实体信息，可以用于提交表单，向新闻组、BBS、邮件群组和数据库发送消息。

SP 表示空格。Request-URI 遵循 URI 格式，在此字段为星号（*）时，说明请求并不用于某个特定的资源地址，而是用于服务器本身。HTTP- Version 表示支持的 HTTP 版本，例如为 HTTP/1.1。CRLF 表示换行回车符。请求头域允许客户端向服务器传递关于请求或者关于客户机的附加信息。请求头域可能包含下列字段 Accept、Accept-Charset、Accept- Encoding、Accept-Language、Authorization、From、Host、If-Modified-Since、If- Match、If-None-Match、If-Range、If-Range、If-Unmodified-Since、Max-Forwards、 Proxy-Authorization、Range、Referer、User-Agent。对请求头域的扩展要求通讯双方都支持，如果存在不支持的请求头域，一般将会作为实体头域处理。

典型的请求消息：

```
GET http://download.microtool.de:80/somedata.exe
Host: download.microtool.de
Accept:*/*
Pragma: no-cache
Cache-Control: no-cache
Referer: http://download.microtool.de/
User-Agent:Mozilla/4.04[en](Win95;I;Nav)
Range:bytes=554554-
```

上例第一行表示 HTTP 客户端（可能是浏览器、下载程序）通过 GET 方法获得指定 URL 下的文件。棕色的部分表示请求头域的信息，绿色的部分表示通用头部分。

Host 头域

Host 头域指定请求资源的 Intenet 主机和端口号，必须表示请求 url 的原始服务器或网关的位置。HTTP /1.1 请求必须包含主机头域，否则系统会以 400 状态码返回。

Referer 头域

Referer 头域允许客户端指定请求 uri 的源资源地址，这可以允许服务器生成回退链表，可用来登陆、优化 cache 等。他也允许废除的或错误的连接由于维护的目的被追踪。如果请求的 uri 没有自己的 uri 地址，Referer 不能被发送。如果指定的是部分 uri 地址，则此地址应该是一个相对地址。

Range 头域

Range 头域可以请求实体的一个或者多个子范围。例如，

表示头 500 个字节：bytes=0-499

表示第二个 500 字节：bytes=500-999

表示最后 500 个字节：bytes=-500

表示 500 字节以后的范围：bytes=500-

第一个和最后一个字节：bytes=0-0,-1

同时指定几个范围：bytes=500-600,601-999

但是服务器可以忽略此请求头，如果无条件 GET 包含 Range 请求头，响应会以状态码 206（PartialContent）返回而不是以 200 （OK）。

User-Agent 头域

User-Agent 头域的内容包含发出请求的用户信息。

响应消息

响应消息的第一行为下面的格式：

HTTP-VersionSPStatus-CodeSPReason-PhraseCRLF

HTTP-Version 表示支持的 HTTP 版本，例如为 HTTP/1.1。Status- Code 是一个三个数字的结果代码。Reason-Phrase 给 Status-Code 提供一个简单的文本描述。Status-Code 主要用于机器自动识别，Reason-Phrase 主要用于帮助用户理解。Status-Code 的第一个数字定义响应的类别，后两个数字没有分类的作用。第一个数字可能取 5 个不同的值：

1xx:信息响应类，表示接收到请求并且继续处理

2xx:处理成功响应类，表示动作被成功接收、理解和接受

3xx:重定向响应类，为了完成指定的动作，必须接受进一步处理

4xx:客户端错误，客户请求包含语法错误或者是不能正确执行

5xx:服务端错误，服务器不能正确执行一个正确的请求

响应头域允许服务器传递不能放在状态行的附加信息，这些域主要描述服务器的信息和 Request-URI 进一步的信息。响应头域包含 Age、Location、Proxy-Authenticate、Public、Retry- After、Server、Vary、Warning、WWW-Authenticate。对响应头域的扩展要求通讯双方都支持，如果存在不支持的响应头域，一般将会作为实体头域处理。

典型的响应消息：

HTTP/1.0200OK

Date:Mon,31Dec200104:25:57GMT

Server:Apache/1.3.14(Unix)

Content-type:text/html

Last-modified:Tue,17Apr200106:46:28GMT

Etag:"a030f020ac7c01:1e9f"

Content-length:39725426

Content-range:bytes554554-40279979/40279980

上例第一行表示 HTTP 服务端响应一个 GET 方法。棕色的部分表示响应头域的信息，绿色的部分表示通用头部分，红色的部分表示实体头域的信息。

Location 响应头

Location 响应头用于重定向接收者到一个新 URI 地址。

Server 响应头

Server 响应头包含处理请求的原始服务器的软件信息。此域能包含多个产品标识和注释，产品标识一般按照重要性排序。

实体

请求消息和响应消息都可以包含实体信息，实体信息一般由实体头域和实体组成。实体头域包含关于实体的原信息，实体头包括 Allow、Content- Base、Content-Encoding、Content-Language、 Content-Length、Content-Location、Content-MD5、Content-Range、Content-Type、 Etag、Expires、Last-Modified、extension-header。extension-header 允许客户端定义新的实体头，但是这些域可能无法未接受方识别。实体可以是一个经过编码的字节流，它的编码方式由 Content-Encoding 或 Content-Type 定义，它的长度由 Content-Length 或 Content-Range 定义。

Content-Type 实体头

Content-Type 实体头用于向接收方指示实体的介质类型，指定 HEAD 方法送到接收方的实体介质类型，或 GET 方法发送的请求介质类型 Content-Range 实体头

Content-Range 实体头用于指定整个实体中的一部分的插入位置，他也指示了整个实体的长度。在服务器向客户返回一个部分响应，它必须描述响应覆盖的范围和整个实体长度。一般格式：

Content-Range:bytes-unitSPfirst-byte-pos-last-byte-pos/entity-legth

例如，传送头 500 个字节次字段的形式：Content-Range:bytes0- 499/1234 如果一个 http 消息包含此节（例如，对范围请求的响应或对一系列范围的重叠请求），Content-Range 表示传送的范围， Content-Length 表示实际传送的字节数。

Last-modified 实体头

Last-modified 实体头指定服务器上保存内容的最后修订时间。

应答头	说明
Allow	服务器支持哪些请求方法（如 GET、POST 等）。
Content-Encoding	文档的编码（Encode）方法。只有在解码之后才可以得到 Content-Type 头指定的内容类型。利用 gzip 压缩文档能够显著地减少 HTML 文档的下载时间。Java 的 GZIPOutputStream 可以很方便地进行 gzip 压缩，但只有 Unix 上的 Netscape 和 Windows 上的 IE 4、IE 5 才支持它。因此，Servlet 应该通过查看 Accept-Encoding 头（即 request.getHeader("Accept-Encoding")）检查浏览器是否支持 gzip，为支持 gzip 的浏览器返回经 gzip 压缩的 HTML 页面，为其他浏览器返回普通页面。
Content-Length	表示内容长度。只有当浏览器使用持久 HTTP 连接时才需要这个数据。如果你想要利用持久连接的优势，可以把输出文档写入 ByteArrayOutputStram，完成后查看其大小，然后把该值放入 Content-Length 头，最后通过 byteArrayStream.writeTo(response.getOutputStream()发送内容。
Content-Type	表示后面的文档属于什么 MIME 类型。Servlet 默认为 text/plain，但通常需要显式地指定为 text/html。由于经常要设置 Content-Type，因此 HttpServletResponse 提供了一个专用的方法 setContentTyep。
Date	当前的 GMT 时间。你可以用 setDateHeader 来设置这个头以避免转换时间格式的麻烦。
Expires	应该在什么时候认为文档已经过期，从而不再缓存它？
Last-Modified	文档的最后改动时间。客户可以通过 If-Modified-Since 请求头提供一个日期，该请求将被视为一个条件 GET，只有改动时间迟于指定时间的文档才会返回，否则返回一个 304（Not Modified）状态。Last-Modified 也可用 setDateHeader 方法来设置。
Location	表示客户应当到哪里去提取文档。Location 通常不是直接设置的，而是通过 HttpServletResponse 的 sendRedirect 方法，该方法同时设置状态代码为 302。

Refresh	表示浏览器应该在多少时间之后刷新文档，以秒计。除了刷新当前文档之外，你还可以通过setHeader("Refresh", "5; URL=http://host/path")让浏览器读取指定的页面。 注意这种功能通常是通过设置 HTML 页面 HEAD 区的＜META HTTP-EQUIV="Refresh" CONTENT="5;URL=http://host/path"＞实现，这是因为，自动刷新或重定向对于那些不能使用 CGI 或 Servlet 的 HTML 编写者十分重要。但是，对于 Servlet 来说，直接设置 Refresh 头更加方便。 注意 Refresh 的意义是“N 秒之后刷新本页面或访问指定页面”，而不是“每隔 N 秒刷新本页面或访问指定页面”。因此，连续刷新要求每次都发送一个 Refresh 头，而发送 204 状态代码则可以阻止浏览器继续刷新，不管是使用 Refresh 头还是＜META HTTP-EQUIV="Refresh" ...＞。 注意 Refresh 头不属于 HTTP 1.1 正式规范的一部分，而是一个扩展，但 Netscape 和 IE 都支持它。
Server	服务器名字。Servlet 一般不设置这个值，而是由 Web 服务器自己设置。
Set-Cookie	设置和页面关联的 Cookie。Servlet 不应使用 response.setHeader("Set-Cookie", ...)，而是应使用 HttpServletResponse 提供的专用方法 addCookie。参见下文有关 Cookie 设置的讨论。
WWW-Authenticate	客户应该在 Authorization 头中提供什么类型的授权信息？在包含 401（Unauthorized）状态行的应答中这个头是必需的。例如，response.setHeader("WWW-Authenticate", "BASIC realm=＼"executives＼"")。 注意 Servlet 一般不进行这方面的处理，而是让 Web 服务器的专门机制来控制受密码保护页面的访问（例如.htaccess）。