互聯網工程任務組 (IETF)
徵求意見：7231
過時：2616
更新：2817
類別：標準軌道
ISSN：2070-1721

R. 菲爾丁，埃德。
Adobe
J. Reschke，編輯。
greenbytes 2014
年 6 月

超文本傳輸協議 (HTTP/1.1)：語義和內容

抽象的

超文本傳輸協議 (HTTP) 是用於分佈式協作超文本信息系統的無狀態應用程序級協議。本文檔定義了 HTTP/1.1 消息的語義，如請求方法、請求標頭字段、響應狀態代碼和響應標頭字段所表達，以及消息的有效負載（元數據和正文內容)和內容協商機制。

本備忘錄的狀態

這是一份 Internet 標準跟踪文檔。

本文檔是 Internet 工程任務組 (IETF) 的產品。它代表了 IETF 社區的共識。它已接受公眾審查，並已被互聯網工程指導小組 (IESG) 批准發布。有關 Internet 標準的更多信息，請參閱 RFC 5741 的第 2 節。

有關本文檔的當前狀態、任何勘誤表以及如何提供反饋的信息，請訪問 http://www.rfc-editor.org/info/rfc7231。

目錄

一、簡介

每個超文本傳輸協議 (HTTP) 消息都是請求或響應。服務器在連接上偵聽請求，解析收到的每條消息，解釋與已識別請求目標相關的消息語義，並使用一個或多個響應消息響應該請求。客戶端構造請求消息來傳達特定意圖，檢查收到的響應以查看意圖是否已執行，並確定如何解釋結果。本文檔根據 [RFC7230] 中定義的體系結構定義了 HTTP/1.1 請求和響應語義。

HTTP 提供了一個統一的接口來與資源交互（第 2 節），無論其類型、性質或實現如何，通過表示的操作和傳輸（第 3 節）。

HTTP 語義包括每個請求方法定義的意圖（第 4 節）、可能在請求標頭字段中描述的那些語義的擴展（第 5 節）、指示機器可讀響應的狀態代碼的含義（第 6 節），以及響應頭字段中可能給出的其他控制數據和資源元數據的含義（第 7 節）。

本文檔還定義了表示元數據，這些元數據描述了接收者打算如何解釋有效負載、可能影響內容選擇的請求標頭字段以及統稱為“內容協商”的各種選擇算法（第 3.4 節）。

1.1.一致性和錯誤處理

本文檔中的關鍵詞“必須”、“不得”、“必需”、“應”、“不應”、“應該”、“不應”、“推薦”、“可以”和“可選”按照 [RFC2119] 中的描述進行解釋。

[RFC7230] 的第 2.5 節中定義了有關錯誤處理的一致性標準和注意事項。

1.2.語法符號

本規範使用 [RFC5234] 的增強巴科斯-諾爾形式 (ABNF) 表示法和 [RFC7230] 第 7 節中定義的列表擴展，允許使用“#”運算符（類似於 * 運算符如何表示重複）。附錄 C 描述了從其他文檔導入的規則。附錄 D 顯示了收集到的語法，其中所有列表運算符都擴展為標準 ABNF 表示法。

本規範使用 [RFC6365] 中定義的術語“字符”、“字符編碼方案”、“字符集”和“協議元素”。

2.資源

HTTP 請求的目標稱為“資源”。 HTTP 不限制資源的性質；它僅僅定義了一個可用於與資源交互的接口。每個資源都由統一資源標識符 (URI) 標識，如 [RFC7230] 的第 2.7 節所述。

當客戶端構造 HTTP/1.1 請求消息時，它會以各種形式之一發送目標 URI，如（[RFC7230] 的第 5.3 節)中所定義。收到請求時，服務器會為目標資源重建一個有效的請求 URI（[RFC7230] 的第 5.5 節）。

HTTP 的一個設計目標是將資源標識與請求語義分開，這可以通過將請求語義賦予請求方法（第 4 節)和一些請求修改標頭字段（第 5 節)來實現。如果方法語義與 URI 本身隱含的任何語義之間存在衝突，如第 4.2.1 節所述，則方法語義優先。

3. 交涉

考慮到資源可以是任何東西，並且 HTTP 提供的統一接口類似於一個窗口，通過該窗口，人們只能通過與另一端的某個獨立參與者的消息通信來觀察和操作這樣的東西，抽像是需要在我們的通信中表示（“取代”）該事物的當前或期望狀態。這種抽象稱為表示 [REST]。

就 HTTP 而言，“表示”是旨在反映給定資源的過去、當前或期望狀態的信息，其格式可以通過協議很容易地進行通信，並且由一組表示組成元數據和潛在無限的表示數據流。

源服務器可能提供或能夠生成多個表示，每個表示都旨在反映目標資源的當前狀態。在這種情況下，源服務器使用某種算法來選擇最適用於給定請求的表示之一，通常基於內容協商。此“選定表示”用於提供

用於評估條件請求 [RFC7232] 和構建對 GET 的 200（OK)和 304（未修改)響應的有效負載（第 4.3.1 節)的數據和元數據。

3.1.表示元數據

表示標題字段提供有關表示的元數據。當消息包含有效負載主體時，表示標頭字段描述如何解釋包含在有效負載主體中的表示數據。在對 HEAD 請求的響應中，表示標頭字段描述了表示數據，如果同一請求是 GET，則表示數據將包含在有效負載主體中。

以下標頭字段傳達表示元數據：

標題字段名稱	定義在...
內容類型	第 3.1.1.5 節
內容編碼	第 3.1.2.2 節
內容語言	第 3.1.3.2 節
內容位置	第 3.1.4.2 節

3.1.1.處理表示數據

3.1.1.1.媒體類型

HTTP 在 Content-Type（第 3.1.1.5 節)和 Accept（第 5.3.2 節)標頭字段中使用 Internet 媒體類型 [RFC2046]，以提供開放和可擴展的數據類型化和類型協商。

媒體類型定義了數據格式和各種處理模型：如何根據接收數據的每個上下文處理數據。

```
media-type = type     /     subtype *( OWS    ;    OWS parameter ) = token 類型 subtype =
                                token
```

類型/子類型後面可以跟以名稱=值對形式的參數。

```
範圍                                = 令牌“=”（令牌/帶引號的字符串）
```

類型、子類型和參數名稱標記不區分大小寫。
參數值可能區分大小寫，也可能不區分大小寫，具體取決於參數名稱的語義。參數的存在與否可能對媒體類型的處理很重要，這取決於它在媒體類型註冊表中的定義。

與令牌生產相匹配的參數值可以作為令牌或在引號字符串中傳輸。引用和未引用的值是等效的。例如，以下示例都是等價的，但為了保持一致性，首選第一個：

```
文本/html；字符集=utf-8 文本/html；字符
集=UTF-8
文本/HTML;Charset= utf-8 文本/html;字
符集 =“utf-8”
```

互聯網媒體類型應該根據 [BCP13] 中定義的程序向 IANA 註冊。

注意：與其他標頭字段中的一些類似構造不同，媒體類型參數不允許在“=”字符周圍使用空格（甚至是“壞”空格）。

3.1.1.2。字符集

HTTP 使用字符集名稱來指示或協商文本表示的字符編碼方案 [RFC6365]。字符集由不區分大小寫的標記標識。

```
字符集 = 標記
```

根據 [RFC2978] 中定義的程序，字符集名稱應該在 IANA“字符集”註冊表 (<http://www.iana.org/assignments/character-sets>) 中註冊。

3.1.1.3.規範化和文本默認值

互聯網媒體類型以規範形式註冊，以便在具有不同本機編碼格式的系統之間進行互操作。

由於多用途 Internet 郵件擴展 (MIME) [RFC2045] 描述的許多相同原因，通過 HTTP 選擇或傳輸的表示應該採用規範形式。然而，電子郵件部署的性能特徵（即，存儲消息並將消息轉發給對等方）與 HTTP 和 Web（基於服務器的信息服務）的常見特徵有很大不同。

此外，為了與舊郵件傳輸協議兼容而對 MIME 進行的限制不適用於 HTTP（請參閱附錄 A）。

MIME 的規範形式要求“文本”類型的媒體子類型使用 CRLF 作為文本換行符。 HTTP 允許傳輸帶有純 CR 或 LF 的文本媒體單獨代表一個換行符，當這樣的換行符對於整個表示是一致的。 HTTP 發送方可以生成，並且接收方必須能夠解析由 CRLF、裸 CR 或裸 LF 組成的文本媒體中的換行符。此外，HTTP 中的文本媒體不限於分別將八位字節 13 和 10 用於 CR 和 LF 的字符集。這種關於換行符的靈活性僅適用於已分配“文本”媒體類型的表示中的文本；它不適用於有效負載主體之外的“多部分”類型或 HTTP 元素（例如，標頭字段）。

如果表示使用內容編碼進行編碼，則基礎數據在編碼之前應該採用上面定義的形式。

3.1.1.4。多部分類型

MIME 提供了多種“多部分”類型 將一個或多個表示封裝在單個消息主體中。所有多部分類型都共享一個通用語法，如 [RFC2046] 的第 5.1.1 節中所定義，並包含一個邊界參數作為媒體類型值的一部分。消息體本身就是一個協議元素；發送者必須只生成 CRLF 來表示正文部分之間的換行符。

HTTP 消息框架不使用多部分邊界作為消息正文長度的指示符，儘管生成或處理有效負載的實現可能會使用它。例如，“multipart/form-data”類型通常用於在請求中攜帶錶單數據，如 [RFC2388] 中所述，而“multipart/ byteranges”類型由該規範定義用於某些 206（部分內容）響應 [RFC7233]。

3.1.1.5。內容類型

“Content-Type”頭字段指示相關表示的媒體類型：消息有效負載中包含的表示或所選表示，由消息語義確定。在 Content-Encoding 指示的任何內容編碼被解碼後，指示的媒體類型定義了數據格式以及接收者打算如何在接收到的消息語義範圍內處理該數據。

內容類型 = 媒體類型

媒體類型在第 3.1.1.1 節中定義。該字段的一個例子是

```
內容類型：文本/html；字符集=ISO-8859-4
```

生成包含有效負載主體的消息的發送方應該在該消息中生成 Content-Type 頭字段，除非發送方不知道所包含的表示的預期媒體類型。如果 Content-Type 頭字段不存在，接收者可以假定媒體類型為“application/octet-stream”

（[RFC2046]，第 4.5.1 節）或檢查數據以確定其類型。

在實踐中，資源所有者並不總是正確配置他們的源服務器來為給定的表示提供正確的內容類型，結果是一些客戶端將檢查有效負載的內容並覆蓋指定的類型。這樣做的客戶端冒著得出錯誤結論的風險，這可能會暴露額外的安全風險（例如，“權限升級”）。此外，不可能通過檢查數據格式來確定發送者的意圖：許多數據格式匹配僅在處理語義上不同的多種媒體類型。鼓勵實施者提供一種在使用時禁用這種“內容嗅探”的方法。

3.1.2.壓縮或完整性編碼

3.1.2.1.內容編碼

內容編碼值指示已經或可以應用於表示的編碼轉換。內容編碼主要用於允許對錶示進行壓縮或以其他方式進行有用的轉換，而不會丟失其底層媒體類型的身份並且不會丟失信息。通常，表示以編碼形式存儲，直接傳輸，並且僅由最終接收者解碼。

```
內容編碼 = 令牌
```

所有內容編碼值都不區分大小寫，並且應該在“HTTP 內容編碼註冊表”中註冊，如第 8.4 節中所定義。它們用於 Accept-Encoding（第 5.3.4 節）和 Content-Encoding（第 3.1.2.2 節）頭字段。

本規範定義了以下內容編碼值：

compress（和 x-compress）：參見 [RFC7230] 的第 4.2.1 節。

deflate：參見 [RFC7230] 的第 4.2.2 節。

gzip（和 x-gzip）：參見 [RFC7230] 的第 4.2.3 節。

3.1.2.2.內容編碼

“Content-Encoding”報頭字段指示哪些內容編碼已應用於表示，超出了媒體類型固有的編碼，因此必須應用哪些解碼機制才能獲得 Content-引用的媒體類型中的數據類型標題字段。 Content-Encoding 主要用於允許壓縮表示的數據而不丟失其底層媒體類型的標識。

內容編碼= 1#內容編碼

它的一個使用例子是

內容編碼：gzip

如果一種或多種編碼已應用於表示，則應用編碼的發送方必須生成一個 Content-Encoding 標頭字段，該字段按應用順序列出內容編碼。關於編碼參數的附加信息可以由本規範未定義的其他頭字段提供。

與 Transfer-Encoding（[RFC7230] 的第 3.3.1 節)不同，Content-Encoding 中列出的編碼是表示的一個特徵；表示是根據編碼形式定義的，除非元數據定義中另有說明，否則關於表示的所有其他元數據都是關於編碼形式的。

通常，表示僅在渲染或類似使用之前才被解碼。

如果媒體類型包含固有編碼，例如始終壓縮的數據格式，則該編碼不會在 Content-Encoding 中重述，即使它恰好與其中一種內容編碼的算法相同。這樣的內容編碼只有在出於某種奇怪的原因被第二次應用以形成表示時才會被列出。同樣，源服務器可能會選擇將相同的數據發佈為多種表示形式，不同之處僅在於編碼是否定義為 Content-Type 的一部分

或內容編碼，因為某些用戶代理在處理每個響應時會有不同的行為（例如，打開“另存為...”對話框而不是自動解壓縮和呈現內容）。

如果請求消息中的表示具有不可接受的內容編碼，則源服務器可以使用狀態代碼 415（不支持的媒體類型）進行響應。

3.1.3.觀眾語言

3.1.3.1.語言標籤

語言標籤，如 [RFC5646] 中所定義，識別人類口頭、書面或以其他方式傳達的自然語言，用於與其他人交流信息。計算機語言被明確排除在外。

HTTP 在 Accept-Language 和 Content-Language 標頭字段中使用語言標籤。 Accept-Language 使用第 5.3.5 節中定義的更廣泛的語言範圍產生式，而 Content-Language 使用下面定義的語言標籤產生式。

language-tag = <語言標籤，參見 [RFC5646]，第 2.1 節>

語言標籤是一個或多個不區分大小寫的子標籤的序列，每個子標籤由連字符（“-”、%x2D）分隔。在大多數情況下，語言標籤由一個主要語言子標籤組成，該子標籤標識一個廣泛的相關語言家族（例如，“en”= 英語），它後面可選地跟有一系列細化或縮小該語言範圍的子標籤（例如，“en-CA”= 在加拿大交流的各種英語）。語言標記中不允許使用空格。示例標籤包括：

fr, en-US, es-419, az-Arab, x-pig-latin, man-Nkoo-GN

有關更多信息，請參閱 [RFC5646]。

3.1.3.2.內容語言

“Content-Language”標題字段描述了表示的預期受眾的自然語言。請注意，這可能不等同於表示中使用的所有語言。

內容語言= 1#language-tag

語言標籤在第 3.1.3.1 節中定義。 Content-Language 的主要目的是允許用戶根據自己的首選語言來識別和區分錶示。

因此，如果內容僅供丹麥語讀者使用，則相應的字段是

內容語言：da

如果未指定 Content-Language，則默認內容適用於所有語言的受眾。這可能意味著發件人不認為它特定於任何自然語言，或者發件人不知道它打算用於哪種語言。

可以為面向多個受眾的內容列出多種語言。例如，同時以毛利語原版和英文版呈現的《懷唐伊條約》的翻版要求

內容語言：mi, en

然而，僅僅因為一種表示中存在多種語言並不意味著它適用於多種語言的受眾。一個例子是初學者的語言入門，例如“拉丁語第一課”，它顯然旨在供英語讀者使用。在這種情況下，Content-Language 應該只包含“en”。

Content-Language 可以應用於任何媒體類型 它不限於文本文檔。

3.1.4.鑑別

3.1.4.1.識別表示

當在消息有效負載中傳輸完整或部分錶示時，通常希望發送方提供或接收方確定與該表示對應的資源的標識符。

對於請求消息：

o 如果請求有一個 Content-Location 頭字段，那麼發送者斷言有效載荷是由 Content-Location 字段值標識的資源的表示。但是，除非可以通過其他方式（本規範未定義）對其進行驗證，否則此類斷言是不可信的。該信息對於修訂歷史鏈接可能仍然有用。

o 否則，有效載荷是無法識別的。

對於響應消息，將按順序應用以下規則，直到找到匹配項：

1. 如果請求方法是 GET 或 HEAD 並且響應狀態代碼是 200（OK）、204（無內容）、206（部分內容)或 304（未修改），則有效負載表示由有效請求 URI（[RFC7230] 的第 5.5 節）。

2. 如果請求方法是 GET 或 HEAD 並且響應狀態代碼是 203（非權威信息），則有效負載是中介提供的目標資源的潛在修改或增強表示。

3. 如果響應有一個 Content-Location 頭字段，並且它的字段值是對與有效請求 URI 相同的 URI 的引用，則有效負載是有效請求 URI 標識的資源的表示。

4. 如果響應有一個 Content-Location 頭字段及其 field-value 是對與有效請求 URI 不同的 URI 的引用，然後發送方斷言有效負載是由 Content-Location field-value 標識的資源的表示。但是，除非可以通過其他方式（本規範未定義)對其進行驗證，否則此類斷言是不可信的。

5.否則，載荷不明。

3.1.4.2.內容位置

“Content-Location”報頭字段引用一個 URI，該 URI 可用作與此消息有效負載中的表示相對應的特定資源的標識符。換句話說，如果在生成此消息時在此 URI 上執行 GET 請求，則 200 (OK) 響應將包含與此消息中作為負載包含的相同表示。

內容位置 = 絕對 URI / 部分 URI

Content-Location 值不能替代有效的 Request URI（[RFC7230] 的第 5.5 節）。它是表示元數據。它與 [RFC2557] 第 4 節中為 MIME 正文部分定義的同名標頭字段具有相同的語法和語義。然而，它在 HTTP 消息中的出現對 HTTP 接收者有一些特殊的含義。

如果 Content-Location 包含在 2xx（成功)響應消息中並且其值引用（在轉換為絕對形式後)與有效請求 URI 相同的 URI ，則接收者可以將有效負載視為當前表示該資源在消息發起日期指示的時間 。對於 GET（第 4.3.1 節)或 HEAD（第 4.3.2 節)請求 ，這與服務器未提供 Content-Location 時的默認語義相同 。對於像 PUT（第 4.3.4 節)或 POST（第 4.3.3 節)這樣的狀態更改請求 ，它意味著服務器的響應包含該資源的新表示，從而將其與可能僅報告有關操作的表示區分開來（例如，“它成功了！”） 。這允許創作應用程序更新其本地副本 ，而無需後續 GET 請求。

如果 Content-Location 包含在 2xx（成功)響應消息中 ，並且其字段值引用與有效請求 URI 不同的 URI ，則源服務器聲稱該 URI 是對應於所包含的不同資源的標識符表示 。只有當兩個標識符共享相同的資源所有者時 ，這樣的聲明才能被信任 ，這不能通過 HTTP 以編程方式確定。

o 對於 GET 或 HEAD 請求的響應 ，這表明有效請求 URI 指的是受內容協商約束的資源 ，Content-Location 字段值是所選表示的更具體的標識符。

o 對於狀態改變方法的 201（已創建)響應 ，a
與 Location 字段值相同的 Content-Location 字段值指示此有效負載是新創建資源的當前表示。

o 否則 ，這樣的 Content-Location 表示此有效負載是報告所請求操作狀態的表示 ，並且相同的報告在給定的 URI 處可用（用於將來使用 GET 訪問） 。例如 ，通過 POST 請求進行的購買交易可能包含收據文檔作為 200 (OK) 響應的有效負載； Content-Location 字段值提供了一個標識符 ，用於將來檢索同一收據的副本。

在請求消息中發送 Content-Location 的用戶代理聲明其值是指用戶代理最初從何處獲得所附表示的內容（在該用戶代理進行任何修改之前） 。換句話說 ，用戶代理正在提供指向原始表示源的反向鏈接。

接收請求消息中的 Content-Location 字段的源服務器必須將信息視為臨時請求上下文，而不是作為表示的一部分逐字保存的元數據。原始服務器可以使用該上下文來指導處理請求或將其保存以供其他用途，例如在源鏈接或版本控制元數據中。但是，源服務器不得使用此類上下文信息來更改請求語義。

例如，如果客戶端對協商資源發出 PUT 請求，並且源服務器接受該 PUT（沒有重定向），則該資源的新狀態應與該 PUT 中提供的一種表示一致；Content-Location 不能用作反向內容選擇標識符的一種形式來僅更新協商表示之一。如果用戶代理想要後一種語義，它會直接將 PUT 應用於 Content-Location URI。

3.2.表示數據

與 HTTP 消息關聯的表示數據要么作為消息的有效負載主體提供，要么由消息語義和有效請求 URI 引用。表示數據採用由表示元數據報頭字段定義的格式和編碼。

表示數據的數據類型通過標題字段 Content-Type 和 Content-Encoding 確定。這些定義了一個兩層的有序編碼模型：

表示數據 := 內容編碼（內容類型（位））

3.3.有效載荷語義

一些 HTTP 消息將完整或部分錶示作為消息“有效負載”進行傳輸。在某些情況下，有效載荷可能僅包含相關表示的頭部字段（例如，對 HEAD 的響應）或僅包含表示數據的某些部分（例如，206（部分內容）狀態代碼）。

請求中有效載荷的用途由方法語義定義。例如，如果成功應用請求，PUT 請求（第 4.3.4 節）的有效負載中的表示表示目標資源的期望狀態，而 POST 請求（第 4.3.3 節）的有效負載中的表示表示目標資源要處理的信息。

在響應中，負載的目的由請求方法和響應狀態代碼定義。例如，對 GET（第 4.3.1 節)的 200（OK)響應的有效負載表示目標資源的當前狀態，正如在消息發起日期（第 7.1.1.2 節)時觀察到的那樣，而有效負載對 POST 的響應中的相同狀態代碼可能表示處理結果或應用處理後目標資源的新狀態。帶有錯誤狀態代碼的響應消息通常包含一個表示錯誤條件的有效負載，這樣它就描述了錯誤狀態以及建議的後續步驟來解決它。

專門描述有效負載而非關聯表示的報頭字段稱為“有效負載報頭字段”。有效負載標頭字段在本規範的其他部分中定義，因為它們對消息解析有影響。

標題字段名稱	定義在...
內容長度	[RFC7230] 第 3.3.2 節
內容範圍	[RFC7233] 第 4.2 節
預告片	[RFC7230] 的第 4.4 節
傳輸編碼	[RFC7230] 第 3.3.1 節

3.4.內容協商

當響應傳遞有效載荷信息時，無論是指示成功還是錯誤，源服務器通常有不同的方式來表示該信息；例如，以不同的格式、語言或編碼。同樣，不同的用戶或用戶代理可能具有不同的能力、特徵或偏好，這些可能會影響在可用的表示中最好交付的表示。為此，HTTP 提供了內容協商機制。

本規範定義了兩種可以在協議中可見的內容協商模式：“主動”模式，其中服務器根據用戶代理聲明的偏好選擇表示，以及“反應式”協商，其中服務器為以下內容提供表示列表可供選擇的用戶代理。內容協商的其他模式包括“條件內容”，其中表示由基於用戶代理參數選擇性呈現的多個部分組成，“活動內容”，其中表示包含一個腳本，該腳本根據用戶代理特徵和“透明內容協商”（[RFC2295]），其中內容

選擇由中介進行。這些模式並不相互排斥，每個模式都在適用性和實用性方面有所取捨。

請注意，在所有情況下，HTTP 都不知道資源語義。
隨著時間的推移和內容協商的不同維度，源服務器響應請求的一致性，以及因此隨著時間的推移觀察到的資源表示的“相同性”，完全由選擇或生成這些響應的任何實體或算法決定。 HTTP 不關注幕後的人。

3.4.1.主動談判

當用戶代理在請求中發送內容協商首選項以鼓勵位於服務器的算法選擇首選表示時，這稱為主動協商（也稱為服務器驅動協商）。選擇是基於與請求中提供的各種信息（包括第 5.3 節的顯式協商字段和隱式協商字段）相比，響應的可用表示（它可能變化的維度，例如語言、內容編碼等）特性，例如客戶端的網絡地址或 User-Agent 字段的一部分。

當從可用表示中進行選擇的算法難以向用戶代理描述時，或者當服務器希望將其“最佳猜測”連同第一個響應（希望避免回合）發送給用戶代理時，主動協商是有利的如果“最佳猜測”對用戶來說足夠好，則後續請求的行程延遲）。為了改進服務器的猜測，用戶代理可以發送描述其偏好的請求標頭字段。

主動協商有嚴重的缺點：

o 服務器不可能準確地確定對於任何給定用戶什麼可能是“最好的”，因為這需要完全了解用戶代理的能力和響應的預期用途（例如，用戶是否想要在屏幕上查看還是打印在紙上？）；

o 讓用戶代理在每個請求中描述它的能力可能是非常低效的（假設只有一小部分響應有多種表示）並且對用戶隱私有潛在風險；

o 它使源服務器的實現和生成對請求的響應的算法變得複雜；和，

o 它限制了共享緩存響應的可重用性。

用戶代理不能依賴於始終如一地遵守主動協商首選項，因為源服務器可能不會對請求的資源實施主動協商，或者可能決定發送不符合用戶代理首選項的響應比發送 406 (不可接受)回應。

Vary 報頭字段（第 7.1.4 節)通常在響應中發送以進行主動協商，以指示在選擇算法中使用了請求信息的哪些部分。

3.4.2.被動談判

對於反應式協商（又名，代理驅動的協商），用戶代理在收到來自源服務器的初始響應後執行最佳響應表示的選擇（無論狀態代碼如何），其中包含用於替代表示的資源列表。如果用戶代理對初始響應表示不滿意，它可以對根據列表中包含的元數據選擇的一個或多個替代資源執行 GET 請求，以獲得該響應的不同形式的表示。備選方案的選擇可能由用戶代理自動執行，或由用戶從生成的（可能是超文本)菜單中選擇手動執行。

請注意，上面提到的是響應的表示，通常不是資源的表示。僅當提供替代表示的響應具有作為目標資源表示的語義（例如，對 GET 請求的 200（OK)響應)或具有以下語義時，替代表示才被視為目標資源的表示提供指向目標資源替代表示的鏈接（例如，對 GET 請求的 300（多項選擇)響應）。

服務器可能會選擇不發送初始表示，而不是備選列表，從而表明用戶代理的反應式協商是首選。例如，在具有 300（多項選擇)和 406（不可接受)狀態代碼的響應中列出的備選方案包括有關可用表示的信息，以便用戶或用戶代理可以通過做出選擇來做出反應。

當響應在常用維度（例如類型、語言或編碼)上發生變化時，當源服務器無法通過檢查請求來確定用戶代理的能力時，以及通常當公共緩存用於分發時，反應式協商是有利的服務器負載並減少網絡使用。

反應式協商的缺點是向用戶代理傳輸替代列表，如果在標頭部分傳輸，這會降低用戶感知的延遲，並且需要第二次請求以獲得替代表示。此外，本規範沒有定義支持自動選擇的機制，儘管它不阻止將這種機制開發為擴展。

4.請求方法

4.1.概述

請求方法令牌是請求語義的主要來源；它表明客戶提出此請求的目的以及客戶對成功結果的期望。

如果這些額外的語義不與方法衝突，則請求方法的語義可能會通過請求中出現的某些標頭字段的語義進一步專門化（第 5 節）。例如，客戶端可以發送條件請求標頭字段（第 5.2 節)以使請求的操作以目標資源的當前狀態為條件（[RFC7232]）。

方法=令牌

HTTP 最初設計為可用作分佈式對象系統的接口。請求方法被設想為將語義應用於目標資源，其方式與在已識別對像上調用定義的方法將應用語義的方式大致相同。方法標記區分大小寫，因為它可能被用作使用區分大小寫方法的基於對象的系統的網關

名字。

與分佈式對像不同，HTTP 中的標準化請求方法不是特定於資源的，因為統一接口在基於網絡的系統 [REST] 中提供了更好的可見性和重用性。一旦定義，標準化方法在應用於任何資源時應該具有相同的語義，儘管每個資源自行決定是否實現或允許這些語義。

本規範定義了 HTTP 中常用的許多標準化方法，如下表所示。按照慣例，標準化方法以全大寫 US-ASCII 字母定義。

方法	描述	秒。
獲取	傳輸目標的當前表示	4.3.1	
頭	與 GET 相同，但只傳輸狀態行	4.3.2	
發布	對	執行特定於資源的處理	4.3.3	
放	替換	目標資源	4.3.4	
刪除	刪除目標資源	4.3.5	
選項	描述目標資源的通信選項	4.3.7	
追蹤	沿著到目標資源的路徑進行消息環回測試	4.3.8	
連接	與由目標資源標識的伺服器建立隧道	4.3.6	

所有通用服務器必須支持 GET 和 HEAD 方法。
所有其他方法都是可選的。

本規范范圍之外的其他方法已標準化用於 HTTP。所有此類方法都應在 IANA 維護的“超文本傳輸協議 (HTTP) 方法註冊表”中註冊，如第 8.1 節中所定義。

目標資源允許的方法集可以在 Allow 頭字段中列出（第 7.4.1 節）。但是，允許的方法集可以動態更改。當接收到原始服務器無法識別或未實現的請求方法時，原始服務器應該使用 501（未實現）狀態代碼進行響應。當接收到源服務器已知但目標資源不允許的請求方法時，源服務器應該響應 405（方法不允許）狀態代碼。

4.2.通用方法屬性

4.2.1.安全方法

如果請求方法定義的語義本質上是只讀的，則請求方法被認為是“安全的”；即，客戶端不請求也不期望由於對目標資源應用安全方法而導致源服務器上的任何狀態更改。同樣，合理使用安全方法預計不會對源服務器造成任何傷害、財產損失或異常負擔。

安全方法的這種定義不會阻止實現包含潛在有害的行為、不完全只讀的行為或在調用安全方法時導致副作用的行為。然而，重要的是客戶沒有要求額外的行為並且不能為此負責。例如，大多數服務器都會在每次響應完成時附加請求信息以訪問日誌文件，無論使用何種方法，這被認為是安全的，即使日誌存儲可能已滿並導致服務器崩潰。同樣，通過選擇 Web 上的廣告發起的安全請求通常會產生向廣告帳戶收費的副作用。

在本規範定義的請求方法中，GET、HEAD、OPTIONS 和 TRACE 方法被定義為安全的。

區分安全和不安全方法的目的是讓自動檢索過程（蜘蛛）和緩存性能優化（預取）工作而不必擔心造成傷害。此外，它允許用戶代理在處理可能不受信任的內容時對不安全方法的自動使用應用適當的約束。

用戶代理在向用戶呈現潛在操作時應該區分安全和不安全方法，以便用戶可以在請求之前意識到不安全操作。

當構建資源時，有效請求 URI 中的參數具有選擇操作的效果，資源所有者有責任確保操作與請求方法語義一致。例如，基於 Web 的內容編輯軟件通常在查詢參數內使用操作，例如“page?do=delete”。如果此類資源的目的是執行不安全的操作，則資源所有者必須在使用安全請求方法訪問時禁用或不允許該操作。如果不這樣做，當自動化進程為了鏈接維護、預取、構建搜索索引等而對每個 URI 引用執行 GET 時，將導致不幸的副作用。

4.2.2.冪等方法

如果使用該方法的多個相同請求對服務器的預期效果與單個此類請求的效果相同，則請求方法被認為是“冪等的”。在本規範定義的請求方法中，PUT、DELETE 和安全請求方法是冪等的。

與安全的定義一樣，冪等屬性僅適用於用戶請求的內容；服務器可以自由地單獨記錄每個請求，保留修訂控制歷史記錄，或為每個冪等請求實現其他非冪等副作用。

冪等方法之所以與眾不同，是因為如果在客戶端能夠讀取服務器響應之前發生通信故障，則可以自動重複請求。例如，如果客戶端發送 PUT 請求，並且在收到任何響應之前關閉了底層連接，則客戶端可以建立新連接並重試冪等請求。它知道重複請求將產生相同的預期效果，即使原始請求成功，但響應可能不同。

4.2.3.可緩存的方法

請求方法可以定義為“可緩存”的，以表明允許存儲對它們的響應以供將來重用；具體要求參見 [RFC7234]。通常，不依賴於當前或權威響應的安全方法被定義為可緩存的；該規範將 GET、HEAD 和 POST 定義為可緩存的，儘管絕大多數緩存實現僅支持 GET 和 HEAD。

4.3.方法定義

4.3.1.得到

GET 方法請求傳輸目標資源的當前選定表示。 GET 是信息檢索的主要機制，也是幾乎所有性能優化的重點。

因此，當人們談到通過 HTTP 檢索一些可識別信息時，他們通常指的是發出 GET 請求。

人們很容易將資源標識符視為遠程文件系統路徑名，並將表示視為此類文件內容的副本。事實上，這就是實現了多少資源（有關安全考慮，請參閱第 9.1 節）。然而，在實踐中沒有這樣的限制。資源的 HTTP 接口很可能被實現為內容對象樹、各種數據庫記錄的編程視圖或其他信息系統的網關。即使當 URI 映射機制綁定到文件系統時，源服務器也可能被配置為將請求作為輸入來執行文件，並將輸出作為表示發送，而不是直接傳輸文件。無論如何，只有源服務器需要知道它的每個資源如何

標識符對應於一個實現以及每個實現如何在對 GET 的響應中設法選擇和發送目標資源的當前表示。

客戶端可以將 GET 的語義更改為“範圍請求”，通過在請求中發送 Range 標頭字段（[RFC7233]），請求僅傳輸所選表示的某些部分。

GET 請求消息中的負載沒有定義的語義；在 GET 請求上發送有效負載主體可能會導致某些現有實現拒絕該請求。

對 GET 請求的響應是可緩存的；緩存可以使用它來滿足後續的 GET 和 HEAD 請求，除非 Cache-Control 頭字段（[RFC7234] 的第 5.2 節）另有說明。

4.3.2.頭

HEAD 方法與 GET 相同，除了服務器不得在響應中發送消息主體（即，響應在標頭部分的末尾終止）。服務器應該發送相同的頭字段來響應 HEAD 請求，如果請求是 GET，它會發送相同的頭字段，除了有效負載頭字段（第 3.3 節)可以省略。此方法可用於在不傳輸表示數據的情況下獲取有關所選表示的元數據，並且通常用於測試超文本鏈接的有效性、可訪問性和最近修改。

HEAD 請求消息中的有效負載沒有定義的語義；在 HEAD 請求上發送有效負載主體可能會導致某些現有實現拒絕該請求。

對 HEAD 請求的響應是可緩存的；緩存可以使用它來滿足後續的 HEAD 請求，除非 Cache-Control 頭字段（[RFC7234] 的第 5.2 節)另有說明。 HEAD 響應也可能對先前緩存的 GET 響應產生影響；請參閱 [RFC7234] 的第 4.3.5 節。

4.3.3.郵政

POST 方法請求目標資源根據資源自己的特定語義處理包含在請求中的表示。例如，POST 用於以下功能（以及其他功能）：

o 提供數據塊，例如輸入到 HTML 中的字段
形式，到數據處理過程；

o 將消息發佈到公告板、新聞組、郵件列表、
博客或類似的文章組；

o 創建一個尚未被源服務器識別的新資源；和

o 將數據附加到資源的現有表示中。

源服務器根據處理POST請求的結果選擇合適的狀態碼來指示響應語義；本規範定義的幾乎所有狀態代碼都可能在對 POST 的響應中收到（例外情況是 206（部分內容）、304（未修改)和 416（範圍不可滿足））。

如果作為成功處理 POST 請求的結果在源服務器上創建了一個或多個資源，源服務器應該發送一個 201（已創建)響應，其中包含一個 Location 頭字段，為創建的主要資源提供標識符（第 7.1 節）.2) 以及在引用新資源時描述請求狀態的表示。

對 POST 請求的響應只有在包含明確的新鮮度信息時才可緩存（請參閱 [RFC7234] 的第 4.2.1 節）。

但是，POST 緩存並未得到廣泛實施。對於源服務器希望客戶端能夠緩存 POST 的結果以供稍後 GET 重用的情況，源服務器可以發送包含結果和 Content-Location 的 200（OK)響應與 POST 的有效請求 URI（第 3.1.4.2 節)具有相同值的標頭字段。

如果處理 POST 的結果等同於現有資源的表示，源服務器可以通過發送 303（參見其他)響應將用戶代理重定向到該資源，並在 Location 字段中包含現有資源的標識符。這具有為用戶代理提供資源標識符並通過更適合共享緩存的方法傳輸表示的好處，但如果用戶代理尚未緩存表示，則會以額外請求為代價。

4.3.4.放

PUT 方法請求創建目標資源的狀態或將其替換為請求消息負載中包含的表示所定義的狀態。給定表示的成功 PUT 表明對同一目標資源的後續 GET 將導致在 200 (OK) 響應中發送等效表示。但是，不能保證

這樣的狀態變化是可以觀察到的，因為在接收到任何後續 GET 之前，目標資源可能會被其他用戶代理並行操作，或者可能會受到源服務器的動態處理。成功的響應僅意味著用戶代理的意圖在原始服務器處理時已實現。

如果目標資源沒有當前表示並且 PUT 成功創建了一個，則源服務器必須通過發送 201（已創建）響應通知用戶代理。如果目標資源確實有當前表示並且該表示已根據封閉表示的狀態成功修改，則源服務器必鬚髮送 200（OK）或 204（無內容）響應以指示成功完成要求。

源服務器應該忽略在 PUT 請求中接收到的無法識別的頭字段（即，不要將它們保存為資源狀態的一部分）。

源服務器應該驗證 PUT 表示是否符合服務器對目標資源的任何約束，這些約束不能或不會被 PUT 更改。當源服務器使用與 URI 相關的內部配置信息以便在 GET 響應上設置表示元數據的值時，這一點尤為重要。當 PUT 表示與目標資源不一致時，源服務器應該通過轉換錶示或更改資源配置使它們一致，或者以包含足夠信息的適當錯誤消息響應，以解釋表示不合適的原因。建議使用 409（衝突）或 415（不支持的媒體類型）狀態代碼，後者特定於對 Content-Type 值的約束。

例如，如果目標資源配置為始終具有
“文本/html”的內容類型和被 PUT 的表示有一個
“圖像/jpeg”的內容類型，源服務器應該執行以下操作之一：

一種。重新配置目標資源以反映新的媒體類型；

b.在將 PUT 表示形式保存為新的資源狀態之前，將其轉換為與資源一致的格式；或者，

C。使用 415（不受支持的媒體類型）響應拒絕請求，指示目標資源僅限於“text/html”，可能包括指向不同資源的鏈接，該資源將是新表示的合適目標。

除了用戶代理請求的意圖和源服務器響應的語義可以表達的內容之外，HTTP 沒有準確定義 PUT 方法如何影響源服務器的狀態。它沒有定義資源可能是什麼，在任何意義上，超出了通過 HTTP 提供的接口。它沒有定義資源狀態是如何“存儲”的，也沒有定義這種存儲如何因資源狀態的變化而改變，也沒有定義源服務器如何將資源狀態轉換為表示。一般來說，資源接口背後的所有實現細節都是服務器有意隱藏的。

源服務器不得在對 PUT 的成功響應中發送驗證器標頭字段（第 7.2 節），例如 ETag 或 Last-Modified 字段，除非請求的表示數據在保存時未對正文應用任何轉換（即，資源的新的表示數據與 PUT 請求中接收到的表示數據相同）並且驗證器字段值反映了新的表示。此要求允許用戶代理知道它在內存中的表示體何時由於 PUT 而保持最新，因此不需要從源服務器再次檢索，並且新的驗證器在響應中收到可用於未來的條件請求，以防止意外覆蓋（第 5.2 節）。

POST 和 PUT 方法之間的根本區別體現在封閉表示的不同意圖上。

POST 請求中的目標資源旨在根據資源自身的語義處理封閉表示，而 PUT 請求中的封閉表示被定義為替換目標資源的狀態。因此，PUT 的意圖是冪等的並且對中介可見，即使確切的效果只有源服務器知道。

PUT 請求的正確解釋假設用戶代理知道需要哪個目標資源。代表客戶端選擇適當 URI 的服務，在收到狀態更改請求後，應該使用 POST 方法而不是 PUT 方法來實現。

如果源服務器不會將請求的 PUT 狀態更改為目標資源，而是希望將其應用於不同的資源，例如當資源已移動到不同的 URI 時，那麼源服務器必鬚髮送適當的 3xx（重定向）響應；然後用戶代理可以自己決定是否重定向請求。

應用於目標資源的 PUT 請求可能會對其他資源產生副作用。例如，一篇文章可能有一個用於標識“當前版本”（資源）的 URI，該 URI 與標識每個特定版本（不同資源）的 URI 是分開的

在某一時刻與當前版本資源共享相同的狀態）。因此，除了更改目標資源的狀態之外，對“當前版本”URI 的成功 PUT 請求可能會創建新版本資源，並且還可能導致在相關資源之間添加鏈接。

允許在給定目標資源上進行 PUT 的源服務器必須向包含 Content-Range 標頭字段（[RFC7233] 的第 4.2 節)的 PUT 請求發送 400（錯誤請求)響應，因為有效負載可能是部分內容已被錯誤地 PUT 為完整表示。部分內容更新可以通過將單獨標識的資源作為目標，其狀態與較大資源的一部分重疊，或者使用專門為部分更新定義的不同方法（例如，[RFC5789] 中定義的 PATCH 方法）。

對 PUT 方法的響應不可緩存。如果一個成功的 PUT 請求通過一個緩存，緩存中存儲了一個或多個有效請求 URI 的響應，這些存儲的響應將失效（參見 [RFC7234] 的第 4.4 節）。

4.3.5.刪除

DELETE 方法請求源服務器刪除目標資源與其當前功能之間的關聯。實際上，這種方法類似於UNIX中的rm命令：它表達了對源站URI映射的刪除操作，而不是期望刪除之前關聯的信息。

如果目標資源具有一個或多個當前表示，它們可能會或可能不會被源服務器銷毀，並且相關存儲可能會或可能不會被回收，這完全取決於資源的性質及其源服務器的實現（這超出了本規範的範圍）。

同樣，作為 DELETE 的結果，資源的其他實現方面可能需要停用或存檔，例如數據庫或網關連接。通常，假設源服務器只允許對它具有完成刪除的規定機制的資源進行 DELETE。

允許使用 DELETE 方法的資源相對較少 它的主要用途是用於遠程創作環境，在這種環境中，用戶對其效果有一定的指導。例如，先前使用 PUT 請求創建的資源，或在對 POST 請求的 201（已創建)響應後通過 Location 標頭字段標識的資源，可能允許相應的 DELETE 請求撤消這些操作。同樣，實現的自定義用戶代理實現

創作功能，例如使用 HTTP 進行遠程操作的版本控制客戶端，可能會基於服務器的 URI 空間已被設計為與版本存儲庫相對應的假設來使用 DELETE。

如果成功應用 DELETE 方法，如果操作可能成功但尚未執行，源服務器應該發送 202（已接受）狀態代碼，如果操作已執行且沒有進一步執行，則發送 204（無內容）狀態代碼將提供信息，或 200 (OK) 狀態代碼（如果操作已執行且響應消息包含描述狀態的表示）。

DELETE 請求消息中的有效載荷沒有定義的語義；在 DELETE 請求上發送有效負載主體可能會導致某些現有實現拒絕該請求。

對 DELETE 方法的響應不可緩存。如果一個 DELETE 請求通過一個緩存，緩存中存儲了一個或多個針對有效請求 URI 的響應，這些存儲的響應將失效（參見 [RFC7234] 的第 4.4 節）。

4.3.6.連接

CONNECT 方法請求接收者建立到由請求目標標識的目標源服務器的隧道，如果成功，此後將其行為限制為雙向盲目轉發數據包，直到隧道關閉。隧道通常用於通過一個或多個代理創建端到端的虛擬連接，然後可以使用 TLS（傳輸層安全性，[RFC5246]）對其進行保護。

CONNECT 僅用於對代理的請求。為自己接收 CONNECT 請求的源服務器可以用 2xx（成功）狀態代碼進行響應，以指示已建立連接。然而，大多數源服務器沒有實現 CONNECT。

發送 CONNECT 請求的客戶端必鬚髮送請求目標的授權形式（[RFC7230] 的第 5.3 節）；即，請求目標僅包含隧道目的地的主機名和端口號，以冒號分隔。例如，

```
連接 server.example.com:80 HTTP/1.1 主機：
server.example.com:80
```

接收代理可以通過直接連接到請求目標來建立隧道，或者如果配置為使用另一個代理，則可以通過將 CONNECT 請求轉發到下一個入站代理來建立隧道。

任何 2xx（成功）響應表明發送者（以及所有

入站代理)將在結束成功響應的標頭部分的空行之後立即切換到隧道模式 ;在該空行之後收到的數據來自請求目標標識的服務器 。除成功響應之外的任何響應都表示隧道尚未形成並且連接仍由 HTTP 管理。

當隧道中介檢測到任何一方已關閉其連接時 ,隧道將關閉 :中介必須嘗試將來自封閉端的任何未完成數據發送到另一端 ,關閉兩個連接 ,然後丟棄任何未交付的剩餘數據。

代理身份驗證可用於建立創建隧道的權限 。例如，

```
連接 server.example.com:80 HTTP/1.1 主機：
server.example.com:80 代理授權 :基本 aGVsbG86d29ybGQ=
```

建立通往任意服務器的隧道存在重大風險 ,特別是當目標是眾所周知或保留的 TCP 端口 ,而不是用於 Web 流量時 。例如 ,連接到“example.com:25”的請求目標將建議代理連接到 SMTP 流量的保留端口 ;如果允許 ,這可能會誘使代理轉發垃圾郵件 。支持 CONNECT 的代理應該將其使用限制在一組有限的已知端口或可配置的安全請求目標白名單中。

服務器不得在 2xx（成功)響應中向 CONNECT 發送任何 Transfer-Encoding 或 Content-Length 標頭字段 。客戶端必須忽略在對 CONNECT 的成功響應中收到的任何 Content-Length 或 Transfer-Encoding 標頭字段。

CONNECT 請求消息中的有效載荷沒有定義的語義 ;在 CONNECT 請求上發送有效負載主體可能會導致某些現有實現拒絕該請求。

對 CONNECT 方法的響應不可緩存。

4.3.7.選項

OPTIONS 方法請求關於目標資源可用的通信選項的信息 ,在源服務器或中間中介 。此方法允許客戶端確定與資源關聯的選項和/或要求，或服務器的功能 ,而不暗示資源操作。

帶有星號（“*”）作為請求目標（[RFC7230] 的第 5.3 節）的 OPTIONS 請求通常適用於服務器而不是特定資源。由於服務器的通信選項通常取決於資源，因此“*”請求僅用作“ping”或“no-op”類型的方法，除了允許客戶端測試服務器的功能外，它什麼都不做。例如，這可用於測試代理是否符合 HTTP/1.1（或不符合）。

如果請求目標不是星號，則 OPTIONS 請求適用於與目標通信時可用的選項

資源。

生成對 OPTIONS 的成功響應的服務器應該發送任何標頭字段，這些字段可能指示服務器實現的可選功能並適用於目標資源（例如，允許），包括本規範未定義的潛在擴展。

響應負載（如果有的話）也可能以機器或人類可讀的表示形式描述通信選項。這種表示的標準格式未由本規範定義，但可能由未來的 HTTP 擴展定義。如果沒有負載主體要在響應中發送，服務器必須生成一個值為“0”的 Content-Length 字段。

客戶端可以在 OPTIONS 請求中發送一個 Max-Forwards 頭字段，以將請求鏈中的特定接收者作為目標（參見第 5.1.2 節）。代理在轉發請求時不得生成 Max-Forwards 標頭字段，除非該請求是使用 Max-Forwards 字段接收的。

生成包含有效負載主體的 OPTIONS 請求的客戶端必鬚髮送描述表示媒體類型的有效 Content-Type 標頭字段。儘管本規範未定義此類負載的任何用途，但未來對 HTTP 的擴展可能會使用 OPTIONS 主體來對目標進行更詳細的查詢

資源。

對 OPTIONS 方法的響應不可緩存。

4.3.8.痕跡

TRACE 方法請求請求消息的遠程應用程序級環回。請求的最終接收者應該將接收到的消息（不包括下面描述的一些字段）作為 200（OK）響應的消息正文反映給客戶端，其中 Content-Type 為“message/http”（第 8.3.1 節）[RFC7230]）。最終接收者是源服務器或第一個在請求中接收到 Max-Forwards 值為零 (0) 的服務器（第 5.1.2 節）。

客戶端不得在包含敏感數據的 TRACE 請求中生成標頭字段，這些數據可能會被響應洩露。

例如，用戶代理在 TRACE 請求中發送存儲的用戶憑證 [RFC7235] 或 cookie [RFC6265] 是愚蠢的。請求的最終接收者應該在該接收者生成響應主體時排除任何可能包含敏感數據的請求標頭字段。

TRACE 允許客戶端查看在請求鏈的另一端接收到的內容，並將該數據用於測試或診斷信息。 Via 標頭字段（[RFC7230] 的第 5.7.1 節）的值特別令人感興趣，因為它充當請求鏈的跟踪。使用 Max-Forwards 標頭字段允許客戶端限制請求鏈的長度，這對於測試在無限循環中轉發消息的代理鏈很有用。

客戶端不得在 TRACE 請求中發送消息體。

對 TRACE 方法的響應不可緩存。

5. 請求頭域

客戶端發送請求標頭字段以提供有關請求上下文的更多信息、根據目標資源狀態使請求有條件、建議響應的首選格式、提供身份驗證憑證或修改預期的請求處理。這些字段充當請求修飾符，類似於編程語言方法調用的參數。

5.1.控件

控件是指示請求的特定處理的請求標頭字段。

標題字段名稱	定義在...
緩存控制	[RFC7234] 的第 5.2 節
預計	第 5.1.1 節
主持人	[RFC7230] 的第 5.4 節
最大前鋒	第 5.1.2 節
語用	[RFC7234] 的第 5.4 節
範圍	[RFC7233] 的第 3.1 節
TE	[RFC7230] 第 4.3 節

5.1.1.預計

請求中的“Expect”標頭字段表示服務器需要支持的一組特定行為（期望)才能正確處理此請求。本規範定義的唯一此類期望是 100-continue。

期望=“100-繼續”

Expect 字段值不區分大小寫。

接收到除 100-continue 之外的 Expect 字段值的服務器可以使用 417（Expectation Failed)狀態代碼進行響應，以指示無法滿足意外的期望。

100-continue 期望通知收件人客戶端將在此請求中發送（可能是大的)消息體，並且如果請求行和標頭字段不足以引起立即響應，則希望接收 100（繼續)臨時響應成功、重定向或錯誤響應。這允許客戶端在實際發送消息體之前等待值得發送消息體的指示，這可以在消息體很大或客戶端預計可能會出錯時（例如，發送狀態時)提高效率- 更改方法，第一次，無需先前驗證的身份驗證憑據）。

例如，以以下開頭的請求

```
PUT /somewhere/fun HTTP/1.1 Host:
origin.example.com Content-Type: video/
h264 Content-Length: 1234567890987
Expect: 100-continue
```

允許源服務器在客戶端開始用不必要的數據傳輸填充管道之前立即響應錯誤消息，例如 401（未授權)或 405（方法不允許）。

對客戶的要求：

o 客戶端不得在請求中生成 100-continue 期望
不包括消息正文。

o 客戶端將等待 100（繼續)響應
發送請求消息體必鬚髮送一個包含 100-continue 期望的 Expect 頭字段。

o 發送 100-continue 期望的客戶端不需要
等待任何特定的時間長度；這樣的客戶端可以繼續發送消息體，即使它還沒有收到響應。

此外，由於 100（繼續）響應不能通過 HTTP/1.0 中介發送，這樣的客戶端不應該在發送消息正文之前無限期地等待。

o 收到 417（預期失敗）狀態代碼的客戶端
對包含 100-continue 期望的請求的響應應該重複沒有 100-continue 期望的請求，因為 417 響應僅表示響應鏈不支持期望（例如，它通過 HTTP/1.0 服務器）。

服務器要求：

o 在 HTTP/1.0 中接收 100-continue 期望的服務器
請求必須忽略該期望。

o 服務器可以忽略發送 100（繼續）響應，如果它有
已經收到相應請求的部分或全部消息體，或者如果幀指示沒有消息體。

o 發送 100（繼續）響應的服務器必須最終發送最終狀態代碼，一旦消息正文被接收和處理，除非連接過早關閉。

o 在讀取整個消息正文之前以最終狀態代碼響應的服務器應該在該響應中指示它是打算關閉連接還是繼續讀取並丟棄請求消息（參見 [RFC7230] 的第 6.6 節）。

源服務器必須在收到 HTTP/1.1（或更高版本）請求行和包含 100-continue 期望並指示請求消息正文將跟隨的完整標頭部分後，發送帶有最終狀態代碼的立即響應，如果可以通過僅檢查請求行和標頭字段來確定該狀態，或者立即發送 100（繼續）響應以鼓勵客戶端發送請求的消息體。源服務器在發送 100（繼續）響應之前不得等待消息體。

代理必須在收到 HTTP/1.1（或更高版本）請求行和包含 100-continue 期望並指示請求消息正文將跟隨的完整標頭部分後，發送帶有最終狀態代碼的立即響應，如果該狀態可以通過僅檢查請求行和標頭字段來確定，或者通過發送

相應的請求行和標題部分到下一個入站服務器。如果代理認為（從配置或過去的交互）下一個入站服務器僅支持 HTTP/1.0，則代理可以立即生成 100（繼續）響應以鼓勵客戶端開始發送消息體。

注意：Expect 標頭字段是在 HTTP/1.1 [RFC2068] 的原始發布之後添加的，作為請求臨時 100（繼續）響應的方法和指示必須理解的擴展的一般機制。但是，擴展機制並沒有被客戶端使用，很多服務端也沒有實現必須了解的需求，導致擴展機制無用武之地。本規範去除了擴展機制以簡化100-continue的定義和處理。

5.1.2.最大前鋒

“Max-Forwards”頭字段提供了一種機制，使用 TRACE（第 4.3.8 節）和 OPTIONS（第 4.3.7 節）請求方法來限制代理轉發請求的次數。當客戶端試圖跟踪一個似乎失敗或在鏈中循環的請求時，這可能很有用。

```
最大轉發 = 1*DIGIT
```

Max-Forwards 值為十進制整數，表示該請求報文可以轉發的剩餘次數。

每個接收包含 Max-Forwards 頭字段的 TRACE 或 OPTIONS 請求的中介必須在轉發請求之前檢查並更新其值。如果接收到的值為零 (0)，中介不得轉發請求；相反，中介必須作為最終接收者響應。如果接收到的 Max-Forwards 值大於零，中介必須在轉發消息中生成一個更新的 Max-Forwards 字段，其字段值是 a) 接收到的值減一 (1) 或 b)收件人對 Max-Forwards 的最大支持值。

接收者可以忽略通過任何其他請求方法接收到的 Max-Forwards 標頭字段。

5.2.條件句

HTTP 條件請求標頭字段 [RFC7232] 允許客戶端在目標資源的狀態上放置一個先決條件，以便如果先決條件的計算結果為假，則不會應用與方法語義對應的操作。每個先決條件由

本規範包含一組從目標資源的先前表示中獲得的驗證器與所選表示的驗證器的當前狀態之間的比較（第 7.2 節）。因此，這些先決條件評估目標資源的狀態自客戶端已知的給定狀態以來是否發生了變化。這種評估的效果取決於方法語義和條件的選擇，如 [RFC7232] 第 5 節中所定義。

標題字段名稱	定義在...
如果匹配	[RFC7232] 的第 3.1 節
如果不匹配	[RFC7232] 的第 3.2 節
If-Modified-Since	[RFC7232] 的第 3.3 節
If-Unmodified-Since	[RFC7232] 第 3.4 節
如果範圍	[RFC7233] 的第 3.2 節

5.3.內容協商

以下請求標頭字段由用戶代理髮送，以參與響應內容的主動協商，如第 3.4.1 節中所定義。在這些字段中發送的首選項適用於響應中的任何內容，包括目標資源的表示、錯誤或處理狀態的表示，甚至可能出現在協議中的雜項文本字符串。

標題字段名稱	定義於...
接受	第 5.3.2 節
接受字符集	第 5.3.3 節
接受編碼	第 5.3.4 節
接受語言	第 5.3.5 節

5.3.1.品質價值觀

許多用於主動協商的請求標頭字段使用一個名為“q”（不區分大小寫）的公共參數來為相關內容類型的偏好分配相對“權重”。這個權重被稱為“質量值”（或“qvalue”），因為在服務器配置中經常使用相同的參數名稱來為可以為資源選擇的各種表示的相對質量分配權重。

權重被歸一化為 0 到 1 範圍內的實數，其中 0.001 是最不優選的，1 是最優選的；值為 0 表示“不可接受”。如果不存在“q”參數，則默認權重為 1。

```
權重 = OWS “;” OWS q= qvalue qvalue =
( 0 [ . 0*3DIGIT ] ) / ( 1 [ . 0*3( 0 ) ] )
```

qvalue 的發送者不得在小數點後生成超過三位數字。這些值的用戶配置應該以相同的方式進行限制。

5.3.2.接受

用戶代理可以使用“Accept”頭字段來指定可接受的響應媒體類型。 Accept 標頭字段可用於指示請求專門限於一小組所需類型，如請求內聯圖像的情況。

```
接受 = #( 媒體範圍 [ 接受參數 ] )

媒體範圍 = ( */* /
( 輸入 / * ) / ( 輸入
/ 子類型 )
) *( OWS ; OWS 參數 ) accept-params =
weight *( accept-ext ) accept-ext = OWS ; OWS token [ = ( token /
quoted-string ) ]
```

星號“*”字符用於將媒體類型分組到範圍內，“*/*”表示所有媒體類型，“type/*”表示該類型的所有子類型。媒體範圍可以包括適用於該範圍的媒體類型參數。

每個媒體範圍後面可能跟著零個或多個適用的媒體類型參數（例如，字符集），一個可選的“q”參數用於指示相對權重（第 5.3.1 節），然後是零個或多個擴展參數。如果存在任何擴展 (accept-ext)，則“q”參數是必需的，因為它充當兩個參數集之間的分隔符。

注意：使用“q”參數名稱將媒體類型參數與接受擴展參數分開是由於歷史慣例。雖然這可以防止任何名為“q”的媒體類型參數與媒體範圍一起使用，但鑑於 IANA 中缺少任何“q”參數，這種事件被認為是不太可能的

媒體類型註冊表和 Accept 中很少使用任何媒體類型參數。不鼓勵未來的媒體類型註冊任何名為“q”的參數。

這個例子

接受：音頻/*； q=0.2，音頻/基本

被解釋為“我更喜歡音頻/基本，但如果它是質量降價 80% 後最好的音頻類型，請發送給我”。

沒有任何 Accept 標頭字段的請求意味著用戶代理將接受任何媒體類型作為響應。如果請求中存在標頭字段，並且沒有可用的響應表示具有列為可接受的媒體類型，則源服務器可以通過發送 406（不可接受）響應來尊重標頭字段，或者忽略標頭通過將響應視為不受內容協商影響來處理字段。

一個更詳細的例子是

接受：文本/純文本； q=0.5，文本/html，文本/x-dvi； q=0.8，文本/xc

在口頭上，這將被解釋為“text/html 和 text/xc 是同樣首選的媒體類型，但如果它們不存在，則發送 text/x-dvi 表示，如果不存在，則發送 text/簡單的表示”。

媒體範圍可以被更具體的媒體範圍或特定的媒體類型覆蓋。如果多個媒體範圍適用於給定類型，則最具體的參考具有優先權。例如，

接受：text/*, text/plain, text/plain;format=flowed, */*

具有以下優先級：

1. 文本/純文本；格式=流式

2. 文本/純文本

3.文字/*

4. */*

與給定類型關聯的媒體類型質量因子是通過查找與該類型匹配的具有最高優先級的媒體範圍來確定的。例如，

接受：text/*;q=0.3, text/html;q=0.7, text/html;level=1, text/html;level=2;q=0.4, */*;q=0.5

將導致關聯以下值：

+----------+----------+
|媒體類型 |品質價值 |
+----------+----------+
|文本/html；級別=1 | 1 |文本/html | 0.7 |文本/純 | 0.3 |圖片/jpeg | 0.5 |文本/html；級別=2 | 0.4 |文本/html；級別=3 | 0.7
+----------+----------+

注意：用戶代理可能會為某些媒體範圍提供一組默認的質量值。然而，除非用戶代理是一個不能與其他渲染代理交互的封閉系統，否則這個默認設置應該可以由用戶配置。

5.3.3.接受字符集

“Accept-Charset”標頭字段可以由用戶代理髮送，以指示在文本響應內容中哪些字符集是可接受的。

該字段允許能夠理解更全面或專用字符集的用戶代理向能夠在這些字符集中表示信息的源服務器發出該能力信號。

接受字符集 = 1#(字符集 / *) [重量]

字符集名稱在第 3.1.1.2 節中定義。用戶代理可以將質量值與每個字符集相關聯，以指示用戶對該字符集的相對偏好，如第 5.3.1 節中所定義。

一個例子是

接受字符集：iso-8859-5、unicode-1-1；q=0.8

特殊值“*”如果出現在 Accept-Charset 字段中，則匹配 Accept-Charset 字段中其他地方未提及的每個字符集。如果 Accept-Charset 字段中沒有“*”，則該字段中未明確提及的任何字符集都被視為客戶端“不可接受”。

沒有任何 Accept-Charset 標頭字段的請求意味著用戶代理將接受任何字符集作為響應。大多數通用用戶代理不發送 Accept-Charset，除非特別說明

配置為這樣做，因為支持的字符集的詳細列表使服務器更容易根據用戶代理的請求特徵（第 9.7 節)識別個人。

如果請求中存在 Accept-Charset 標頭字段，並且沒有可用的響應表示具有列為可接受的字符集，則源服務器可以通過發送 406（不可接受)響應來尊重標頭字段，或者通過將資源視為不受內容協商影響來忽略標頭字段。

5.3.4.接受編碼

用戶代理可以使用“Accept-Encoding”頭字段來指示響應中可接受的響應內容編碼（第 3.1.2.1 節）。 “身份”令牌用作“無編碼”的同義詞，以便在首選無編碼時進行通信。

```
Accept-Encoding = #( 編碼 [ 權重 ] 編碼 )
                  = 內容編碼/“身份”/“*”
```

如第 5.3.1 節中所定義，每個編碼值都可以被賦予一個相關聯的質量值，代表該編碼的偏好。 Accept-Encoding 字段中的星號“*”符號匹配任何未在標題字段中明確列出的可用內容編碼。

例如，

```
接受編碼：壓縮，gzip 接受編碼：接受編碼：* 接受編碼：壓縮；
q=0.5，gzip；q=1.0 接受編碼：gzip；q=1.0，身份； q=0.5, *;q=0
```

沒有 Accept-Encoding 標頭字段的請求意味著用戶代理沒有關於內容編碼的偏好。雖然這允許服務器在響應中使用任何內容編碼，但這並不意味著用戶代理能夠正確處理所有編碼。

服務器使用以下規則測試給定表示的內容編碼是否可接受：

1.如果請求中沒有Accept-Encoding字段，任何content-coding
被用戶代理認為是可以接受的。

2.如果表示沒有內容編碼，那麼它是
默認情況下可接受，除非 Accept-Encoding 字段明確排除說明“identity;q=0”或“*;q=0”而沒有更具體的“identity”條目。

3.如果表示的內容編碼是其中之一
Accept-Encoding 字段中列出的內容編碼，那麼它是可以接受的，除非它伴隨著 qvalue 為 0。（如第 5.3.1 節中所定義，qvalue 為 0 表示“不可接受”。）

4.如果多個內容編碼是可接受的，那麼具有最高非零qvalue的可接受內容編碼是優選的。

具有空組合字段值的 Accept-Encoding 標頭字段意味著用戶代理不希望任何內容編碼作為響應。如果請求中出現 Accept-Encoding 頭字段，並且響應的可用表示都沒有列為可接受的內容編碼，則源服務器應該發送沒有任何內容編碼的響應。

注意：大多數 HTTP/1.0 應用程序不識別或遵守與內容編碼關聯的 qvalues。這意味著 qvalues 可能不起作用並且不允許與 x-gzip 或 x-compress 一起使用。

5.3.5.接受語言

用戶代理可以使用“Accept-Language”標頭字段來指示響應中首選的自然語言集。語言標籤在第 3.1.3.1 節中定義。

Accept-Language = 1#(language-range [weight] language-range =)

<語言範圍，參見 [RFC4647]，第 2.1 節>

每個語言範圍都可以被賦予一個相關的質量值，表示用戶對該範圍指定的語言的偏好估計，如第 5.3.1 節中所定義。例如，

接受語言：da, en-gb;q=0.8, en;q=0.7

意思是：“我更喜歡丹麥語，但會接受英式英語和其他類型的英語”。

沒有任何 Accept-Language 標頭字段的請求意味著用戶代理將接受任何語言作為響應。如果請求中存在標頭字段，並且響應的可用表示都沒有匹配的語言標記，則源服務器可以通過將響應視為好像它一樣來忽略標頭字段

不受內容協商或通過發送 406（不可接受）響應來遵守標頭字段。但是，不鼓勵使用後者，因為這樣做會阻止用戶訪問他們可能會使用的內容（例如，使用翻譯軟件）。

請注意，一些收件人將語言標籤列出的順序視為優先級遞減的指示，特別是對於分配了相同質量值的標籤（沒有值與 q=1 相同）。

但是，不能依賴此行為。為了一致性和最大化互操作性，許多用戶代理為每個語言標籤分配一個唯一的質量值，同時還按質量遞減的順序列出它們。可以在 [RFC4647] 的第 2.3 節中找到關於語言優先級列表的其他討論。

對於匹配，[RFC4647] 的第 3 節定義了幾種匹配方案。實現可以根據他們的要求提供最合適的匹配方案。 “基本過濾”方案（[RFC4647]，第 3.3.1 節）與先前在 [RFC2616] 的第 14.4 節中為 HTTP 定義的匹配方案相同。

在每個請求中發送包含用戶完整語言偏好的 Accept-Language 標頭字段可能有悖於用戶的隱私期望（第 9.7 節）。

由於可理解性高度依賴於單個用戶，因此用戶代理需要允許用戶控制語言偏好（通過用戶代理本身的配置或默認為用戶可控的系統設置）。不向用戶提供此類控制的用戶代理不得發送 Accept-Language 標頭字段。

注意：用戶代理應該在設置首選項時為用戶提供指導，因為用戶很少熟悉上述語言匹配的細節。例如，用戶可能會假設在選擇“en-gb”時，如果英式英語不可用，他們將獲得任何類型的英文文檔。在這種情況下，用戶代理可能會建議將“en”添加到列表中以獲得更好的匹配行為。

5.4.身份驗證憑據

兩個標頭字段用於攜帶身份驗證憑據，如 [RFC7235] 中所定義。請注意，用於用戶身份驗證的各種自定義機制為此目的使用 Cookie 標頭字段，如 [RFC6265] 中所定義。

標題字段名稱	定義在...
授權	[RFC7235] 的第 4.2 節
代理授權	[RFC7235] 第 4.4 節

5.5.請求上下文

以下請求標頭字段提供有關請求上下文的其他信息，包括有關用戶、用戶代理和請求背後的資源的信息。

標題字段名稱	定義於...
從	第 5.5.1 節
推薦人	第 5.5.2 節
用戶代理	第 5.5.3 節

5.5.1.從

“發件人”標題字段包含控制請求用戶代理的人類用戶的 Internet 電子郵件地址。地址應該是機器可用的，如 [RFC5322] 第 3.4 節中的“郵箱”所定義：

```
  從 =郵箱

  郵箱 = <郵箱，請參閱 [RFC5322]，第 3.4 節>
```

一個例子是：

```
  來自：webmaster@example.org
```

From 頭字段很少由非機器人用戶代理髮送。用戶代理不應該在沒有用戶明確配置的情況下發送 From 標頭字段，因為這可能與用戶的隱私利益或他們站點的安全策略發生衝突。

機器人用戶代理應該發送一個有效的 From 頭字段，以便在服務器出現問題時可以聯繫負責運行機器人的人員，例如機器人發送過多的、不需要的或無效的請求。

服務器不應該使用 From 標頭字段進行訪問控製或身份驗證，因為大多數接收者會假定該字段值是公共信息。

5.5.2.推薦人

“Referer”[sic] 標頭字段允許用戶代理為從中獲取目標 URI 的資源指定 URI 引用（即“referrer”，儘管字段名稱拼寫錯誤）。在生成 Referer 字段值時，用戶代理不得包含 URI 引用 [RFC3986] 的片段和用戶信息組件（如果有的話）。

Referer = 絕對 URI / 部分 URI

Referer 標頭字段允許服務器生成指向其他資源的反向鏈接，以進行簡單的分析、日誌記錄、優化緩存等。它還允許找到過時或輸入錯誤的鏈接以進行維護。一些服務器使用 Referer 標頭字段作為拒絕來自其他站點的鏈接（所謂的“深度鏈接”）或限制跨站點請求偽造 (CSRF) 的手段，但並非所有請求都包含它。

例子：

推薦人：http://www.example.org/hypertext/Overview.html

如果目標 URI 是從沒有自己的 URI 的來源獲得的（例如，從用戶鍵盤輸入，或用戶書籤/收藏夾中的條目），則用戶代理必須排除 Referer 字段或將其與“關於：空白”的價值。

Referer 字段有可能洩露有關用戶請求上下文或瀏覽歷史的信息，如果引用資源的標識符洩露個人信息（例如帳戶名）或本應保密的資源（例如在防火牆後面或安全服務的內部）。當引用資源是本地“文件”或“數據”URI 時，大多數通用用戶代理不會發送 Referer 標頭字段。如果引用頁面是使用安全協議接收的，則用戶代理不得在不安全的 HTTP 請求中發送 Referer 標頭字段。有關其他安全注意事項，請參閱第 9.4 節。

眾所周知，一些中介會不加區別地從傳出請求中刪除 Referer 標頭字段。這有一個不幸的副作用，即乾擾對 CSRF 攻擊的保護，這可能對他們的用戶造成更大的傷害。

希望在 Referer 中限制信息披露的中介和用戶代理擴展應該限制他們對特定編輯的更改，例如用假名替換內部域名或截斷查詢和/或路徑組件。當字段值與請求目標共享相同的方案和主機時，中間人不應該修改或刪除 Referer 頭字段。

5.5.3.用戶代理

“User-Agent”標頭字段包含有關發起請求的用戶代理的信息，服務器通常使用這些信息來幫助識別報告的互操作性問題的範圍，解決或定制響應以避免特定的用戶代理限制，以及用於分析關於瀏覽器或操作系統的使用。用戶代理應該在每個請求中發送一個 User-Agent 字段，除非特別配置為不這樣做。

User-Agent = product *(RWS (產品 / 評論))

User-Agent 字段值由一個或多個產品標識符組成，每個產品標識符後跟零個或多個註釋（[RFC7230] 的第 3.2 節），它們共同標識用戶代理軟件及其重要的子產品。按照慣例，產品標識符按其識別用戶代理軟件的重要性降序排列。每個產品標識符都包含名稱和可選版本。

產品 = 令牌 [/ 產品版本] 產品版本 = 令牌

發件人應該將生成的產品標識符限制為識別產品所必需的；發件人不得在產品標識符中生成廣告或其他非必要信息。發件人不應在產品版本中生成不是版本標識符的信息（即，相同產品名稱的連續版本應該僅在產品標識符的產品版本部分有所不同）。

例子：

用戶代理：CERN-LineMode/2.15 libwww/2.17b3

用戶代理不應生成包含不必要的細粒度細節的用戶代理字段，並且應限制第三方添加子產品。過長和詳細的 User-Agent 字段值會增加請求延遲和用戶被識別為違背其意願的風險（“指紋識別”）。

同樣，鼓勵實現不要使用其他實現的產品令牌來聲明與它們的兼容性，因為這會規避該字段的目的。如果用戶代理偽裝成不同的用戶代理，接收者可以假設用戶有意希望看到為該識別的用戶代理量身定制的響應，即使它們可能不適用於正在使用的實際用戶代理。

6.響應狀態碼

status-code 元素是一個三位整數代碼，給出嘗試理解和滿足請求的結果。

HTTP 狀態代碼是可擴展的。 HTTP 客戶端不需要理解所有已註冊狀態代碼的含義，儘管這種理解顯然是可取的。但是，客戶端必須理解任何狀態代碼的類，如第一個數字所示，並將無法識別的狀態代碼視為等同於該類的 x00 狀態代碼，但接收者不得緩存響應無法識別的狀態代碼。

例如，如果客戶端收到無法識別的狀態代碼 471，則客戶端可以假設其請求有問題並將響應視為收到 400（錯誤請求）狀態代碼。響應消息通常包含解釋狀態的表示。

狀態代碼的第一位數字定義了響應的類別。
最後兩位數字沒有任何分類作用。第一個數字有五個值：

- o 1xx（信息）：收到請求，繼續處理
- o 2xx（成功）：請求被成功接收、理解和接受
- o 3xx（重定向）：需要採取進一步的行動以完成請求
- o 4xx（客戶端錯誤）：請求包含錯誤的語法或無法履行

o 5xx（服務器錯誤）：服務器未能完成明顯有效的請求

6.1.狀態碼概覽

下面列出的狀態代碼在本規範、[RFC7232] 的第 4 節、[RFC7233] 的第 4 節和 [RFC7235] 的第 3 節中定義。此處列出的短語僅是建議的
原因 它們可以在不影響協議的情況下替換為本地等效項。

具有默認定義為可緩存的狀態代碼的響應（例如，本規範中的 200、203、204、206、300、301、404、405、410、414 和 501）可以由具有啟發式過
期的緩存重用，除非方法定義或顯式緩存控制 [RFC7234] 另有說明；默認情況下，所有其他狀態代碼都不可緩存。

代碼	原因短語	定義在...
100	繼續	第 6.2.1 節
101	交換協議	第 6.2.2 節
200	好的	第 6.3.1 節
201	已創建	第 6.3.2 節
202	已接受	第 6.3.3 節
203	非權威信息	第 6.3.4 節
204	沒有內容	第 6.3.5 節
205	重置內容	第 6.3.6 節
206	部分內容	[RFC7233] 第 4.1 節
300	多種選擇	第 6.4.1 節
301	永久移動	第 6.4.2 節
302	找到	第 6.4.3 節
303	查看其他	第 6.4.4 節
304	未修改	[RFC7232] 第 4.1 節
305	使用代理	第 6.4.5 節
306	（未使用）	第 6.4.6 節
307	臨時重定向	第 6.4.7 節
400	錯誤請求	第 6.5.1 節
401	未經授權	[RFC7235] 第 3.1 節
402	需要付款	第 6.5.2 節
403	禁止	第 6.5.3 節
404	未找到	第 6.5.4 節
405	方法不允許	第 6.5.5 節
406	不可接受	第 6.5.6 節
407	需要代理身份驗證	[RFC7235] 第 3.2 節
408	請求超時	第 6.5.7 節
409	衝突	第 6.5.8 節
410	走了	第 6.5.9 節
411	要求長度	第 6.5.10 節
412	前提條件失敗	[RFC7232] 第 4.2 節
413	有效載荷太大	第 6.5.11 節
414	URI 太長	第 6.5.12 節
415	不支持的媒體類型	第 6.5.13 節
416	範圍不滿足	[RFC7233] 第 4.4 節
417	期望失敗	第 6.5.14 節
426	需要升級	第 6.5.15 節
500	內部服務器錯誤	第 6.6.1 節
501	未實施	第 6.6.2 節
502	壞網關	第 6.6.3 節
503	服務不可用	第 6.6.4 節
504	網關超時	第 6.6.5 節
505	HTTP 版本不支持	第 6.6.6 節

請注意，此列表並不詳盡 它不包括其他規範中定義的擴展狀態代碼。狀態代碼的完整列表由 IANA 維護。有關詳細信息，請參閱第 8.2 節。

6.2.信息 1xx

1xx（信息）類狀態代碼表示在完成請求的操作並發送最終響應之前，用於通信連接狀態或請求進度的臨時響應。 1xx 響應由狀態行之後的第一個空行終止（表示標題部分結束的空行）。由於 HTTP/1.0 沒有定義任何 1xx 狀態代碼，服務器不得向 HTTP/1.0 客戶端發送 1xx 響應。

客戶端必須能夠解析在最終響應之前收到的一個或多個 1xx 響應，即使客戶端不希望收到一個。用戶代理可以忽略意外的 1xx 響應。

代理必須轉發 1xx 響應，除非代理本身請求生成 1xx 響應。例如，如果代理在轉發請求時添加了“Expect: 100-continue”字段，則它不需要轉發相應的 100 (Continue) 響應。

6.2.1. 100 繼續

100（Continue）狀態碼表示請求的初始部分已經收到，還沒有被服務器拒絕。服務器打算在完全接收並執行請求後發送最終響應。

當請求包含包含 100-continue 期望的 Expect 標頭字段時，100 響應表示服務器希望接收請求有效負載主體，如第 5.1.1 節所述。客戶端應該繼續發送請求並丟棄 100 響應。

如果請求不包含包含 100-continue 期望的 Expect 標頭字段，則客戶端可以簡單地丟棄此中間值

回复。

6.2.2. 101 交換協議

101（切換協議）狀態代碼表示服務器理解並願意通過升級標頭字段（[RFC7230] 的第 6.7 節）遵守客戶端的請求，以更改此連接上使用的應用程序協議。服務器

必須在響應中生成一個 Upgrade 標頭字段，指示將在終止 101 響應的空行之後立即切換到哪個協議。

假設服務器只會在有利時才同意切換協議。例如，切換到較新版本的 HTTP 可能優於舊版本，而切換到實時同步協議可能在交付使用此類功能的資源時更有優勢。

6.3.成功 2xx

2xx (Successful) 類的狀態碼表示客戶端的請求被成功接收、理解和接受。

6.3.1. 200 好

200（OK)狀態碼表示請求成功。
200 響應中發送的負載取決於請求方法。
對於本規範定義的方法，有效載荷的預期含義可以概括為：

GET 目標資源的表示；

HEAD 與 GET 相同的表示，但沒有表示
數據;

發布狀態的表示或從中獲得的結果，
那個行動;

PUT、DELETE 表示動作的狀態；

OPTIONS 通信選項的表示；

TRACE 表示最後收到的請求消息
服務器。

除了對 CONNECT 的響應之外，200 響應總是有一個有效負載，儘管源服務器可能會生成一個零長度的有效負載主體。
如果不需要負載，源服務器應該發送 204（無內容）。對於 CONNECT，不允許任何負載，因為成功的結果是一個隧道，它在 200 響應頭部分之後立即開始。

默認情況下，200 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.3.2. 201 創建

201（已創建）狀態代碼表示請求已完成並導致創建一個或多個新資源。請求創建的主要資源由響應中的 Location 標頭字段標識，或者如果未收到 Location 字段，則由有效請求 URI 標識。

201 響應負載通常描述並鏈接到創建的資源。請參閱第 7.2 節，了解 201 響應中驗證器標頭字段（例如 ETag 和 Last-Modified）的含義和用途的討論。

6.3.3. 202 已接受

202（已接受）狀態碼表示請求已被接受處理，但處理尚未完成。

該請求最終可能會或可能不會被執行，因為在實際進行處理時可能會拒絕該請求。 HTTP 中沒有用於從異步操作重新發送狀態代碼的工具。

202 響應是故意不置可否的。它的目的是允許服務器接受對其他進程（可能是每天只運行一次的面向批處理的進程）的請求，而不需要用戶代理與服務器的連接持續到進程完成。與此響應一起發送的表示應該描述請求的當前狀態並指向（或嵌入）狀態監視器，該監視器可以為用戶提供請求何時完成的估計。

6.3.4. 203非權威信息

203（非權威信息）狀態代碼表示請求成功，但包含的負載已被轉換代理（[RFC7230] 的第 5.7.2 節）從源服務器的 200（OK）響應中修改。此狀態代碼允許代理在應用轉換時通知收件人，因為該信息可能會影響以後有關內容的決策。例如，未來對內容的緩存驗證請求可能僅適用於相同的請求路徑（通過相同的代理）。

203 響應類似於 214 Transformation Applied（[RFC7234] 的第 5.5 節）的警告代碼，其優點是適用於任何狀態代碼的響應。

默認情況下，203 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.3.5. 204 無內容

204（無內容）狀態代碼表示服務器已成功完成請求，並且響應有效負載正文中沒有要發送的其他內容。響應標頭字段中的元數據指的是目標資源及其在應用請求的操作後選擇的表示。

例如，如果收到 204 狀態代碼以響應 PUT 請求並且響應包含 ETag 標頭字段，則 PUT 成功並且 ETag 字段值包含該目標資源的新表示的實體標籤。

204 響應允許服務器指示操作已成功應用於目標資源，同時暗示用戶代理不需要遍歷其當前的“文檔視圖”（如果有）。服務器假定用戶代理將根據其自己的界面向其用戶提供一些成功指示，並在對其活動表示的響應中應用任何新的或更新的元數據。

例如，204 狀態碼通常用於對應於“保存”操作的文檔編輯界面，使得正在保存的文檔仍然可供用戶編輯。它還經常與期望自動數據傳輸流行的接口一起使用，例如在分佈式版本控制系統中。

204 響應由標頭字段後的第一個空行終止，因為它不能包含消息正文。

默認情況下，204 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.3.6. 205重置內容

205（重置內容）狀態代碼表示服務器已完成請求並希望用戶代理將導致發送請求的“文檔視圖”重置為從源服務器接收到的原始狀態。

此響應旨在支持常見的數據輸入用例，其中用戶接收支持數據輸入的內容（表單、記事本、畫布等），在該空間中輸入或操作數據，

導致在請求中提交輸入的數據，然後為下一個條目重置數據輸入機制，以便用戶可以輕鬆發起另一個輸入操作。

由於 205 狀態代碼暗示不會提供額外的內容，因此服務器不得在 205 響應中生成有效負載。換句話說，服務器必須為 205 響應執行以下操作之一： a) 通過包含值為 0 的 Content-Length 標頭字段來指示響應的零長度主體； b) 通過包含值為 chunked 的 Transfer-Encoding 標頭字段和由單個零長度塊組成的消息正文來指示響應的零長度有效載荷；或者，c) 在發送終止標題部分的空行後立即關閉連接。

6.4.重定向 3xx

狀態代碼的 3xx（重定向）類表示用戶代理需要採取進一步的操作才能完成請求。如果提供 Location 頭字段（第 7.1.2 節），用戶代理可以自動將其請求重定向到 Location 字段值引用的 URI，即使不理解特定的狀態代碼。自動重定向需要小心處理未知的安全方法，如第 4.2.1 節中所定義，因為用戶可能不希望重定向不安全的請求。

有幾種類型的重定向：

1. 指示資源可能在不同 URI 可用的重定向，如位置字段提供的，如狀態代碼 301（永久移動）、302（找到）和 307（臨時重定向）。

2. 提供匹配資源選擇的重定向，每個
   能夠表示原始請求目標，如 300（多項選擇）狀態代碼。

3. 重定向到由 Location 標識的不同資源
   字段，可以表示對請求的間接響應，如 303（參見其他）狀態代碼。

4. 重定向到先前緩存的結果，如 304 (Not
   修改）狀態碼。

   注意：在 HTTP/1.0 中，狀態代碼 301（永久移動）和 302（找到）是為第一種重定向定義的（[RFC1945]，第 9.3 節）。早期的用戶代理在應用於重定向目標的方法是否與

原始請求或將被重寫為 GET。儘管 HTTP 最初為 301 和 302 定義了前者語義（以匹配其在 CERN 的原始實現），並定義了 303（參見其他)以匹配後者語義，但流行的做法也逐漸向 301 和 302 的後者語義收斂。 HTTP/1.1 的第一次修訂增加了 307（臨時重定向)來指示以前的語義，而不受不同實踐的影響。 10 多年過去了，大多數用戶代理仍然對 301 和 302 進行方法重寫；因此，當原始請求是 POST 時，本規範使該行為符合要求。

客戶端應該檢測並干預循環重定向（即“無限”重定向循環）。

注意：本規範的早期版本推薦最多五個重定向（[RFC2068]，第 10.3 節）。內容開發人員需要注意某些客戶端可能會實施此類固定限制。

6.4.1. 300多項選擇

300（多項選擇)狀態代碼表示目標資源有多個表示，每個都有自己更具體的標識符，並且正在提供有關備選方案的信息，以便用戶（或用戶代理)可以通過以下方式選擇首選表示將其請求重定向到這些標識符中的一個或多個。換句話說，服務器希望用戶代理參與反應性協商以選擇最適合其需要的表示（第 3.4 節）。

如果服務器有首選，服務器應該生成一個 Location 頭字段，其中包含首選的 URI 引用。

用戶代理可以使用 Location 字段值進行自動重定向。

對於 HEAD 以外的請求方法，服務器應該在 300 響應中生成一個負載，其中包含表示元數據和 URI 引用的列表，用戶或用戶代理可以從中選擇最喜歡的一個。如果用戶代理理解提供的媒體類型，它可以自動從該列表中進行選擇。本規範未定義用於自動選擇的特定格式，因為 HTTP 試圖與其有效負載的定義保持正交。在實踐中，表示以一些易於解析的格式提供，這些格式被認為是用戶代理可以接受的，如共享設計或內容協商所確定的，或者以一些普遍接受的超文本格式提供。

默認情況下，300 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

注意：300 狀態代碼的原始提案將 URI 標頭字段定義為提供替代表示列表，以便它可用於 200、300 和 406 響應，並在響應 HEAD 方法時傳輸。

然而，缺乏部署和對語法的分歧導致 URI 和 Alternates（後續提案）被從該規範中刪除。可以使用一組鏈接頭字段 [RFC5988] 來傳達列表，每個字段都具有“交替”關係，儘管部署是先有雞還是先有蛋的問題。

6.4.2. 301 永久移動

301（永久移動）狀態代碼表示目標資源已分配了一個新的永久 URI，並且將來對該資源的任何引用都應該使用其中一個包含的 URI。

如果可能，具有鏈接編輯功能的客戶端應該自動將對有效請求 URI 的引用重新鏈接到服務器發送的一個或多個新引用。

服務器應該在響應中生成一個 Location 頭字段，其中包含新永久 URI 的首選 URI 引用。用戶代理可以使用 Location 字段值進行自動重定向。服務器的響應負載通常包含一個簡短的超文本註釋，其中包含指向新 URI 的超鏈接。

注意：由於歷史原因，用戶代理可以將後續請求的請求方法從 POST 更改為 GET。如果不需要此行為，則可以改用 307（臨時重定向）狀態代碼。

默認情況下，301 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.4.3. 302 找到

302（已找到）狀態碼表示目標資源暫時駐留在不同的 URI 下。由於有時可能會更改重定向，因此客戶端應該繼續為以後的請求使用有效的請求 URI。

服務器應該在包含不同 URI 的 URI 引用的響應中生成 Location 頭域。用戶代理可以使用 Location 字段值進行自動重定向。服務器的響應負載通常包含一個簡短的超文本註釋，帶有指向不同 URI 的超鏈接。

注意：由於歷史原因，用戶代理可以將後續請求的請求方法從 POST 更改為 GET。如果不需要此行為，則可以改用 307（臨時重定向）狀態代碼。

6.4.4. 303 看其他

303 (See Other) 狀態代碼表示服務器正在將用戶代理重定向到不同的資源，如 Location 標頭字段中的 URI 所示，旨在提供對原始請求的間接響應。用戶代理可以執行針對該 URI 的檢索請求（如果使用 HTTP，則為 GET 或 HEAD 請求），該請求也可能被重定向，並將最終結果作為對原始請求的答復。請注意，Location 標頭字段中的新 URI 不被視為等同於有效請求 URI。

此狀態代碼適用於任何 HTTP 方法。它主要用於允許 POST 操作的輸出將用戶代理重定向到選定的資源，因為這樣做以一種可以單獨識別、添加書籤和緩存的形式提供與 POST 響應對應的信息，獨立於原始請求。

對 GET 請求的 303 響應表示源服務器沒有可以由服務器通過 HTTP 傳輸的目標資源的表示。但是，Location 字段值指的是描述目標資源的資源，因此對該其他資源發出檢索請求可能會導致對接收者有用的表示，而不暗示它代表原始目標資源。請注意，可以表示什麼、什麼表示是合適的以及什麼可能是有用的描述等問題的答案超出了 HTTP 的範圍。

除了對 HEAD 請求的響應之外，303 響應的表示應該包含一個簡短的超文本註釋，其中包含指向 Location 標頭字段中提供的相同 URI 引用的超鏈接。

6.4.5. 305 使用代理

305（使用代理）狀態代碼在本規範的先前版本中定義，現在已棄用（附錄 B）。

6.4.6. 306（未使用）

306 狀態碼在本規範的前一版本中定義，不再使用，該代碼被保留。

6.4.7. 307臨時重定向

307（臨時重定向）狀態代碼表示目標資源暫時駐留在不同的 URI 下，如果用戶代理執行自動重定向到該 URI，則用戶代理不得更改請求方法。由於重定向會隨著時間的推移而改變，因此客戶端應該繼續使用原始有效的請求 URI 來處理未來的請求。

服務器應該在包含不同 URI 的 URI 引用的響應中生成 Location 頭字段。用戶代理可以使用 Location 字段值進行自動重定向。服務器的響應負載通常包含一個簡短的超文本註釋，帶有指向不同 URI 的超鏈接。

> 注意：這個狀態碼類似於302（Found），只是它不允許將請求方式從POST改為GET。本規範沒有定義 301（永久移動）的等效對應物（[RFC7238]，但是，為此目的定義了狀態代碼 308（永久重定向））。

6.5.客戶端錯誤 4xx

4xx (Client Error) 類狀態碼表示客戶端似乎出錯了。除了響應 HEAD 請求時，服務器應該發送一個包含對錯誤情況的解釋的表示，以及它是臨時的還是永久的條件。這些狀態代碼適用於任何請求方法。

用戶代理應該向用戶顯示任何包含的表示。

6.5.1. 400 錯誤請求

400（Bad Request）狀態代碼表示服務器不能或不會處理請求，因為某些事情被認為是客戶端錯誤（例如，格式錯誤的請求語法、無效的請求消息框架或欺騙性請求路由）。

6.5.2. 402 需要付款

402（需要付款）狀態代碼保留供將來使用。

6.5.3. 403禁止訪問

403（禁止訪問）狀態碼表示服務器理解請求但拒絕授權。希望公開請求被禁止的原因的服務器可以在響應有效負載（如果有）中描述該原因。

如果請求中提供了身份驗證憑據，則服務器認為它們不足以授予訪問權限。客戶端不應該使用相同的憑據自動重複請求。客戶端可以使用新的或不同的憑據重複請求。但是，由於與憑據無關的原因，請求可能會被禁止。

希望“隱藏”當前存在的禁止目標資源的源服務器可以用狀態代碼 404（未找到）來響應。

6.5.4. 404 未找到

404（未找到）狀態代碼表示源服務器未找到目標資源的當前表示或不願意公開存在該表示。 404 狀態代碼不表示這種缺乏表示是暫時的還是永久的； 410（Gone）狀態代碼比 404 更受歡迎，如果源服務器知道（可能通過一些可配置的方式）這種情況可能是永久性的。

默認情況下，404 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.5.5. 405 方法不允許

405（Method Not Allowed）狀態代碼表示請求行中收到的方法為源服務器所知，但目標資源不支持。源服務器必須在 405 響應中生成一個 Allow 頭字段，其中包含目標資源當前支持的方法列表。

默認情況下，405 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.5.6. 406 不可接受

406（Not Acceptable)狀態碼表示根據請求中收到的主動協商標頭字段（第 5.3 節），目標資源沒有用戶代理可以接受的當前表示，並且服務器不願意提供默認表示。

服務器應該生成一個包含可用表示特徵列表和相應資源標識符的有效負載，用戶或用戶代理可以從中選擇最合適的。用戶代理可以自動從該列表中選擇最合適的選項。然而，本規範並未為此類自動選擇定義任何標準，如第 6.4.1 節所述。

6.5.7. 408請求超時

408（Request Timeout)狀態碼表示服務器在準備等待的時間內沒有收到完整的請求消息。服務器應該在響應中發送“關閉”連接選項（[RFC7230] 的第 6.1 節），因為 408 意味著服務器已決定關閉連接而不是繼續等待。如果客戶端在傳輸過程中有未完成的請求，則客戶端可以在新連接上重複該請求。

6.5.8. 409 衝突

409（衝突)狀態碼表示由於與目標資源的當前狀態發生衝突，請求無法完成。此代碼用於用戶可能能夠解決衝突並重新提交請求的情況。服務器應該生成一個負載，其中包含足夠的信息供用戶識別衝突的來源。

衝突最有可能在響應 PUT 請求時發生。例如，如果正在使用版本控制並且正在 PUT 的表示包含對與早期（第三方)請求所做的衝突的資源的更改，則源服務器可能會使用 409 響應來指示它無法完成要求。在這種情況下，響應表示可能包含對基於修訂歷史合併差異有用的信息。

6.5.9. 410 沒了

410（Gone)狀態代碼表示在源服務器上不再提供對目標資源的訪問，並且這種情況可能是永久性的。如果源服務器沒有

知道或無法確定條件是否是永久性的，應該使用狀態代碼 404（未找到）。

410 響應主要旨在通過通知接收者資源有意不可用以及服務器所有者希望刪除指向該資源的遠程鏈接來協助 Web 維護任務。對於限時促銷服務和屬於不再與原始服務器站點相關聯的個人的資源，此類事件很常見。沒有必要將所有永久不可用的資源標記為“消失”或將標記保留任何時間長度 這由服務器所有者自行決定。

默認情況下，410 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.5.10。 411 長度要求

411（需要長度）狀態代碼表示服務器拒絕接受沒有定義內容長度的請求（[RFC7230] 的第 3.3.2 節）。如果客戶端在請求消息中添加了包含消息正文長度的有效 Content-Length 標頭字段，則客戶端可以重複該請求。

6.5.11。 413 負載太大

413（負載太大）狀態代碼表示服務器拒絕處理請求，因為請求負載大於服務器願意或能夠處理的負載。服務器可以關閉連接以防止客戶端繼續請求。

如果條件是臨時的，服務器應該生成一個 Retry-After 頭字段來指示它是臨時的，並且在什麼時間之後客戶端可以重試。

6.5.12。 414 URI 太長

414（URI 太長）狀態代碼表示服務器拒絕為請求提供服務，因為請求目標（[RFC7230] 的第 5.3 節）比服務器願意解釋的要長。

這種罕見的情況只有當客戶端不正確地將 POST 請求轉換為具有長查詢信息的 GET 請求時，當客戶端陷入重定向的“黑洞”時（例如，重定向的 URI 前綴指向本身的後綴）或當服務器受到試圖利用潛在安全漏洞的客戶端的攻擊時。

默認情況下，414 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.5.13。 415 不支持的媒體類型

415（不支持的媒體類型）狀態代碼表示源服務器拒絕為請求提供服務，因為有效負載的格式不受此方法在目標資源上的支持。

格式問題可能是由於請求指定的 Content-Type 或 Content-Encoding，或者是直接檢查數據的結果。

6.5.14。 417 期待失敗

417（Expectation Failed）狀態碼表示請求的 Expect 標頭字段（第 5.1.1 節）中給出的期望至少有一個入站無法滿足

服務器。

6.5.15。 426 需要升級

426（需要升級）狀態碼表示服務器拒絕使用當前協議執行請求，但在客戶端升級到不同的協議後可能願意這樣做。服務器必須在 426 響應
中發送 Upgrade 標頭字段以指示所需的協議（[RFC7230] 的第 6.7 節）。

例子：

```
HTTP/1.1 426 需要升級升級：HTTP/3.0 連接：升級內容長度：
53 內容類型：文本/純文本

此服務需要使用 HTTP/3.0 協議。
```

6.6.服務器錯誤 5xx

5xx（服務器錯誤）類狀態代碼表示服務器知道它有錯誤或無法執行請求的方法。除了響應 HEAD 請求時，服務器應該發送一個包含對錯誤情況的解釋
的表示，以及它是臨時的還是永久的

健康)狀況。用戶代理應該向用戶顯示任何包含的表示。這些響應代碼適用於任何請求方法。

6.6.1. 500內部服務器錯誤

500（內部服務器錯誤)狀態代碼表示服務器遇到了阻止它完成請求的意外情況。

6.6.2. 501 未實現

501（未實現)狀態代碼表示服務器不支持完成請求所需的功能。當服務器無法識別請求方法並且無法支持任何資源時，這是適當的響應。

默認情況下，501 響應是可緩存的；即，除非方法定義或顯式緩存控制另有說明（參見 [RFC7234] 的第 4.2.2 節）。

6.6.3. 502錯誤的網關

502（錯誤的網關)狀態代碼表示服務器在充當網關或代理時，在嘗試完成請求時從其訪問的入站服務器收到了無效響應。

6.6.4. 503服務不可用

503（Service Unavailable)狀態碼表示服務器當前由於臨時過載或定期維護而無法處理請求，延遲一段時間後可能會得到緩解。服務器可以發送一個 Retry-After 頭字段（第 7.1.3 節)來建議客戶端在重試請求之前等待的適當時間。

注意：503 狀態碼的存在並不意味著服務器在超載時必須使用它。一些服務器可能只是拒絕連接。

6.6.5. 504網關超時

504（網關超時)狀態代碼表示服務器在充當網關或代理時，沒有從它需要訪問以完成請求的上游服務器及時收到響應。

6.6.6. 505 不支持 HTTP 版本

505（不支持 HTTP 版本）狀態代碼表示服務器不支持或拒絕支持請求消息中使用的 HTTP 主要版本。服務器表明它不能或不願意使用與客戶端相同的主要版本來完成請求，如 [RFC7230] 的第 2.6 節所述，除了此錯誤消息。服務器應該為 505 響應生成一個表示，描述為什麼不支持該版本以及該服務器支持哪些其他協議。

7.響應頭域

響應標頭字段允許服務器傳遞有關響應的附加信息，超出狀態行中的信息。這些標頭字段提供有關服務器的信息、有關對目標資源的進一步訪問或有關的相關信息

資源。

儘管每個響應標頭字段都有定義的含義，但一般來說，精確的語義可能會通過請求方法和/或響應狀態代碼的語義進一步細化。

7.1.控制數據

響應標頭字段可以提供補充狀態代碼、指導緩存或指示客戶端下一步去向的控制數據。

標題字段名稱	定義在...
年齡	[RFC7234] 的第 5.1 節
緩存控制	[RFC7234] 的第 5.3 節
過期	[RFC7234] 的第 5.2 節
日期	第 7.1.1.2 節
地點	第 7.1.2 節
之後重試	第 7.1.3 節
各不相同	第 7.1.4 節
警告	[RFC7234] 的第 5.5 節

7.1.1.發起日期

7.1.1.1.日期/時間格式

在 1995 年之前，服務器通常使用三種不同的格式來傳達時間戳。為了與舊實現兼容，所有三個都在此處定義。首選格式是 Internet 消息格式 [RFC5322] 使用的日期和時間規範的固定長度和單區域子集。

```
HTTP 日期 = IMF-fixdate / 觀察日期
```

首選格式的一個例子是

```
1994 年 11 月 6 日，星期日 08:49:37 GMT ;國際貨幣基金組織固定日期
```

兩種過時格式的示例是

```
格林威治標準時間 2094 年 11 月 6 日星期日 08:49:37 ;過時的 RFC 850 格式
1994 年 11 月 6 日星期日 08:49:37 ; ANSI C 的 asctime() 格式
```

解析 HTTP 標頭字段中的時間戳值的接收者必須接受所有三種 HTTP 日期格式。當發送方生成包含一個或多個定義為 HTTP-date 的時間戳的標頭字段時，發送方必須以 IMF-fixdate 格式生成這些時間戳。

HTTP 日期值將時間表示為協調世界時 (UTC) 的一個實例。前兩種格式通過格林威治標準時間的三個字母縮寫“GMT”表示 UTC，這是 UTC 名稱的前身；假定 asctime 格式的值採用 UTC。從本地時鐘生成 HTTP 日期值的發送方應該使用 NTP（[RFC5905])或一些類似的協議將其時鐘同步到 UTC。

首選格式：

```
IMF-fixdate = day-name , SP date1 SP time-of-day SP GMT ;格式的固定長度/區域/大寫子集；參見 [RFC5322] 的
第 3.3 節

日名 = %x4D.6F.6E ; “週一”，區分大小寫 / %x54.75.65 ； “星期
二”，區分大小寫 / %x57.65.64 ； “星期三”，區分大小寫 /
%x54.68.75 ； “週四”，區分大小寫 / %x46.72.69 ； “星期
五”，區分大小寫 / %x53.61.74 ； “星期六”，區分大小寫 /
%x53.75.6E ； “太陽”，區分大小寫
```

日期 1 = 日 SP 月 SP 年 ;例如 ,1982 年 6 月 2 日

日月 = 2DIGIT =
%x4A.61.6E ;“一月” ,區分大小寫 / %x46.65.62 ;“二月” ,區分大小寫 / %x4D.61.72 ;“Mar” ,區分大小寫 / %x41.70.72 ;“四月” ,區分大小寫 / %x4D.61.79 ;“可能” ,區分大小寫 / %x4A.75.6E ;“君” ,區分大小寫 / %x4A.75.6C ;“七月” ,區分大小寫 / %x41.75.67 ;“八月” ,區分大小寫 / %x53.65.70 ;“九月” ,區分大小寫 / %x4F.63.74 ;“Oct” ,區分大小寫 / %x4E.6F.76 ;“11 月” ,區分大小寫 / %x44.65.63 ;“Dec” ,區分大小寫 = 4DIGIT

年

格林威治標準時間 = %x47.4D.54 ;“GMT” ,區分大小寫

time-of-day = 小時 : 分鐘 : 秒; 00:00:00 - 23:59:60（閏秒）

時分秒 = 2 位數字
= 2 位數字
= 2 位數字

過時的格式：

觀察日期 = rfc850-日期/asctime-日期

rfc850-date = day-name-l , SP date2 SP time-of-day SP GMT date2
=日“-”月“-”2DIGIT ;例如 ,02-Jun-82

日名-l = %x4D.6F.6E.64.61.79 ;“星期一” ,區分大小寫 / %x54.75.65.73.64.61.79 ;“星期二” ,區分大小寫 / %x57.65.64.6E.65.73.64.61.79 ;“星期三” ,區分大小寫 / %x54.68.75.72.73.64.61.79 ;“星期四” ,區分大小寫 / %x46.72.69.64.61.79 ;“星期五” ,區分大小寫 / %x53.61.74.75.72.64.61.79 ;“星期六” ,區分大小寫 / %x53.75.6E.64.61.79 ;“星期日” ,區分大小寫

asctime-date = day-name SP date3 SP time-of-day SP year date3 = month SP (2DIGIT / (SP 1DIGIT)) ;例如，6 月 2 日

HTTP 日期區分大小寫。發送者不得在 HTTP 日期中生成額外的空格，超出語法中作為 SP 明確包含的空格。 day-name、day、month、year 和 time-of-day 的語義與為具有相應名稱的 Internet 消息格式構造定義的語義相同（[RFC5322]，第 3.3 節）。

使用兩位數年份的 rfc850-date 格式的時間戳值的接收者必須將未來 50 年以上的時間戳解釋為代表過去具有相同最後兩位數的最近一年.

除非字段定義另有限制，否則鼓勵時間戳值的接收者在解析時間戳時保持健壯。例如，消息偶爾會從非 HTTP 源通過 HTTP 轉發，這可能會生成 Internet 消息格式定義的任何日期和時間規範。

注意：日期/時間戳格式的 HTTP 要求僅適用於它們在協議流中的使用。實現不需要將這些格式用於用戶呈現、請求記錄等。

7.1.1.2.日期

“Date”頭字段表示消息發出的日期和時間，與 [RFC5322] 第 3.6.1 節中定義的發出日期字段 (orig-date) 具有相同的語義。字段值是 HTTP 日期，如第 7.1.1.1 節中所定義。

日期 = HTTP 日期

一個例子是

日期：1994 年 11 月 15 日星期二 08:12:31 GMT

當生成 Date 頭字段時，發送方應該生成其字段值作為消息生成日期和時間的最佳可用近似值。理論上，日期應該代表生成有效載荷之前的時刻。實際上，日期可以在消息發起期間的任何時間生成。

如果原始服務器沒有能夠提供協調世界時當前實例的合理近似值的時鐘，則它不得發送 Date 標頭字段。如果響應是 1xx（信息）或 5xx（服務器錯誤）狀態代碼類，源服務器可以發送日期頭字段。在所有其他情況下，源服務器必鬚髮送日期頭字段。

接收沒有 Date 頭字段的響應消息的帶有時鐘的接收者必須記錄它被接收的時間，並且如果它被緩存或轉發到下游，則將相應的 Date 頭字段附加到消息的頭部分。

用戶代理可以在請求中發送 Date 標頭字段，但通常不會這樣做，除非它被認為可以向服務器傳達有用的信息。例如，如果期望服務器根據用戶代理和服務器時鐘之間的差異調整其對用戶請求的解釋，則 HTTP 的自定義應用程序可能會傳達一個日期。

7.1.2.地點

“Location”頭字段在某些響應中用於引用與響應相關的特定資源。關係的類型由請求方法和狀態代碼語義的組合定義。

位置 = URI 參考

字段值由單個 URI 引用組成。當它具有相對引用的形式（[RFC3986]，第 4.2 節）時，最終值是通過根據有效請求 URI（[RFC3986]，第 5 節）解析它來計算的。

對於 201（已創建）響應，Location 值指的是請求創建的主要資源。對於 3xx（重定向）響應，Location 值是指用於自動重定向請求的首選目標資源。

如果在 3xx（重定向）響應中提供的 Location 值沒有片段組件，用戶代理必須處理重定向，就好像該值繼承了用於生成請求目標的 URI 引用的片段組件（即，重定向繼承原始參考的片段，如果有的話）。

例如，為 URI 引用“http://www.example.org/~tim”生成的 GET 請求可能會導致包含標頭字段的 303（參見其他）響應：

位置：/People.html#tim

這表明用戶代理重定向到“http://www.example.org/People.html#tim”

同樣，為 URI 引用“http://www.example.org/index.html#larry”生成的 GET 請求可能會導致包含標頭字段的 301（永久移動）響應：

```
位置：http://www.example.net/index.html
```

這表明用戶代理重定向到“http://www.example.net/index.html#larry”，保留原始片段標識符。

在某些情況下，位置值中的片段標識符是不合適的。例如，201（已創建）響應中的 Location 標頭字段應該提供特定於已創建資源的 URI。

注意：一些收件人試圖從不是有效 URI 引用的位置字段中恢復。本規範不強製或定義此類處理，但出於穩健性考慮允許這樣做。

注意：Content-Location 頭字段（第 3.1.4.2 節）與 Location 的不同之處在於 Content-Location 指的是與所附表示相對應的最具體的資源。

因此，響應可能同時包含 Location 和 Content-Location 標頭字段。

7.1.3.之後重試

服務器發送“Retry-After”標頭字段以指示用戶代理在發出後續請求之前應該等待多長時間。當與 503（服務不可用）響應一起發送時，Retry-After 指示預計服務對客戶端不可用的時間。

當與任何 3xx（重定向）響應一起發送時，Retry-After 指示用戶代理在發出重定向請求之前被要求等待的最短時間。

此字段的值可以是 HTTP 日期或收到響應後延遲的秒數。

```
Retry-After = HTTP-date / delay-seconds
```

delay-seconds 值是一個非負十進制整數，表示以秒為單位的時間。

```
延遲秒數 = 1*DIGIT
```

它的兩個使用示例是

重試之後：1999 年 12 月 31 日星期五 23:59:59 GMT
重試後：120

在後一個示例中，延遲為 2 分鐘。

7.1.4.各不相同

響應中的“Vary”頭字段描述了請求消息的哪些部分，除了方法、主機頭字段和請求目標之外，可能會影響源服務器選擇和表示此響應的過程。該值由單個星號（ * ）或標題字段名稱列表（不區分大小寫）組成。

變化 = * / 1#字段名

“*”的 Vary 字段值表示關於請求的任何內容都可能在選擇響應表示中發揮作用，可能包括消息語法之外的元素（例如，客戶端的網絡地址）。如果不將請求轉發到源服務器，接收者將無法確定此響應是否適合以後的請求。代理不得生成具有

“*“價值。

由逗號分隔的名稱列表組成的 Vary 字段值指示命名的請求標頭字段（稱為選擇標頭字段）可能在選擇表示中起作用。潛在的選擇報頭字段不限於本規範定義的那些。

例如，包含的響應

變化：接受編碼，接受語言

表示源服務器在選擇此響應的內容時可能已使用請求的 Accept-Encoding 和 Accept-Language 字段（或缺少）作為決定因素。

源服務器可能會發送帶有兩個字段列表的 Vary
用途：

1. 通知緩存接收者他們不得使用此響應
   以滿足後來的請求，除非後來的請求對列出的字段具有與原始請求相同的值（[RFC7234] 的第 4.1 節）。換句話說，Vary 擴展了將新請求與存儲的緩存條目匹配所需的緩存鍵。

2. 通知用戶代理接收者此響應受內容協商（第 5.3 節)的約束，並且如果在列出的標頭字段中提供附加參數（主動協商），則可能會在後續請求中發送不同的表示。

當源服務器選擇表示的算法根據請求消息的方面而不是方法和請求目標而變化時，源服務器應該發送一個 Vary 頭字段，除非不能跨越差異或源服務器已被故意配置為防止緩存透明.例如，無需在 Vary 中發送授權字段名稱，因為跨用戶的重用受字段定義的限制（[RFC7235] 的第 4.2 節）。同樣，源服務器可能會使用 Cache-Control 指令（[RFC7234] 的第 5.2 節)來取代 Vary，如果它認為差異不如 Vary 對緩存的影響的性能成本那麼重要的話。

7.2.驗證器標頭字段

驗證器標頭字段傳達有關所選表示的元數據（第 3 節）。在響應安全請求時，驗證器字段描述了源服務器在處理響應時選擇的選定表示。請注意，根據狀態代碼語義，為給定響應選擇的表示不一定與作為響應負載包含的表示相同。

在對狀態更改請求的成功響應中，驗證器字段描述了作為處理請求的結果已替換先前選擇的表示的新表示。

例如，201（已創建)響應中的 ETag 標頭字段傳達新創建資源表示的實體標籤，以便它可以在以後的條件請求中使用，以防止“丟失更新”問題 [RFC7232]。

標題字段名稱	定義在...
電子標籤	[RFC7232] 的第 2.3 節
最後修改	[RFC7232] 的第 2.2 節

7.3.身份驗證挑戰

身份驗證挑戰指示客戶端可以使用哪些機制在未來的請求中提供身份驗證憑據。

標題字段名稱	定義在...
WWW-認證	[RFC7235] 第 4.1 節
代理驗證	[RFC7235] 第 4.3 節

7.4.響應上下文

其餘的響應標頭字段提供了有關目標資源的更多信息，以便在以後的請求中使用。

標題字段名稱	定義在...	
接受範圍	[RFC7233] 的第 2.3 節	第 7.4.1 節
允許		第 7.4.2 節
服務器		

7.4.1.允許

“允許”標頭字段列出了目標資源支持的一組方法。這個字段的目的是嚴格地通知接收者與資源相關的有效請求方法。

允許=#method

使用示例：

允許：GET、HEAD、PUT

實際允許的方法集由源服務器在每次請求時定義。源服務器必須在 405（方法不允許）響應中生成允許字段，並且可以在任何其他響應中這樣做。空的 Allow 字段值表示該資源不允許任何方法，如果該資源已被配置暫時禁用，則可能會在 405 響應中出現這種情況。

代理不得修改 Allow 頭字段 它不需要理解所有指定的方法來根據通用消息處理規則處理它們。

7.4.2.服務器

“服務器”頭字段包含有關源服務器用於處理請求的軟件的信息，客戶端通常使用這些信息來幫助確定報告的互操作性問題的範圍，解決或定制請求以避免特定的服務器限制，以及用於有關服務器或操作系統使用情況的分析。源服務器可以在其回應。

服務器 = 產品 *(RWS (產品 / 評論))

服務器字段值由一個或多個產品標識符組成，每個產品標識符後跟零個或多個註釋（[RFC7230] 的第 3.2 節），它們共同標識源服務器軟件及其重要的子產品。按照慣例，產品標識符按照其識別原始服務器軟件的重要性的降序排列。每個產品標識符都包含一個名稱和可選版本，如第 5.5.3 節中所定義。

例子：

服務器：CERN/3.0 libwww/2.17

源服務器不應該生成包含不必要的細粒度細節的服務器字段，並且應該限制第三方添加子產品。過長和詳細的服務器字段值會增加響應延遲並可能洩露內部實現細節，這可能使攻擊者（稍微）更容易找到和利用已知的安全漏洞。

8. IANA 考慮

8.1.方法註冊

“超文本傳輸協議 (HTTP) 方法註冊表”定義了請求方法令牌的名稱空間（第 4 節）。方法註冊表已創建，現在維護在 <http://www.iana.org/assignments/http-methods>。

8.1.1.程序

HTTP 方法註冊必須包含以下字段：

o 方法名稱（見第 4 節）

o 安全（“是”或“否”，參見第 4.2.1 節）

o 冪等（“是”或“否”，參見第 4.2.2 節）

o 指向規範文本的指針

要添加到此名稱空間的值需要 IETF 審查（參見 [RFC5226]，第 4.1 節）。

8.1.2.新方法的注意事項

標準化方法是通用的；也就是說，它們可能適用於任何資源，而不僅僅是一種特定的媒體類型、資源種類或應用程序。因此，最好在不特定於單個應用程序或數據格式的文檔中註冊新方法，因為正交技術值得正交規範。

由於消息解析（[RFC7230] 的第 3.3 節)需要獨立於方法語義（除了對 HEAD 的響應），新方法的定義不能更改解析算法或禁止請求或響應中出現消息體信息。新方法的定義可以通過要求 Content-Length 標頭字段值為“0”來指定僅允許零長度的消息正文。

一個新的方法定義需要指明它是否安全（第 4.2.1 節）、冪等（第 4.2.2 節）、可緩存（第 4.2.3 節），如果有效載荷主體中存在任何語義，則與有效載荷主體相關聯的語義是什麼請求以及該方法對標頭字段或狀態代碼語義進行了哪些改進。如果新方法是可緩存的，它的定義應該描述緩存如何以及在什麼條件下可以存儲響應並使用它來滿足後續請求。新方法應該描述它是否可以成為有條件的（第 5.2 節），如果可以，當條件為假時服務器如何響應。同樣，如果新方法可能對部分響應語義有一些用途（[RFC7233]），它也應該對此進行記錄。

注意：避免定義以“M-”開頭的方法名稱，因為該前綴可能會被誤解為具有 [RFC2774] 分配給它的語義。

8.1.3.登記

“超文本傳輸協議 (HTTP) 方法註冊表”已填充以下註冊：

方法	安全	冪等	參考
連接	沒有	不	第 4.3.6 節
刪除	沒有	是的	第 4.3.5 節
獲取	是	是的	第 4.3.1 節
頭	是	是的	第 4.3.2 節
選項	是	是的	第 4.3.7 節
發布	沒有	不	第 4.3.3 節
放	沒有	是的	第 4.3.4 節
追蹤	是	是的	第 4.3.8 節

8.2.狀態代碼註冊表

“超文本傳輸協議 (HTTP) 狀態代碼註冊表”定義了響應狀態代碼令牌的命名空間（第 6 節）。狀態代碼註冊表維護在 <http://www.iana.org/assignments/http-status-codes>。

本節取代了先前在 [RFC2817] 的第 7.1 節中定義的 HTTP 狀態代碼的註冊過程。

8.2.1.程序

註冊必須包括以下字段：

o 狀態代碼（3 位數字）

o 簡短說明

o 指向規範文本的指針

要添加到 HTTP 狀態代碼命名空間的值需要 IETF 審查（參見 [RFC5226]，第 4.1 節）。

8.2.2.新狀態代碼的注意事項

當需要表達當前狀態代碼未定義的響應的語義時，可以註冊一個新的狀態代碼。

狀態代碼是通用的；它們可能適用於任何資源，而不僅僅是一種特定的媒體類型、資源種類或 HTTP 應用程序。因此，最好在不特定於單個應用程序的文檔中註冊新的狀態代碼。

新的狀態代碼需要屬於第 6 節中定義的類別之一。為了允許現有的解析器處理響應消息，新的狀態代碼不能禁止有效負載，儘管它們可以強制使用零長度的有效負載主體。

尚未廣泛部署的新狀態代碼的提案應避免為代碼分配特定編號，直到明確同意將其註冊為止；相反，早期的草案可以使用諸如“4NN”或“3N0”..“3N9”之類的符號來指示提議的狀態代碼的類別，而不會過早地使用數字。

新狀態代碼的定義應該解釋導致包含該狀態代碼的響應的請求條件（例如，請求標頭字段和/或方法的組合）以及對響應標頭字段的任何依賴性（例如，什麼字段是必需的，哪些字段可以修改語義，以及在與新狀態代碼一起使用時進一步細化哪些頭字段語義）。

新狀態碼的定義應該指定它是否可緩存。請注意，如果出現的響應具有明確的新鮮度信息，則可以緩存所有狀態代碼；然而，定義為可緩存的狀態代碼允許在沒有顯式新鮮度信息的情況下被緩存。同樣，狀態代碼的定義可以對緩存行為施加約束。有關詳細信息，請參閱 [RFC7234]。

最後，新狀態碼的定義應該表明有效負載是否與已識別資源有任何隱含關聯（第 3.1.4.1 節）。

8.2.3.登記

狀態代碼註冊表已更新為以下註冊：

價值	描述	參考
100	繼續	第 6.2.1 節
101	交換協議	第 6.2.2 節
200	好的	第 6.3.1 節
201	已創建	第 6.3.2 節
202	已接受	第 6.3.3 節
203	非權威信息	第 6.3.4 節
204	無內容	第 6.3.5 節
205	重置內容	第 6.3.6 節
206	部分內容	第 4.1 節
300	多種選擇	第 6.4.1 節
301	永久移動	第 6.4.2 節
302	找到	第 6.4.3 節
303	參見其他	第 6.4.4 節
304	未修改	第 4.1 節
305	使用代理	第 6.4.5 節
306	（未使用）	第 6.4.6 節
307	臨時重定向	第 6.4.7 節
400	錯誤請求	第 6.5.1 節
401	未經授權	第 3.1 節
402	需要付款	第 6.5.2 節
403	禁止	第 6.5.3 節
404	未找到	第 6.5.4 節
405	不允許的方法	第 6.5.5 節
406	不可接受	第 6.5.6 節
407	需要代理身份驗證	第 3.2 節
408	請求超時	第 6.5.7 節
409	衝突	第 6.5.8 節
410	消失了	第 6.5.9 節
411	長度要求	第 6.5.10 節
412	前提條件失敗	第 4.2 節
413	請求實體太大	第 6.5.11 節
414	URI 太長	第 6.5.12 節
415	不支持的媒體類型	第 6.5.13 節
416	請求範圍不滿足	第 4.4 節
417	期望失敗	第 6.5.14 節
426	需要升級	第 6.5.15 節
500	內部服務器錯誤	第 6.6.1 節
501	未實現	第 6.6.2 節
502	錯誤的網關	第 6.6.3 節
503	服務不可用	第 6.6.4 節
504	網關超時	第 6.6.5 節
505	不支持 HTTP 版本	第 6.6.6 節

8.3.標頭字段註冊表

HTTP 標頭字段在位於 <http://www.iana.org/assignments/message-headers> 的“消息標頭”註冊表中註冊，如 [BCP90] 所定義。

8.3.1.新標題字段的注意事項

標頭字段是鍵值對，可用於傳遞有關消息、消息有效負載、目標資源或連接（即控制數據）的數據。有關 HTTP 消息中標頭字段語法的一般定義，請參閱 [RFC7230] 的第 3.2 節。

[BCP90] 中定義了頭域名稱的要求。

建議定義新字段的規範的作者使名稱盡可能短，並且不要在名稱前加上“X-”，除非標題字段永遠不會在 Internet 上使用。（“X-”前綴習語在實踐中被廣泛濫用；它的目的只是用作避免專有軟件或內部網處理中的名稱衝突的機制，因為前綴將確保私有名稱永遠不會與新註冊的名稱衝突Internet 名稱；有關詳細信息，請參閱 [BCP178]）。

新的標頭字段值通常使用 ABNF ([RFC5234]) 定義語法，必要時使用 [RFC7230] 第 7 節中定義的擴展，並且通常限制在 US-ASCII 字符範圍內。需要更大範圍字符的標頭字段可以使用 [RFC5987] 中定義的編碼。

原始字段值中的前導和尾隨空格在字段解析時被刪除（[RFC7230] 的第 3.2.4 節）。值中前導或尾隨空格很重要的字段定義將必須使用容器語法，例如引號字符串（[RFC7230] 的第 3.2.6 節）。

由於逗號（，）用作字段值之間的通用分隔符，因此如果允許在字段值中使用逗號，則需要謹慎對待。通常，可能包含逗號的組件使用帶引號的字符串 ABNF 產生式用雙引號保護。

例如，文本日期和 URI（其中任何一個都可能包含逗號）可以安全地包含在如下字段值中：

示例 URI 字段：“http://example.com/a.html,foo”、“http://without-a-comma.example.com/”
示例日期字段：“1996 年 5 月 4 日星期六”、“2005 年 9 月 14 日星期三”

請注意，雙引號定界符幾乎總是用於帶引號的字符串生成；在雙引號內使用不同的語法可能會造成不必要的混淆。

許多標題字段使用一種格式，包括（不區分大小寫)命名參數（例如，Content-Type，在第 3.1.1.5 節中定義）。

允許參數值的不帶引號（令牌)和帶引號（帶引號的字符串)語法使接收者能夠使用現有的解析器組件。當允許這兩種形式時，參數值的含義應該獨立於用於它的語法（例如，參見第 3.1.1.1 節中關於媒體類型的參數處理的註釋）。

建議定義新標頭字段的規範作者考慮記錄：

o 該字段是單個值還是可以是列表（以逗號分隔；參見 [RFC7230] 的第 3.2 節）。

如果它不使用列表語法，請記錄如何處理該字段多次出現的消息（明智的默認設置是忽略該字段，但這可能並不總是正確的選擇）。

請注意，中介和軟件庫可能會將多個標頭字段實例組合成一個，儘管該字段的定義不允許使用列表語法。穩健的格式使接收者能夠發現這些情況（好的例子：“Content-Type”，因為逗號只能出現在帶引號的字符串內；不好的例子：“Location”，因為逗號可以出現在 URI 內）。

o 在什麼條件下可以使用頭字段；例如，僅在響應或請求中，在所有消息中，僅在對特定請求方法的響應中，等等。

o 該字段是否應由原始服務器存儲
根據 PUT 請求理解它。

o 字段語義是否通過上下文進一步細化，例如通過現有的請求方法或狀態代碼。

o 在 Connection 報頭字段中列出字段名稱是否合適（即，如果報頭字段是逐跳的；請參閱 [RFC7230] 的第 6.1 節）。

o 在什麼條件下允許中介機構插入，
刪除或修改字段的值。

o 在 Vary 中列出 field-name 是否合適
響應頭字段（例如，當請求頭字段被源服務器的內容選擇算法使用時；參見第 7.1.4 節）。

o 標題字段在預告片中是否有用或允許（參見
[RFC7230] 的第 4.1 節）。

o 標題字段是否應該在重定向中保留。

o 它是否引入了任何額外的安全考慮，例如
作為隱私相關數據的披露。

8.3.2.登記

“消息標題”註冊表已更新為以下永久註冊：

標題字段名稱	協議	狀態	參考
接受	網址	標準	第 5.3.2 節
接受字符集	網址	標準	第 5.3.3 節
接受編碼	網址	標準	第 5.3.4 節
接受語言	網址	標準	第 5.3.5 節
允許	網址	標準	第 7.4.1 節
內容編碼	網址	標準	第 3.1.2.2 節
內容語言	網址	標準	第 3.1.3.2 節
內容位置	網址	標準	第 3.1.4.2 節
內容類型	網址	標準	第 3.1.1.5 節
日期	網址	標準	第 7.1.1.2 節
期待	網址	標準	第 5.1.1 節
來自	網址	標準	第 5.5.1 節
地點	網址	標準	第 7.1.2 節
最大前鋒	網址	標準	第 5.1.2 節
MIME 版本	網址	標準	附錄 A.1
推薦人	網址	標準	第 5.5.2 節
之後重試	網址	標準	第 7.1.3 節
服務器	網址	標準	第 7.4.2 節
用戶代理	網址	標準	第 5.5.3 節
變化	網址	標準	第 7.1.4 節

上述註冊的變更控制者是：“IETF (iesg@ietf.org) - Internet Engineering Task Force”。

8.4.內容編碼註冊表

“HTTP 內容編碼註冊表”定義了內容編碼名稱的名稱空間（[RFC7230] 的第 4.2 節）。內容編碼註冊表維護在 <http://www.iana.org/assignments/http-parameters>。

8.4.1.程序

內容編碼註冊必須包括以下字段：

姓名

o 說明

o 指向規範文本的指針

內容編碼的名稱不得與傳輸編碼的名稱重疊（[RFC7230] 的第 4 節），除非編碼轉換是相同的（如 [RFC7230] 第 4.2 節中定義的壓縮編碼的情況）。

添加到此命名空間的值需要 IETF 審查（參見 [RFC5226] 的第 4.1 節)並且必須符合本節中定義的內容編碼的目的。

8.4.2.登記

“HTTP 內容編碼註冊表”已更新為以下註冊：

名稱	描述	參考
身份	保留（| 第 5.3.4 節中“無編碼”的同義詞		
	接受編碼）	

9. 安全注意事項

本節旨在告知開發人員、信息提供者和用戶與 HTTP 語義及其在 Internet 上傳輸信息的用途相關的已知安全問題。

[RFC7230] 的第 9 節討論了與消息語法、解析和路由相關的注意事項。

以下注意事項列表並不詳盡。大多數與 HTTP 語義相關的安全問題都是關於保護服務器端應用程序（HTTP 接口背後的代碼）、保護用戶代理

處理通過 HTTP 接收的有效負載，或一般安全使用 Internet，而不是協議的安全性。各種組織維護主題信息和當前 Web 應用程序安全研究的鏈接（例如，[OWASP]）。

9.1.基於文件名和路徑名的攻擊

源服務器經常使用它們的本地文件系統來管理從有效請求 URI 到資源表示的映射。大多數文件系統並不是為了防止惡意文件或路徑名而設計的。因此，源站在將請求目標映射到文件、文件夾或目錄時，需要避免訪問對系統有特殊意義的名稱。

例如，UNIX、Microsoft Windows 等操作系統使用“..”作為路徑組成部分，表示當前目錄的上一級目錄，它們使用特殊命名的路徑或文件名向系統設備發送數據。類似的命名約定可能存在於其他類型的存儲系統中。同樣，在處理無效或意外字符、分解字符的重組以及不區分大小寫的名稱的大小寫規範化時，本地存儲系統有一個令人討厭的傾向，即更傾向於用戶友好而不是安全。

基於此類特殊名稱的攻擊往往側重於拒絕服務（例如，告訴服務器從 COM 端口讀取)或洩露不應提供的配置和源文件。

9.2.基於命令、代碼或查詢注入的攻擊

源服務器通常使用 URI 中的參數作為識別系統服務、選擇數據庫條目或選擇數據源的手段。但是，請求中收到的數據是不可信的。攻擊者可以構造任何請求數據元素（方法、請求目標、標頭字段或正文)以包含在通過命令調用、語言解釋器或數據庫傳遞時可能被誤解為命令、代碼或查詢的數據界面。

例如，SQL 注入是一種常見的攻擊，其中在請求目標或標頭字段（例如，Host、Referer 等)的某些部分中插入了額外的查詢語言。如果接收到的數據直接在 SELECT 語句中使用，則查詢語言可能被解釋為數據庫命令而不是簡單的字符串值。

這種類型的實施漏洞非常普遍，儘管很容易預防。

一般來說，資源實現應該避免在處理或解釋為指令的上下文中使用請求數據。

參數應該與固定字符串進行比較，並根據比較的結果採取行動，而不是通過未為不受信任的數據準備的接口傳遞。接收到的不是基於固定參數的數據應該仔細過濾或編碼以避免被誤解。

類似的注意事項適用於存儲和稍後處理請求數據時，例如在日誌文件、監控工具中，或者當包含在允許嵌入腳本的數據格式中時。

9.3.個人信息的披露

客戶通常不了解大量個人信息，包括用戶提供的與資源交互的信息（例如，用戶的姓名、位置、郵件地址、密碼、加密密鑰等）和有關用戶瀏覽活動的信息時間（例如歷史、書籤等）。實施需要防止無意中洩露個人信息。

9.4. URI 中敏感信息的洩露

URI 旨在共享，而不是保護，即使它們標識安全資源。 URI 通常顯示在顯示器上，打印頁面時添加到模板，並存儲在各種未受保護的書籤列表中。因此，在 URI 中包含敏感的、個人可識別的或有洩露風險的信息是不明智的。

服務的作者應該避免使用基於 GET 的表單來提交敏感數據，因為該數據將被放置在請求目標中。許多現有的服務器、代理和用戶代理在第三方可能可見的地方記錄或顯示請求目標。此類服務應該改用基於 POST 的表單提交。

由於 Referer 標頭字段告訴目標站點有關導致請求的上下文，它有可能揭示有關用戶的即時瀏覽歷史的信息以及可能在引用資源的 URI 中找到的任何個人信息。 Referer 頭字段的限制在第 5.5.2 節中描述，以解決它的一些安全問題。

9.5.重定向後片段的披露

雖然在 URI 引用中使用的片段標識符不會在請求中發送，但實施者應該知道它們將對用戶代理以及作為響應結果運行的任何擴展或腳本可見。特別是，當發生重定向並且原始請求的片段標識符被 Location 中的新引用繼承時（第 7.1.2 節），這可能會產生將一個站點的片段洩露給另一個站點的效果。如果第一個站點在片段中使用個人信息，它應該確保重定向到其他站點包括一個（可能是空的)片段組件以阻止該繼承。

9.6.產品信息披露

User-Agent（第 5.5.3 節）、Via（[RFC7230] 的第 5.7.1 節)和服務器（第 7.4.2 節)標頭字段通常會顯示有關各個發件人軟件系統的信息。從理論上講，這可以使攻擊者更容易利用已知的安全漏洞；實際上，攻擊者傾向於嘗試所有潛在的漏洞，而不管所使用的明顯軟件版本如何。

作為通過網絡防火牆的門戶的代理應該採取特殊的預防措施來傳輸可能識別防火牆後面主機的標頭信息。 Via 標頭字段允許中間人用假名替換敏感的機器名稱。

9.7.瀏覽器指紋識別

瀏覽器指紋識別是一組技術，用於通過其獨特的特徵集隨時間識別特定用戶代理。這些特徵可能包括與其 TCP 行為、特性功能和腳本環境相關的信息，儘管這里特別令人感興趣的是可能通過 HTTP 進行通信的一組獨特特徵。指紋識別被認為是一個隱私問題，因為它可以跟踪用戶代理隨時間的行為，而無需用戶可能對其他形式的數據收集（例如 cookie)進行相應的控制。許多通用用戶代理（即 Web 瀏覽器)已採取措施減少其指紋。

有許多請求標頭字段可能會向服務器顯示足夠唯一以啟用指紋識別的信息。 From 報頭字段是最明顯的，儘管預計只有在用戶需要自我識別時才會發送 From。同樣，Cookie 標頭字段是故意的

旨在啟用重新識別，因此指紋問題僅適用於 cookie 被用戶代理的配置禁用或限制的情況。

User-Agent 標頭字段可能包含足夠的信息來唯一標識特定設備，通常與其他特徵結合使用時，尤其是當用戶代理髮送有關用戶系統或擴展的過多詳細信息時。然而，用戶最不期望的唯一信息來源是主動協商（第 5.3 節），包括 Accept、Accept-Charset、Accept-Encoding 和 Accept-Language 標頭字段。

除了指紋問題之外，Accept-Language 標頭字段的詳細使用可以揭示用戶可能認為具有隱私性質的信息。例如，理解給定的語言集可能與特定種族群體的成員身份密切相關。一種限制這種隱私損失的方法是用戶代理省略發送 Accept-Language 除了已列入白名單的站點，可能是在檢測到指示語言協商可能有用的 Vary 標頭字段後通過交互。

在使用代理來增強隱私的環境中，用戶代理在發送主動協商標頭字段時應該保守。提供高度標頭字段可配置性的通用用戶代理應該告知用戶如果提供過多的細節可能會導致隱私丟失。

作為一種極端的隱私措施，代理可以過濾中繼請求中的主動協商標頭字段。

10.致謝

請參閱 [RFC7230] 的第 10 節。

11. 參考資料

11.1.規範性參考文獻

[RFC2045] Freed, N. 和 N. Borenstein，“多用途互聯網郵件
擴展 (MIME) 第一部分：Internet 消息體的格式”，RFC 2045，1996 年 11 月。

[RFC2046] Freed, N. 和 N. Borenstein，“多用途互聯網郵件
擴展 (MIME) 第二部分：媒體類型”，RFC 2046，1996 年 11 月。

[RFC2119] Bradner, S.，“RFC 中用於指示的關鍵詞
要求級別”，BCP 14，RFC 2119，1997 年 3 月。

[RFC3986] Berners-Lee, T.、Fielding, R. 和 L. Masinter，“制服
資源標識符 (URI)：通用語法”，STD 66，RFC 3986，2005 年 1 月。

[RFC4647] Phillips, A., Ed.和 M. Davis, Ed.，“語言匹配
標籤”，BCP 47，RFC 4647，2006 年 9 月。

[RFC5234] Crocker, D., Ed.和 P. Overell，“用於語法規範的增強 BNF：ABNF”，STD 68，RFC 5234，2008 年 1 月。

[RFC5646] Phillips, A., Ed.和 M. Davis, Ed.，“用於識別的標籤
語言”，BCP 47，RFC 5646，2009 年 9 月。

[RFC6365] Hoffman, P. 和 J. Klensin，“術語用於
IETF 中的國際化”，BCP 166，RFC 6365，2011 年 9 月。

[RFC7230] 菲爾丁，R.，Ed。和 J. Reschke, Ed.，“超文本傳輸協議 (HTTP/1.1)：消息語法和路由”，RFC 7230，2014 年 6 月。

[RFC7232] 菲爾丁，R.，Ed。和 J. Reschke, Ed.，“超文本傳輸
協議 (HTTP/1.1)：條件請求”，RFC 7232，2014 年 6 月。

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed.,
“超文本傳輸協議 (HTTP/1.1)：範圍請求”，RFC 7233，2014 年 6 月。

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed. 和 J. Reschke, Ed.，“超文本傳輸協議 (HTTP/1.1)：緩存”，RFC 7234，2014 年 6
月。

[RFC7235] 菲爾丁，R.，Ed。和 J. Reschke, Ed.，“超文本傳輸協議 (HTTP/1.1)：身份驗證”，RFC 7235，2014 年 6 月。

11.2.信息參考

[BCP13] Freed, N.、Klensin, J. 和 T. Hansen，“媒體類型規範和註冊程序”，BCP 13，RFC 6838，2013 年 1 月。

[BCP178] Saint-Andre, P.、Crocker, D. 和 M. Nottingham，
“棄用應用程序協議中的“X-”前綴和類似結構”，BCP 178，RFC 6648，2012 年 6 月。

[BCP90] Klyne, G. 、Nottingham, M. 和 J. Mogul，“消息標頭字段的註冊程序” ，BCP 90 ，RFC 3864 ，2004 年 9 月。

[OWASP] van der Stock, A., Ed.，“構建安全 Web 應用程序和 Web 服務指南” ，開放 Web 應用程序安全項目 (OWASP) 2.0.1 ，2005 年 7 月 ，<https://www.owasp.org/ >。

[休息] Fielding, R.，“架構風格和基於網絡的軟件架構設計” ，博士論文 ，加州大學爾灣分校 ，2000 年 9 月 ，<http://roy.gbiv.com/pubs/dissertation/top.htm> .

[RFC1945] Berners-Lee, T. 、Fielding, R. 和 H. Nielsen，“超文本傳輸協議 HTTP/1.0” ，RFC 1945 ，1996 年 5 月。

[RFC2049] Freed, N. 和 N. Borenstein，“多用途 Internet 郵件擴展 (MIME) 第五部分：一致性標準和示例” ，RFC 2049 ，1996 年 11 月。

[RFC2068] Fielding, R. 、Gettys, J. 、Mogul, J. 、Nielsen, H. 和 T. Berners-Lee，“超文本傳輸協議 HTTP/1.1” ，RFC 2068 ，1997 年 1 月。

[RFC2295] Holtman, K. 和 A. Mutz，“HTTP 中的透明內容協商” ，RFC 2295 ，1998 年 3 月。

[RFC2388] Masinter, L.，“從表單返回值：多部分/表單數據” ，RFC 2388 ，1998 年 8 月。

[RFC2557] Palme, F. 、Hopmann, A. 、Shelness, N. 和 E. Stefferud，“聚合文檔的 MIME 封裝 ，例如 HTML (MHTML)” ，RFC 2557 ，1999 年 3 月。

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L. 、Leach, P. 和 T. Berners-Lee，“超文本傳輸協議 HTTP/1.1” ，RFC 2616 ，1999 年 6 月。

[RFC2774] Frystyk, H. 、Leach, P. 和 S. Lawrence，“HTTP 擴展框架” ，RFC 2774 ，2000 年 2 月。

[RFC2817] Khare, R. 和 S. Lawrence，“在內部升級到 TLS HTTP/1.1” ，RFC 2817 ，2000 年 5 月。

[RFC2978] Freed, N. 和 J. Postel，“IANA 字符集註冊程序” ，BCP 19 ，RFC 2978 ，2000 年 10 月。

[RFC5226] Narten, T. 和 H. Alvestrand，“編寫
RFC 中的 IANA Considerations Section”，BCP 26，RFC 5226，2008 年 5 月。

[RFC5246] Dierks, T. 和 E. Rescorla，“傳輸層安全
(TLS) 協議版本 1.2”，RFC 5246，2008 年 8 月。

[RFC5322] Resnick, P.，“Internet 消息格式”，RFC 5322，
2008 年 10 月。

[RFC5789] Dusseault, L. 和 J. Snell，“HTTP 的 PATCH 方法”，RFC 5789，2010 年 3 月。

[RFC5905] Mills, D.、Martin, J.、Ed.、Burbank, J. 和 W. Kasch，
“網絡時間協議版本 4：協議和算法規範”，RFC 5905，2010 年 6 月。

[RFC5987] Reschke, J.，“字符集和語言編碼
超文本傳輸協議 (HTTP) 標頭字段參數”，RFC 5987，2010 年 8 月。

[RFC5988] Nottingham, M.，“網絡鏈接”，RFC 5988，2010 年 10 月。

[RFC6265] Barth, A.，“HTTP 狀態管理機制”，RFC 6265，2011 年 4 月。

[RFC6266] Reschke, J.，“在超文本傳輸協議 (HTTP) 中使用內容配置標頭字段”，RFC 6266，2011 年 6 月。

[RFC7238] Reschke, J.，“超文本傳輸協議 (HTTP)
狀態代碼 308（永久重定向）”，RFC 7238，2014 年 6 月。

附錄 A. HTTP 和 MIME 之間的區別

HTTP/1.1 使用許多為 Internet 消息格式 [RFC5322] 和多用途 Internet 郵件擴展 (MIME) 定義的結構

[RFC2045] 允許消息體以開放的多種表示形式和可擴展的報頭字段進行傳輸。

然而，RFC 2045 只關注電子郵件； HTTP 的應用程序有許多不同於電子郵件的特性；因此，HTTP 具有不同於 MIME 的特性。這些差異經過精心選擇，以優化二進制連接的性能，允許更大的自由使用新媒體類型，使日期比較更容易，並承認一些早期 HTTP 服務器和客戶端的做法。

本附錄描述了 HTTP 與 MIME 不同的特定領域。
進出嚴格 MIME 環境的代理和網關需要了解這些差異，並在必要時提供適當的轉換。

A.1. MIME 版本

HTTP 不是 MIME 兼容協議。但是，消息可以包含單個 MIME-Version 標頭字段以指示使用哪個版本的 MIME 協議來構造消息。使用 MIME-Version 標頭字段表示消息完全符合 MIME 協議（如 [RFC2045] 中定義）。

將 HTTP 消息導出到嚴格的 MIME 環境時，發件人有責任確保完全符合（如果可能）。

A2。轉換為規範形式

MIME 要求 Internet 郵件正文部分在傳輸之前轉換為規範格式，如 [RFC2049] 第 4 節所述。本文檔的第 3.1.1.3 節描述了通過
HTTP 傳輸時“文本”媒體類型的子類型所允許的形式。 [RFC2046] 要求具有“文本”類型的內容將換行符表示為 CRLF，並禁止在換行符序列之外使用 CR 或 LF。 HTTP 允許 CRLF、裸 CR 和裸 LF 指示文本內容中的換行符。

從 HTTP 到嚴格 MIME 環境的代理或網關應該將本文檔第 3.1.1.3 節中描述的文本媒體類型中的所有換行符轉換為 CRLF 的 RFC 2049 規範形式。但是請注意，由於內容編碼的存在以及 HTTP 允許使用某些不使用八位字節 13 和 10 分別表示 CR 和 LF 的字符集，這可能會變得複雜。

除非原始內容已經是規範形式，否則轉換將破壞應用於原始內容的任何加密校驗和。因此，對於在 HTTP 中使用此類校驗和的任何內容，建議使用規範形式。

A.3.日期格式轉換

HTTP/1.1 使用一組受限制的日期格式（第 7.1.1.1 節）來簡化日期比較過程。來自其他協議的代理和網關應該確保消息中出現的任何日期標頭字段符合 HTTP/1.1 格式之一，並在必要時重寫日期。

A.4.內容編碼的轉換

MIME 不包含任何等同於 HTTP/1.1 的 Content-Encoding 標頭字段的概念。由於這充當媒體類型的修飾符，從 HTTP 到 MIME 兼容協議的代理和網關應該更改 Content-Type 標頭字段的值或在轉發消息之前解碼表示。

（某些用於 Internet 郵件的 Content-Type 實驗性應用程序使用媒體類型參數“;conversions=<content-coding>”來執行與 Content-Encoding 等效的功能。但是，此參數不是 MIME 標準的一部分）.

A.5. Content-Transfer-Encoding 的轉換

HTTP 不使用 MIME 的 Content-Transfer-Encoding 字段。
從 MIME 兼容協議到 HTTP 的代理和網關需要在將響應消息傳遞給 HTTP 客戶端之前刪除任何內容傳輸編碼。

從 HTTP 到 MIME 兼容協議的代理和網關負責確保消息採用正確的格式和編碼，以便在該協議上安全傳輸，其中“安全傳輸”由所使用協議的限制定義。

這樣的代理或網關應該使用適當的內容傳輸編碼來轉換和標記數據，如果這樣做會提高通過目標協議安全傳輸的可能性。

A.6. MHTML 和行長度限制

與 MHTML [RFC2557] 實現共享代碼的 HTTP 實現需要注意 MIME 行長度限制。

由於 HTTP 沒有這個限制，所以 HTTP 不會折疊長行。由 HTTP 傳輸的 MHTML 消息遵循 MHTML 的所有約定，包括行長度限制和折疊、規範化等，因為 HTTP 將消息體傳輸為

有效載荷，並且除了“multipart/byteranges”類型（[RFC7233] 的附錄 A）之外，不解釋其中可能包含的內容或任何 MIME 標題行。

附錄 B. RFC 2616 的變化

此修訂版中的主要更改本質上是編輯性的：提取消息傳遞語法並將 HTTP 語義劃分為核心功能、條件請求、部分請求、緩存和身份驗證的單獨文檔。一致性語言已被修訂以明確針對需求，並且術語已得到改進以區分有效載荷與表示和表示與資源。

添加了一項新要求，即當 URI 中嵌入的語義與請求方法不一致時，禁用這些語義，因為這是互操作性失敗的常見原因。

（第 2 節）

添加了一種算法，用於確定有效載荷是否與特定標識符相關聯。（第 3.1.4.1 節）

文本媒體類型的默認 ISO-8859-1 字符集已被刪除；默認值現在是媒體類型定義所說的任何內容。

同樣，ISO-8859-1 的特殊處理已從 Accept-Charset 標頭字段中刪除。（第 3.1.1.3 節和第 5.3.3 節）

Content-Location 的定義已更改為不再影響用於解析相對 URI 引用的基本 URI，這是由於實施支持不佳以及可能破壞內容協商資源中的相對鏈接的不良影響。

（第 3.1.4.2 節）

為了與 [RFC7230] 的方法中立解析算法保持一致，GET 的定義已經放寬，因此請求可以有一個主體，即使主體對 GET 沒有意義。

（第 4.3.1 節）

服務器不再需要處理所有 Content-* 標頭字段，並且在 PUT 請求中明確禁止使用 Content-Range。

（第 4.3.4 節）

CONNECT 方法的定義已從 [RFC2817] 移至本規範。（第 4.3.6 節）

OPTIONS 和 TRACE 請求方法已被定義為安全的。（第 4.3.7 節和第 4.3.8 節）

由於廣泛部署的損壞實現，Expect 標頭字段的擴展機制已被刪除。 （第 5.1.1 節）

Max-Forwards 頭域被限制在 OPTIONS 和 TRACE 方法中，以前，擴展方法也可以使用它。 （第 5.1.2 節）

當沒有適用的引用 URI 時，“about:blank”URI 已被建議作為 Referer 標頭字段的值，這將這種情況與 Referer 字段未發送或已被刪除的其他情況區分開來。 （第 5.5.2 節）

以下狀態代碼現在是可緩存的（也就是說，它們可以在沒有顯式新鮮度信息的情況下被緩存存儲和重用）：204、404、405、414、501。（第 6 節）

201（已創建）狀態描述已更改，以允許創建多個資源的可能性。

（第 6.3.2 節）

203（非權威信息）的定義已經擴展到包括負載轉換的情況。

（第 6.3.4 節）

安全自動重定向的請求方法集不再關閉；用戶代理能夠根據請求方法語義做出決定。重定向狀態碼 301、302 和 307 不再對響應負載和用戶交互有規範要求。 （第 6.4 節）

狀態代碼 301 和 302 已更改為允許用戶代理將方法從 POST 重寫為 GET。 （第 6.4.2 和 6.4.3 節）

303（查看其他）狀態代碼的描述已更改為允許在提供明確的新鮮度信息時對其進行緩存，並且已為 GET 的 303 響應添加了特定定義。

（第 6.4.4 節）

由於代理的帶內配置方面的安全問題，305（使用代理）狀態代碼已被棄用。 （第 6.4.5 節）

400（錯誤請求）狀態代碼已放寬，因此它不僅限於語法錯誤。 （第 6.5.1 節）

426（需要升級）狀態代碼已從 [RFC2817] 中合併。 （第 6.5.15 節）

HTTP-date 和 Date 標頭字段的要求目標已減少到那些生成日期的系統，而不是所有發送日期的系統。（第 7.1.1 節）

Location 標頭字段的語法已更改為允許所有 URI 引用，包括相對引用和片段，以及關於何時不適合使用片段的一些說明。（第 7.1.2 節）

Allow 已被重新歸類為響應標頭字段，刪除了在 PUT 請求中指定它的選項。放寬了對Allow內容的要求；相應地，客戶不需要總是相信它的價值。（第 7.4.1 節）

已定義方法註冊表。（第 8.1 節）

本規範重新定義了 Status Code Registry；以前，它是在 [RFC2817] 的第 7.1 節中定義的。

（第 8.2 節）

內容編碼的註冊已更改為需要 IETF 審核。（第 8.4 節）

Content-Disposition 標頭字段已被刪除，因為它現在由 [RFC6266] 定義。

Content-MD5 標頭字段已被刪除，因為它在部分響應方面的實現不一致。

附錄 C. 導入的 ABNF

以下核心規則通過引用包含在 [RFC5234] 的附錄 B.1 中定義：ALPHA（字母）、CR（回車）、CRLF（CR LF）、CTL（控件）、DIGIT（十進制 0-9）、DQUOTE（雙引號）、HEXDIG（十六進制 0-9/AF/af）、HTAB（水平製表符）、LF（換行）、OCTET（任意 8 位數據序列）、SP（空格)和 VCHAR (任何可見的 US-ASCII 字符）。

[RFC7230] 中定義了以下規則：

```
BWS           = <BWS，請參閱 [RFC7230]，第 3.2.3 節> = <OWS，請參閱
OWS           [RFC7230]，第 3.2.3 節> = <RWS，請參閱 [RFC7230]，第 3.2.3 節>
2.7> absolute-URI = <absolute-URI，參見 [RFC7230]，Section 2.7> comment = <comment，參見 [RFC7230]，Section
[RFC7230]，Section 3.2.6> comment field-name = <comment，參見[RFC7230]，第 3.2 節> partial-URI =
<partial-URI，參見 [RFC7230]，第 2.7 節>
```

quoted-string = <quoted-string, see [RFC7230], Section 3.2.6> 標記
= <令牌 ,參見 [RFC7230] ,第 3.2.6 節>

附錄 D. 收集的 ABNF

在下面收集的 ABNF 中 ,列表規則根據 [RFC7230] 的第 1.2 節進行了擴展。

接受= [(“ , ” / (媒體範圍[接受參數])) *(OWS , [
OWS（媒體範圍[接受參數]）]）]
Accept-Charset = *(, OWS) ((字符集 / *) [重量]) *(
, [OWS ((字符集 / *) [權重]]))
Accept-Encoding = [(, / (編碼 [權重] (編碼 [權重]))) *(OWS , [OWS
)]]
接受語言 = *(, OWS) (語言範圍 [權重]) *(
“ , ”[OWS（語言範圍[權重]]）（ “ , ” /方法)*（OWS “ , ”[OWS方法] ）
允許 = [ )]

BWS = <BWS ,參見 [RFC7230] ,第 3.2.3 節>

內容編碼 = *(, OWS) 內容編碼 *(OWS , [OWS 內容編碼]
)
內容語言 = *(, OWS) 語言標籤 *(OWS , [OWS 語言標籤]
)
內容位置 = 絕對 URI / 部分 URI
內容類型 = 媒體類型

日期 = HTTP 日期

期望=“100-繼續”

來自=郵箱

GMT = %x47.4D.54 ;格林威治標準時間

HTTP-date = IMF-fixdate / obs-date

IMF-fixdate = day-name , SP date1 SP time-of-day SP GMT

位置 = URI 參考

最大轉發 = 1*DIGIT

OWS = <OWS ,參見 [RFC7230] ,第 3.2.3 節>

RWS = <RWS ,參見 [RFC7230] ,第 3.2.3 節> Referer = absolute-URI / partial-URI
Retry-After = HTTP-date / delay-seconds

服務器 = 產品 *(RWS (產品 / 評論))

URI-reference = <URI-reference, 參見 [RFC7230], Section 2.7> User-Agent = product *(RWS (product / comment))

變化 =)) * / (*(, OWS) 字段名 *(OWS , [OWS 字段名]

absolute-URI = <absolute-URI, see [RFC7230], Section 2.7> accept-ext = OWS ; OWS token [= (token / quoted-string) accept-params = weight *accept-ext asctime-date = day-name SP date3 SP time-of-day SP year]

charset = token codings = content-coding / “identity” / “*” comment = <comment, see [RFC7230], Section 3.2.6> content-coding = token

date1 = day SP month SP year date2 = day - month - 2DIGIT date3 = month SP (2DIGIT / (SP DIGIT) day = 2DIGIT day-name = %x4D.6F.6E ; Mon)

/ %x54.75.65 ;週二 / %x57.65.64 ;週三 / %x54.68.75 ;週四 / %x46.72.69 ;週五 / %x53.61.74 ;週六 / %x53.75.6E ;星期日名稱-l = %x4D.6F.6E.64.61.79 ;星期一 / %x54.75.65.73.64.61.79 ;週二 / %x57.65.64.6E.65.73.64.61.79 ;星期三 / %x54.68.75.72.73.64.61.79 ;星期四 / %x46.72.69.64.61.79 ;星期五 / %x53.61.74.75.72.64.61.79 ;星期六 / %x53.75.6E.64.61.79 ;週日延遲秒數 = 1*DIGIT

field-name = <註釋 ,參見 [RFC7230] ,第 3.2 節>

小時 = 2DIGIT

language-range = <語言範圍 ,參見 [RFC4647] ,第 2.1 節> language-tag = <語言標籤 ,參見 [RFC5646] ,第 2.1 節>

mailbox = <mailbox, see [RFC5322], Section 3.4> media-range = (*/* / (type /*) / (type / 子類型)) *(“;” OWS參數）

media-type = type / subtype *(OWS ; OWS parameter) method = token minute = 2DIGIT month = %x4A.61.6E ;一月 / %x46.65.62 ;二月 / %x4D.61.72 ;三月 / %x41.70.72 ;四月 / %x4D.61.79 ;五月 / %x4A.75.6E ;六月 / %x4A.75.6C ;七月 / %x41.75.67 ;八月 / %x53.65.70 ;九月 / %x4F.63.74 ;八月 / %x4E.6F.76 ;十一月 / %x44.65.63 ;十二月

obs-date = rfc850-date / asctime-date

parameter = token = (token / quoted-string) partial-URI = <partial-URI, see [RFC7230], Section 2.7> product = token [/ product-version] product-version = token quoted-string = <quoted-string, see [RFC7230], Section 3.2.6> qvalue = (0 [. *3DIGIT]) / (1 [. *3 0]

)

rfc850-date = day-name-l , SP date2 SP time-of-day SP GMT

第二 = 2DIGIT 子類型 = 令牌

time-of-day = hour : minute : second token = <token, see [RFC7230], Section 3.2.6> 類型 = token

權重 = OWS “;” OWS q= qvalue

年份 = 4DIGIT

指數

作者地址

Roy T. Fielding（編輯）
Adobe Systems Incorporated 345 Park Ave San
Jose, CA 95110 USA

電子郵件 :fielding@gbiv.com URI :http://
roy.gbiv.com/

Julian F. Reschke（編輯)greenbytes GmbH
Hafenweg 16 Muenster, NW 48155 德國

電子郵件 :julian.reschke@greenbytes.de URI :http://greenbytes.de/
tech/webdav/