- 在接收端，如果节点 B 需要确定报文确实是节点 A 写的，而且没有被篡改过，节点 B 就可以对签名进行检查。节点 B 接收经私有密钥扰码的签名，并应用了使用公开密钥的反函数。如果拆包后的摘要与节点 B 自己的摘要版本不匹配，要么就是报文在传输过程中被篡改了，要么就是发送端没有节点 A 的私有密钥（也
318 就是说它不是节点 A）。

14.6 数字证书

本节将介绍因特网上的“ID 卡”——数字证书。数字证书（通常被称作“certs”，有点像 certs 牌薄荷糖）中包含了由某个受信任组织担保的用户或公司的相关信息。

我们每个人都有很多形式的身份证明。有些 ID，比如护照和驾照，都足以在很多场合证明某人的身份。例如，你可以用美国的驾照在新年前夜搭乘前往纽约的航班，在你到那儿之后，接着用它来证明你的年龄，这样你就能和朋友们一起喝酒了。

受信程度更高的身份证明，比如护照，是由政府在特殊的纸上签发并盖章的。很难伪造，因此可以承载较高的信任度。有些公司的徽章和智能卡中包含有电子信息，以强化使用者的身份证明。有些绝密的政府组织甚至会对你的指纹或视网膜毛细血管模式进行匹配以便确认你的 ID！

有些形式的 ID，比如名片，相对来说更容易伪造，因此人们不太信任这些信息。虽然足以应付职场交流，但申请住房贷款时，可能就不足以证明你的就业情况了。

14.6.1 证书的主要内容

数字证书中还包含一组信息，所有这些信息都是由一个官方的“证书颁发机构”以数字方式签发的。基本的数字证书中通常包含一些纸质 ID 中常见的内容，比如：

- 对象的名称（人、服务器、组织等）；
319 • 过期时间；
- 证书发布者（由谁为证书担保）；
- 来自证书发布者的数字签名。

而且，数字证书通常还包括对象的公开密钥，以及对象和所用签名算法的描述性信息。任何人都可以创建一个数字证书，但并不是所有人都能够获得受人尊敬的签发权，从而为证书信息担保，并用其私有密钥签发证书。典型的证书结构如图 14-11 所示。

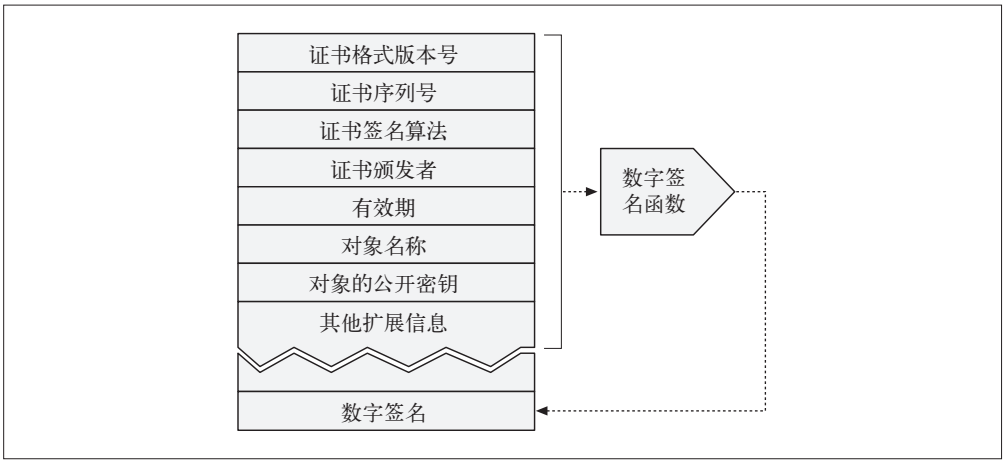


图 14-11　典型的数字签名格式

14.6.2　X.509 v3证书

不幸的是，数字证书没有单一的全球标准。就像不是所有印刷版 ID 卡都在同样的位置包含了同样的信息一样，数字证书也有很多略有不同的形式。 不过好消息就是现在使用的大多数证书都以一种标准格式——X.509 v3，来存储它们的信息。X.509 v3 证书提供了一种标准的方式，将证书信息规范至一些可解析字段中。不同类型的证书有不同的字段值，但大部分都遵循 X.509 v3 结构。表 14-2 介绍了 X.509 证书中的字段信息。

表14-2　X.509证书字段

字　　段	描　　述
版本	这个证书的 X.509 证书版本号。现在使用的通常都是版本 3
序列号	证书颁发机构（CA）生成的唯一整数。CA 生成的每个证书都要有一个唯一的序列号
签名算法 ID	签名所使用的加密算法。例如，“用 RSA 加密的 MD2 摘要”
证书颁发者	发布并签署这个证书的组织名称，以 X.500 格式表示
有效期	此证书何时有效，由一个起始日期和一个结束日期来表示
对象名称	证书中描述的实体，比如一个人或一个组织。对象名称是以 X.500 格式表示的
对象的公开密钥信息	证书对象的公开密钥，公开密钥使用的算法，以及所有附加参数
发布者唯一的 ID（可选）	可选的证书发布者唯一标识符，这样就可以重用相同的发布者名称
对象唯一的 ID（可选）	可选的证书对象唯一标识符，这样就可以重用相同的对象名称了

（续）

字　段	描　述
扩展	可选的扩展字段集（在版本 3 及更高的版本中使用）。每个扩展字段都被标识为关键或非关键的。关键扩展非常重要，证书使用者一定要能够理解。如果证书使用者无法识别出关键扩展字段，就必须拒绝这个证书。目前在使用的常用扩展字段包括： 基本约束 对象与证书颁发机构的关系 证书策略 授予证书的策略 密钥的使用 对公开密钥使用的限制
证书的颁发机构签名	证书颁发机构用指定的签名算法对上述所有字段进行的数字签名

基于 X.509 证书的签名有好几种，（其中）包括 Web 服务器证书、客户端电子邮件证书、软件代码签名证书和证书颁发机构证书。

14.6.3　用证书对服务器进行认证

通过 HTTPS 建立了一个安全 Web 事务之后，现代的浏览器都会自动获取所连接服务器的数字证书。如果服务器没有证书，安全连接就会失败。服务器证书中包含很多字段，其中包括：

- Web 站点的名称和主机名；
- Web 站点的公开密钥；
- 签名颁发机构的名称；
- 来自签名颁发机构的签名。

浏览器收到证书时会对签名颁发机构进行检查。[10] 如果这个机构是个很有权威的公
321 共签名机构，浏览器可能已经知道其公开密钥了（浏览器会预先安装很多签名颁发机构的证书）。这样，就可以像前面的 14.5 节中所讨论的那样验证签名了。图 14-12 说明了如何通过其数字签名来验证证书的完整性。

如果对签名颁发机构一无所知，浏览器就无法确定是否应该信任这个签名颁发机构，它通常会向用户显示一个对话框，看看他是否相信这个签名发布者。签名发布者可能是本地的 IT 部门或软件厂商。

注 10：浏览器和其他因特网应用程序都会尽量隐藏大部分证书管理的细节，使得浏览更加方便。但通过安全连接进行浏览时，所有主要的浏览器都允许你自己去检查所要对话站点的证书，以确保所有内容都是诚实可信的。

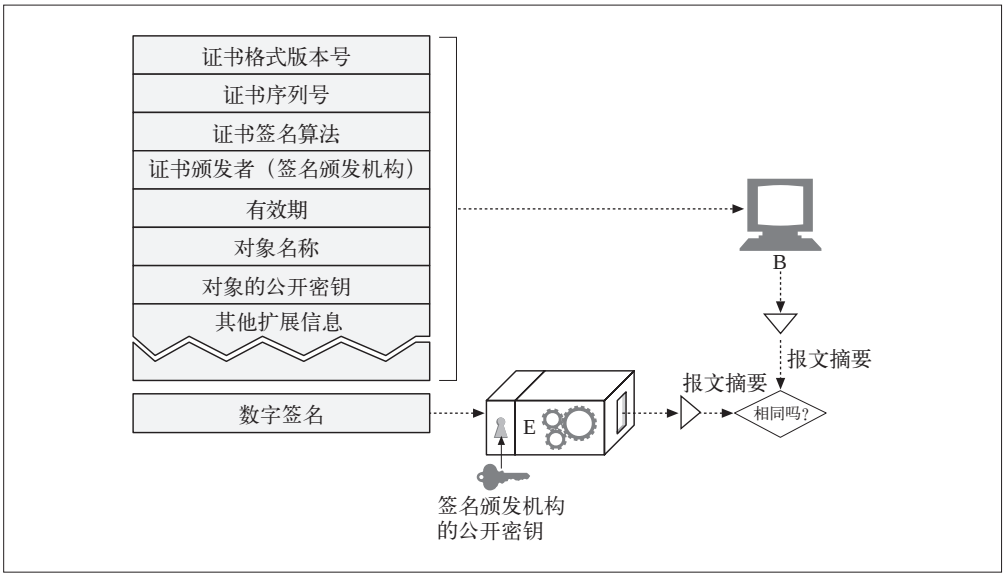


图 14-12　验证签名是真的

14.7　HTTPS——细节介绍

HTTPS 是最常见的 HTTP 安全版本。它得到了很广泛的应用，所有主要的商业浏览器和服务器上都提供 HTTPS。HTTPS 将 HTTP 协议与一组强大的对称、非对称和基于证书的加密技术结合在一起，使得 HTTPS 不仅很安全，而且很灵活，很容易在处于无序状态的、分散的全球互联网上进行管理。

HTTPS 加速了因特网应用程序的成长，已经成为基于 Web 的电子商务快速成长的主要推动力。在广域网中对分布式 Web 应用程序的安全管理方面，HTTPS 也是非常重要的。

14.7.1　HTTPS概述

HTTPS 就是在安全的传输层上发送的 HTTP。HTTPS 没有将未加密的 HTTP 报文发送给 TCP，并通过世界范围内的因特网进行传输（参见图 14-13a），它在将 HTTP 报文发送给 TCP 之前，先将其发送给了一个安全层，对其进行加密（参见图 14-13b）。

322

现在，HTTP 安全层是通过 SSL 及其现代替代协议 TLS 来实现的。我们遵循常见的用法，用术语 SSL 来表示 SSL 或者 TLS。

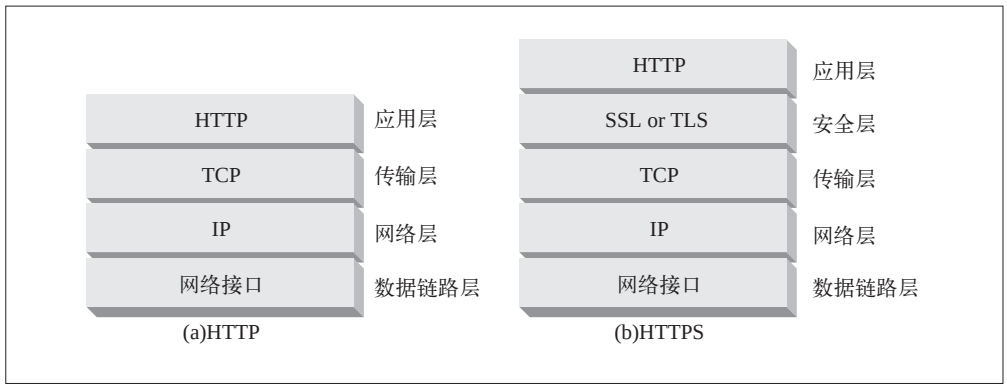


图 14-13　HTTP 传输层安全

14.7.2　HTTPS方案

现在，安全 HTTP 是可选的。因此，对 Web 服务器发起请求时，我们需要有一种方式来告知 Web 服务器去执行 HTTP 的安全协议版本。这是在 URL 的方案中实现的。

通常情况下，非安全 HTTP 的 URL 方案前缀为 http，如下所示：

http://www.joes-hardware.com/index.html

在安全 HTTPS 协议中，URL 的方案前缀为 https，如下所示：

https://cajun-shop.securesites.com/Merchant2/merchant.mv?Store_Code=AGCGS

请求一个客户端（比如 Web 浏览器）对某 Web 资源执行某事务时，它会去检查 URL 的方案。

- 如果 URL 的方案为 http，客户端就会打开一条到服务器端口 80（默认情况下）的连接，并向其发送老的 HTTP 命令（参见图 14-14a）。
- 如果 URL 的方案为 https，客户端就会打开一条到服务器端口 443（默认情况下）的连接，然后与服务器“握手”，以二进制格式与服务器交换一些 SSL 安全参数，附上加密的 HTTP 命令（参见图 14-14b）。

SSL 是个二进制协议，与 HTTP 完全不同，其流量是承载在另一个端口上的（SSL 通常是由端口 443 承载的）。如果 SSL 和 HTTP 流量都从端口 80 到达，大部分 Web 服务器会将二进制 SSL 流量理解为错误的 HTTP 并关闭连接。将安全服务进一步整合到 HTTP 层中去就无需使用多个目的端口了，在实际中这样不会引发严重的问题。

323　我们来详细介绍下 SSL 是如何与安全服务器建立连接的。

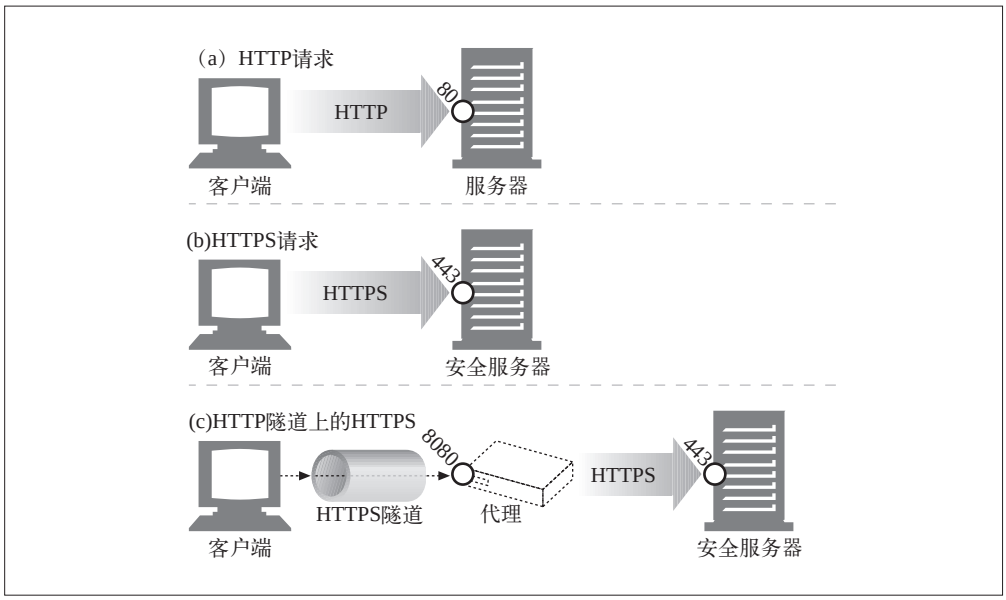


图 14-14　HTTP 和 HTTPS 端口号

14.7.3　建立安全传输

在未加密 HTTP 中，客户端会打开一条到 Web 服务器端口 80 的 TCP 连接，发送一条请求报文，接收一条响应报文，关闭连接。图 14-15a 对此序列进行了说明。

由于 SSL 安全层的存在，HTTPS 中这个过程会略微复杂一些。在 HTTPS 中，客户端首先打开一条到 Web 服务器端口 443（安全 HTTP 的默认端口）的连接。一旦建立了 TCP 连接，客户端和服务器就会初始化 SSL 层，对加密参数进行沟通，并交换密钥。握手完成之后，SSL 初始化就完成了，客户端就可以将请求报文发送给安全层了。在将这些报文发送给 TCP 之前，要先对其进行加密。图 14-15b 对此过程进行了说明。

14.7.4　SSL握手

在发送已加密的 HTTP 报文之前，客户端和服务器要进行一次 SSL 握手，在这个握手过程中，它们要完成以下工作：

- 交换协议版本号；
- 选择一个两端都了解的密码；
- 对两端的身份进行认证；
- 生成临时的会话密钥，以便加密信道。

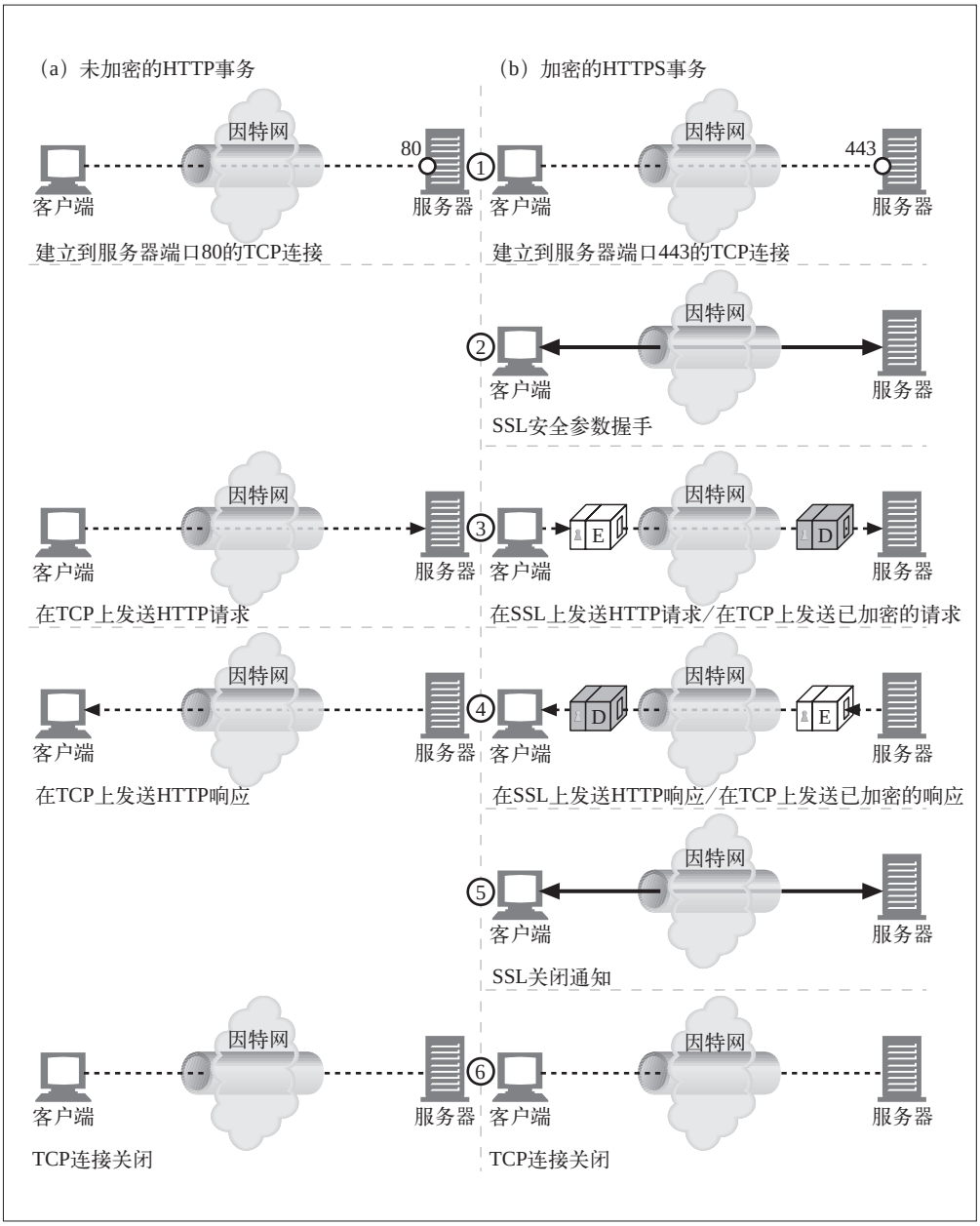


图 14-15　HTTP 和 HTTPS 事务

在通过网络传输任何已加密的 HTTP 数据之前，SSL 已经发送了一组握手数据来建立通信连接了。图 14-16 显示了 SSL 握手的基本思想。

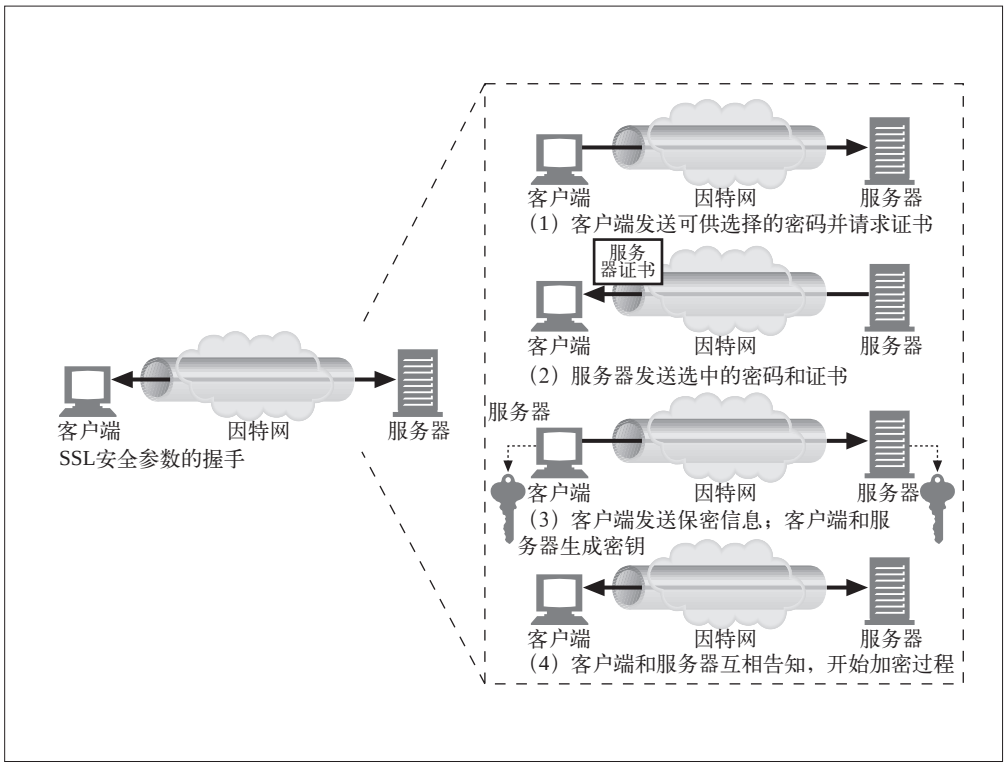


图 14-16 （简化版）SSL 握手

这是 SSL 握手的简化版本。根据 SSL 的使用方式，握手过程可能会复杂一些，但总的思想就是这样。

325

14.7.5 服务器证书

SSL 支持双向认证，将服务器证书承载回客户端，再将客户端的证书回送给服务器。而现在，浏览时并不经常使用客户端证书。大部分用户甚至都没有自己的客户端证书。[11] 服务器可以要求使用客户端证书，但实际中很少出现这种情况。[12]

另一方面，安全 HTTPS 事务总是要求使用服务器证书的。在一个 Web 服务器上执行安全事务，比如提交信用卡信息时，你总是希望是在与你所认为的那个组织对话。由知名权威机构签发的服务器证书可以帮助你在发送信用卡或私人信息之前评估你对服务器的信任度。

注 11：在某些公司的网络设置中会将客户端证书用于 Web 浏览，客户端证书还被用于安全电子邮件。未来，客户端证书可能会更经常地用于 Web 浏览，但现在它们发展的速度非常慢。

注 12：有些组织的内部网络会使用客户端证书来控制雇员对信息的访问。

服务器证书是一个显示了组织的名称、地址、服务器 DNS 域名以及其他信息的 X.509 v3 派生证书（参见图 14-17）。你和你所用的客户端软件可以检查证书，以确保所有的信息都是可信的。

326

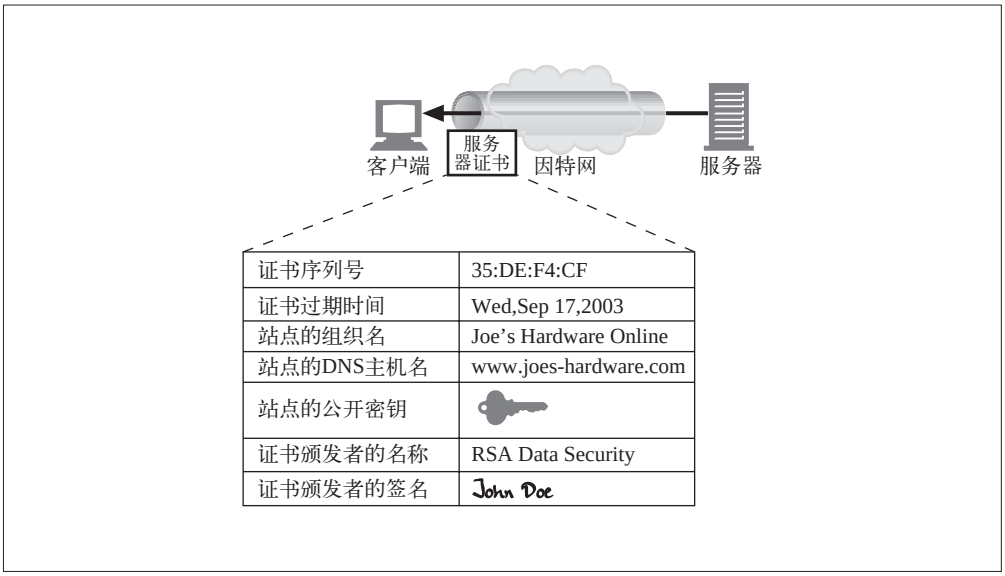


图 14-17　HTTPS 证书是带有站点信息的 X.509 证书

14.7.6　站点证书的有效性

SSL 自身不要求用户检查 Web 服务器证书，但大部分现代浏览器都会对证书进行简单的完整性检查，并为用户提供进行进一步彻查的手段。网景公司提出的一种 Web 服务器证书有效性算法是大部分浏览器有效性验证技术的基础。验证步骤如下所述。

- 日期检测

  首先，浏览器检查证书的起始日期和结束日期，以确保证书仍然有效。如果证书过期了，或者还未被激活，则证书有效性验证失败，浏览器显示一条错误信息。

- 签名颁发者可信度检测

  每个证书都是由某些证书颁发机构（CA）签发的，它们负责为服务器担保。证书有不同的等级，每种证书都要求不同级别的背景验证。比如，如果申请某个电子商务服务器证书，通常需要提供一个营业的合法证明。

  任何人都可以生成证书，但有些 CA 是非常著名的组织，它们通过非常清晰的流程来验证证书申请人的身份及商业行为的合法性。因此，浏览器会附带一个签名颁发机构的受信列表。如果浏览器收到了某未知（可能是恶意的）颁发机构

签发的证书，那它通常会显示一条警告信息。有些证书会携带到受信 CA 的有效签名路径，浏览器可能会选择接受所有此类证书。换句话说，如果某受信 CA 为“Sam 的签名商店”签发了一个证书，而 Sam 的签名商店也签发了一个站点证书，浏览器可能会将其作为从有效 CA 路径导出的证书接受。 327

- *签名检测*
  一旦判定签名授权是可信的，浏览器就要对签名使用签名颁发机构的公开密钥，并将其与校验码进行比较，以查看证书的完整性。

- *站点身份检测*
  为防止服务器复制其他人的证书，或拦截其他人的流量，大部分浏览器都会试着去验证证书中的域名与它们所对话的服务器的域名是否匹配。服务器证书中通常都包含一个域名，但有些 CA 会为一组或一群服务器创建一些包含了服务器名称列表或通配域名的证书。如果主机名与证书中的标识符不匹配，面向用户的客户端要么就去通知用户，要么就以表示证书不正确的差错报文来终止连接。

14.7.7 虚拟主机与证书

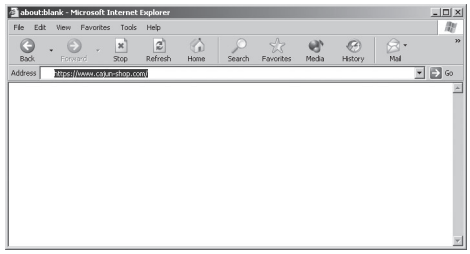
对虚拟主机（一台服务器上有多个主机名）站点上安全流量的处理有时是很棘手的。有些流行的 Web 服务器程序只支持一个证书。如果用户请求的是虚拟主机名，与证书名称并不严格匹配，浏览器就会显示警告框。

比如，我们来看以路易斯安那州为主题的电子商务网站 Cajun-Shop.com。站点的托管服务提供商提供的官方名称为 cajun-shop.securesites.com。用户进入 http://www.cajun-shop.com 时，服务器证书中列出的官方主机名（*.securesites.com）与用户浏览的虚拟主机名（www.cajun-shop.com）不匹配，以致出现图 14-18 中的警告。

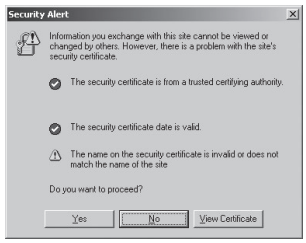
为防止出现这个问题，Cajun-Shop.com 的所有者会在开始处理安全事务时，将所有用户都重定向到 cajun-shop.securesites.com。虚拟主机站点的证书管理会稍微棘手一些。

14.8 HTTPS客户端实例

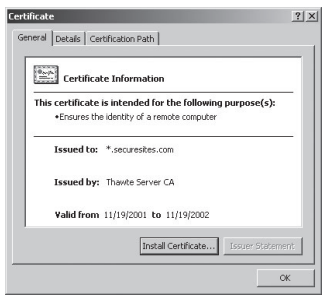
SSL 是个复杂的二进制协议。除非你是密码专家，否则就不应该直接发送原始的 SSL 流量。幸运的是，借助一些商业或开源的库，编写 SSL 客户端和服务器并不十分困难。



（a）由于站点是虚拟主机站点，而证书的主机名为*.securesites.com，所以这个URL（www.cajun-shop.com）中的主机名与证书中的名称不匹配。



(b)对话框警告用户站点证书的日期有效，而且来自有效的证书颁发机构，但证书中所列名称与URL所请求的站点不相符。



（c）为了获取更详细的信息，用户点击了“查看证书”按钮，看到证书是一个通配证书，主机名为*.securesites.com。有此信息之后，用户就可以判定是该接受还是该拒绝这个证书了。



(d)接受证书，通过安全HTTPS协议装载页面。为避免此类用户错误，这个特定的站点将所有 HTTPS流量都导向了主机别名cajun-shop.securesites.com。这个虚拟主机名与ISP在其商业包中提供的证书名字相符。

图 14-18　证书名不匹配引发的证书错误对话框

14.8.1　OpenSSL

OpenSSL 是 SSL 和 TLS 最常见的开源实现。OpenSSL 项目由一些志愿者合作开发，目标是开发一个强壮的、具有完备功能的商业级工具集，以实现 SSL 和 TLS 协议以及一个全功能的通用加密库。可以从 http://www.openssl.org 上获得 OpenSSL 的相关信息，并下载相应软件。

328

你可能还听说过 SSLeay（读作 S-S-L-e-a-y）。OpenSSL 是 SSLeay 库的后继者，接口非常相似。SSLeay 最初是由 Eric A. Young（就是 SSLeay 中的“eay”）开发的。

14.8.2 简单的HTTPS客户端

本节我们将用 OpenSSL 包来编写一个非常初级的 HTTPS 客户端。这个客户端与服务器建立一条 SSL 连接，打印一些来自站点服务器的标识信息，通过安全信道发送 HTTP GET 请求，接收 HTTP 响应，并将响应打印出来。 329

下面显示的 C 程序是普通 HTTPS 客户端的 OpenSSL 实现。为了保持其简洁性，程序中没有包含差错处理和证书处理逻辑。

这个示例程序中删除了差错处理功能，所以只能将其用于示例。在一般的有差错存在的环境中，软件会崩溃或者无法正常运行。 330

```
/**********************************************************************
* https_client.c --- very simple HTTPS client with no error checking
*      usage: https_client servername
**********************************************************************/

#include <stdio.h>
#include <memory.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#include <openssl/crypto.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

void main(int argc, char **argv)
{

    SSL *ssl;
    SSL_CTX *ctx;
    SSL_METHOD *client_method;
    X509 *server_cert;
    int sd,err;
    char *str,*hostname,outbuf[4096],inbuf[4096],host_header[512];
    struct hostent *host_entry;
    struct sockaddr_in server_socket_address;
    struct in_addr ip;

    /*=======================================*/
    /* (1) initialize SSL library */
    /*=======================================*/
```

```
SSLeay_add_ssl_algorithms( );
client_method = SSLv2_client_method( );
SSL_load_error_strings( );
ctx = SSL_CTX_new(client_method);
printf("(1) SSL context initialized\n\n");
/*============================================*/
/* (2) convert server hostname into IP address */
/*============================================*/

hostname = argv[1];
host_entry = gethostbyname(hostname);
bcopy(host_entry->h_addr, &(ip.s_addr), host_entry->h_length);

printf("(2) '%s' has IP address '%s'\n\n", hostname, inet_ntoa(ip));
/*================================================*/
/* (3) open a TCP connection to port 443 on server */
/*================================================*/

sd = socket (AF_INET, SOCK_STREAM, 0);

memset(&server_socket_address, '\0', sizeof(server_socket_address));
server_socket_address.sin_family = AF_INET;
server_socket_address.sin_port = htons(443);
memcpy(&(server_socket_address.sin_addr.s_addr),
       host_entry->h_addr, host_entry->h_length);

err = connect(sd, (struct sockaddr*) &server_socket_address,
              sizeof(server_socket_address));
if (err < 0) { perror("can't connect to server port"); exit(1); }

printf("(3) TCP connection open to host '%s', port %d\n\n",
       hostname, server_socket_address.sin_port);

/*=========================================================*/
/* (4) initiate the SSL handshake over the TCP connection */
/*=========================================================*/

ssl = SSL_new(ctx);      /* create SSL stack endpoint */
SSL_set_fd(ssl, sd);     /* attach SSL stack to socket */
err = SSL_connect(ssl); /* initiate SSL handshake */

printf("(4) SSL endpoint created & handshake completed\n\n");

/*============================================*/
/* (5) print out the negotiated cipher chosen */
/*============================================*/

printf("(5) SSL connected with cipher: %s\n\n", SSL_get_cipher(ssl));

/*=======================================*/
/* (6) print out the server's certificate */
/*=======================================*/

server_cert = SSL_get_peer_certificate(ssl);
printf("(6) server's certificate was received:\n\n");
str = X509_NAME_oneline(X509_get_subject_name(server_cert), 0, 0);
printf("      subject: %s\n", str);
```

```
    str = X509_NAME_oneline(X509_get_issuer_name(server_cert), 0, 0);
    printf("      issuer: %s\n\n", str);

    /* certificate verification would happen here */

    X509_free(server_cert);

    /*---------------------------------------------------*/
    /* (7) handshake complete --- send HTTP request over SSL */
    /*---------------------------------------------------*/

    sprintf(host_header,"Host: %s:443\r\n",hostname);
    strcpy(outbuf,"GET / HTTP/1.0\r\n");
    strcat(outbuf,host_header);
    strcat(outbuf,"Connection: close\r\n");
    strcat(outbuf,"\r\n");

    err = SSL_write(ssl, outbuf, strlen(outbuf));
    shutdown (sd, 1); /* send EOF to server */

    printf("(7) sent HTTP request over encrypted channel:\n\n%s\n",outbuf);

    /*-----------------------------------------------*/
    /* (8) read back HTTP response from the SSL stack */
    /*-----------------------------------------------*/

    err = SSL_read(ssl, inbuf, sizeof(inbuf) - 1);
    inbuf[err] = '\0';
    printf ("(8) got back %d bytes of HTTP response:\n\n%s\n",err,inbuf);

    /*-----------------------------------------------*/
    /* (9) all done, so close connection & clean up */
    /*-----------------------------------------------*/

    SSL_shutdown(ssl);
    close (sd);
    SSL_free (ssl);
    SSL_CTX_free (ctx);

    printf("(9) all done, cleaned up and closed connection\n\n");
}
```

这个例子是在 Sun Solaris 上面编译运行的，但它说明了 SSL 在很多 OS 平台上的工作原理。由于 OpenSSL 提供了一些强有力的特性，包括所有加密、密钥以及证书管理在内的整个程序都可以在一个几页左右的 C 程序中实现。

331 ≀ 332

下面按部分分析下这个程序。

- 程序的顶端包含了一些用于支持 TCP 联网和 SSL 的支撑文件。
- 第 1 部分通过调用 `SSL_CTX_new` 创建了本地上下文，以记录握手参数及与 SSL 连接有关的其他状态。
- 第 2 部分通过 Unix 的 `gethostbyname` 函数将（由一个命令行变元提供的）输入主机名转换成了 IP 地址。其他平台可能会通过其他方式来提供这项功能。

- 第 3 部分通过创建本地套接字、设置远端地址信息并连接到远端服务器，建立了一条到服务器端口 443 的 TCP 连接。
- 一旦 TCP 连接建立起来，就用 SSL_new 和 SSL_set_fd 将 SSL 层附加到 TCP 连接之上，并调用 SSL_connect 与服务器进行 SSL 握手。第 4 部分完成时，我们就建立了一个已选好密码且交换过证书的可运行的 SSL 信道。
- 第 5 部分打印了选中的批量加密密码值。
- 第 6 部分打印了服务器回送的 X.509 证书中包含的部分信息，其中包括与证书持有者和颁发证书的组织有关的信息。OpenSSL 库没有对服务器证书中的信息作任何特殊的处理。实际的 SSL 应用程序，比如 Web 浏览器会对证书进行一些完整性检查，以确保证书是正确签发的，且是来自正确主机的。我们在 14.7.6 节讨论了浏览器对服务器证书所做的处理。
- 此时，我们的 SSL 连接就已经可以用于安全数据的传输了。在第 7 部分中，用 SSL_write 在 SSL 信道上发送了简单的 HTTP 请求 GET / HTTP/1.0，然后关闭了连接的输出端。
- 在第 8 部分中，用 SSL_read 从连接上读回响应，并将其打印到屏幕上。SSL 层负责所有的加密和解密工作，因此可以直接读写普通的 HTTP 命令。
- 最后，在第 9 部分进行了一些清理工作。

更多与 OpenSSL 库有关的信息请参见 http://www.openssl.org。

14.8.3 执行OpenSSL客户端

下面显示了指向安全服务器时这个简单 HTTP 客户端的输出。在这个例子中，客户端指向了摩根士丹利的在线证券主页。在线交易公司都在广泛使用 HTTPS。

333

```
% https_client clients1.online.msdw.com
(1) SSL context initialized

(2) 'clients1.online.msdw.com' has IP address'63.151.15.11'

(3) TCP connection open to host 'clients1.online.msdw.com', port 443

(4) SSL endpoint created & handshake completed

(5) SSL connected with cipher: DES-CBC3-MD5

(6) server's certificate was received:

      subject: /C=US/ST=Utah/L=Salt Lake City/O=Morgan Stanley/OU=Online/CN=
        clients1.online.msdw.com
      issuer: /C=US/O=RSA Data Security, Inc./OU=Secure Server Certification
        Authority
(7) sent HTTP request over encrypted channel:
```

```
GET / HTTP/1.0
Host: clients1.online.msdw.com:443
Connection: close

(8) got back 615 bytes of HTTP response:

HTTP/1.1 302 Found
Date: Sat, 09 Mar 2002 09:43:42 GMT
Server: Stronghold/3.0 Apache/1.3.14 RedHat/3013c (Unix) mod_ssl/2.7.1 OpenSSL/0.9.6
Location: https://clients.online.msdw.com/cgi-bin/ICenter/home
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>302 Found</TITLE>
</HEAD><BODY>
<H1>Found</H1>
The document has moved <A HREF="https://clients.online.msdw.com/cgi-bin/ICenter/
home">here</A>.<P>
<HR>
<ADDRESS>Stronghold/3.0Apache/1.3.14 RedHat/3013c Server at clients1.online.msdw.com
Port 443</ADDRESS>
</BODY></HTML>

(9) all done, cleaned up and closed connection
```

只要完成了前面 4 个部分，客户端就有了一条打开的 SSL 连接。这样它就可以查询连接的状态，选择参数，检查服务器证书了。

在这个例子中，客户端和服务器对 DES-CBC3-MD5 批量加密密码进行了沟通。你还能看到服务器站点证书属于美国犹他州盐湖城的摩根士丹利组织。证书由 RSA 数据安全组织授予，主机名为 clients1.online.msdw.com，与请求相符。 334

只要建立起了 SSL 信道，并且客户端对站点的证书没有异议，就可以通过安全信道来发送其 HTTP 请求了。在我们这个例子中，客户端发送了一条简单的“GET / HTTP/1.0”HTTP 请求，并收到了 302 Redirect 响应，请求用户去获取另一个 URL。

14.9 通过代理以隧道形式传输安全流量

客户端通常会用 Web 代理服务器（参见第 6 章）代表它们来访问 Web 服务器。比如，很多公司都会在公司网络和公共因特网的安全边界上放置一个代理（参见图 14-19）。代理是防火墙路由器唯一允许进行 HTTP 流量交换的设备，它可能会进行病毒检测或其他的内容控制工作。

但只要客户端开始用服务器的公开密钥对发往服务器的数据进行加密，代理就再也不能读取 HTTP 首部了！代理不能读取 HTTP 首部，就无法知道应该将请求转向何处了（参见图 14-20）。

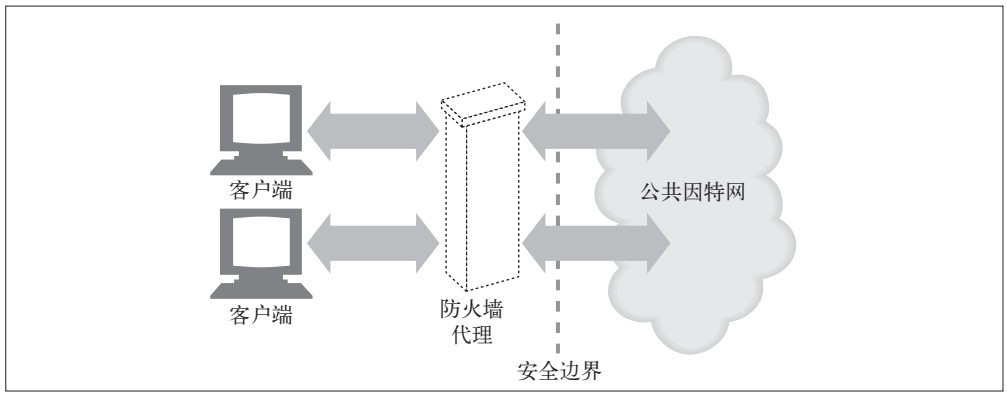


图 14-19　公司防火墙代理

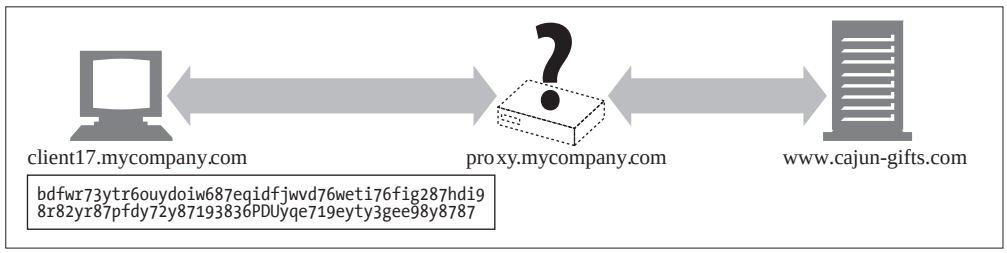


图 14-20　代理无法转发加密请求

为了使 HTTPS 与代理配合工作，要进行几处修改以告知代理连接到何处。一种常
335 用的技术就是 HTTPS SSL 隧道协议。使用 HTTPS 隧道协议，客户端首先要告知代理，它想要连接的安全主机和端口。这是在开始加密之前，以明文形式告知的，所以代理可以理解这条信息。

HTTP 通过新的名为 CONNECT 的扩展方法来发送明文形式的端点信息。CONNECT 方法会告诉代理，打开一条到所期望主机和端口号的连接。这项工作完成之后，直接在客户端和服务器之间以隧道形式传输数据。CONNECT 方法就是一条单行的文本命令，它提供了由冒号分隔的安全原始服务器的主机名和端口号。host:port 后面跟着一个空格和 HTTP 版本字符串，再后面是 CRLF。接下来是零个或多个 HTTP 请求首部行，后面跟着一个空行。空行之后，如果建立连接的握手过程成功完成，就可以开始传输 SSL 数据了。下面是一个例子：

```
CONNECT home.netscape.com:443 HTTP/1.0
User-agent: Mozilla/1.1N

<raw SSL-encrypted data would follow here...>
```

在请求中的空行之后，客户端会等待来自代理的响应。代理会对请求进行评估，确保它是有效的，而且用户有权请求这样一条连接。如果一切正常，代理会建立一条

到目标服务器的连接。如果成功，就向客户端发送一条 200 Connection Established 响应。

```
HTTP/1.0 200 Connection established
Proxy-agent: Netscape-Proxy/1.1
```

更多有关安全隧道和安全代理的信息，请回顾 8.5 节。

14.10 更多信息

安全和密码问题是非常重要，也非常复杂的问题。如果想学习更多有关 HTTP 安全性、数字加密技术、数字证书以及公开密钥基础设施方面的内容，可以从下面这几个地方开始。

14.10.1 HTTP安全性

- *Web security, Privacy & Commerce*[13]（《Web 安全与电子商务》）
  Simson Garfinkel 著，O'Reilly & Associates 公司。这是 Web 安全以及 SSL/TLS 和数字证书方面最好、最可读的入门型书籍之一。

- http://www.ietf.org/rfc/rfc2818.txt
  RFC 2818，“HTTP Over TLS”（“TLS 上的 HTTP”），说明了如何在 SSL 的后继协议——TLS 协议之上实现安全 HTTP。

- http://www.ietf.org/rfc/rfc2817.txt
  RFC 2817，“Upgrading to TLS Within HTTP/1.1”（“在 HTTP/1.1 中升级到 TLS”），说明了如何使用 HTTP/1.1 中的升级机制在现存的 TCP 连接上启动 TLS。这样非安全和安全 HTTP 流量就可以共享相同的知名端口了（在这种情况下，使用的是 http: 的 80 端口，而不是 https: 的 443 端口）。还可以使用虚拟主机技术。这样，使用一台 HTTP+TLS 服务器就可以区分出发往同一个 IP 地址上不同主机名的流量了。

14.10.2 SSL与TLS

- http://www.ietf.org/rfc/rfc2246.txt
  RFC 2246，“The TLS Protocol Version 1.0”（“TLS 协议版本 1.0”），对（SSL 的后继协议）TLS 协议的版本 1.0 进行了规范。TLS 提供了因特网上通信的私密性。协议允许客户端 / 服务器应用程序以防止窃听、篡改以及伪造报文的方式进行通信。

- http://developer.netscape.com/docs/manuals/security/sslin/contents.htm
  “Introduction to SSL”（“SSL 简介”）介绍了 SSL 协议。SSL 最初是由网景公司

336

注 13：本书中文版由中国电力出版社出版。（编者注）

开发的，已广泛应用于万维网上客户端和服务器间的认证及加密通信。

- http://www.netscape.com/eng/ssl3/draft302.txt
  “The SSL Protocol Version 3.0”（“SSL 协议版本 3.0”）是网景公司 1996 年的 SSL 规范。

- http://developer.netscape.com/tech/security/ssl/howitworks.html
  “How SSL Works”（“SSL 是如何工作的”）是网景公司对密钥加密技术的介绍。

- http://www.openssl.org
  OpenSSL 项目是一个合作开发项目，目的是开发一个强壮的、全功能的、商业级开源工具集，以实现安全套接字层（SSL v2/v3）和传输层安全（TLS v1）协议以及强大的通用密码库。这个项目由全世界范围内的志愿者社区管理，那些志愿者通过因特网进行交流、制定计划、开发 OpenSSL 工具集并撰写相关文档。OpenSSL 基于 Eric A. Young 和 Tim J. Hudson 开发的优秀 SSLeay 库。OpenSSL 工具集有一个 Apache 风格的许可证，这基本上就意味着只要遵循一些基本的许可条件，就可免费获得并将其用于商业或非商业目的。

14.10.3 公开密钥基础设施

- http://www.ietf.org/html.charters/pkix-charter.html
  IETF PKIX 工作组组建于 1995 年，目的是开发一些因特网标准，支持基于
337 X.509 的公开密钥基础设施。这是对此小组活动很好的总结。

- http://www.ietf.org/rfc/rfc2459.txt
  RFC 2459，“Internet X.509 Public Key Infrastructure Certificate and CRL Profile”（“因特网 X.509 公开密钥基础设施证书及 CRL 概述”），详细介绍了 X.509 v3 数字证书。

14.10.4 数字密码

- *Applied Cryptography*[14]（《应用密码学》）
  Bruce Schneier 著，John Wiley & Sons 公司出版。这是为实现者编写的经典密码学书籍。

- *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*[15]（《密码故事——人类智力的另类较量》）
  Simon Singh 著，Anchor Books 公司出版。这是一本有趣的密码学入门书籍。它
338 不是为技术专家编写的，而是一本生动的密码学历史读物。

注 14：本书中文版由机械工业出版社出版。（编者注）
注 15：本书中文版由海南出版社出版。（编者注）

第四部分

实体、编码和国际化

第四部分讲述的内容都与 HTTP 报文的实体主体和被实体主体作为货物承载的内容有关。

- 第 15 章讲述了 HTTP 内容的格式和语法。
- 第 16 章探讨了允许世界各地的人们相互交换内容的各种 Web 标准，这些内容由各种不同语言和不同字符集构成。
- 第 17 章讲解了各种用于协商可接受内容的机制。

339
~
340

第15章

实体和编码

每天都有数以亿计的各种媒体对象经由 HTTP 传送，如图像、文本、影片以及软件程序等。只要你能叫出名字，HTTP 就可以传送。HTTP 还会确保它的报文被正确传送、识别、提取以及适当处理。具体说来，HTTP 要确保它所承载的“货物”满足以下条件。

- 可以被正确地识别（通过 Content-Type 首部说明媒体格式，Content-Language 首部说明语言），以便浏览器和其他客户端能正确处理内容。
- 可以被正确地解包（通过 Content-Length 首部和 Content-Encoding 首部）。
- 是最新的（通过实体验证码和缓存过期控制）
- 符合用户的需要（基于 Accept 系列的内容协商首部）
- 在网络上可以快速有效地传输（通过范围请求、差异编码以及其他数据压缩方法）
- 完整到达、未被篡改（通过传输编码首部和 Content-MD5 校验和首部）

为了实现这些目标，HTTP 使用了完善的标签来描述承载内容的实体。

本章讨论各种实体、与它们相关的实体首部以及它们是如何运作来传送网站“货物”的。我们将展示 HTTP 是如何提供必需的内容大小、类型以及编码的。我们还要解释 HTTP 实体的某些更复杂和强大的特性，比如范围请求、差异编码、摘要以及分块编码等。

本章涵盖了下列内容。

- 作为 HTTP 数据的容器，HTTP 报文实体有哪些格式和行为。
- HTTP 如何描述实体的主体大小，HTTP 为确定大小制定了哪些规则。
- 为了使客户端正确处理内容，使用了哪些实体首部来描述内容的格式、字母和
341 语言。
- 可逆的内容编码，发送方可以在发送之前用它来转换内容的数据格式，使其占用更小的空间，或者更安全。
- 传输编码和分块编码。传输编码可以改变 HTTP 传输数据的方式，以改善某些类型内容的通信能力。分块编码是一种特殊的传输编码，它把数据切分为若干块，这样可以更可靠地传输长度未知的内容。
- 标记、标签、时间以及校验和等一整套机制，帮助客户端获取所请求内容的最新版本。
- 可用作内容版本号的验证码，网站应用可以通过它确保接收最新的内容。还有设计用来控制对象新鲜度的各种 HTTP 首部字段。
- 范围，在恢复中断的传输方面很有用。
- HTTP 差异编码扩展，它使客户端只需要请求网页中和前一次相比有改变的部分。
- 实体主体的校验和，可以用来检测经过若干代理之后，实体的内容是否发生了改变。

15.1 报文是箱子，实体是货物

如果把 HTTP 报文想象成因特网货运系统中的箱子，那么 HTTP 实体就是报文中实际的货物。图 15-1 展示了一个简单的实体，装在 HTTP 响应报文中。

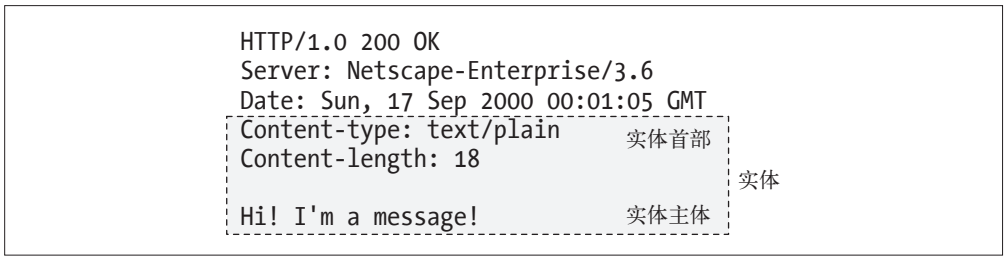


图 15-1 报文实体由实体首部和实体主体组成

实体首部指出这是一个纯文本文档（`Content-Type:text/plain`），它只有 18 个字节长（`Content-Length:18`）。和往常一样，一个空白行（CRLF）把首部字段同主体的开始部分分隔开来。

HTTP 实体首部（参见第 3 章）描述了 HTTP 报文的内容。HTTP/1.1 版定义了以下 10 个基本字体首部字段。

- `Content-Type`
  实体中所承载对象的类型。
- `Content-Length`
  所传送实体主体的长度或大小。
- `Content-Language`
  与所传送对象最相配的人类语言。
- `Content-Encoding`
  对象数据所做的任意变换（比如，压缩）。
- `Content-Location`
  一个备用位置，请求时可通过它获得对象。
- `Content-Range`
  如果这是部分实体，这个首部说明它是整体的哪个部分。
- `Content-MD5`
  实体主体内容的校验和。

342

- `Last-Modified`
  所传输内容在服务器上创建或最后修改的日期时间。

- `Expires`
  实体数据将要失效的日期时间。

- `Allow`
  该资源所允许的各种请求方法，例如，GET 和 HEAD。

- `ETag`
  这份文档特定实例（参见 15.7 节）的唯一验证码。`ETag` 首部没有正式定义为实体首部，但它对许多涉及实体的操作来说，都是一个重要的首部。

- `Cache-Control`
  指出应该如何缓存该文档。和 `ETag` 首部类似，`Cache-Control` 首部也没有正式定义为实体首部。

实体主体

实体主体中就是原始货物啦。[1] 任何其他描述性的信息都包含在首部中。因为货物（也就是实体主体）只是原始数据，所以需要实体首部来描述数据的意义。例如，`Content-Type` 实体首部告诉我们如何去解释数据（是图像还是文本等），而 `Content-Encoding` 实体首部告诉我们数据是不是已被压缩或者重编码。我们将在随后的小节中讨论所有这些方面及更多的内容。

首部字段以一个空白的 CRLF 行结束，随后就是实体主体的原始内容。不管内容是什么，文本或二进制的、文档或图像、压缩的或未压缩的、英语、法语或日语，都
343 紧随这个 CRLF 之后。

图 15-2 展示了两个实际的 HTTP 报文的例子。一个携带着文本实体，另一个承载的是图像实体。十六进制的数值中展示的是报文的实际内容。

- 在图 15-2a 中，实体主体从第 65 个字节开始，紧随首部末尾的 CRLF。实体主体中包含了“Hi! I'm a message!”这句话的 ASCII 编码字符。
- 在图 15-2b 中，实体主体从第 67 字节开始。实体主体包含了一个 GIF 格式图像的二进制内容。GIF 文件以 6 个字节的版本标志开头，后面是 16 位的宽度和 16 位的高度。可以在实体主体中直接看到这 3 项内容。

注 1：如果有 `Content-Encoding` 首部的话，实体主体的内容就已经被指定的内容编码算法进行过编码了，第一个字节就是编码（比如，压缩）后的货物的第一个字节。

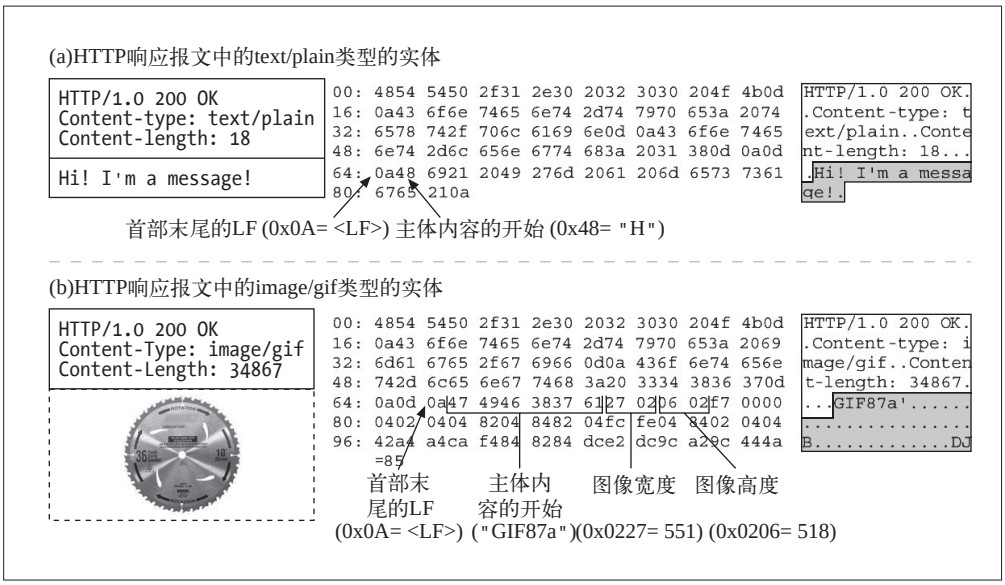


图 15-2　实际报文内容（紧随空白的 CRLF 之后的原始报文内容）的十六进制转储显示

15.2　Content-Length：实体的大小

Content-Length 首部指示出报文中实体主体的字节大小。这个大小是包含了所有内容编码的，比如，对文本文件进行了 gzip 压缩的话，Content-Length 首部就是压缩后的大小，而不是原始大小。

除非使用了分块编码，否则 Content-Length 首部就是带有实体主体的报文必须使用的。使用 Content-Length 首部是为了能够检测出服务器崩溃而导致的报文截尾，并对共享持久连接的多个报文进行正确分段。

15.2.1　检测截尾

HTTP 的早期版本采用关闭连接的办法来划定报文的结束。但是，没有 Content-Length 的话，客户端无法区分到底是报文结束时正常的连接关闭，还是报文传输中由于服务器崩溃而导致的连接关闭。客户端需要通过 Content-Length 来检测报文截尾。

344

报文截尾的问题对缓存代理服务器来说尤其严重。如果缓存服务器收到被截尾的报文却没有识别出截尾的话，它可能会存储不完整的内容并多次使用它来提供服务。缓存代理服务器通常不会为没有显式 Content-Length 首部的 HTTP 主体做缓存，以此来减小缓存已截尾报文的风险。

15.2.2 错误的Content-Length

错误的 Content-Length 比缺少 Content-Length 还要糟糕。因为某些早期的客户端和服务器在 Content-Length 计算上存在一些众所周知的错误，有些客户端、服务器以及代理中就包含了特别的算法，用来检测和纠正与有缺陷服务器的交互过程。HTTP/1.1 规定用户 Agent 代理应该在接收且检测到无效长度时通知用户。

15.2.3 Content-Length与持久连接

Content-Length 首部对于持久连接是必不可少的。如果响应通过持久连接传送，就可能有另一条 HTTP 响应紧随其后。客户端通过 Content-Length 首部就可以知道报文在何处结束，下一条报文从何处开始。因为连接是持久的，客户端无法依赖连接关闭来判别报文的结束。如果没有 Content-Length 首部，HTTP 应用程序就不知道某个实体主体在哪里结束，下一条报文从哪里开始。

我们将在 15.6 节看到，有一种情况下，使用持久连接时可以没有 Content-Length 首部，即采用*分块编码*（chunked encoding）时。在分块编码的情况下，数据是分为一系列的块来发送的，每块都有大小说明。哪怕服务器在生成首部的时候不知道整个实体的大小（通常是因为实体是动态生成的），仍然可以使用分块编码传输若干已知大小的块。

15.2.4 内容编码

HTTP 允许对实体主体的内容进行编码，比如可以使之更安全或进行压缩以节省空间（本章稍后将详细解释压缩的问题）。如果主体进行了内容编码，Content-Length 首部说明的就是*编码后*（encoded）的主体的字节长度，而*不是*未编码的原始主体的长度。

某些 HTTP 应用程序在这方面搞错了，发送的是数据编码之前的大小，这会导致严
345 重的错误，尤其是用在持久连接上。不幸的是，HTTP/1.1 规范中没有首部可以用来说明原始的、未编码的主体的长度，这就让客户端难以验证*解码过程*的完整性。[2]

15.2.5 确定实体主体长度的规则

下面列出的规则说明了在若干不同的情况下如何正确计算主体的长度和结束位置。这些规则应当按顺序应用，谁先匹配就用谁。

注 2：Content-MD5 首部也不行，它用来说明文档的 128 位 MD5 值，但这也是针对编码后的文档的。本章后面会说明 Content-MD5 首部。

（1）如果特定的 HTTP 报文类型中不允许带有主体，就忽略 Content-Length 首部，它是对（没有实际发送出来的）主体进行计算的。这种情况下，Content-Length 首部是提示性的，并不说明实际的主体长度。（考虑不周的 HTTP 应用程序会认为有了 Content-Length 就有主体存在，这样就会出问题。）

最重要的例子就是 HEAD 响应。HEAD 方法请求服务器发送等价的 GET 请求中会出现的首部，但不要包括主体。因为对 GET 的响应会带有 Content-Length 首部，所以 HEAD 响应里面也有；但和 GET 响应不同的是，HEAD 响应中不会有主体。1XX、204 以及 304 响应也可以有提示性的 Content-Length 首部，但是也都没有实体主体。那些规定不能带有实体主体的报文，不管带有什么首部字段，都必须在首部之后的第一个空行终止。

（2）如果报文中含有描述传输编码的 Transfer-Encoding 首部（不采用默认的 HTTP“恒等”编码），那实体就应由一个称为“零字节块”（zero-byte chunk）的特殊模式结束，除非报文已经因连接关闭而结束。我们将在本章后面讨论传输编码和分块编码。

（3）如果报文中含有 Content-Length 首部（并且报文类型允许有实体主体），而且没有非恒等的 Transfer-Encoding 首部字段，那么 Content-Length 的值就是主体的长度。如果收到的报文中既有 Content-Length 首部字段又有非恒等的 Transfer-Encoding 首部字段，那就必须忽略 Content-Length，因为传输编码会改变实体主体的表示和传输方式（因此可能就会改变传输的字节数）。

（4）如果报文使用了 multipart/byteranges（多部分 / 字节范围）媒体类型，并且没有用 Content-Length 首部指出实体主体的长度，那么多部分报文中的每个部分都要说明它自己的大小。这种多部分类型是唯一的一种自定界的实体主体类型，因此除非发送方知道接收方可以解析它，否则就不能发送这种媒体类型。[3] 346

（5）如果上面的规则都不匹配，实体就在连接关闭的时候结束。实际上，只有服务器可以使用连接关闭来指示报文的结束。客户端不能用关闭连接来指示客户端报文的结束，因为这样会使服务器无法发回响应。[4]

为了和使用 HTTP/1.0 的应用程序兼容，任何带有实体主体的 HTTP/1.1 请求都必须带有正确的 Content-Length 首部字段（除非已经知道服务器兼容 HTTP/1.1）。

注 3：因为 Range 首部可能会被不理解多部分 / 字节范围的更原始的代理所转发，所以如果发送方不能确定接收方是否理解这种自定界的格式的话，就必须用本节的方法（1）、（3）或（5）来对报文定界。

注 4：客户端可以使用半关闭，也就是只把连接的输出端关闭，但很多服务器应用程序设计的时候没有考虑到处理这种情况，会把半关闭当作客户端要从服务器断开连接来处理。HTTP 没有对连接管理进行良好的规范。详情请参见第 4 章。

HTTP/1.1 规范中建议对于带有主体但没有 Content-Length 首部的请求，服务器如果无法确定报文的长度，就应当发送 400 Bad Request 响应或 411 Length Required 响应，后一种情况表明服务器要求收到正确的 Content-Length 首部。

15.3 实体摘要

尽管 HTTP 通常都是在像 TCP/IP 这样的可靠传输协议之上实现的，但仍有很多因素会导致报文的一部分在传输过程中被修改，比如有不兼容的转码代理，或者中间代理有误，等等。为检测实体主体的数据是否被不经意（或不希望有）地修改，发送方可以在生成初始的主体时，生成一个数据的校验和，这样接收方就可以通过检查这个校验和来捕获所有意外的实体修改了。[5]

服务器使用 Content-MD5 首部发送对实体主体运行 MD5 算法的结果。只有产生响应的原始服务器可以计算并发送 Content-MD5 首部。中间代理和缓存不应当修改或添加这个首部，否则就会与验证端到端完整性的这个最终目的相冲突。Content-MD5 首部是在对内容做了所有需要的内容编码之后，还没有做任何传输编码之前，计算出来的。为了验证报文的完整性，客户端必须先进行传输编码的解码，然后计算所得到的未进行传输编码的实体主体的 MD5。举个例子吧，如果一份文档使用 gzip 算法进行压缩，然后用分块编码发送，那么就对整个经 gzip 压缩的主体进行 MD5 计算。

347

除了检查报文的完整性之外，MD5 还可以当作散列表的关键字，用来快速定位文档并消除不必要的重复内容存储。除了这些可能的用法，一般不常用到 Content-MD5 首部。

作为对 HTTP 的扩展，在 IETF 的草案中提出了其他一些摘要算法。这些扩展建议增加新的 Want-Digest 首部，它允许客户端说明期望响应中使用的摘要类型，并使用质量值来建议多种摘要算法并说明优先顺序。

15.4 媒体类型和字符集

Content-Type 首部字段说明了实体主体的 MIME 类型。[6]MIME 类型是标准化的名字，用以说明作为货物运载实体的基本媒体类型（比如：HTML 文件、Microsoft Word 文档或是 MPEG 视频等）。客户端应用程序使用 MIME 类型来解释和处理其内容。

注 5：当然，这种方法对同时替换报文主体和摘要首部的恶意攻击无效。这只是为了检测不经意的修改。对付恶意篡改，需要使用别的机制，比如摘要认证。

注 6：在 HEAD 请求中，Content-Type 说明如果请求是 GET 时，将要发送的主体的类型。

Content-Type 的值是标准化的 MIME 类型，都在互联网号码分配机构（Internet Assigned Numbers Authority，简称 IANA）中注册。MIME 类型由一个主媒体类型（比如：text、image 或 audio 等）后面跟一条斜线以及一个子类型组成，子类型用于进一步描述媒体类型。表 15-1 中列出了一些 Content-Type 首部中常用的 MIME 类型。附录 D 中列出了更多的 MIME 类型。

表15-1　常用媒体类型

媒体类型	描　　述
text/html	实体主体是 HTML 文档
text/plain	实体主体是纯文本文档
image/gif	实体主体是 GIF 格式的图像
image/jpeg	实体主体是 JPEG 格式的图像
audio/x-wav	实体主体包含 WAV 格式声音数据
model/vrml	实体主体是三维的 VRML 模型
application/vnd.ms-powerpoint	实体主体是 Microsoft PowerPoint 演示文档
multipart/byteranges	实体主体有若干部分，每个部分都包含了完整文档中不同的字节范围
message/http	实体主体包含完整的 HTTP 报文（参见 TRACE）

要着重注意的是，Content-Type 首部说明的是原始实体主体的媒体类型。例如，如果实体经过内容编码的话，Content-Type 首部说明的仍是编码之前的实体主体的类型。

348

15.4.1　文本的字符编码

Content-Type 首部还支持可选的参数来进一步说明内容的类型。charset（字符集）参数就是个例子，它说明把实体中的比特转换为文本文件中的字符的方法：

```
Content-Type: text/html; charset=iso-8859-4
```

我们将在第 16 章详细讨论字符集。

15.4.2　多部分媒体类型

MIME 中的 multipart（多部分）电子邮件报文中包含多个报文，它们合在一起作为单一的复杂报文发送。每一部分都是独立的，有各自的描述其内容的集；不同的部分之间用分界字符串连接在一起。

HTTP 也支持多部分主体。不过，通常只用在下列两种情形之一：提交填写好的表格，或是作为承载若干文档片段的范围响应。

15.4.3 多部分表格提交

当提交填写的 HTTP 表格时，变长的文本字段和上传的对象都作为多部分主体里面独立的部分发送，这样表格中就可以填写各种不同类型和长度的值。比如，你可能选择用昵称和小照片来填写询问你的名字和介绍信息的表格，而你的朋友可能填了她的全名并在介绍信息表内抱怨了一堆大众汽车的修理问题。

HTTP 使用 `Content-Type:multipart/form-data` 或 `Content-Type:multipart/mixed` 这样的首部以及多部分主体来发送这种请求，举例如下：

```
Content-Type: multipart/form-data; boundary=[abcdefghijklmnopqrstuvwxyz]
```

其中的 `boundary` 参数说明了分割主体中不同部分所用的字符串。

349 下面的例子展示了 `multipart/form-data` 编码。假设我们有这样的表格：

```
<FORM action=" http://server.com/cgi/handle"
      enctype="multipart/form-data"
      method="post">
<P>
What is your name? <INPUT type="text" name="submit-name"><BR>
What files are you sending? <INPUT type="file" name="files"><BR>
<INPUT type="submit" value="Send"> <INPUT type="reset">
</FORM>
```

如果用户在文本输入字段中键入 Sally，并选择了文本文件 essayfile.txt，用户 Agent 代理可能会发回下面这样的数据：

```
Content-Type: multipart/form-data; boundary=AaB03x
--AaB03x
Content-Disposition: form-data; name="submit-name"
Sally
--AaB03x
Content-Disposition: form-data; name="files"; filename="essayfile.txt"
Content-Type: text/plain
...contents of essayfile.txt...
--AaB03x--
```

如果用户还选了另一个（图像）文件 imagefile.gif，用户 Agent 代理可能像下面这样构造这个部分：

```
Content-Type: multipart/form-data; boundary=AaB03x
--AaB03x
Content-Disposition: form-data; name="submit-name"
Sally
--AaB03x
Content-Disposition: form-data; name="files"
Content-Type: multipart/mixed; boundary=BbC04y
```

```
--BbC04y
Content-Disposition: file; filename="essayfile.txt"
Content-Type: text/plain
...contents of essayfile.txt...
--BbC04y
Content-Disposition: file; filename="imagefile.gif"
Content-Type: image/gif
Content-Transfer-Encoding: binary
...contents of imagefile.gif...
--BbC04y--
--AaB03x--
```

15.4.4 多部分范围响应

HTTP 对范围请求的响应也可以是多部分的。这样的响应中有 Content-Type: multipart/byteranges 首部和带有不同范围的多部分主体。下面是一个例子，展示了对文档不同范围的请求产生的响应：

```
HTTP/1.0 206 Partial content
Server: Microsoft-IIS/5.0
Date: Sun, 10 Dec 2000 19:11:20 GMT
Content-Location: http://www.joes-hardware.com/gettysburg.txt
Content-Type: multipart/x-byteranges; boundary=--[abcdefghijklmnopqrstu
vwxyz]--
Last-Modified: Sat, 09 Dec 2000 00:38:47 GMT

--[abcdefghijklmnopqrstuvwxyz]--
Content-Type: text/plain
Content-Range: bytes 0-174/1441
Fourscore and seven years ago our fathers brought forth on this
continent a new nation, conceived in liberty and dedicated to the
proposition that all men are created equal.
--[abcdefghijklmnopqrstuvwxyz]--
Content-Type: text/plain
Content-Range: bytes 552-761/1441

But in a larger sense, we can not dedicate, we can not consecrate,
we can not hallow this ground. The brave men, living and dead who
struggled here have consecrated it far above our poor power to add
or detract.
--[abcdefghijklmnopqrstuvwxyz]--
Content-Type: text/plain
Content-Range: bytes 1344-1441/1441

and that government of the people, by the people, for the people shall
not perish from the earth.

--[abcdefghijklmnopqrstuvwxyz]--
```

350

本章后面将详细讨论范围请求。

15.5 内容编码

HTTP 应用程序有时在发送之前需要对内容进行编码。例如，在把很大的 HTML 文档发送给通过慢速连接连上来的客户端之前，服务器可能会对它进行压缩，这样有助于减少传输实体的时间。服务器还可以把内容搅乱或加密，以此来防止未经授权的第三方看到文档的内容。

这种类型的编码是在发送方应用到内容之上的。当内容经过内容编码之后，编好码的数据就放在实体主体中，像往常一样发送给接收方。

15.5.1 内容编码过程

内容编码的过程如下所述。

（1）网站服务器生成原始响应报文，其中有原始的 `Content-Type` 和 `Content-Length` 首部。

（2）内容编码服务器（也可能就是原始的服务器或下行的代理）创建编码后的报文。编码后的报文有同样的 `Content-Type` 但 `Content-Length` 可能不同（比如主体被压缩了）。内容编码服务器在编码后的报文中增加 `Content-Encoding` 首部，这样接收的应用程序就可以进行解码了。

（3）接收程序得到编码后的报文，进行解码，获得原始报文。

图 15-3 给出了内容编码的梗概示例。

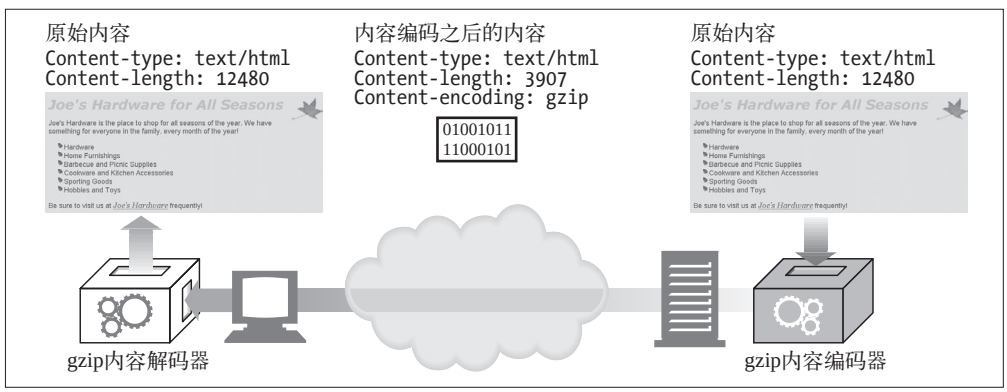


图 15-3 内容编码示例

在这个例子中，通过 gzip 内容编码函数对 HTML 页面处理之后，得到一个更小的、压缩的主体。经过网络发送的是压缩的主体，并打上了 gzip 压缩的标志。接收的客

户端使用 gzip 解码器对实体进行解压缩。

下面给出的响应片段是另一个编码响应的例子（一个压缩的图像）：

```
HTTP/1.1 200 OK
Date: Fri, 05 Nov 1999 22:35:15 GMT
Server: Apache/1.2.4
Content-Length: 6096
Content-Type: image/gif
Content-Encoding: gzip
[...]
```

注意，Content-Type 首部可以且还应当出现在报文中。它说明了实体的原始格式，一旦实体被解码，要显示的时候，可能还是需要该信息才行的。记住，Content-Length 首部现在代表的是*编码之后*的主体长度。

15.5.2 内容编码类型

HTTP 定义了一些标准的内容编码类型，并允许用扩展编码的形式增添更多的编码。由互联网号码分配机构（IANA）对各种编码进行标准化，它给每个内容编码算法分配了唯一的代号。Content-Encoding 首部就用这些标准化的代号来说明编码时使用的算法。

表 15-2 列出了一些常用的内容编码代号。 352

表15-2 内容编码代号

Content-Encoding值	描　述
gzip	表明实体采用 GNU zip 编码 [a]
compress	表明实体采用 Unix 的文件压缩程序
deflate	表明实体是用 zlib 的格式压缩的 [b]
identity	表明没有对实体进行编码。当没有 Content-Encoding 首部时，就默认为这种情况

a：RFC 1952 中说明了 gzip 编码。

b：RFC 1950 和 1951 中讲解了 zlib 格式和 deflate 压缩算法。

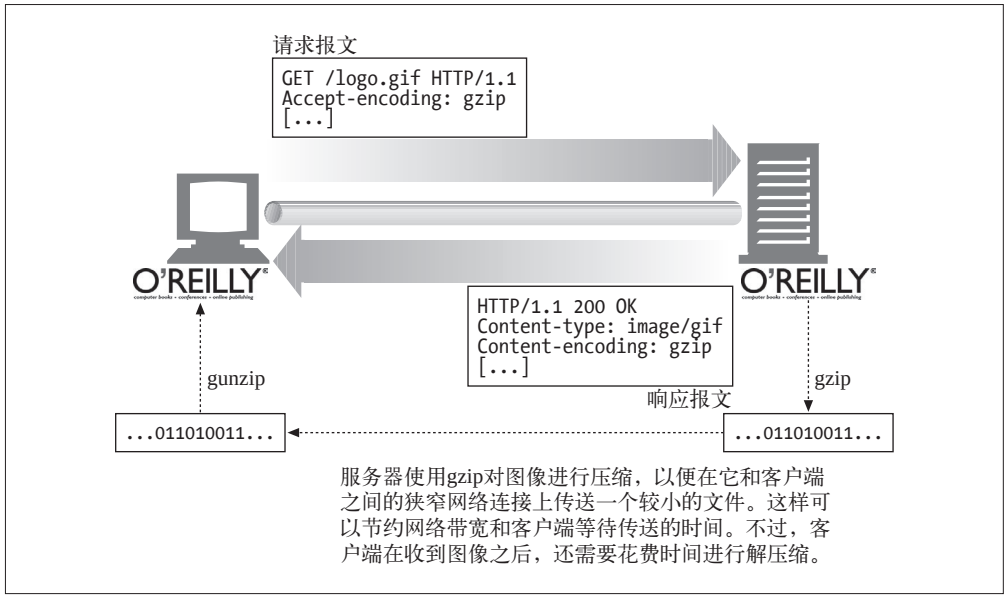
gzip、compress 以及 deflate 编码都是无损压缩算法，用于减少传输报文的大小，不会导致信息损失。这些算法中，gzip 通常是效率最高的，使用最为广泛。

15.5.3 Accept-Encoding首部

毫无疑问，我们不希望服务器用客户端无法解码的方式来对内容进行编码。为了避免服务器使用客户端不支持的编码方式，客户端就把自己支持的内容编码方式

列表放在请求的 Accept-Encoding 首部里发出去。如果 HTTP 请求中没有包含 Accept-Encoding 首部，服务器就可以假设客户端能够接受任何编码方式（等价于发送 Accept-Encoding: *）。

图 15-4 展示了 HTTP 事务中的 Accept-Encoding 首部。



353

图 15-4 展示了 HTTP 事务中的 Accept-Encoding 首部

Accept-Encoding 字段包含用逗号分隔的支持编码的列表，下面是一些例子：

```
Accept-Encoding: compress, gzip
Accept-Encoding:
Accept-Encoding: *
Accept-Encoding: compress;q=0.5, gzip;q=1.0
Accept-Encoding: gzip;q=1.0, identity; q=0.5, *;q=0
```

客户端可以给每种编码附带 Q（质量）值参数来说明编码的优先级。Q 值的范围从 0.0 到 1.0，0.0 说明客户端不想接受所说明的编码，1.0 则表明最希望使用的编码。“*”表示“任何其他方法”。决定在响应中回送什么内容给客户端是个更通用的过程，而选择使用何种内容编码则是此过程的一部分。第 17 章将详细讨论这个过程，以及 Content-Encoding 首部和 Accept-Encoding 首部。

identity 编码代号只能在 Accept-Encoding 首部中出现，客户端用它来说明相对于其他内容编码算法的优先级。

15.6 传输编码和分块编码

前一节讨论的内容编码，是对报文的主体进行的可逆变换。内容编码是和内容的具体格式细节紧密相关的。例如，你可能会用 gzip 压缩文本文件，但不是 JPEG 文件，因为 JPEG 这类东西用 gzip 压缩的不够好。

本节讨论传输编码。传输编码也是作用在实体主体上的可逆变换，但使用它们是由于架构方面的原因，同内容的格式无关。如图 15-5 所示，使用传输编码是为了改变报文中的数据在网络上传输的方式。

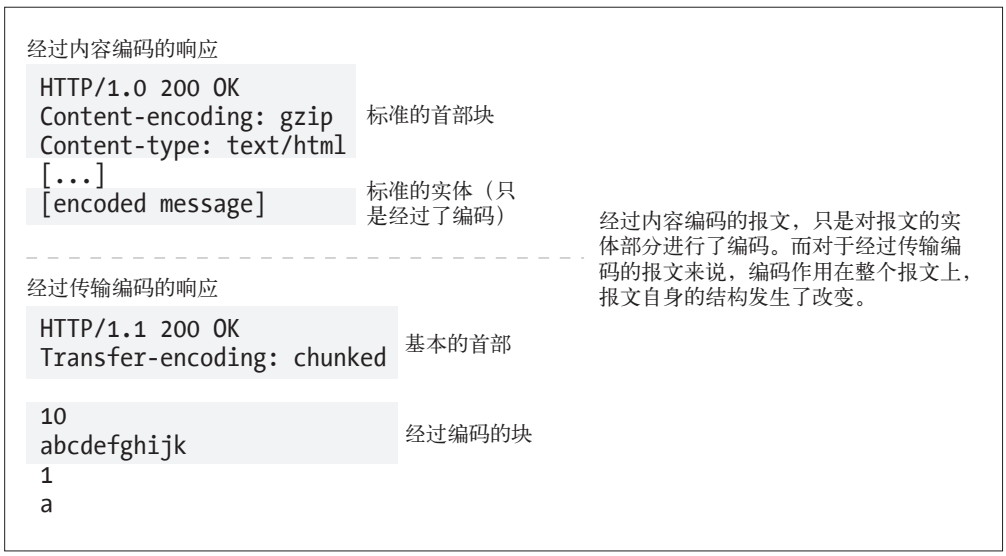


图 15-5　内容编码和传输编码的对比

15.6.1 可靠传输

长久以来，在其他一些协议中会用传输编码来保证报文经过网络时能得到“可靠传输”。在 HTTP 协议中，可靠传输关注的焦点有所不同，因为底层的传输设施已经标准化并且容错性更好。在 HTTP 中，只有少数一些情况下，所传输的报文主体可能会引发问题。其中两种情况如下所述。

- 未知的尺寸

  如果不先生成内容，某些网关应用程序和内容编码器就无法确定报文主体的最终大小。通常，这些服务器希望在知道大小之前就开始传输数据。因为 HTTP 协议 354
  要求 `Content-Length` 首部必须在数据之前，有些服务器就使用传输编码来发

送数据，并用特别的结束脚注表明数据结束。[7]

- 安全性
  你可以用传输编码来把报文内容扰乱，然后在共享的传输网络上发送。不过，由于像 SSL 这样的传输层安全体系的流行，就很少需要靠传输编码来实现安全性了。

15.6.2 Transfer-Encoding首部

HTTP 协议中只定义了下面两个首部来描述和控制传输编码。

- Transfer-Encoding
  告知接收方为了可靠地传输报文，已经对其进行了何种编码。

- TE
  用在请求首部中，告知服务器可以使用哪些传输编码扩展。[8]

355

下面的例子中，请求使用了 TE 首部来告诉服务器它可以接受分块编码（如果是 HTTP/1.1 应用程序的话，这就是必须的）并且愿意接受附在分块编码的报文结尾上的拖挂：

```
GET /new_products.html HTTP/1.1
Host: www.joes-hardware.com
User-Agent: Mozilla/4.61 [en] (WinNT; I)
TE: trailers, chunked
...
```

对它的响应中包含 Transfer-Encoding 首部，用于告诉接收方已经用分块编码对报文进行了传输编码：

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Server: Apache/3.0
...
```

在这个起始首部之后，报文的结构就将发生改变。

传输编码的值都是大小写无关的。HTTP/1.1 规定在 TE 首部和 Transfer-Encoding 首部中使用传输编码值。最新的 HTTP 规范只定义了一种传输编码，就是分块编码。

与 Accept-Encoding 首部类似，TE 首部也可以使用 Q 值来说明传输编码的优先顺序。不过，HTTP/1.1 规范中禁止将分块编码关联的 Q 值设为 0.0。

注 7：尽管可以因陋就简地用关闭连接作为报文结束的信号，但这种方法不能用于持久连接。

注 8：如果这个首部起名叫 Accept-Transfer-Encoding，它的意义就会更直白。

HTTP 将来的扩展可能会推动对更多传输编码的需求。如果真的如此，那分块编码仍应始终作用在其他传输编码之上，这样就保证数据可以像隧道那样“穿透”那些只理解分块编码但不理解其他传输编码的 HTTP/1.1 应用程序。

15.6.3 分块编码

分块编码把报文分割为若干个大小已知的块。块之间是紧挨着发送的，这样就不需要在发送之前知道整个报文的大小了。

要注意的是，分块编码是一种传输编码，因此是报文的属性，而不是主体的属性。本章前面部分讨论过的多部分编码，就是主体的属性，它和分块编码是完全独立的。

1. 分块与持久连接

若客户端和服务器之间不是持久连接，客户端就不需要知道它正在读取的主体的长度，而只需要读到服务器关闭主体连接为止。

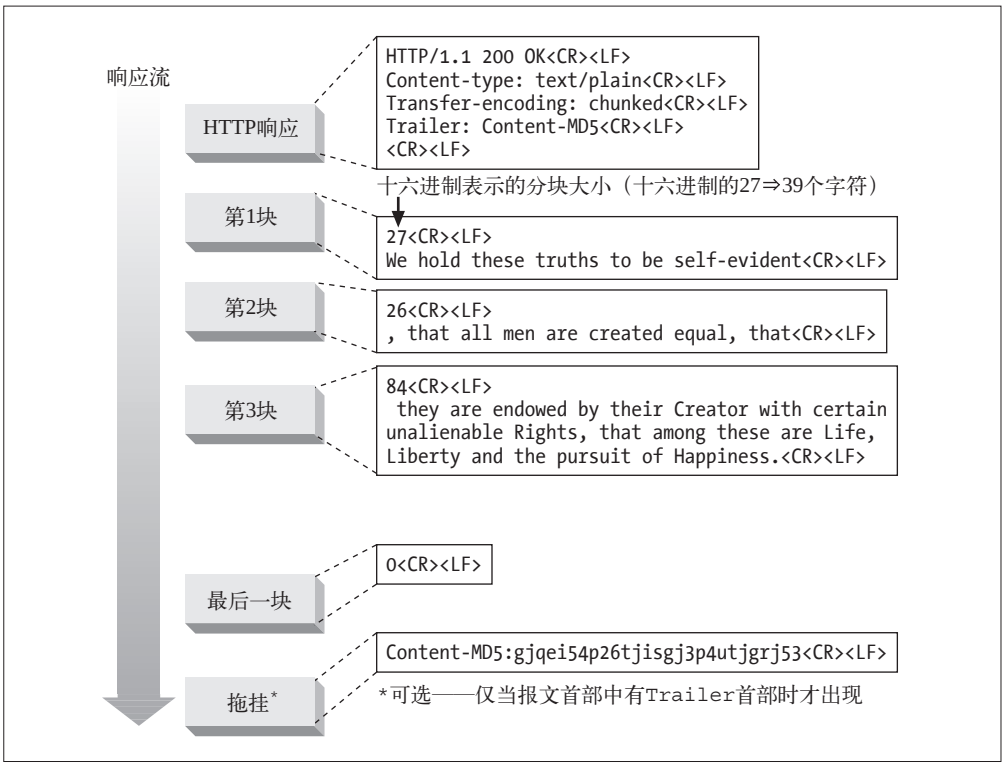
356

当使用持久连接时，在服务器写主体之前，必须知道它的大小并在 `Content-Length` 首部中发送。如果服务器动态创建内容，就可能在发送之前无法知道主体的长度。

分块编码为这种困难提供了解决方案，只要允许服务器把主体逐块发送，说明每块的大小就可以了。因为主体是动态创建的，服务器可以缓冲它的一部分，发送其大小和相应的块，然后在主体发送完之前重复这个过程。服务器可以用大小为 0 的块作为主体结束的信号，这样就可以继续保持连接，为下一个响应做准备。

分块编码是相当简单的。图 15-6 展示了一个分块编码报文的基本结构。它由起始的 HTTP 响应首部块开始，随后就是一系列分块。每个分块包含一个长度值和该分块的数据。长度值是十六进制形式并将 CRLF 与数据分隔开。分块中数据的大小以字节计算，不包括长度值与数据之间的 CRLF 序列以及分块结尾的 CRLF 序列。最后一个块有点特别，它的长度值为 0，表示“主体结束”。

客户端也可以发送分块的数据给服务器。因为客户端事先不知道服务器是否接受分块编码（这是因为服务器不会在给客户端的响应中发送 `TE` 首部），所以客户端必须做好服务器用 411 Length Required（需要 `Content-Length` 首部）响应来拒绝分块请求的准备。



357 图 15-6 剖析分块编码报文

2. 分块报文的拖挂

如果客户端的 TE 首部中说明它可以接受拖挂的话，就可以在分块的报文最后加上拖挂。产生原始响应的服务器也可以在分块的报文最后加上拖挂。拖挂的内容是可选的元数据，客户端不一定需要理解和使用（客户端可以忽略并丢弃拖挂中的内容）。[9]

拖挂中可以包含附带的首部字段，它们的值在报文开始的时候可能是无法确定的（例如，必须要先生成主体的内容）。Content-MD5 首部就是一个可以在拖挂中发送的首部，因为在文档生成之前，很难算出它的 MD5。图 15-6 中展示了拖挂的使用方式。报文首部中包含一个 Trailer 首部，列出了跟在分块报文之后的首部列表。在 Trailer 首部中列出的首部就紧接在最后一个分块之后。

除了 Transfer-Encoding、Trailer 以及 Content-Length 首部之外，其他 HTTP 首部都可以作为拖挂发送。

注 9：trailer（拖挂）首部是在最初的分块编码被加入到 HTTP/1.1 规范的草案之后才加入的，因此有些应用程序可能不理解这个首部（或者不理解拖挂），尽管它们声称是兼容 HTTP/1.1 的。

15.6.4 内容编码与传输编码的结合

内容编码与传输编码可以同时使用。例如，图 15-7 中展示了发送方如何用内容编码压缩 HTML 文件，再使用传输编码分块发送。接收方“重构”主体的过程和发送方相反。

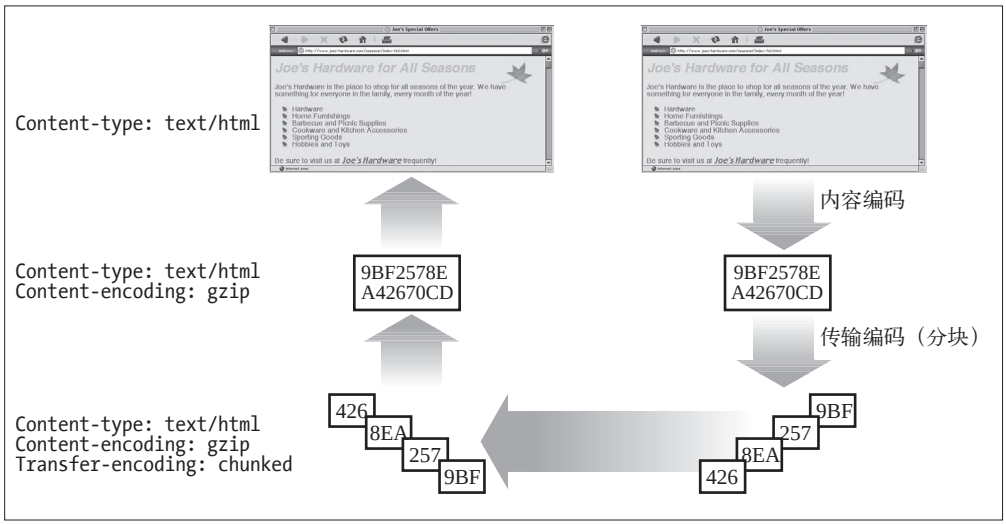


图 15-7 内容编码和传输编码结合

15.6.5 传输编码的规则

对报文主体使用传输编码时，必须遵守以下规则。

- 传输编码集合中必须包括“分块”。唯一的例外是使用关闭连接来结束报文。
- 当使用分块传输编码时，它必须是最后一个作用到报文主体之上的。
- 分块传输编码不能多次作用到一个报文主体上。

358

这些规则使得接收方能够确定报文的传输长度。

传输编码是 HTTP 1.1 版中引入的一个相对较新的特性。实现传输编码的服务器必须特别注意不要把经传输编码后的报文发送给非 HTTP/1.1 的应用程序。同样地，如果服务器收到无法理解的经过传输编码的报文，它应当用 501 Unimplemented 状态码来回复。不过，所有的 HTTP/1.1 应用程序至少都必须支持分块编码。

15.7 随时间变化的实例

网站对象并不是静态的。同样的 URL 会随着时间变化而指向对象的不同版本。以

CNN 的主页为例，同一天里多次访问 http://www.cnn.com，可能每次得到的返回页面都会略有不同。

可以把 CNN 的主页当作一个对象来考虑，其不同版本就可以看作这个对象的不同*实例*（参见图 15-8）。在图中，客户端多次请求同一个资源（URL），但得到的是该资源的不同实例，因为它是随时间而变化的。在时间（a）和时间（b）具有相同的实例，而在时间（c）则是不同的实例。

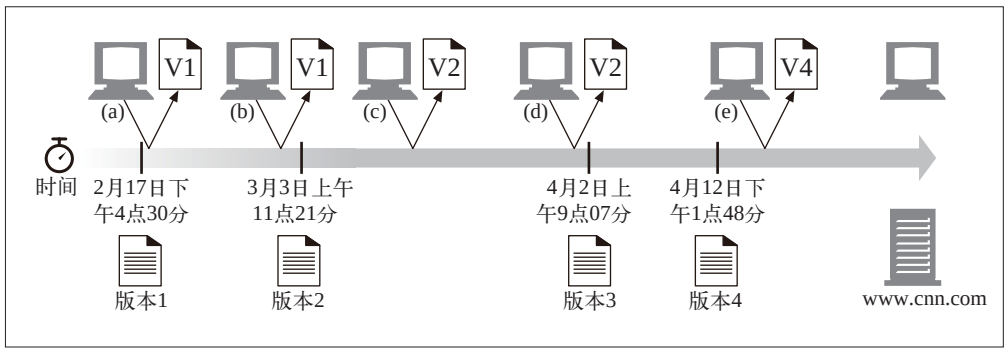


图 15-8　实例是资源在某个时间的“快照”

HTTP 协议规定了称为*实例操控*（instance manipulations）的一系列请求和响应操作，用以操控对象的实例。两个主要的实例操控方法是范围请求和差异编码。这两种方法都要求客户端能够标识它所拥有（如果有的话）的资源的特定副本，并在一
359 定的条件下请求新的实例。本章后面将讨论这些机制。

15.8　验证码和新鲜度

现在再回顾前面的图 15-8。客户端起初没有该资源的副本，因此它发送请求给服务器要求得到一份。服务器用该资源的版本 1 给以响应。客户端现在可以缓存这份副本，但是要缓存多长时间呢？

当文档在客户端“过期”之后（也就是说，客户端不再认为该副本有效），客户端必须从服务器请求一份新的副本。不过，如果该文档在服务器上并未发生改变，客户端也就不需要再接收一次了——继续使用缓存的副本即可。

这种特殊的请求，称为*有条件的请求*（conditional request），要求客户端使用*验证码*（validator）来告知服务器它当前拥有的版本号，并仅当它的当前副本不再有效时才要求发送新的副本。让我们进一步详细研究这 3 个关键概念——新鲜度、验证码以及有条件的请求。

15.8.1 新鲜度

服务器应当告知客户端能够将内容缓存多长时间，在这个时间之内就是新鲜的。服务器可以用这两个首部之一来提供这种信息：Expires（过期）和Cache-Control（缓存控制）。

Expires首部规定文档“过期”的具体时间——此后就不应当认为它还是最新的。Expires首部的语法如下：

```
Expires: Sun Mar 18 23:59:59 GMT 2001
```

客户端和服务器为了能正确使用Expires首部，它们的时钟必须同步。这并不总是很容易的，因为它们可能都没有运行像Network Time Protocol（网络时间协议，NTP）这样的时钟同步协议。用相对时间来定义过期的机制会更有用。Cache-Control首部可以用秒数来规定文档最长使用期——从文档离开服务器之后算起的总计时间。使用期不与时钟同步，因此可以给出更精确的结果。

360

实际上，Cache-Control首部功能很强大。服务器和客户端都可以用它来说明新鲜度，并且除了使用期或过期时间之外，还有很多指令可用。表15-3列出了Cache-Control首部的一些指令。

表15-3 Cache-Control首部的指令

指　令	报文类型	描　述
no-cache	请求	在重新向服务器验证之前，不要返回文档的缓存副本
no-store	请求	不要返回文档的缓存副本。不要保存服务器的响应
max-age	请求	缓存中的文档不能超过指定的使用期
max-stale	请求	文档允许过期（根据服务器提供的过期信息计算），但不能超过指令中指定的过期值
min-fresh	请求	文档的使用期不能小于这个指定的时间与它的当前存活时间之和。换句话说，响应必须至少在指定的这段时间之内保持新鲜
no-transform	请求	文档在发送之前不允许被转换
only-if-cached	请求	只有当文档在缓存中才发送，不要联系原始服务器
public	响应	响应可以被任何服务器缓存
private	响应	响应可以被缓存，但只能被单个客户端访问
no-cache	响应	如果该指令伴随一个首部列表的话，那么内容可以被缓存并提供给客户端，但必须先删除所列出的首部。如果没有指定首部，缓存中的副本在没有重新向服务器验证之前不能提供给客户端
no-store	响应	响应不允许被缓存
no-transform	响应	响应在提供给客户端之前不能做任何形式的修改

（续）

指　令	报文类型	描　述
must-revalidate	响应	响应在提供给客户端之前必须重新向服务器验证
proxy-revalidate	响应	共享的缓存在提供给客户端之前必须重新向原始服务器验证。私有的缓存可以忽略这条指令
max-age	响应	指定文档可以被缓存的时间以及新鲜度的最长时间
s-max-age	响应	指定文档作为共享缓存时的最长使用时间（如果有 max-age 指令的话，以本指令为准）。私有的缓存可以忽略本指令

缓存和新鲜度在第 7 章中曾有详细讨论。

15.8.2　有条件的请求与验证码

当请求缓存服务器中的副本时，如果它不再新鲜，缓存服务器就需要保证它有一个新鲜的副本。缓存服务器可以向原始服务器获取当前的副本。但在很多情况下，原始服务器上的文档仍然与缓存中已过期的副本相同。我们在图 15-8b 中看到过这种
361 情况；缓存的副本或许已经过期了，但原始服务器上的内容与缓存的内容仍然相同。如果服务器上的文档和已过期的缓存副本相同，而缓存服务器还是要从原始服务器上取文档的话，那缓存服务器就是在浪费网络带宽，给缓存服务器和原始服务器增加不必要的负载，使所有事情都变慢了。

为了避免这种情况，HTTP 为客户端提供了一种方法，仅当*资源改变*时才请求副本，这种特殊请求称为有条件的请求。有条件的请求是标准的 HTTP 请求报文，但仅当某个特定条件为真时才执行。例如，某个缓存服务器可能发送下面的有条件 GET 报文给服务器，仅当文件 /announce.html 从 2002 年 6 月 29 日（这是缓存的文档最后被作者修改的时间）之后发生改变的情况下才发送它：

```
GET /announce.html HTTP/1.0
If-Modified-Since: Sat, 29 Jun 2002, 14:30:00 GMT
```

有条件的请求是通过以“If-”开头的有条件的首部来实现的。在上面的例子中，有条件的首部是 If-Modified-Since（如果 – 从……之后 – 修改过）。有条件的首部使得方法仅在条件为真时才执行。如果条件不满足，服务器就发回一个 HTTP 错误码。

每个有条件的请求都通过特定的*验证码*来发挥作用。验证码是文档实例的一个特殊属性，用它来测试条件是否为真。从概念上说，你可以把验证码看作文件的序列号、版本号，或者最后发生改变的日期时间。在图 15-8b 中，那个智能的客户端发送给服务器的有条件的验证请求是在说：“我有版本 1，如果这个资源不再是版本 1 就把它发给我。”我们在第 7 章已经讨论过有条件的缓存再验证了，而本章会更仔细地研

究实体验证码的细节。

有条件的首部 If-Modified-Since 测试的是文档实例最后被修改的日期时间，因此我们说最后被修改的日期时间就是验证码。有条件的首部 If-None-Match 测试的是文档的 ETag 值，它是与实体相关联的一个特殊的关键字，或者说是版本识别标记。Last-Modified 和 ETag 是 HTTP 使用的两种主要验证码。表 15-4 中列出了用于有条件请求的 4 种 HTTP 首部。每个有条件的首部之后就是这种首部所用的验证码类型。

表15-4　有条件的请求类型

请求类型	验　证　码	描　　述
If-Modified-Since	Last-Modified	如果在前一条响应的 Last-Modified 首部中说明的时间之后，资源的版本发生变化，就发送其副本
If-Unmodified-Since	Last-Modified	仅在前一条响应的 Last-Modified 首部中说明的时间之后，资源的版本没有变化，才发送其副本
If-Match	ETag	如果实体的标记与前一次响应首部中的 ETag 相同，就发送该资源的副本
If-None-Match	ETag	如果实体的标记与前一次响应首部中的 ETag 不同，就发送该资源的副本

362

HTTP 把验证码分为两类：*弱验证码*（weak validators）和*强验证码*（strong validators）。弱验证码不一定能唯一标识资源的一个实例，而强验证码必须如此。弱验证码的一个例子是对象的大小字节数。有可能资源的内容改变了，而大小还保持不变，因此假想的字节计数验证码与改变是弱相关的。而资源内容的加密校验和（比如 MD5）就是强验证码，当文档改变时它总是会改变。

最后修改时间被当作弱验证码，因为尽管它说明了资源最后被修改的时间，但它的描述精度最大就是 1 秒。因为资源在 1 秒内可以改变很多次，而且服务器每秒可以处理数千个请求，最后修改日期时间并不总能反应变化情况。ETag 首部被当作强验证码，因为每当资源内容改变时，服务器都可以在 ETag 首部放置不同的值。版本号和摘要校验和也是很好的 ETag 首部候选，但它们不能带有任意的文本。ETag 首部很灵活，它可以带上任意的文本值（以标记的形式），这样就可以用来设计出各种各样的客户端和服务器验证策略。

有时候，客户端和服务器可能需要采用不那么精确的实体标记验证方法。例如，某服务器可能想对一个很大、被广泛缓存的文档进行一些美化修饰，但不想在缓存服务器再验证时产生很大的传输流量。在这种情况下，该服务器可以在标记前面加上“W/”前缀来广播一个“弱”实体标记。对于弱实体标记来说，只有当关联的实体

在语义上发生了重大改变时，标记才会变化。而强实体标记则不管关联的实体发生了什么性质的变化，标记都一定会改变。

下面的例子展示了客户端如何用弱实体标记向服务器请求再验证。服务器仅当文档的内容从版本 4.0 算起发生了显著变化时，才返回主体：

```
GET /announce.html HTTP/1.1
If-None-Match: W/"v4.0"
```

概括一下，当客户端多次访问同一个资源时，首先需要判断它当前的副本是不是仍然新鲜。如果不再新鲜，它们就必须从服务器获取最新的版本。为了避免在资源没有改变的情况下收到一份相同的副本，客户端可以向服务器发送有条件的请求，说明能唯一标识客户端当前副本的验证码。只在资源和客户端的副本不同的情况下服务器才会发送其副本。更多关于缓存再验证的细节，请回顾 7.7 节。

15.9 范围请求

关于客户端如何要求服务器只在资源的客户端副本不再有效的情况下才发送其副本，我们已经清楚地理解了。HTTP 还进一步锦上添花：它允许客户端实际上只请求文
363 档的一部分，或者说某个范围。

假设你正通过慢速的调制解调器连接下载最新的热门软件，已经下了四分之三，忽然因为一个网络故障，连接中断了。你已经为等待下载完成耽误了很久，而现在被迫要全部重头再来，祈祷着别再发生这样的倒霉事了。

有了范围请求，HTTP 客户端可以通过请求曾获取失败的实体的一个范围（或者说一部分），来恢复下载该实体。当然这有一个前提，那就是从客户端上一次请求该实体到这次发出范围请求的时段内，该对象没有改变过。例如：

```
GET /bigfile.html HTTP/1.1
Host: www.joes-hardware.com
Range: bytes=4000-
User-Agent: Mozilla/4.61 [en] (WinNT; I)
...
```

在本例中，客户端请求的是文档开头 4000 字节之后的部分（不必给出结尾字节数，因为请求方可能不知道文档的大小）。在客户端收到了开头的 4000 字节之后就失败的情况下，可以使用这种形式的范围请求。还可以用 Range 首部来请求多个范围（这些范围可以按任意顺序给出，也可以相互重叠）。例如，假设客户端同时连接到多个服务器，为了加速下载文档而从不同的服务器下载同一个文档的不同部分。对于客户端在一个请求内请求多个不同范围的情况，返回的响应也是单个实体，它有

一个多部分主体及 Content-Type: multipart/byteranges 首部。

并不是所有服务器都接受范围请求，但很多服务器可以。服务器可以通过在响应中包含 Accept-Ranges 首部的形式向客户端说明可以接受的范围请求。这个首部的值是计算范围的单位，通常是以字节计算的。[10] 例如：

```
HTTP/1.1 200 OK
Date: Fri, 05 Nov 1999 22:35:15 GMT
Server: Apache/1.2.4
Accept-Ranges: bytes
...
```

图 15-9 展示了涉及范围请求的一系列 HTTP 事务的例子。

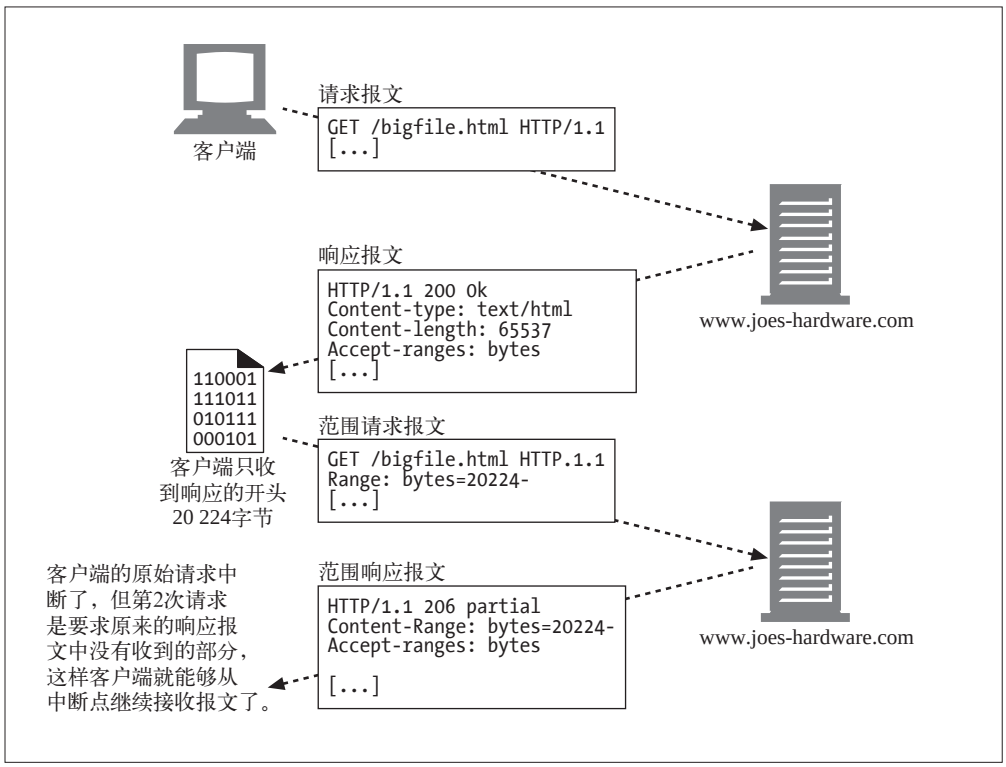


图 15-9　实体范围请求示例

Range 首部在流行的点对点（Peer-to-Peer，P2P）文件共享客户端软件中得到广泛应用，它们从不同的对等实体同时下载多媒体文件的不同部分。

注 10：HTTP/1.1 规范中只定义了 bytes 记号，但服务器和客户端的具体实现可以用它们自己认定的单位来衡量或切分实体。

注意，范围请求也属于一类实例操控，因为它们是在客户端和服务器之间针对特定的对象实例来交换信息的。也就是说，客户端的范围请求仅当客户端和服务器拥有
364 文档的同一个版本时才有意义。

15.10 差异编码

我们曾把网站页面的不同版本看作页面的不同实例。如果客户端有一个页面的已过期副本，就要请求页面的最新实例。如果服务器有该页面更新的实例，就要把它发给客户端，哪怕页面上只有一小部分发生了改变，也要把完整的新页面实例发给客户端。

若改变的地方比较少，与其发送完整的新页面给客户端，客户端更愿意服务器只发送页面发生改变的部分，这样就可以更快地得到最新的页面。差异编码是 HTTP 协议的一个扩展，它通过交换对象改变的部分而不是完整的对象来优化传输性能。差异编码也是一类实例操控，因为它依赖客户端和服务器之间针对特定的对象实例来交换信息。RFC 3229 描述了差异编码。

图 15-10 更清楚地展示了差异编码的结构，包括请求、生成、接收和装配文档的全过程。客户端必须告诉服务器它有页面的哪个版本，它愿意接受页面最新版的差异（delta），它懂得哪些将差异应用于现有版本的算法。服务器必须检查它是否有这个
365 页面的客户端现有版本，计算客户端现有版本与最新版之间的差异（有若干算法可以计算两个对象之间的差异）。然后服务器必须计算差异，发送给客户端，告知客户端所发送的是差异，并说明最新版页面的新标识（`ETag`），因为客户端将差异应用于其老版本之后就会得到这个版本。

客户端在 `If-None-Match` 首部中使用的是它所持有页面版本的唯一标识，这个标
366 识是服务器之前响应客户端时在 `ETag` 首部中发送的。客户端是在对服务器说：“如果你那里页面的最新版本标识和这个 `ETag` 不同，就把这个页面的最新版本发给我。”如果只有 `If-None-Match` 首部，服务器将会把该页面的最新版本完整地发给客户端。（假设最新版和客户端持有的版本不同。）

不过，如果客户端想告诉服务器它愿意接受该页面的差异，只要发送 `A-IM` 首部就可以了。`A-IM` 是 Accept-Instance-Manipulation（接受实例操控）的缩写。形象比喻的话，客户端相当于这样说：“哦对了，我能接受某些形式的实例操控，如果你会其中一种的话，就不用发送完整的文档给我了。”在 `A-IM` 首部中，客户端会说明它知道哪些算法可以把差异应用于老版本而得到最新版本。服务端发送回下面这些内容：一个特殊的响应代码——226 IM Used，告知客户端它正在发送的是所请求对象

的实例操控，而不是那个完整的对象自身；一个 IM（Instance-Manipulation 的缩写）首部，说明用于计算差异的算法；新的 ETag 首部和 Delta-Base 首部，说明用于计算差异的基线文档的 ETag（理论上，它应该和客户端之前请求里的 If-None-Match 首部中的 ETag 相同！）。表 15-5 中总结了差异编码使用的首部。

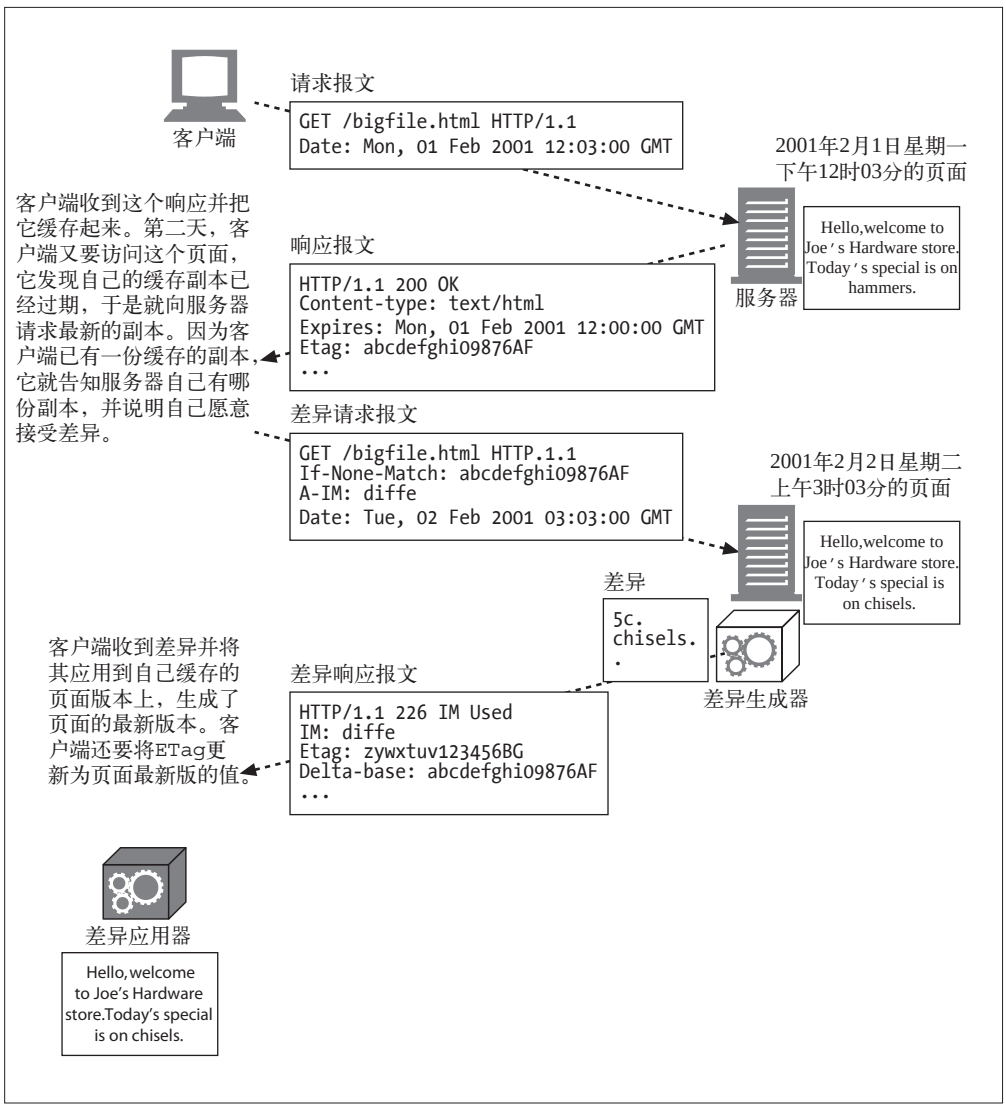


图 15-10　差异编码的结构

表15-5　差异编码所用的首部

首　部	描　述
ETag	文档每个实例的唯一标识符。由服务器在响应中发送；客户端在后继请求的 If-Match 首部和 If-None-Match 首部中可以使用它
If-None-Match	客户端发送的请求首部，当且仅当客户端的文档版本与服务器不同时，才向服务器请求该文档
A-IM	客户端请求首部，说明可以接受的实例操控类型
IM	服务器响应首部，说明作用在响应上的实例操控的类型。当响应代码是 226 IM Used 时，会发送这个首部
Delta-Base	服务器响应首部，说明用于计算差异的基线文档的 ETag 值（应当与客户端请求中的 If-None-Match 首部里的 ETag 相同）

实例操控、差异生成器和差异应用器

客户端可以使用 A-IM 首部说明可以接受的一些实例操控的类型。服务器在 IM 首部中说明使用的是何种实例操控。不过到底哪些实例操控类型是可接受的呢？它们又是做什么的呢？表 15-6 中列出了一些在 IANA 注册的实例操控类型。

表 15-6　在IANA注册的实例操控类型

类　型	说　明
vcdiff	用 vcdiff 算法计算差异[a]
diffe	用 Unix 系统的 diff-e 命令计算差异
gdiff	用 gdiff 算法计算差异[b]
gzip	用 gzip 算法压缩
deflate	用 deflate 算法压缩
range	用在服务器的响应中，说明响应是针对范围选择得到的部分内容
identity	用在客户端请求中的 A-IM 首部中，说明客户端愿意接受恒等实例操控

a：因特网草案 draft-korn-vcdiff-01 中描述了 vcdiff 算法。该规范在 2002 年初期由 IESG（Internet Engineering Steering Group，因特网工程指导组）批准，将很快以 RFC 的形式发布。（译注：vcdiff 的规范由 RFC3284 发布。）

b：http://www.w3.org/TR/NOTE-gdiff-19970901.html 描述了 gdiff 算法。

367

图 15-10 中，服务器侧的“差异生成器”根据基线文档和该文档的最新实例，用客户端在 A-IM 首部中指明的算法计算它们之间的差异。客户端侧的“差异应用器”得到差异，将其应用于基线文档，得到文档的最新实例。例如，如果产生差异的算法是 Unix 系统的 diff-e 命令，客户端就可以用 Unix 系统中的文本编辑器 ed 提供的功能来应用差异，因为 diff-e <file1> <file2> 产生了一系列 ed 命令来把 <file1>

转化为 <file2>。ed 是一个非常简单的编辑器，支持一些命令。在图 15-10 的例子中，5c 说明要删除基线文档的第 5 行，而 chisels.<cr>. 说明要添加 chisels.，就这么简单。对于更大的改动，会产生更复杂的指令。Unix 系统的 diff-e 算法是对文件进行逐行比较的，这对于文本文件没问题，但并不适合二进制文件。vcdiff 算法更强大，对于非文本文件也适用，并且产生的差异比 diff-e 要小。

差异编码的规范中详细定义了 A-IM 和 IM 首部的格式。在这里，我们只要知道这些首部中可以说明多个实例操控（并可以带有相关的质量值）就够了。在返回给客户端之前，文档可以经过多种实例操控，这样可以获得最大程度的压缩。例如，用 vcdiff 算法产生的差异随后可以再用 gzip 算法压缩。于是服务器的响应中就含有 IM:vcdiff, gzip 首部。客户端应当先对内容进行 gunzip，再把得到的差异应用到自己的基线页面上，这样才能生成最终的文档。

差异编码可以减少传输次数，但实现起来可能比较麻烦。设想一下页面改动频繁，而且有很多不同的人都在访问的情形。支持差异编码的服务器必须保存页面随时间变化的所有不同版本，这样才能指出最新版本与所请求的客户端持有的任意版本之间的差异。（如果文档变化频繁，而且有很多客户端都在请求文档，那它们就会获得 368
文档的不同实例。随后当它们再向服务器发起请求时，它们将请求它们所持有的版本与最新版本之间的差异。为了能够只向它们发送变化的部分，服务器必须保存所有客户端曾经持有过的版本。）要降低提交文档时的延迟时间，服务器必须增加磁盘空间来保存文档的各种旧的实例。实现差异编码所需的额外磁盘空间可能很快就会将减少传输量获得的好处抵消掉。

15.11 更多信息

关于实体和编码方面的更多信息，请参考以下资源。

- http://www.ietf.org/rfc/rfc2616.txt
  RFC 2616，也就是 HTTP/1.1 版的规范，是实体主体管理和编码方面的主要参考。

- http://www.ietf.org/rfc/rfc3229.txt
  RFC 3229，“Delta Encoding in HTTP”（“HTTP 中的差异编码”），说明了如何通过扩展 HTTP/1.1 来支持差异编码。

- *Introduction to Data Compression*[11]（《数据压缩导论》）
  这本书的作者是 Khalid Sayood，出版商为 Morgan Kaufmann Publishers。该书介绍了几种 HTTP 内容编码支持的压缩算法。

注 11：本书影印版由人民邮电出版社出版。（编者注）

- http://www.ietf.org/rfc/rfc1521.txt
  RFC 1521，“Multipurpose Internet Mail Extensions, Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies”（“多用途因特网邮件扩展，第一部分：规定和描述因特网报文主体格式的机制”），描述了MIME主体的格式。这份参考材料很有用，因为HTTP从MIME中借用了大量内容。设计这份文档的目的，就是为了提供在单一报文中包含多个对象的各种设施，比如用US-ASCII之外的字符集来表示主体文本，表示多种字体格式的文本消息以及表示非文本类的信息，比如图像和声音片段等。

- http://www.ietf.org/rfc/rfc2045.txt
  RFC 2045，“Multipurpose Internet Mail Extensions, Part One: Format of Internet Message Bodies”（“多用途因特网邮件扩展，第一部分：因特网报文主体的格式”），规定了用来描述MIME格式报文结构的各种首部，其中许多都和HTTP中的用法类似或相同。

- http://www.ietf.org/rfc/rfc1864.txt
  RFC 1864，“The `Content-MD5` Header Field”（“`Content-MD5`首部字段”），提供了用`Content-MD5`首部字段来做报文完整性检查的行为及用途方面的一些历史细节。

- http://www.ietf.org/rfc/rfc3230.txt
  RFC 3230，“Instance Digests in HTTP”（“HTTP中的实例摘要”），描述了对HTTP实体摘要处理的改进，解决了`Content-MD5`中存在的各种问题。

369

第16章

国际化

每天有上亿的人用数百种语言写着各种文档。为了真正实现万维网的目标，HTTP要能够传输和处理用多种语言和字母表编写的国际性文档。

本章涵盖网站国际化方面的两个主要问题：*字符集编码*（character set encoding）和*语言标记*（language tag）。HTTP 应用程序使用字符集编码请求和显示不同字母表中的文本，它们使用语言标记根据用户所理解的语言来说明并限制内容。而在本章的最后将讨论多语言 URI 和日期格式。

本章主要内容：

- 讲解 HTTP 如何与多语言字母表的方案和相关标准进行交互；
- 快速概览术语、技术和标准，以帮助 HTTP 编程人员正确理解（熟悉字符编码的读者可以跳过本节）；
- 解释对各种语言的标准命名系统，以及标准化的语言标记如何描述和选择内容；
- 概述国际性的 URI 要遵循的规则和注意事项；
- 简要讨论日期格式和其他国际化方面的问题。

16.1 HTTP对国际性内容的支持

HTTP 报文中可以承载以任何语言表示的内容，就像它能承载图像、影片，或任何类型的媒体那样。对 HTTP 来说，实体主体只是二进制信息的容器而已。

为了支持国际性的内容，服务器需要告知客户端每个文档的字母表和语言，这样客
370 户端才能正确地把文档中的信息解包为字符并把内容呈现给用户。

服务器通过 HTTP 协议的 `Content-Type` 首部中的 `charset` 参数和 `Content-Language` 首部告知客户端文档的字母表和语言。这些首部描述了实体主体的“信息盒子”里面装的是什么，如何把内容转换成合适的字符以便显示在屏幕上以及里面的词语表示的是哪种语言。

同时，客户端需要告知服务器用户理解何种语言，浏览器上安装了何种字母表编码算法。客户端发送 `Accept-Charset` 首部和 `Accept-Language` 首部，告知服务器它理解哪些字符集编码算法和语言以及其中的优先顺序。

下面的 HTTP 报文中的这些 `Accept` 首部可能是母语为法语的人发出的。他喜欢使用母语，但也会说一点儿英语，他的浏览器支持 iso-8859-1 西欧字符集编码和 UTF-8 Unicode 字符集编码：

```
Accept-Language: fr, en;q=0.8
Accept-Charset: iso-8859-1, utf-8
```

参数“q=0.8”是质量因子（quality factor），说明英语的优先级（0.8）比法语低（默认值是 1.0）。

16.2 字符集与HTTP

现在我们进入主题，开始研究网站国际化中最重要（且令人困惑）的方面——各国的字母表和它们的字符集编码。

Web 字符集标准很有些令人迷惑。由于必须阅读很多标准文档，其中术语复杂且不一致，再加上对外语不太熟悉，很多人首次尝试编写国际化的网站软件时，都被搞糊涂了。本节和下一节应该能让读者更容易地学会在 HTTP 中使用字符集。

16.2.1 字符集是把字符转换为二进制码的编码

HTTP 字符集的值说明如何把实体内容的二进制码转换为特定字母表中的字符。每个字符集标记都命名了一种把二进制码转换为字符的算法（反之亦然）。字符集标记在由 IANA 维护（参见 http://www.iana.org/assignments/character-sets）的 MIME 字符集注册机构进行了标准化。附录 H 中概述了其中的很多字符集。

下面的 Content-Type 首部告知接收者，传输的内容是一份 HTML 文件，用 charset 参数告知接收者使用 iso-8859-6 阿拉伯字符集的解码算法把内容中的二进制码转换为字符：

371

```
Content-Type: text/html; charset=iso-8859-6
```

iso-8859-6 的编码算法把 8 位值域映射为拉丁字母和阿拉伯字母，以及数字，标点和其他符号[1]。例如，在图 16-1 中，突出显示的二进制码的值是 225，它在 iso-8859-6 中被映射到阿拉伯字母“FEH”（读音类似英语字母 F）。

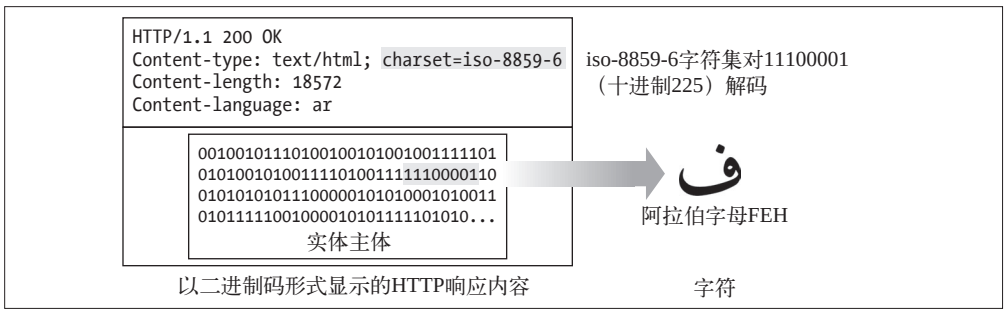


图 16-1 charset 参数告知客户端如何把内容中的二进制码转换为字符

注 1：与汉语、日语不同的是，阿拉伯语中只有 28 个字符。8 位空间有 256 个不同的值，足以容纳拉丁字符、阿拉伯字符以及其他符号。

有些字符编码（比如 UTF-8 和 iso-2022-jp）更加复杂，它们是可变长（variable-length）编码，也就是说每个字符的位数都是可变的。这种类型的编码允许使用额外的二进制位表示拥有大量字符的字母表（比如汉语和日语），仅用较少的二进制位来表示标准的拉丁字符。

16.2.2 字符集和编码如何工作

现在我们来看看字符集和编码到底做了什么。

我们想把文档中的二进制码转换为字符以便显示在屏幕上。但由于有很多不同的字母表，也有很多不同的方法把字符编码成二进制码（这些方法各有优缺点），我们需要一种标准方法来描述并应用把二进制码转换为字符的解码算法。

把二进制码转换为字符要经过两个步骤，如图 16-2 所示。

- 在图 16-2a 中，文档中的二进制码被转换成字符代码，它表示了特定编码字符集中某个特定编号的字符。在这个例子里，解码后的字符代码是数字编号 225。
- 在图 16-2b 中，字符代码用于从编码的字符集中选择特定的元素。在 iso-8859-6 中，值 225 对应阿拉伯字母“FEH”。在步骤 a 和 b 中使用的算法取决于 MIME 的 `charset` 标记。

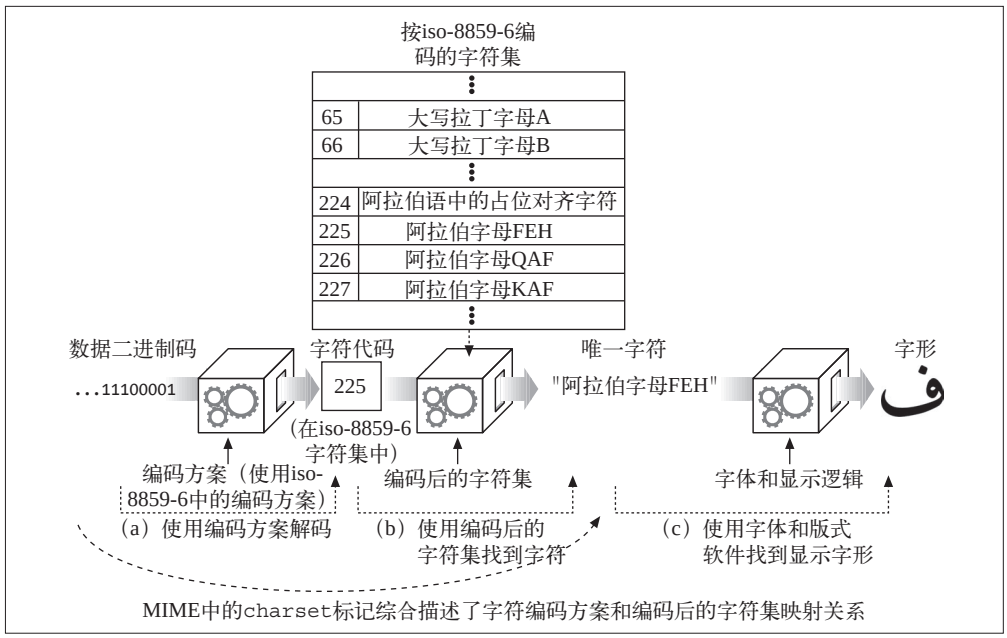


图 16-2 HTTP 协议中的 `charset` 是字符编码方案和编码后字符集的组合

国际化字符系统的关键目标是把语义（字母）和表示（图形化的显示形式）隔离开来。HTTP 只关心字符数据和相关语言及字符集标签的传输。字符形状的显示是由用户的图形显示软件（包括浏览器、操作系统、字体等）完成的，如图 16-2c 所示。 372

16.2.3 字符集不对，字符就不对

如果客户端使用了错误的字符集参数，客户端就会显示一些奇怪的错乱字符。假设浏览器从主体中获得值 225（二进制为 11100001）。

- 如果浏览器认为主体是用 iso-8859-1 西欧字符编码的，它将会显示带有重音符号的小写拉丁字母“a”：



- 如果浏览器使用 iso-8859-6 阿拉伯编码，它将会显示阿拉伯字母“FEH”：



- 如果浏览器使用 iso-8859-7 希腊编码，它将会显示小写的希腊字母“Alpha”： 373



- 如果浏览器使用 iso-8859-8 希伯来编码，它将会显示希伯来字母“BET”：



16.2.4 标准化的MIME `charset`值

特定的字符编码方案和特定的已编码字符集组合成一个 MIME 字符集（MIME charset）。HTTP 在 `Content-Type` 和 `Accept-Charset` 首部中使用标准化的 MIME `charset` 标记。MIME `charset` 的值都会在 IANA 注册。[2] 表 16-1 列出了文档和浏览器所使用的一些 MIME `charset` 编码方案。更完整的列表参见附录 H。

注 2：请从 http://www.iana.org/numbers.htm 获取注册的 charset 值的完整列表。

表16-1 MIME charset 编码标记

MIME charset 值	描　述
us-ascii	这是个著名的字符编码，在 1968 年就已标准化，称为 ANSI_X3.4-1968。它也称为 ASCII，但最好还是加上“US”前缀，因为 ISO 646 中有某些国际化的变体，它们修改了一些字符。US-ASCII 把 7 位数值映射到 128 个字符上。最高位未使用
iso-8859-1	iso-8859-1 是对 ASCII 的 8 位扩展，以支持西欧的多种语言。它使用了最高位以包含更多西欧字符，而保持 ASCII 的编码部分（0 ～ 127）没有变。它也称为 iso-latin-1，或简称为 Latin1
iso-8859-2	对 ASCII 扩展以包括中欧和东欧语言中的字符，包括捷克、波兰、罗马尼亚。它也称为 iso-latin-2
iso-8859-5	对 ASCII 扩展以包括斯拉夫语字符，使用这些字符的语言包括俄语、塞尔维亚语和保加利亚语
iso-8859-6	对 ASCII 扩展以包括阿拉伯语字符。因为阿拉伯语字符的显示形状会随它在单词中的位置而变化，阿拉伯语的显示引擎需要分析上下文来为每个字符生成正确的形状
iso-8859-7	对 ASCII 扩展以包括现代希腊语字符。以前称为 ELOT-928 或 ECMA-118:1986
iso-8859-8	对 ASCII 扩展以包括希伯来语和意第绪语（这两种语言都是犹太人所用的）的字符
iso-8859-15	更新了 iso-8859-1，用遗漏的法语和芬兰语字母替换了一些不太常用的标点符号和分数符号，并用新的欧元符号替换国际货币符号。这种字符集简称为 Latin0，可能将来会替代 iso-8859-1，作为欧洲的首选默认字符集
iso-2022-jp	iso-2022-jp 是在日语的电子邮件和网页内容中广泛使用的编码。它是一种变长编码方案，支持用单字节表示 ASCII 字符，但使用 3 字符的模态转义序列在 3 种日语字符集中切换
euc-jp	euc-jp 是遵循 ISO 2022 的变长编码，它用显式的二进制码模式来标识每个字符，不需要模态及转义序列。它使用单字节、2 字节以及 3 字节的序列来标识多个日语字符集中的字符
Shift_JIS	该编码起初是由微软公司开发的，有时称为 SJIS 或 MS Kanji。出于保持历史兼容性方面的原因，它有点儿复杂，并且不能映射所有的字符，不过它还是用的很普遍
koi8-r	KOI8-R 是为俄语设计的流行的 8 位因特网字符集编码，在 IETF RFC 1489 中定义。 这些大写字母是 Code for Information Exchange, 8 bit, Russian（俄语 8 位信息交换代码）的首字母缩略形式
utf-8	UTF-8 是一种用来表示 UCS（Unicode）的常用变长字符编码方案，UCS 的意思是 Universal Character Set of the world’s characters（世界字符统一字符集）。UTF-8 使用变长的编码来表示字符代码值，每个字符使用 1 ～ 6 个字节。UTF-8 的主要特点之一就是保持对普通的 7 位 ASCII 文本的后向兼容性
windows-1252	微软公司把它编码后的字符集称为 code page（代码页）。Windows 的代码页 1252（也称为 CP1252 或 WinLatin1）是对 iso-8859-1 的扩展

16.2.5 Content-Type首部和Charset首部以及META标志

Web 服务器通过在 Content-Type 首部中使用 charset 参数把 MIME 字符集标记发送给客户端：

```
Content-Type: text/html; charset=iso-2022-jp
```

如果没有显式地列出字符集，接收方可能就要设法从文档内容中推断出字符集。对于 HTML 内容来说，可以在描述 charset 的 <META HTTP-EQUIV="Content-Type"> 标记中找到字符集。

例 16-1 中展示了 HTML META 标记如何把字符集设置为日语编码 iso-2022-jp。如果文档不是 HTML 类型，或其中没有 META Content-Type 标记，软件可以设法扫描实际的文本，看看能否找出语言和编码的常见模式，以此推断字符编码。

例 16-1　可以在 HTML META 标记中规定字符编码

```
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-2022-jp">
    <META LANG="jp">
    <TITLE>A Japanese Document</TITLE>
</HEAD>
<BODY>
...
```

如果客户端无法推断出字符编码，就假定使用的是 iso-8859-1。

16.2.6 Accept-Charset首部

在过去的几十年间，人们开发了成千上万种字符编解码方法。大多数客户端不可能支持所有这些不同的字符编码和映射系统。

HTTP 客户端可以使用 Accept-Charset 请求首部来明确告知服务器它支持哪些字符系统。Accept-Charset 首部的值列出了客户端支持的字符编码方案。例如，下面的 HTTP 请求首部表明，客户端接受西欧字符系统 iso-8859-1 和 UTF-8 变长的 Unicode 兼容系统。服务器可以随便选择这两种字符编码方案之一来返回内容。

375

```
Accept-Charset: iso-8859-1, utf-8
```

注意，没有 Content-Charset 这样的响应首部和 Accept-Charset 请求首部匹配。为了和 MIME 标准兼容，响应的字符集是由服务器通过 Content-Type 响应首部的 charset 参数带回来的。不对称真是太糟了，不过需要的信息倒是都有了。

16.3 多语言字符编码入门

前一节描述了客户端和服务器是如何分别在 HTTP 的 Accept-Charset 首部和 Content-Type 首部的 charset 参数中携带字符编码信息的。对于工作中要开发大量国际化应用的 HTTP 程序员来说，为了能理解技术规范和相应的实现软件，需要深入地理解多语言字符系统。

由于术语很复杂且不一致，学习多语言字符系统不太容易。常常需要阅读标准文档，而且我们可能对工作涉及的那些语言还不太熟悉。本节是对字符系统及其标准的概览。如果读者对字符编码很熟悉，或者对这部分细节不感兴趣，可以直接跳到 16.4 节。

16.3.1 字符集术语

以下是应当了解的电子化字符系统的 8 个术语。

- *字符*
  字符是指字母、数字、标点、表意文字（比如汉语）、符号，或其他文本形式的书写“原子”。由统一字符集（Universal Character Set，UCS, 它的非正式的名字是 Unicode[3]）首创，为多种语言中的很多字符开发了一系列标准化的文本名称，它们常用来便捷地命名字符，而且不会与其他字符冲突。[4]

- *字形*
  描述字符的笔画图案或唯一的图形化形状。如果一个字符有多种不同的写法，就有多个字形（参见图 16-3）。

- *编码后的字符*
  分配给字符的唯一数字编号，这样我们就可以操作它了。

- *代码空间*
  计划用于字符代码值的整数范围。

- *代码宽度*
  每个（固定大小的）字符代码所用的位数。

- *字符库*
  特定的工作字符集（全体字符的一个子集）。

注 3：Unicode 是一个以 UCS 为基础而成立的商业化联合组织，致力推广商业产品。
注 4：字符的名称看起来类似 LATIN CAPITAL LETTER S 和 ARABIC LETTER QAF 的形式。

- *编码后的字符集*
  组成字符库（从全球的字符中选出若干字符）的已编码字符集，并为每个字符分配代码空间中的一个代码。换句话说，它把数字化的字符代码映射为实际的字符。

- *字符编码方案*
  把数字化的字符代码编码成一系列二进制码（并能相应地反向解码）的算法。字符编码方案可用来减少识别字符所需要的数据总量（压缩）、解决传输限制、统一重叠编码字符集。

16.3.2 字符集的命名很糟糕

从技术上说，MIME 中的 `charset` 标记（用在 `Content-Type` 首部的 `charset` 参数中和 `Accept-Charset` 首部中）描述的压根就不是字符集。MIME 中的 charset 值所命名的是把数据位映射为唯一的字符的一整套算法。它是*字符编码方案*（character encoding scheme）和*编码后的字符集*（coded character set）这两种概念的组合（参见图 16-2）。

因为关于字符编码方案和编码后的字符集方面的标准都已经发布过了，所以，这个术语的使用是很草率的，很容易引起混淆[5]。下面是 HTTP/1.1 的作者们对于他们如何使用这些术语的介绍（在 RFC2616 中）。

> 术语“字符集”在本文档中是指一种方法，它可以把一系列8位字节转换为一系列字符。注意：术语“字符集”经常被称为“字符编码”。但由于HTTP和MIME共享同样的注册信息，术语也能共享是很重要的。

IETF 在 RFC 2277 中也采用了非标准的术语。

> 本文档中使用术语“字符集”来表示一组把一系列 8 位字节转换为一系列字符的规则的集合，比如编码后的字符集与字符编码方案的组合。这与 MIME 的“`charset=`”参数中标识符的用法相同，并且已在 IANA 的字符集注册表中注册。（注意这不是在其他标准主体，比如在国际标准化组织 ISO 中使用的术语）。

注 5：更糟糕的是，MIME 中的 `charset` 标记经常会从特定的编码后字符集的名称或编码方案的名称里面选取。例如，iso-8859-1 是一个编码后字符集（它为一个包含 256 个欧洲字符的集合分配了数字化的代码），但 MIME 用 `charset` 值 iso-8859-1 来表示一种 8 位的、对编码后的字符集恒等的编码。这种不精确的术语并不是致命的问题，但在阅读标准文档的时候，需要对其假设用法保持清醒的头脑。

因此，在阅读标准文档的时候，要保持清醒，这样才能确切地知道它所定义的到底是什么。现在我们已经将相关的术语介绍完了，接下来更仔细地研究一下字符、字
377 形、字符集以及字符编码。

16.3.3 字符

字符是书写的最基本的构建单元。字符可以表示字母、数字、标点、表意符号（比如在汉语中）、数学符号，或其他书写的基本单元。

字符和字体以及风格无关。图 16-3 显示了同一个字符（UCS 中的命名是 LATIN SMALL LETTER A）的若干变体。尽管它们的笔画图案和风格有很大的不同，但母语是西欧语言的读者都能立刻辨认出这 5 个形状是同一个字符。



图 16-3 一个字符可以有很多不同的书写形式

在很多书面语体系中，根据一个字符在单词中位置的不同，同一个字符也会有不同的笔画形状。例如，图 16-4 中的 4 种笔画都表示字符 ARABIC LETTER AIN。[6] 图 16-4a 显示了 AIN 作为一个单独的字符时是如何书写的。图 16-4d 显示的是 AIN 在单词开头时的情形。图 16-4c 显示了 AIN 在单词中间的情形，而图 16-4b 显示的是 AIN 在单词结尾处的情形。[7]

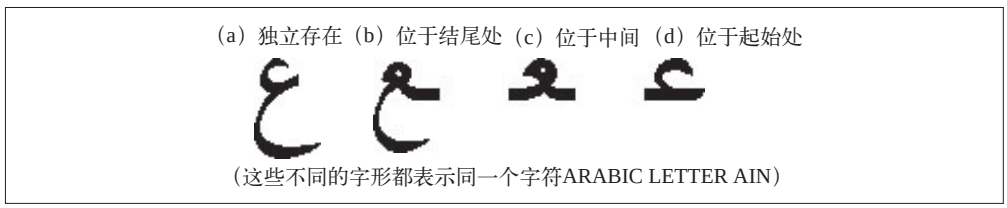


图 16-4 单个字符 ARABIC LETTER AIN 的 4 种与位置相关的书写形式

16.3.4 字形、连笔以及表示形式

不要把字符和字形混淆。字符是唯一的、抽象的语言“原子”。字形是画出每个字符时使用的特定方式。根据艺术形式和手法的不同，每个字符可以有很多不同的字形。[8]

注 6：AIN 的读音有些像 ayine 的发音，但发音朝向喉咙后部。

注 7：注意，阿拉伯语的单词是从右向左书写的。

注 8：很多人用术语“字形”来表示最终生成的位图图像，但从技术上说，字形是字符的内在形状，与字体和次要的艺术形式无关。进行这种区分不太容易，对我们的目的也没有什么用处。

同样，也不要把字符与表示形式混淆起来。为了让书法作品更好看，很多手写体和字体允许人们把相邻的字符漂亮地连写起来，称为连笔（ligatures），这样两个字符就平滑地连接在一起了。母语为英语的作者常把 F 和 I 结合为 FI 连笔（参见图 16-5a 和图 16-5b），而阿拉伯语的作者常把字符“LAM”和“ALIF”结合为一种很优雅的连笔（参见图 16-5c 和图 16-5d）。 378



图 16-5　连笔是相邻字符的另一种风格的表示形式，并非新的字符

这里给出一般的规则：如果用一种字形替代另一种的时候，文本的意思变了，那这些字形就是不同的字符。否则，它们就是同一个字符的不同风格的表示形式。[9]

16.3.5　编码后的字符集

根据 RFC 2277 和 2130 的定义，编码后的字符集把整数映射到字符。编码后的字符集经常用数组来实现[10]，通过代码数值来索引（参见图 16-6）。数组的元素就是字符[11]。

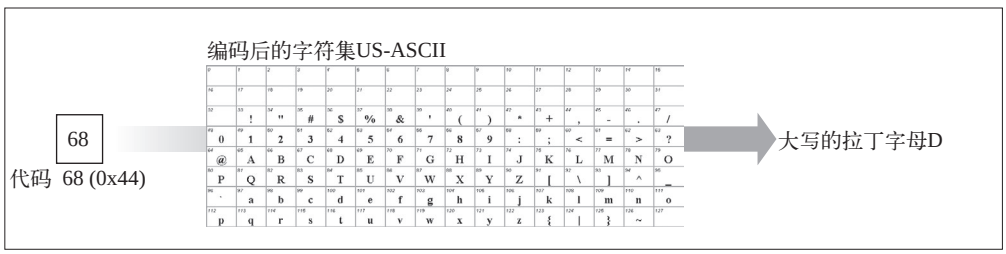


图 16-6　可以把编码后的字符集看作数组，把数值化的代码映射到字符

下面我们来看一些重要的编码后的字符集标准，包括具有历史意义的 US-ASCII 字符集、ASCII 的 iso-8859 扩展、日文的 JIS X 0201 字符集以及统一字符集（Universal Character Set，Unicode）。

注 9：语义和表示方式之间的区别并不总是很清晰的。为了实现的方便，已经为同一个字符的某些表示变体分配了不同的字符。不过我们还是尽量要避免这种做法。

注 10：数组可以是多维的，这样代码数字中的不同二进制码就可以索引到数组的不同维。

注 11：图 16-6 使用了一个网格来表示编码后的字符集。网格中的每个元素都包含一个字符图像。这些图像应看作是符号，图像“D”是字符 LATIN CAPITAL LETTER D 的简称，而不是任何特定的图形化字形。

1. US-ASCII:所有字符集的始祖

ASCII 是最著名的编码后字符集，早在 1968 年就由 ANSI 在标准 X3.4，“美国标准信
379 息交换代码”（American Standard Code for Information Interchange）中进行了标准化。ASCII 的代码值只是从 0 到 127，因此只需要 7 个二进制码就可以覆盖代码空间。ASCII 的推荐名称是 US-ASCII，这样可以和那些 7 位字符集的一些国际化变体区分开来。

HTTP 报文（首部、URI 等）使用的字符集是 US-ASCII。

2. iso-8859

iso-8859 字符集标准是 US-ASCII 的 8 位超集，使用二进制码的高位增加了一些国际化书面字符。由额外的二进制码提供的附加空间（多了 128 个代码）还不够大，甚至都不够所有的欧洲字符使用，更不用说亚洲字符了。因此 iso-8859 为不同地区定制了不同的字符集，如下所示。

iso-8859-1	西欧语言（例如，英语、法语）
iso-8859-2	中欧和东欧语言（例如，捷克、波兰）
iso-8859-3	南欧语言
iso-8859-4	北欧语言（例如，拉托维亚，立陶宛，格陵兰）
iso-8859-5	斯拉夫语（例如，保加利亚、俄罗斯、塞尔维亚）
iso-8859-6	阿拉伯语
iso-8859-7	希腊语
iso-8859-8	希伯来语
iso-8859-9	土耳其语
iso-8859-10	日耳曼和斯堪的纳维亚语言（例如，冰岛、因纽特）
iso-8859-15	对 iso-8859-1 的修改，包括了新的欧元字符

iso-8859-1 也称为 Latin1，是 HTML 的默认字符集。可以用它来表示大多数西欧语言的文本。因为 iso-8859-15 中包含了新的欧元符号，有过一些用它来代替 iso-8859-1 并作为 HTTP 默认编码后字符集的讨论。然而，由于 iso-8859-1 已经被广泛采用，要大范围地变更到 iso-8859-15 恐怕不是短时间内可以完成的。

3. JIS X 0201

JIS X 0201 是把 ASCII 扩展到日文半宽片假名字符的一个极小化的字符集。半宽片假名字符最早用在日文电报系统中。JIS X 0201 常常被称作 JIS Roman，JIS 是“Japanese Industrial Standard”（日文工业化标准）的缩写。

4. JIS X 0208与JIS X 0212

日文中包括数千个来自几个书面语系统中的字符。尽管可以（很痛苦地）勉强只使用 JIS X 0201 中的那 63 个基本的片假名字符，但实际使用中需要远比这个更完整
380 的字符集。

JIS X 0208 字符集是首个多字节日文字符集，它定义了 6879 个编码的字符，其中大多数是来源于中文的日本汉字。JIS X 0212 字符集又扩充了 6067 个字符。

5. UCS

UCS（Universal Character Set，统一字符集）是把全世界的所有字符整合到单一的编码后字符集的环球标准化成果。UCS 由 ISO 10646 定义。Unicode 是遵循 UCS 标准的商业化联合组织。UCS 具有能容纳百万以上字符的代码空间，不过基本集合只有大约 5 万个字符。

16.3.6 字符编码方案

字符编码方案规定如何把字符的代码数字打包装入内容比特，以及在另一端如何将其解包回字符代码（参见图 16-7）。字符编码方案有以下 3 种主要类型。

- 固定宽度
  固定宽度方式的编码用固定数量的比特表示每个编码后的字符。它们能被快速处理，但可能会浪费空间。

- 可变宽度（无模态）
  可变宽度方式的编码对不同的字符代码数字采用不同数量的比特。对于常用字符，这样可以减少需要的位数，而且还能在允许使用多字节来表示国际性字符的同时，保持对传统 8 位字符集的兼容性。

- 可变宽度（有模态）
  有模态的编码使用特殊的“转义”模式在不同的模态之间切换。例如，可以用有模态的编码在文本中使用多个互相有重叠的字符集。有模态的编码处理起来比较复杂，但它们可以有效地支持复杂的书写系统。

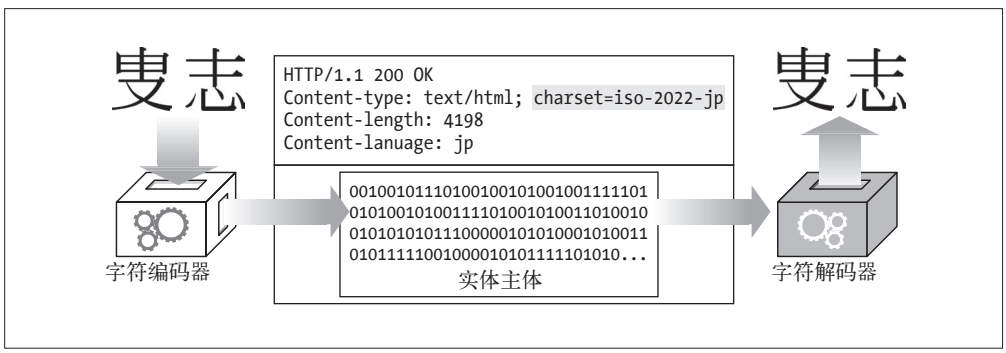


图 16-7　字符编码方案把字符代码编码为比特，并负责反向变换

381 下面我们来看一些常见的编码方案。

1. 8位

8 位固定宽度恒等编码把每个字符代码编码为相应的 8 位二进制值。它只能支持有 256 个字符代码范围的字符集。iso-8859 字符集家族系列使用的就是 8 位恒等编码。

2. UTF-8

UTF-8 是一种流行的为 UCS 设计的字符编码方案，UTF 表示 UCS 变换格式（UCS Transformation Format）。UTF-8 为字符代码值使用的是无模态的变宽编码，第一字节的高位表示编码后的字符所用的字节数，所需的每个后续字节都含有 6 位的代码值（参见表 16-2）。

如果编码后的第 1 字节的最高位是 0，长度就是 1 字节，剩余的 7 位就包含字符的代码。这样带来的美妙结果就是和 ASCII 兼容（但和 iso-8859 系列不兼容，因为 iso-8859 系列使用了最高位）。

表16-2 UTF-8 变宽无模态编码

字符代码的二进制位	字节1	字节2	字节3	字节4	字节5	字节 6
0-7	0ccccccc	-	-	-	-	-
8-11	110ccccc	10cccccc	-	-	-	-
12-16	1110cccc	10cccccc	10cccccc	-	-	-
17-21	11110ccc	10cccccc	10cccccc	10cccccc	-	-
22-26	111110cc	10cccccc	10cccccc	10cccccc	10cccccc	-
27-31	1111110c	10cccccc	10cccccc	10cccccc	10cccccc	10cccccc

例如，字符代码 90（ASCII 的“Z”）会被编码为 1 个字节（01011010），而代码 5073（13 位二进制值为 1001111010001）会被编码为 3 个字节：

11100001 10**001111** 10**010001**

3. iso-2022-jp

iso-2022-jp 是互联网上的日文文档中广泛使用的编码。它是变宽、有模态的，所有值都不超过 128，以避免和不支持 8 位字符的软件出现兼容性问题。

编码上下文始终被设置为 4 种预设的字符集之一[12]，使用特殊的“转义序列”（escape sequence）在字符集之间切换。iso-2022-jp 的初始状态使用 US-ASCII 字符

注 12：iso-2022-jp 编码和这 4 种字符集是紧密绑定的，而其他一些编码是和特定的字符集无关的。

集，使用 3 个字节的转义序列可以切换到 JIS X 0201（JIS-Roman）字符集或大得多的 JIS X 0208-1978 和 JIS X 0208-1983 字符集。

表 16-3 中列出了这些转义序列。实际上，日文文本以 ESC $ @ 或 ESC $ B 开始，以 ESC (B 或 ESC (J 结束。

表16-3　iso-2022-jp的字符集切换转义序列

转义序列	转义后的字符集	每个代码的字节数
ESC (B	US-ASCII	1
ESC (J	JIS X 0201-1976 (JIS Roman)	1
ESC $ @	JIS X 0208-1978	2
ESC $ B	JIS X 0208-1983	2

在 US-ASCII 或 JIS-Roman 模态下，每个字符使用单个字节。当使用更大的 JIS X 0208 系列的字符集时，每个字符代码使用 2 个字节。该编码把发送的字节的值域范围限制在 33~126 之间 [13]。

4. euc-jp

euc-jp 是另一种流行的日文编码。EUC 代表“Extended Unix Code”（扩展 Unix 代码），最早是为了在 Unix 操作系统上支持亚洲字符而开发的。

和 iso-2022-jp 类似，euc-jp 编码也是变长的，允许使用几种标准的日文字符集。但和 iso-2022-jp 不同的是，euc-jp 编码不是模态的。没有转义序列可以在不同模态之间切换。

euc-jp 支持 4 种编码后的字符集：JIS X 0201（JIS-Roman，对 ASCII 进行一些日文替换）、JIS X 0208、半宽片假名（最早在日文电报系统中使用的 63 个字符）以及 JIS X 0212。

编码 JIS Roman（它和 ASCII 兼容）的时候使用 1 个字节，对 JIS X 0208 和半宽片假名则使用 2 个字节，而对 JIS X 0212 使用 3 个字节。这种编码有点浪费空间但处理起来很简单。

表 16-4 概括了此编码的格局。

注 13：尽管每个字节只能有94个不同的值(33~126)，这也足够覆盖JIS X 0208系列字符集里面的所有字符了，因为这些字符集是按照 94×94 的网格来组织代码值的，所以 2 个字节足以覆盖 JIS X 0208 字符集中的全部字符代码。

表16-4　euc-jp编码值

哪个字节	编　码　值
JIS X 0201（有 94 个编码后的字符）	
第 1 个字节	33~126
JIS X 0208（有 6879 个编码后的字符）	
第 1 个字节	161~254
第 2 个字节	161~254
半宽片假名（有 63 个编码后的字符）	
第 1 个字节	142
第 2 个字节	161~223
JIS X 0212（有 6067 个编码后的字符）	
第 1 个字节	143
第 2 个字节	161~254
第 3 个字节	161~254

现在我们就介绍完了字符集和编码。下面一节将解释语言标记和 HTTP 如何使用语言标记把合适的内容传给受众。请参见附录 H，那里列出了详细的标准化字符集。

16.4　语言标记与HTTP

语言标记是命名口语的标准化字符串短语。

名字需要标准化，不然的话，有些人会把法语文档打上 French 标记，而有些其他人会用 Français，还有些人可能会用 France，更有些懒人可能会用 Fra 甚至是 F。标准化语言标记就可以避免这些混乱。

英语的标记是 en，德语的标记是 de，韩语的标记是 ko，等等。语言标记能够描述语言的地区变种和方言，比如巴西葡萄牙语的标记是 pt-BR、美式英语的标记是 en-US，汉语中的湖南话的标记是 zh-xiang。甚至还有个标准语言标记 i-klingon 是描述克林根语 [14] 的！

16.4.1　Content-Language首部

实体的 `Content-Language` 首部字段描述实体的目标受众语言。如果内容主要是给

注 14：科幻名著《星际迷航》（*Star Trek*）中的外星种族，加州大学伯克利分校的语言学博士马克乌克兰为其发明了一套语言。（译者注）

法语受众的，其 Content-Language 首部字段就将包含：


```
Content-Language: fr
```

Content-Language 首部不仅限于文本文档。音频片段、电影以及应用程序都有可能是面向特定语言受众的。任何面向特定语言受众的媒体类型都可以有 Content-Language 首部。在图 16-8 中，音频文件标记为面向纳瓦霍[15]（Navajo）听众。

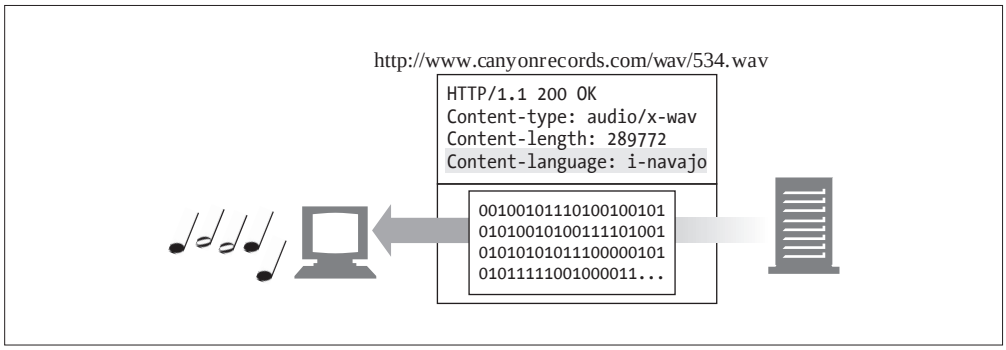


图 16-8 Content-Language 首部把名为 *Rain Song*（雨之歌）的音频片段标记为面向纳瓦霍听众

如果内容是面向多种语言受众的，可以列出多种语言。就像在 HTTP 规范中建议的，一份同时用英语和毛利语写的“Treaty of Waitangi”[16]（怀唐伊条约）译稿，可以这样描述：

```
Content-Language: mi, en
```

不过，不能仅根据有多种语言在实体中出现就认为它是面向多种语言受众的。为初学者编写的语言入门教材，比如“A First Lesson in Latin”（拉丁语第一课），显然是为英语受众准备的，应当只用 en 来描述。

16.4.2 Accept-Language首部

我们绝大多数人至少懂一种语言。HTTP 允许我们把语言方面的限制和优先选择都发送给网站服务器。如果网站服务器有以多种语言表示的资源版本，它就能把内容用我们最优选的语言表示出来。[17]

注 15：美国最大的印地安部落。（译者注）

注 16：英国政府与新西兰毛利酋长们在 1840 年签署的条约，是新西兰的“建国文献”。（译者注）

注 17：服务器也可以根据 Accept-Language 首部生成适合用户语言的动态内容，或据此选择图像，或选择适合目标语言的商业促销等。

这里有个例子，客户端请求西班牙语内容：

```
Accept-Language: es
```

可以在 Accept-Language 首部中放入多个语言标记以枚举所支持的全部语言及其优先顺序（从左到右）。这里有个例子，客户端首选英语，但也接受瑞士德语（标准语言标记是 de-CH）或其他德语变种（标记是 de）：

```
Accept-Language: en, de-CH, de
```

客户端使用 Accept-Language 首部和 Accept-Charset 首部请求可以理解的内容。第 17 章将研究这些机制的细节。

16.4.3 语言标记的类型

在 RFC 3066，“Tags for the Identification of Languages”（标识语言的标记）中记录了语言标记的标准化语法。可以用语言标记来表示：

- 一般的语言分类（比如 es 代表西班牙语）；
- 特定国家的语言（比如 en-GB 代表英国英语）；
- 语言的方言（比如 no-bok 指挪威的书面语）；
- 地区性的语言（比如 sgn-US-MA 代表美国马撒葡萄园岛上的手语）；
- 标准化的非变种语言（比如 i-navajo）；
- 非标准的语言（比如 x-snowboarder-slang[18]）。

385

16.4.4 子标记

语言标记有一个或多个部分，用连字号分隔，称为*子标记*：

- 第一个子标记称为*主子标记*，其值是标准化的；
- 第二个子标记是可选的，遵循它自己的命名标准；
- 其他尾随的子标记都是未注册的。

主子标记中只能含有字母（A ~ Z）。其后的子标记可以含有字母和数字，长度最多 8 个字符。图 16-9 中给出了一个示例。

注 18：描述由“酷爱滑雪者们”（shredders）使用的一种俚语。（译者注）

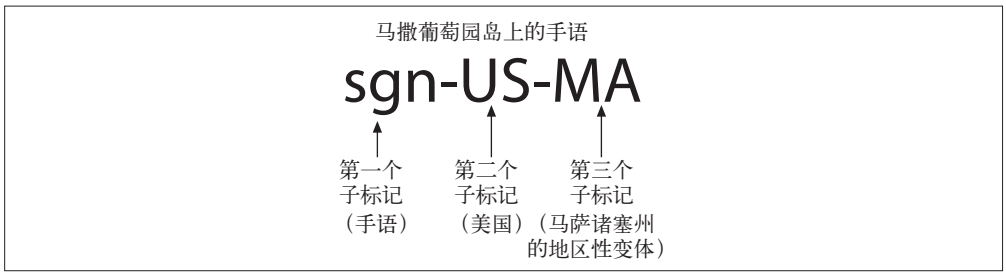


图 16-9　语言标记被分隔为子标记

16.4.5　大小写

所有的标记都是不区分大小写的，也就是说，标记 en 和 eN 是等价的。但是，习惯上用全小写来表示一般的语言，而用全大写来表示特定的国家。例如，fr 表示所有分类为法语的语言，而 FR 表示国家法国 [19]。

16.4.6　IANA语言标记注册

第一个和第二个语言子标记的值由各种标准文档以及相关的维护组织定义。IANA[20] 依据 RFC 3066 中概括的规则来管理标准的语言标记列表。

如果语言标记由标准的国家和语言值组成，标记就不需要专门注册。只有那些无法用标准的国家和语言值构成的语言标记才需要专门向 IANA 注册 [21]。下面几节概括了 RFC 3066 中关于第一子标记和第二子标记的标准。

386

16.4.7　第一个子标记——名字空间

第一个子标记通常是标准化的语言记号，选自 ISO 639 中的语言标准集合。不过也可以用字母 i 来标识在 IANA 中注册的名字，或用 x 表示私有的或者扩展的名字。下面是各种规则。

如果第一个子标记含有：

- 2 个字符，那就是来自 ISO 639[22] 和 639-1 标准的语言代码；

注 19：这种惯用法是 ISO 标准 3166 中推荐的。

注 20：参见 http://www.iana.org 和 RFC 2860。

注 21：编写本书的时候，IANA 中只显式注册了 21 种语言标记，包括汉语中的广东话（标记是 zh-yue）、新挪威语（标记是 no-nyn）、卢森堡语（标记是 i-lux）以及克林根语（语言标记是 i-klingon）。在因特网上使用的其余数百种口语的标记均由标准的组件构成。

注 22：参见 ISO 标准 639，“Codes for the representation of names of languages—Part2：Alpha-3code”（表示语言名字的代码——第 2 部分：Alpha-3 代码）。

- 3 个字符，那就是来自 ISO 639-2[23] 标准及其扩展的语言代码；
- 字母 i，该语言标记是在 IANA 显式注册的；
- 字母 x，该语言标记是私有的、非标准的，或扩展的子标记。

附录 G 中总结了 ISO 639 和 639-2 中的名字。表 16-5 中给出了一些示例。

表16-5　ISO 639和ISO 639-2中的语言代码示例

语　　言	ISO 639	ISO 639–2
阿拉伯语	ar	ara
汉语	zh	chi/zho
荷兰语	nl	dut/nla
英语	en	eng
法语	fr	fra/fre
德语	de	deu/ger
现代希腊语	el	ell/gre
希伯来语	he	heb
意大利语	it	ita
日语	ja	jpn
韩语	ko	kor
挪威语	no	nor
俄语	ru	rus
西班牙语	es	esl/spa
瑞典语	sv	sve/swe
土耳其语	tr	tur

387

16.4.8　第二个子标记——名字空间

第二个子标记通常是标准化的国家记号，选自 ISO 3166 中的国家代码和地区标准集合。不过也可以是在 IANA 注册过的其他字符串。下面是各种规则。

如果第二个子标记含有：

- 2 个字符，那就是 ISO 3166[24] 中定义的国家 / 地区；

注 23：参见 ISO 639-2，“Codes for the representation of names of languages—Part 2: Alpha-3 code”（表示语言名字的代码——第 2 部分：Alpha-3 代码）。

注 24：国家代码 AA、QM-QZ、XA-XZ 以及 ZZ 是在 ISO 3166 中保留的作为用户分配的代码。一定不能用这些值来构造语言标记。

- 3~8 个字符，可能是在 IANA 中注册的值；
- 单个字符，这是非法的情况。

表 16-6 中列出了 ISO 3166 中的部分国家代码。附录 G 中列出了完整的国家代码。

表16-6　ISO 3166中的国家代码示例

国　　家	代　　码
巴西	BR
加拿大	CA
中国	CN
法国	FR
德国	DE
梵蒂冈	VA
印度	IN
意大利	IT
日本	JP
黎巴嫩	LB
墨西哥	MX
巴基斯坦	PK
俄罗斯联邦	RU
英国	GB
美国	US

388

16.4.9　其余子标记——名字空间

除了最长可以到 8 个字符（字母和数字）之外，第三个和其后的子标记没有特殊规则。

16.4.10　配置和语言有关的首选项

可以在浏览器的配置文件中配置和语言有关的首选项。

网景公司的 Navigator 的设置方法是：编辑→首选项→语言，而微软公司的 Internet Explorer 浏览器的设置方法是：工具→ Internet 选项→语言。

16.4.11　语言标记参考表

为便于使用，附录 G 中给出了语言标记的参考表。

- 表 G-1 列出了在 IANA 注册的语言标记。
- 表 G-2 列出了 ISO 639 中的语言代码。
- 表 G-3 列出了 ISO 3166 中的国家代码。

16.5 国际化的URI

直到今天，URI 还没有为国际化提供足够的支持。除了少数（定义得很糟的）例外，URI 如今还是由 US-ASCII 字符的一个子集组成的。人们正在努力使主机名和 URL 的路径中能包含更丰富的集合中的字符，但直到现在，这些标准还没有被广泛接受和部署。现在让我们来回顾一下当前的一些尝试。

16.5.1 全球性的可转抄能力与有意义的字符的较量

URI 的设计者们希望世界上每个人都能通过电子邮件、电话、公告板，甚至无线电来共享 URI。他们还希望 URI 容易使用和记忆，但这两个目标是相互冲突的。

为了让世界各地的人们都能够便捷地输入、操控，以及共享 URI，设计者们为 URI 选择了常用字符的一个很有限的子集（基本的拉丁字母表中的字母、数字以及少数特殊符号）。世界上绝大多数软件和键盘都支持这个小的字符集合。

但不幸的是，限制了字符集的话，URI 就无法被全球的人们方便地使用和记忆。世
389 界上有很大一部分人甚至都不认识拉丁字母，他们几乎无法把 URI 当作抽象模式来记忆。

URI 的设计者们觉得确保资源标识符的*可转抄能力*（transcribability）和共享能力比让它们由最有意义的字符组成更加重要，因此（如今的）URI 基本上是由 ASCII 字符的受限子集构成的。

16.5.2 URI字符集合

URI 中允许出现的 US-ASCII 字符的子集，可以被分成*保留*、*未保留*以及*转义*字符这几类。未保留的字符可用于 URI 允许其出现的任何部分。保留的字符在很多 URI 中都有特殊的含义，因此一般来说不能使用它们。表 16-7 中列出了全部未保留、保留，以及转义字符。

表16-7 URI字符语法

字符类别	字符列表
未保留	[A-Za-z0-9] | "-" | "_" | "." | "!" | "~" | "*" | " ' " | "(" | ")"
保留	";" | "/" | "?" | ":" | "@" | "&" | "=" | "" | "$" | ","
转义	"%" <HEX> <HEX>

16.5.3 转义和反转义

URI 转义提供了一种安全的方式，可以在 URI 内部插入保留字符以及原本不支持的字符（比如各种空白）。每个转义是一组 3 字符序列，由百分号（%）后面跟上两个十六进制数字的字符。这两个十六进制数字就表示一个 US-ASCII 字符的代码。

例如，要在 URL 中插入一个空白（ASCII 32），可以用转义 %20，因为 20 是 32 的十六进制表示。类似地，如果想插入一个百分号并且不想让它被当作转义，就可以输入 %25，25 是百分号的 ASCII 代码的十六进制值。

图 16-10 展示了 URI 中的概念性字符是如何转换为当前字符集中字符的代码字节的。需要处理 URI 时，转义会被反转义回来，产生它们代表的 ASCII 代码的字节。

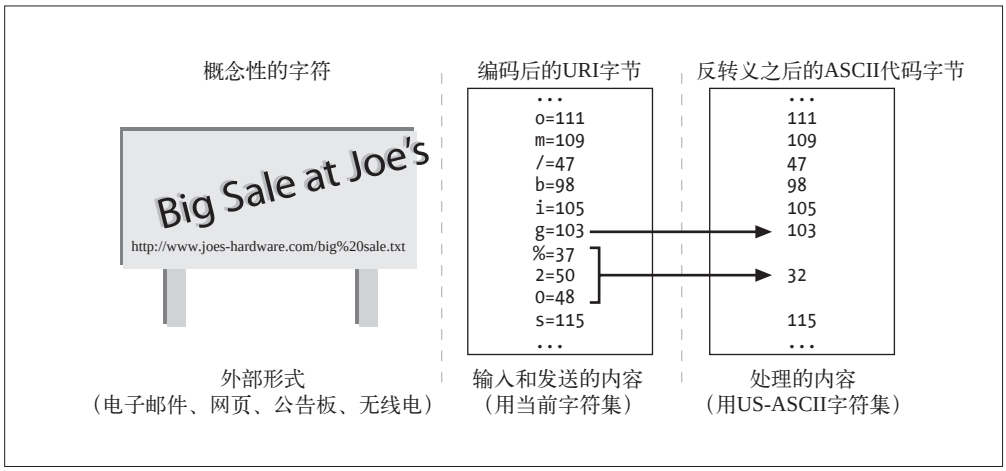


图 16-10　URI 中的字符在传输的时候要转义，但处理的时候要反转义

在内部处理时，HTTP 应用程序应当在传输和转发 URI 的时候保持转义不变。HTTP 应用程序应该仅在需要数据的时候才对 URI 进行转义。更重要的是，应用程序应该确保任何 URI 都不会被反转义 2 次，因为在转义的时候可能会把百分号编码进去，反转义出来之后，再转一次就会导致数据丢失。

16.5.4 转义国际化字符

需要注意的是，要转义的值本身应该在 US-ASCII 代码值的范围内（0 ~ 127）。某些应用程序试图用转义值来表示 iso-8859-1 中扩展的字符（代码范围在 128 ~ 255）。例如，网站服务器可能会错误地用转义来对包含了国际字符的文件名进行编码。这样做是不对的，可能会使别的应用出问题。

390

例如，文件名 Sven Ölssen.html（包含了一个元音变音）可能被网站服务器编码为 Sven%20%D6lssen.html。把空格编码为 %20 是对的，但从技术上说，把 Ö 编码为 %D6 是非法的，因为代码 D6（十进制值 214）落在了 ASCII 代码范围之外。ASCII 只定义了最大值为 0x7F（十进制值 127）的代码。

16.5.5 URI中的模态切换

有些 URI 也用 ASCII 字符的序列来表示其他字符集中的字符。例如，可能使用 iso-2022-jp 编码插入“ESC (J”，切换到 JIS-Roman 字符集，用“ESC (B”切换回 ASCII 字符集。这在一些本地化的环境中可以工作，但这种方式没有进行良好的定义，而且没有标准化的方案来识别 URL 所使用的特定编码。正如 RFC 2396 的作者所说的那样：

> 不过，对于含有非ASCII字符的原始字符序列来说，境况更加复杂。如果可能用到多个字符集的话，传输表示字符序列的8位字节序列的因特网协议期待能有办法来识别所用的字符集[RFC 2277]。
>
> 然而，在通用的URI语法中没有提供进行这种识别的手段。个别的URI方案可以请求单一的字符集，定义默认的字符集，或提供指示所用字符集的方法。期待将来对这个规范的修改能为URI中的字符编码提供一种系统化的处理方案。

目前，URI 对国际化应用还不是非常友好。URI 的可移植性目标比语言灵活性方面的目标更重要。人们正在尽最大努力使 URI 更加国际化，但在短期内，HTTP 应用程序还是应当坚持使用 ASCII。它从 1968 年就出现了，所以只用它的话，一切还不至于太糟。

391

16.6 其他需要考虑的地方

本节讨论在编写国际化的 HTTP 应用程序时，必须牢记的其他一些东西。

16.6.1 首部和不合规范的数据

HTTP 首部必须由 US-ASCII 字符集中的字符构成。不过，并不是所有的客户端和服务器都正确地实现了这一点，你可能会时不时收到一些代码值大于 127 的非法字符。

很多 HTTP 应用程序使用操作系统和库例程来处理字符（比如 Unix 中的字符分类库 ctype），但不是所有这些库都支持 ASCII 范围（0 ~ 127）之外的字符代码。

在某些情况下（一般来说，是较老的实现），当输入非 ASCII 字符时，这些库可能会返回不正确的结果，或者使应用程序崩溃。假设报文中含有非法数据，在使用这

些字符分类库来处理 HTTP 报文之前，要仔细阅读它们的文档。

16.6.2 日期

HTTP 的规范中明确定义了合法的 GMT 日期格式，但要知道并非所有 Web 服务器和客户端都遵守这些规则。例如，我们曾见过 Web 服务器发送的无效 HTTP Date（日期）首部中的月份是用本地语言表示的。

HTTP 应用程序应当尝试容忍一些不合规矩的日期，不能在接收的时候崩溃。不过也不是所有发送出来的日期都能被正确解释，如果日期无法解析，服务器应当谨慎处理。

16.6.3 域名

DNS 目前还不支持在域名中使用国际化的字符。现在正在进行支持多语言的域名的相关标准化工作，但还没有被广泛部署。

16.7 更多信息

万维网的极大成功意味着 HTTP 应用程序要继续在不同的语言和字符集之间交换更多的内容。更多关于多语言的多媒体这个重要但有些复杂的话题的信息，请参考下列资料来源。

392

16.7.1 附录

- 表 H-1 列出了在 IANA 注册的字符集标记。
- 表 G-1 列出了在 IANA 注册的语言标记。
- 表 G-2 列出了 ISO 639 中的语言代码。
- 表 G-3 列出了 ISO 3166 中的国家代码。

16.7.2 互联网的国际化

- http://www.w3.org/International/
  “Making the WWW Truly World Wide”（“使 WWW 真正遍布全球”）——W3C 国际化和本地化网站。

- http://www.ietf.org/rfc/rfc2396.txt
  RFC 2396，“Uniform Resource Identifiers（URI）：Generic Syntax”（“统一资源描述符：一般语法”），是 URI 的定义文档。该文档中包括了描述国际化 URI 中的字符集限制方面的章节。

- *CJKV Information Processing*（《中日韩越信息处理》）
  Ken Lunde 著，由 O'Reilly & Associates 公司出版。CJKV 是亚洲语言的电子化字符处理方面的权威经典。亚洲语言的字符集多种多样、复杂难懂，但这本书详细介绍了大型字符集的各种标准技术。

- http://www.ietf.org/rfc/rfc2277.txt
  RFC 2277，“IETF Policy on Character Sets and Languages”（“IETF 关于字符集和语言的策略”），其中记录了互联网工程指导组（Internet Engineering Steering Group，IESG）应用的当前策略，这些策略是为了配合互联网工程任务组（Internet Engineering Task Force，IETF）帮助各种互联网协议使用多种语言和字符交换数据而做的标准化努力。

16.7.3 国际标准

- http://www.iana.org/numbers.htm
  IANA 中有已注册的各种名字和数字编号的库。其中的 Protocol Numbers and Assignments Directory（协议编号和分配目录）中包含了因特网上使用的已注册字符集记录。因为有关国际间通信的大部分工作都是在 ISO 的领域内，而不是因特网的团体完成的，所以 IANA 的这份列表算不上详尽。

- http://www.ietf.org/rfc/rfc3066.txt
393 RFC 3066，“Tags for the Identification of Languages”（“标识语言的标记”），描述了语言标记、它们的值，以及如何构造这些标记。

- “Codes for the representation of names of languages”（“表示语言名称的代码”）
  ISO 639:1988 (E/F)，国际标准化组织，第 1 版。

- “Codes for the representation of names of languages—Part 2: Alpha-3 code”（“表示语言名称的代码，第 2 部分：Alpha-3 代码”）
  ISO 639-2:1998，ISO TC46/SC4 和 TC37/SC2 联合工作组，第 1 版。

- “Codes for the representation of names of countries”（“表示国家名称的代码”）
394 ISO 3166:1988 (E/F)，国际标准化组织，第 3 版。

第17章

内容协商与转码

一个 URL 常常需要代表若干不同的资源。例如那种需要以多种语言提供其内容的网站站点。如果某个站点（比如 Joe 的五金商店这样的站点）有说法语的和说英语的两种用户，它可能想用这两种语言提供网站站点信息。但在这种情况下，当用户请求 http://www.joes-hardware.com 时，服务器应当发送哪种版本呢？法文版还是英文版？

理想情况下，服务器应当向英语用户发送英文版，向法语用户发送法文版——用户只要访问 Joe 的五金商店的主页就可以得到相应语言的内容。幸运的是，HTTP 提供了*内容协商*方法，允许客户端和服务器作这样的决定。通过这些方法，单一的 URL 就可以代表不同的资源（比如，同一个网站页面的法语版和英语版）。这些不同的版本称为*变体*。

对于特定的 URL 来说，服务器还可以根据其他原则来决定发送什么内容给客户端最合适。在有些场合下，服务器甚至可以自动生成定制的页面。比如，服务器可以为手持设备把 HTML 页面转换成 WML 页面。这类动态内容变换被称为*转码*。这些变换动作是 HTTP 客户端和服务器之间进行内容协商的结果。

本章，我们将讨论内容协商和网站应用程序应该如何担负内容协商的责任。

17.1 内容协商技术

共有 3 种不同的方法可以决定服务器上哪个页面最适合客户端：让客户端来选择、服务器自动判定，或让中间代理来选。这 3 种技术分别称为客户端驱动的协商、服
395 务器驱动的协商以及透明协商（参见表 17-1）。本章，我们将研究每种技术的机制及其优缺点。

表17-1 内容协商技术概要

技　术	工作原理	优　点	缺　点
客户端驱动	客户端发起请求，服务器发送可选项的列表，客户端选择	在服务器端的实现最容易。客户端可以选择最合适的内容	增加了时延：为了获得正确的内容，至少要发送两次请求
服务器驱动	服务器检查客户端的请求首部集并决定提供哪个版本的页面	比客户端驱动的协商方式要快。HTTP 提供了 q 值机制，允许服务器近似匹配，还提供了 Vary 首部供服务器告知下游的设备如何对请求估值	如果结论不是很明确（比如首部集不匹配），服务器要做猜测
透明	某个中间设备（通常是缓存代理）代表客户端进行请求协商	免除了 Web 服务器的协商开销。比客户端驱动的协商要快	关于如何进行透明协商，还没有正式的规范

17.2 客户端驱动的协商

对于服务器来说，收到客户端请求时只是发回响应，在其中列出可用的页面，让客户端决定要看哪个，这是最容易的事情。很显然，这是服务器最容易实现的方式，而且客户端很可能选择到最佳的版本（只要列表中有让客户端选择的足够信息）。不利之处是每个页面都需要两次请求：第一次获取列表，第二次获取选择的副本。这种技术速度很慢且过程枯燥乏味，让用户厌烦。

从实现原理上来说，服务器实际上有两种方法为客户端提供选项：一是发送回一个HTML 文档，里面有到该页面的各种版本的链接和每个版本的描述信息；另一种方法是发送回 HTTP/1.1 响应时，使用 300 Multiple Choices 响应代码。客户端浏览器收到这种响应时，在前一种情况下，会显示一个带有链接的页面；在后一种情况下，可能会弹出对话窗口，让用户做选择。不管怎么样，决定是由客户端的浏览器用户作出的。

除了增加时延并且对每个页面都要进行繁琐的多次请求之外，这种方法还有一个缺点：它需要多个 URL：公共页面要一个，其他每种特殊页面也都要一个。因此，比如说原始的请求地址是 www.joes-hardware.com，Joe 的服务器可能会回复某个页面，该页面里面有到 www.joes-hardware.com/english 和 www.joes-hardware.com/french 的链接。如果客户端想加书签的话，是要加在原始的公共页面上呢，还是加在选中的页面上呢？如果用户想把这个网站推荐给他的朋友，是告知 www.joes-hardware.com 这个地址好呢，还是只告诉他们讲英语的朋友 www.joes-hardware.com/english 这个地址？

396

17.3 服务器驱动的协商

在前一节中，我们了解了客户端驱动的协商存在的若干缺点。大部分缺点都涉及客户端和服务器之间通信量的增长，这些通信量用来决定什么页面才是对请求的最佳响应。减少额外通信量的一种方法是让服务器来决定发送哪个页面回去，但为了做到这一点，客户端必须发送有关客户偏好的足够信息，以便服务器能够作出准确的决策。服务器通过客户端请求的首部集来获得这方面的信息。

有以下两种机制可供 HTTP 服务器评估发送什么响应给客户端比较合适。

- 检查内容协商首部集。服务器察看客户端发送的 `Accept` 首部集，设法用相应的响应首部与之匹配。
- 根据其他（非内容协商）首部进行变通。例如，服务器可以根据客户端发送的 `User-Agent` 首部来发送响应。

后面的小节将详细介绍这两种机制。

17.3.1 内容协商首部集

客户端可以用表 17-2 中列出的 HTTP 首部集发送用户的偏好信息。

表17-2 Accept首部集

首　部	描　述
Accept	告知服务器发送何种媒体类型
Accept-Language	告知服务器发送何种语言
Accept-Charset	告知服务器发送何种字符集
Accept-Encoding	告知服务器采用何种编码

注意，这些首部与第 15 章讨论的那些实体首部非常类似。不过，这两种首部的用途截然不同。正如第 15 章中所述，实体首部集像运输标签，它们描述了把报文从服务器传输给客户端的过程中必须的各种报文主体属性。而内容协商首部集是由客户端发送给服务器用于交换偏好信息的，以便服务器可以从文档的不同版本中选择出最符合客户端偏好的那个来提供服务。

397 服务器用表 17-3 中列出的实体首部集来匹配客户端的 Accept 首部集。

表17-3 Accept首部集和匹配的文档首部集

Accept首部	实体首部
Accept	Content-Type
Accept-Language	Content-Language
Accept-Charset	Content-Type
Accept-Encoding	Content-Encoding

注意，由于 HTTP 是无状态的协议（表示服务器不会在不同的请求之间追踪客户端的偏好），所以客户端必须在每个请求中都发送其偏好信息。

如果两个客户端都发送了 Accept-Language 首部，描述它们感兴趣的语言信息，服务器就能够决定发送 www.joes-hardware.com 的何种版本给哪个客户端了。让服务器自动选择发送回去的文档，减少了往返通信的时延，这种时延是客户端驱动模型中无法避免的。

然而，假设某个客户端偏好西班牙文，那服务器应当回送哪个版本的页面呢？英语还是法语？服务器只有两种选择：猜测或回退到客户端驱动模型，问客户端选择哪个。假如这个西班牙人碰巧懂一点英语，他可能会选择英文页面，这不是最理想的，

但它能解决问题。在这种情况下，这个西班牙人需要有办法传达更多与其偏好有关的信息，也就是他的确对英语略知一二，在没有西班牙语的时候，英语也行。

幸运的是，HTTP 提供了一种机制，可以让与这个西班牙人情况类似的客户端更详细地描述其偏好。这种机制就是*质量值*（简称 q 值）。

17.3.2 内容协商首部中的质量值

HTTP 协议中定义了质量值，允许客户端为每种偏好类别列出多种选项，并为每种偏好选项关联一个优先次序。例如，客户端可以发送下列形式的 Accept-Language 首部：

```
Accept-Language: en;q=0.5, fr;q=0.0, nl;q=1.0, tr;q=0.0
```

其中 q 值的范围从 0.0 ~ 1.0（0.0 是优先级最低的，而 1.0 是优先级最高的）。上面列出的那个首部，说明该客户端最愿意接收荷兰语（缩写为 nl）文档，但英语（缩写为 en）文档也行；无论如何，这个客户端都不愿意收到法语（缩写为 fr）或土耳其语（缩写为 tr）的版本。注意，偏好的排列顺序并不重要，只有与偏好相关的 q 值才是重要的。

服务器偶尔也会碰到找不到文档可以匹配客户端的任何偏好的情况。对于这种情况，服务器可以修改文档，也就是对文档进行转码，以匹配客户端的偏好。我们将在本章后面讨论这种机制。

398

17.3.3 随其他首部集而变化

服务器也可以根据其他客户端请求首部集来匹配响应，比如 User-Agent 首部。例如，服务器知道老版本的浏览器不支持 JavaScript 语言，这样就可以向其发送不含有 JavaScript 的页面版本。

在这种情况下，没有 q 值机制可供查找“最近似”的匹配。服务器或者去找完全匹配，或者简单地有什么就给什么，这取决于服务器的实现。

由于缓存需要尽力提供所缓存文档中正确的“最佳”版本，HTTP 协议定义了服务器在响应中发送的 Vary 首部。这个首部告知缓存（还有客户端和所有下游的代理）服务器根据哪些首部来决定发送响应的最佳版本。本章后面会更详细地讨论 Vary 首部。

17.3.4 Apache中的内容协商

这里概括了著名的 Web 服务器 Apache 是如何支持内容协商的。网站的内容提供者，

比如说 Joe，要负责为 Joe 的索引页面提供不同的版本。Joe 还必须把这些索引页面文件放在和站点相关的 Apache 服务器的适当目录下。用以下两种方式可以启用内容协商。

- 在网站目录中，为网站中每个有变体的 URI 创建一个 type-map（类型映射）文件。这个 type-map 文件列出了每个变体和其相关的内容协商首部集。
- 启用 MultiViews 指令，这样会使 Apache 自动为目录创建 type-map 文件。

1. 使用type-map文件

Apache 服务器需要知道 type-map 文件的命名规则。可以在服务器的配置文件中设置 handler 来说明 type-map 文件的后缀名。例如：

```
AddHandler type-map .var
```

这行就说明了后缀是 .var 的文件就是 type-map 文件。

399 这里给出一个 type-map 文件示例：

```
URI: joes-hardware.html

URI: joes-hardware.en.html
Content-type: text/html
Content-language: en
URI: joes-hardware.fr.de.html
Content-type: text/html;charset=iso-8859-2
Content-language: fr, de
```

根据这个 type-map 文件，Apache 服务器就知道要发送 joes-hardware.en.html 给请求英语版的客户端，发送 joes-hardware.fr.de.html 给请求法语版的客户端。Apache 服务器也支持质量值，具体信息请参阅它的文档。

2. 使用MultiView

为了使用 MultiView，必须在网站目录下的 access.conf 文件中的适当小节（<*Directory*>、<*Location*>，或 <*Files*>）使用 OPTION 指令来启用它。

如果启用了 MultiView，而浏览器又请求了名为 joes-hardware 的资源，服务器就会查找所有名字中含有 joes-hardware 的文件，并为它们创建 type-map 文件。服务器会根据名字猜测其对应的内容协商首部集。例如，法语版的 joes-hardware 应当含有 *.fr*。

17.3.5 服务器端扩展

另一种在服务器端实现内容协商的方法是使用服务器端扩展，比如微软的动态服务

器页面（Microsoft's Active Server Pages，ASP)。参见第 8 章中关于服务器端扩展的综述。

17.4 透明协商

透明协商机制试图从服务器上去除服务器驱动协商所需的负载，并用中间代理来代表客户端以使与客户端的报文交换最小化。假定代理了解客户端的预期，这样就可以代表客户端与服务器协商（在客户端请求内容的时候，代理已经收到了客户端的预期)。为了支持透明内容协商，服务器必须有能力告知代理，服务器需要检查哪些请求首部，以便对客户端的请求进行最佳匹配。HTTP/1.1 规范中没有定义任何透明协商机制，但定义了 `Vary` 首部。服务器在响应中发送了 `Vary` 首部，以告知中间节点需要使用哪些请求首部进行内容协商。

代理缓存可以为通过单个 URL 访问的文档保存不同的副本。如果服务器把它们的决策过程传给缓存，这些代理就能代表服务器与客户端进行协商。缓存同时也是进行内容转码的好地方，因为部署在缓存里的通用转码器能对任意服务器，而不仅仅是一台服务器传来的内容进行转码。图 17-3 中展示了缓存对内容进行转码的情况，本章后面会更详细地探讨。

400

17.4.1 进行缓存与备用候选

对内容进行缓存的时候是假设内容以后还可以重用。然而，为了确保对客户端请求回送的是正确的已缓存响应，缓存必须应用服务器在回送响应时所用到的大部分决策逻辑。

前一节描述了客户端发送的 `Accept` 首部集，以及为了给每条请求选择最佳的响应，服务器使用的与这些首部集匹配的相应实体首部集。缓存也必须使用相同的首部集来决定回送哪个已缓存的响应。

图 17-1 展示了涉及缓存的正确及错误的操作序列。缓存把第一个请求转发给服务器，并存储其响应。对于第二个请求，缓存根据 URL 查找到了匹配的文档。但是，这份文档是法语版的，而请求者想要的是西班牙语版的。如果缓存只是把文档的法语版本发给请求者的话，它就犯了错误。

因此，缓存也应该把第二条请求转发给服务器，并保存该 URL 的响应与“备用候选”响应。缓存现在就保存了同一个 URL 的两份不同的文档，与服务器上一样。这些不同的版本称为*变体*（variant）或*备用候选*（alternate)。内容协商可看成是为客户端请求选择最合适变体的过程。

401

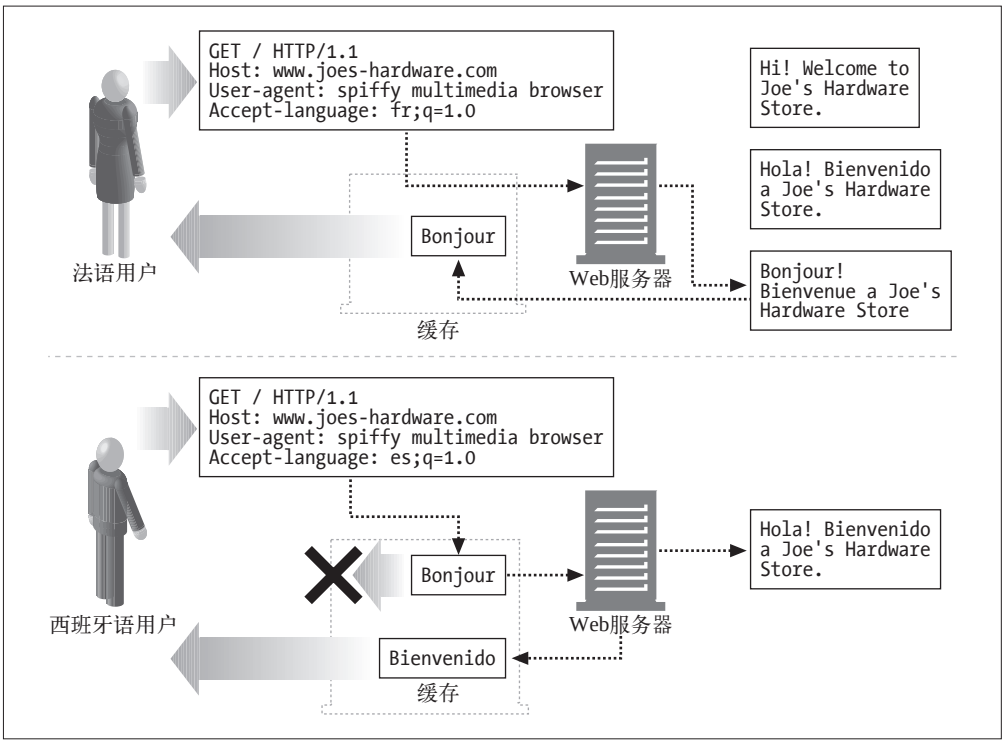


图 17-1　缓存根据内容协商首部发送给客户端正确的响应

17.4.2　Vary首部

这里是浏览器和服务器发送的一些典型的请求及响应首部：

```
GET http://www.joes-hardware.com/ HTTP/1.0
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.73 [en] (WinNT; U)
Host: www.joes-hardware.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/
    png, */*
Accept-Encoding: gzip
Accept-Language: en, pdf
Accept-Charset: iso-8859-1, *, utf-8

HTTP/1.1 200 OK
Date: Sun, 10 Dec 2000 22:13:40 GMT
Server: Apache/1.3.12 OpenSSL/0.9.5a (Unix) FrontPage/4.0.4.3
Last-Modified: Fri, 05 May 2000 04:42:52 GMT
Etag: "1b7ddf-48-3912514c"
Accept-Ranges: Bytes
Content-Length: 72
Connection: close
Content-Type: text/html
```

然而，如果服务器的决策不是依据 Accept 首部集，而是比如 User-Agent 首部的话，情况会如何？这不像听起来这么极端。例如，服务器可能知道老版本的浏览器不支持 JavaScript 语言，因此可能会回送不包含 JavaScript 的页面版本。如果服务器是根据其他首部来决定发送哪个页面的话，缓存必须知道这些首部是什么，这样才能在选择回送的页面时做出同样的逻辑判断。

HTTP 的 Vary 响应首部中列出了所有客户端请求首部，服务器可用这些首部来选择文档或产生定制的内容（在常规的内容协商首部集之外的内容）。例如，若所提供的文档取决于 User-Agent 首部，Vary 首部就必须包含 User-Agent。

当新的请求到达时，缓存会根据内容协商首部集来寻找最佳匹配。但在把文档提供给客户端之前，它必须检查服务器有没有在已缓存响应中发送 Vary 首部。如果有 Vary 首部，那么新请求中那些首部的值必须与旧的已缓存请求里相应的首部相同。因为服务器可能会根据客户端请求的首部来改变响应，为了实现透明协商，缓存必须为每个已缓存变体保存客户端请求首部和相应的服务器响应首部，参见图 17-2。 402

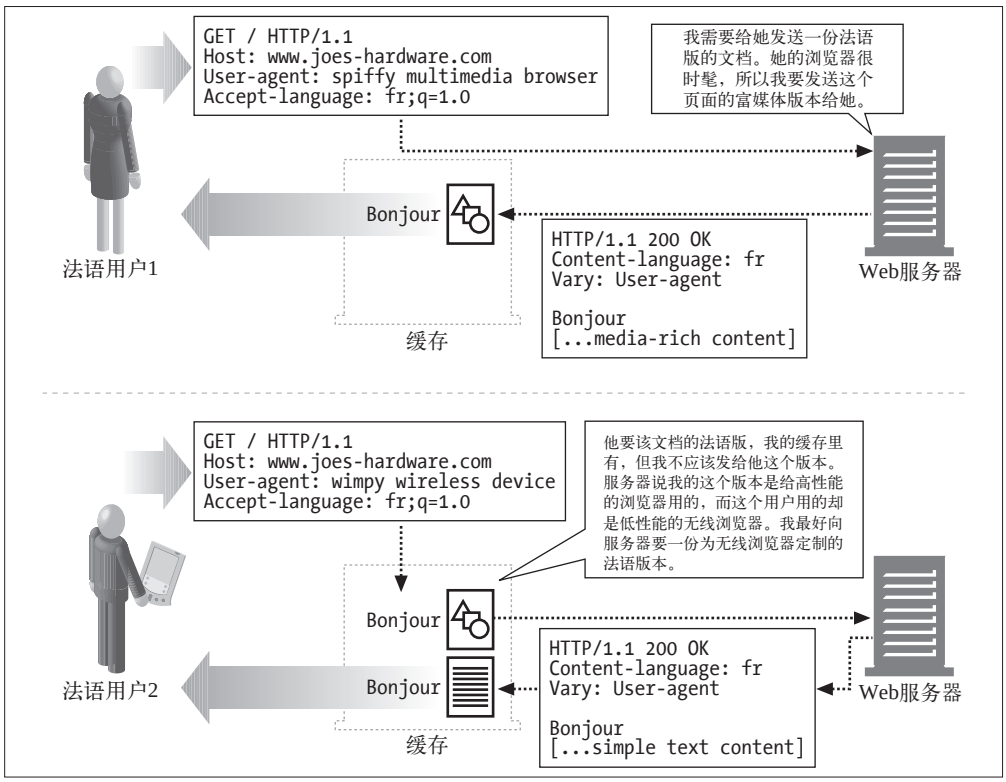


图 17-2　如果服务器根据特定的请求首部集来选择变体，缓存必须在发送回缓存的响应之前，检查常规的内容协商首部集和这些请求首部

如果某服务器的 Vary 首部看起来像下面这样，大量不同的 User-Agent 和 Cookie 值将会产生非常多的变体：

```
Vary: User-Agent, Cookie
```

缓存必须为每个变体保存其相应的文档版本。当缓存执行查找时，首先会对内容协商首部集进行内容匹配，然后比较请求的变体与缓存的变体。如果无法匹配，缓存就从原始服务器获取文档。

17.5 转码

我们已经讨论了一个机制，该机制可以让客户端和服务器从某个 URL 的一系列文档
403 中挑选出最适合客户端的文档。实现这些机制的前提是，存在一些满足客户端需求的文档——不管是完全满足还是在一定程度上满足。

然而，如果服务器没有能满足客户端需求的文档会怎么样呢？服务器可以给出一个错误响应。但理论上，服务器可以把现存的文档转换成某种客户端可用的文档。这种选项称为*转码*。

表 17-4 列出了一些假设的转码。

表17-4 假设的转码

转换之前	转换之后
HTML 文档	WML 文档
高分辨率图像	低分辨率图像
彩色图像	黑白图像
有多个框架的复杂页面	没有很多框架或图像的简单文本页面
有 Java 小应用程序的 HTML 页面	没有 Java 小应用程序的 HTML 页面
有广告的页面	去除广告的页面

有 3 种类别的转码：格式转换、信息综合以及内容注入。

17.5.1 格式转换

*格式转换*是指将数据从一种格式转换成另一种格式，使之可以被客户端查看。通过 HTML 到 WML 的转换，无线设备就可以访问通常供桌面客户端查看的文档了。通过慢速连接访问 Web 页面的客户端并不需要接收高分辨率图像，如果通过格式转换降低图像分辨率和颜色来减小图像文件大小的话，这类客户端就能更容易地查看图像比较丰富的页面了。

格式转换可以由表 17-2 中列出的内容协商首部集来驱动，但也能由 User-Agent 首部来驱动。注意，内容转换或转码与内容编码或传输编码是不同的，后两者一般用于更高效或安全地传输内容，而前两者则可使访问设备能够查看内容。

17.5.2 信息综合

从文档中提取关键的信息片段称为*信息综合*（information synthesis），这是一种有用的转码操作。这种操作的例子包括根据小节标题生成文档的大纲，或者从页面中删除广告和商标。

404

根据内容中的关键字对页面分类是更精细的技术，有助于总结文档的精髓。这种技术常用于 Web 页面分类系统中，比如门户网站的 Web 页面目录。

17.5.3 内容注入

前面描述的两类转码通常会减少 Web 文档的内容，但还有另一类转换会增加文档的内容，即*内容注入*转码。内容注入转码的例子有自动广告生成器和用户追踪系统。

设想一下，一个能往途经的每个 HTML 页面中自动添加广告的广告植入转码器是多么的诱人（当然也很烦人）。这类转码操作只能动态进行——它必须即时添加与当前的特定用户有关，或针对特定用户的广告。也可以构建用户追踪系统，在页面中动态增加内容，用于收集用户查看页面和客户端浏览方式的统计信息。

17.5.4 转码与静态预生成的对比

转码的替代做法是在 Web 服务器上建立 Web 页面的不同副本，例如一个是 HTML，一个是 WML；一个图像分辨率高，一个图像分辨率低；一个有多媒体内容，一个没有。但是，这种方法不是很切合实际，原因很多：某个页面中的任何小改动都会牵扯很多页面，需要很多空间来存储各页面的不同版本，而且使页面编目和 Web 服务器编程（以提供正确的版本）变得更加困难。有些转码操作，比如广告插入（尤其是定向广告插入），就不能静态实现——因为插入什么广告和请求页面的用户有关。

对单一的根页面进行即时转换，是比静态的预生成更容易的解决方案。但这样会在提供内容时增加时延。不过有时候其中一些计算可以由第三方进行，这样就减少了 Web 服务器上的计算负荷——比如可以由代理或缓存中的外部 Agent 完成转换。图 17-3 显示了在代理缓存中进行的转码。

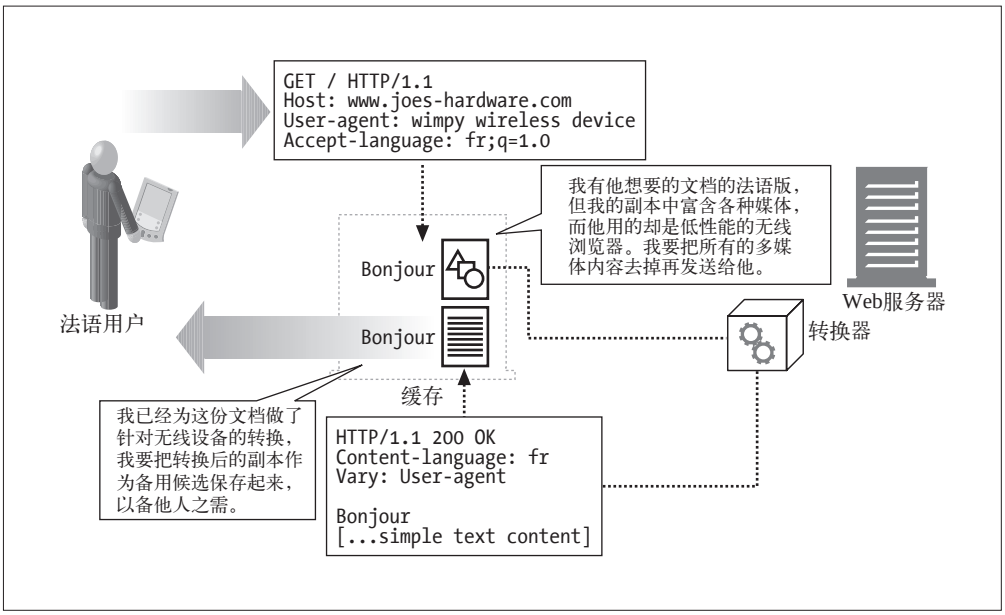


图 17-3　在代理缓存中进行转换或转码

17.6　下一步计划

由于以下两个原因，内容协商这个话题不只限于 Accept 和 Content 这两个首部集。

- HTTP 中的内容协商受到一些性能方面的限制。在各种变体中搜索合适的内容，或尽力“猜测”最佳匹配，都会有很大开销。有没有什么办法能专注内容协商协议以使这个过程更高效？ RFC 2295 和 RFC2296 尝试着对这个问题进行了研究，以提供透明的 HTTP 内容协商。
- HTTP 不是唯一需要进行内容协商的协议。在其他一些情况下，客户端也需要和服务器交互以便获得对客户端请求来说最好的答案，流媒体和传真就是另外两个例子。能否在 TCP/IP 应用层协议之上开发出通用的内容协商协议呢？内容协商工作组（Content Negotiation Working Group）就是专门为这个问题而成立的。这个工作组目前已经停止工作了，不过它提出了若干个 RFC。在下一节中，我们给出了这个组的网站链接。

405

17.7　更多信息

从下面的因特网草案和在线文档中可以获得更多内容协商方面的信息。

- http://www.ietf.org/rfc/rfc2616.txt
  RFC 2616，“Hypertext Transfer Protocol–HTTP/1.1”（“超文本传输协议 HTTP/1.1”），这是 HTTP 协议的当前版本，也是 HTTP/1.1 的官方规范。这份规范行文流畅、组织良好，是份详实的 HTTP 参考文献。不过对那些希望学习 HTTP 背后的各种概念和决策动机、弄清理论与实践不同之处的读者来说，它就不是很理想了。我们希望本书能补足背后的这些概念，使读者能更好地利用这份规范。 406

- http://www.ietf.org/rfc/rfc2295.txt
  RFC 2295，“Transparent Content Negotiation in HTTP”（“HTTP 中的透明内容协商”），这是一份备忘录，描述了建立在 HTTP 之上的透明内容协商协议。这份备忘录目前还是实验性的。

- http://www.ietf.org/rfc/rfc2296.txt
  RFC 2296，“HTTP Remote Variant Selection Algorithm RVSA 1.0”（“HTTP 远程变体选择算法 RVSA1.0”），这份备忘录描述了为特定的 HTTP 请求透明地选择“最佳”内容的算法。这份备忘录目前还是实验性的。

- http://www.ietf.org/rfc/rfc2936.txt
  RFC 2936，“HTTP MIME Type Handler Detection”（“HTTP MIME 类型处理器检测”），这份备忘录描述了一种用来判定浏览器支持的 MIME 类型处理器的方法。如果 Accept 首部不够明确的话，这种方法就能派上用场。

- http://www.imc.org/ietf-medfree/index.htm
  这个链接指向内容协商（简称 CONNEG）工作组网站。该工作组专注于 HTTP、传真和打印方面的透明内容协商。这个工作组目前已停止工作。 407

第五部分

内容发布与分发

第五部分讲述了 Web 内容发布和传播的各种技术。

- 第 18 章介绍了在现代的 Web 托管环境中部署服务器的若干方法，HTTP 对虚拟 Web 托管的支持以及如何在地理上相距遥远的服务器之间复制内容。
- 第 19 章讨论了创建 Web 内容并将其放置到 Web 服务器上去的各种技术。
- 第 20 章探讨了各种将来访的 Web 流量分发到一组服务器上的技术和工具。
- 第 21 章解释了日志的各种格式和各种常见问题。

408
~
410

第18章

Web主机托管

当你把资源放在公共的 Web 服务器上时，因特网社区就可以使用它们了。这些资源可以是简单的文本文件或图像，也可以是复杂的实时导航地图或电子商务购物网关。能够将这些由不同组织拥有的种类繁多的资源便利地发布到网站上，并将其放置在能以合理价格提供很好性能的 Web 服务器上，是很关键的。

对内容资源的存储、协调以及管理的职责统称为 Web *主机托管*。主机托管是 Web 服务器的主要功能之一。保存并提供内容，记录对内容的访问以及管理内容都离不开服务器。如果不想自行管理服务器所需的软硬件，就需要主机托管服务，即*托管者*。托管者出租服务和网站管理维护业务，并提供各种不同程度的安全级别、报告及易用性。托管者通常把很多网站放在一些强大的 Web 服务器上联合运行，这样可以获得更高的成本效益、可靠性和性能。

本章讲解 Web 主机托管服务中的某些重要特征和它们如何与 HTTP 应用程序交互。本章的主要内容包括：

- 不同的网站如何被“虚拟地托管”在同一个服务器上，这样会对 HTTP 产生怎样的影响；
- 在很大的流量压力下，如何确保网站更可靠；
- 如何使网站加载更快。

18.1 主机托管服务

在万维网的早期，每个组织自行购买自己的计算机硬件，搭建自己的计算机房，申请自己的网络连接，并管理自己的 Web 服务器软件。

随着 Web 迅速成为主流，每人都想要一个网站，但很少有人有能力或时间来搭建带
411 空调的服务器机房，注册域名，或购买网络带宽。为了满足人们的迫切需求，出现了很多新的企业，提供了专业化管理的 Web 主机托管服务。服务级别有多种，从物理上的设备管理（提供空间、空调以及线缆）到完整的 Web 主机托管，顾客只需要提供内容就行了。

本章主要探讨托管 Web 服务器要提供什么服务。网站运作需要的很多东西（例如，它支持不同语言的能力和进行安全的电子商务交易的能力）都取决于托管 Web 服务器提供的功能。

简单例子——专用托管

假设 Joe 的五金商店和 Mary 的古董拍卖店都需要大容量的网站。Irene 网络服务提供商那里有很多机架，机架上全是一样的高性能 Web 服务器，可以租给 Joe 和 Mary，

这样，他俩就不用自行购买自己的服务器并管理服务器软件了。

在图 18-1 中，Joe 和 Mary 都签约使用 Irene 的网络服务提供商提供的专用 Web 托管服务。Joe 租了专用的 Web 服务器，该服务器是 Irene 网络服务提供商购买和维护的。Mary 也从 Irene 网络服务提供商那里租了另一个专用服务器。Irene 网络服务提供商大批量地购买服务器硬件，它们选择的硬件经久耐用且相对便宜。如果 Joe 或 Mary 的网站变得更受欢迎，Irene 网络服务提供商可以立刻给 Joe 或 Mary 提供更多的服务器。

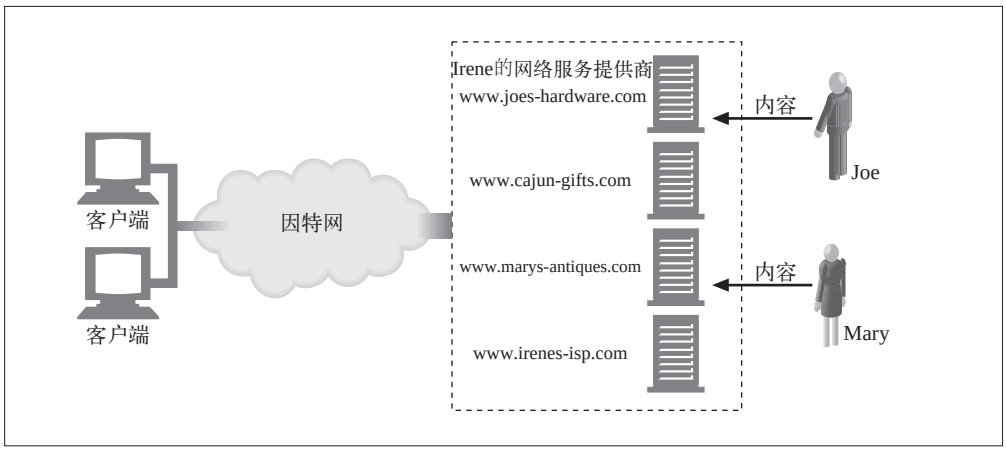


图 18-1　外包的专用托管服务

在这个例子中，浏览器向 Joe 服务器的 IP 地址发送对 www.joes-hardware.com 的 HTTP 请求，向 Mary 服务器（不同于 Joe）的 IP 地址发送对 www.marys-antiques.com 的请求。

412

18.2　虚拟主机托管

许多人想要在 Web 上展现自己，但他们的网站流量都不大。对这些人来说，使用专用的 Web 服务器可能有点儿大材小用，因为他们每月花费数百美元租来的服务器大部分时间都是空闲的！

许多 Web 托管者通过让一些顾客共享一台计算机来提供便宜的 Web 主机托管服务。这称为*共享主机托管*或*虚拟主机托管*。每个网站看起来是托管在不同的服务器上，但实际上是托管在同一个物理服务器上。从最终用户的角度来看，被虚拟托管的网站应当和托管在专用服务器上的网站没什么区别。

从成本效益、空间以及管理方面考虑，提供虚拟主机托管的公司希望能在同一个

服务器上托管数十、上百，甚至上千个网站——但这不一定意味着上千个网站是用一台 PC 机来提供服务的。托管者可以创建成排同样的服务器，称为*服务器集群*（server farm），把负载分摊在群里的服务器上。因为群里的每台服务器都一样，并且托管了许多虚拟网站，所以管理起来更加方便。（我们将在第 20 章更详细地介绍服务器集群。）

当 Joe 和 Mary 刚开始商务运作时，他们可能会选择虚拟主机托管，以节省费用，直到他们网站的流量规模达到值得使用专用服务器的水平为止（参见图 18-2）。

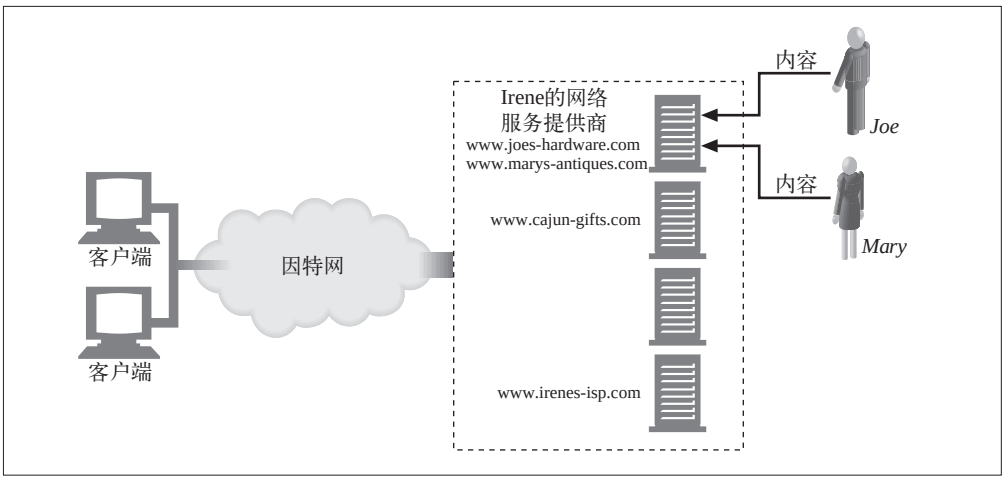


图 18-2　外包的虚拟主机托管

18.2.1　虚拟服务器请求缺乏主机信息

不幸的是，HTTP/1.0 中的一个设计缺陷会使虚拟主机托管者抓狂。HTTP/1.0 规范中
413 没有为共享的 Web 服务器提供任何方法来识别要访问的是所托管的哪个虚拟网站。

回想一下，HTTP/1.0 请求在报文中只发送了 URL 的路径部分。如果要访问 http://www.joes-hardware.com/index.html，浏览器会连接到服务器 www.joes-hardware.com，但 HTTP/1.0 请求中只提到 GET /index.html，没有提到主机名。如果服务器虚拟托管了多个站点，就没有足够的信息能指出要访问的是哪个虚拟网站。图 18-3 就是这样的一个示例。

- 如果客户端 A 试图访问 http://www.joes-hardware.com/index.html，请求 GET /index.html 将被发送到共享的 Web 服务器。
- 如果客户端 B 试图访问 http://www.marys-antiques.com/index.html，同样的请求 GET /index.html 也将被发送到共享的 Web 服务器。

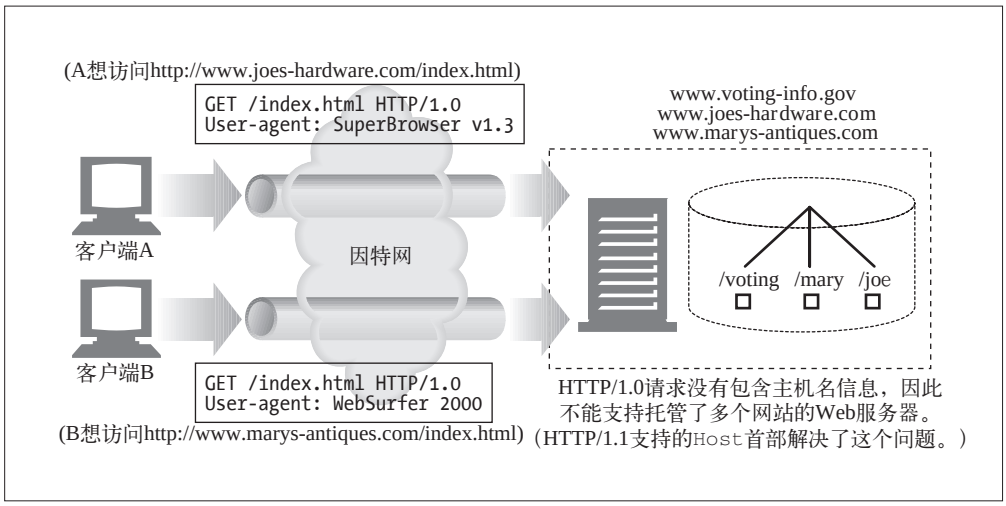


图 18-3　HTTP/1.0 服务器请求中没有主机名信息

就 Web 服务器而言，没有足够的信息可供其判断究竟要访问的是哪个网站。尽管请求的是完全不同的文档（来自不同的网站），但这两个请求看起来是一样的，这是因为网站的主机信息已经从请求中剥离了。

我们已经在第 6 章中介绍过，HTTP 替代物（反向代理）和拦截代理也都需要明确的站点信息。

18.2.2　设法让虚拟主机托管正常工作

缺失的主机信息是原始 HTTP 规范的疏忽，它错误地假设了每个 Web 服务器上只托管了一个网站。HTTP 的设计者没有为进行虚拟主机托管的共享服务器提供支持。正因为如此，URL 中的主机名信息被当作冗余信息剥离了，只要求发送路径部分。

因为早期的规范没有考虑到虚拟主机托管，Web 托管者需要开发变通的方案和约定来支持共享的虚拟主机托管。这个问题本可以通过要求所有 HTTP 请求报文发送完整的 URL 而不只是路径部分来简单地解决。而 HTTP/1.1 的确要求服务器能够处理 HTTP 报文请求行上的完整 URL，但将现存的应用程序都升级到这个规范还需要很长时间。在此期间，涌现了以下 4 种技术。

414

- 通过 URL 路径进行虚拟主机托管
  在 URL 中增添专门的路径部分，以便服务器判断是哪个网站。

- 通过端口号进行主机托管
  为每个站点分配不同的端口号，这样请求就由 Web 服务器的单独实例来处理。

- 通过 IP 地址进行主机托管
  为不同的虚拟站点分配专门的 IP 地址，把这些地址都绑定到一台单独的机器上。这样，Web 服务器就可以通过 IP 地址来识别网站名了。

- 通过 Host 首部进行主机托管
  很多 Web 托管者向 HTTP 的设计者施压，要求解决这个问题。HTTP/1.0 的增强版和 HTTP/1.1 的正式版定义了 Host 请求首部来携带网站名称。Web 服务器可以通过 Host 首部识别虚拟站点。

接下来详细介绍每种技术。

1. 通过URL路径进行虚拟主机托管

可以通过分配不同的 URL 路径，用这种笨方法把共享服务器上的虚拟站点隔离开。例如，可以给每个逻辑网站一个专门的路径前缀。

- Joe 的五金商店可以是：http://www.joes-hardware.com/joe/index.html。
- Mary 的古董拍卖店可以是：http://www.marys-antiques.com/mary/index.html。

当请求到达服务器时，其中并没有主机名信息，但服务器可以通过路径来区分它们。

- 请求 Joe 的五金商店的网址是 GET /joe/index.html。
- 请求 Mary 的古董拍卖店的网址是 GET /mary/index.html。

这不是一个好办法。/joe 和 /mary 这样的前缀是多余的（主机名中已经提到 joe 了）。更糟的是，描述主页链接的常见约定：http://www.joes-hardware.com 或 http://www.joes-hardware.com/index.html 都不能用了。

总之，按 URL 来进行虚拟主机托管是一个糟糕的解决方案，很少会用到。

2. 通过端口号进行虚拟主机托管

除了修改路径名，还可以在 Web 服务器上为 Joe 和 Mary 的网站分配不同的端口号。
415 不再使用端口 80，而是采用其他端口号，例如，Joe 用 82 Mary，用 83。但这个解决方案也有同样的问题：终端用户不会乐意在 URL 中指定非标准的端口号。

3. 通过IP地址进行虚拟主机托管

一个更常用的、更好的方法是通过 IP 地址进行虚拟化。每个虚拟网站都分配一个或多个唯一的 IP 地址。所有虚拟网站的 IP 地址都绑定到同一个共享的服务器上。服务器可以查询 HTTP 连接的目的 IP 地址，并以此来判断客户端的目标网站。

比方说，托管者把 IP 地址 209.172.34.3 分配给 www.joes-hardware.com，把 IP 地址 209.172.34.4 分配给 www.marys-antiques.com，把这两个 IP 地址都绑定到同一个物理服务器上。Web 服务器就能使用目的 IP 地址来识别用户请求的是哪个虚拟站点了，参见图 18-4。

- 客户端 A 获取 http://www.joes-hardware.com/index.html。
- 客户端 A 查询 www.joes-hardware.com 的 IP 地址，得到 209.172.34.3。
- 客户端 A 打开到共享服务器的 TCP 连接，目的地址是 209.172.34.3。
- 客户端 A 发送请求，内容为 GET /index.html HTTP/1.0。
- 在 Web 服务器提供响应之前，它注意到实际的目的 IP 地址（209.172.34.3），判断出这是 Joe 的五金网站的虚拟 IP 地址，就根据子目录 /joe 来完成请求。返回的是文件 /joe/index.html。

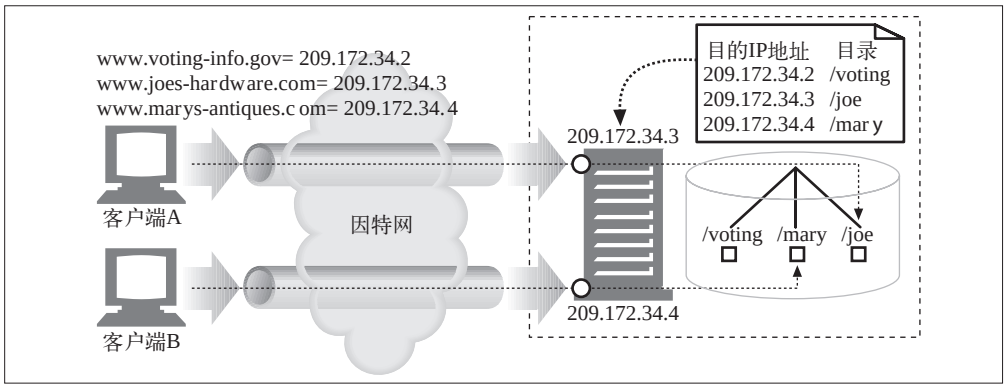


图 18-4　虚拟 IP 的主机托管

类似地，如果客户端 B 请求 http://www.marys-antiques.com/index.html。

- 客户端 B 查询 www.marys-antiques.com 的 IP 地址，得到 209.172.34.4。
- 客户端 B 打开到 Web 服务器的 TCP 连接，目的地址是 209.172.34.4。
- 客户端 B 发送请求，内容是 GET /index.html HTTP/1.0。
- Web 服务器判断出 209.172.34.4 是 Mary 的网站，根据 /mary 目录来完成请求，返回的是文件 /mary/index.html。

416

对大的托管者来说，虚拟 IP 的主机托管能够工作，但它会带来一些麻烦。

- 在计算机系统上能绑定的虚拟 IP 地址通常是有限制的。想在共享的服务器上托管成百上千的虚拟站点的服务商不一定能实现愿望。
- IP 地址是稀缺资源。有很多虚拟站点的托管者不一定能为被托管的网站获取足够多的 IP 地址。

- 托管者通过复制服务器来增加容量时，IP 地址短缺的问题就更严重了。随负载均衡体系的不同，可能会要求每个复制的服务器上有不同的虚拟 IP 地址，因此 IP 地址的需求量可能会随复制服务器的数量而倍增。

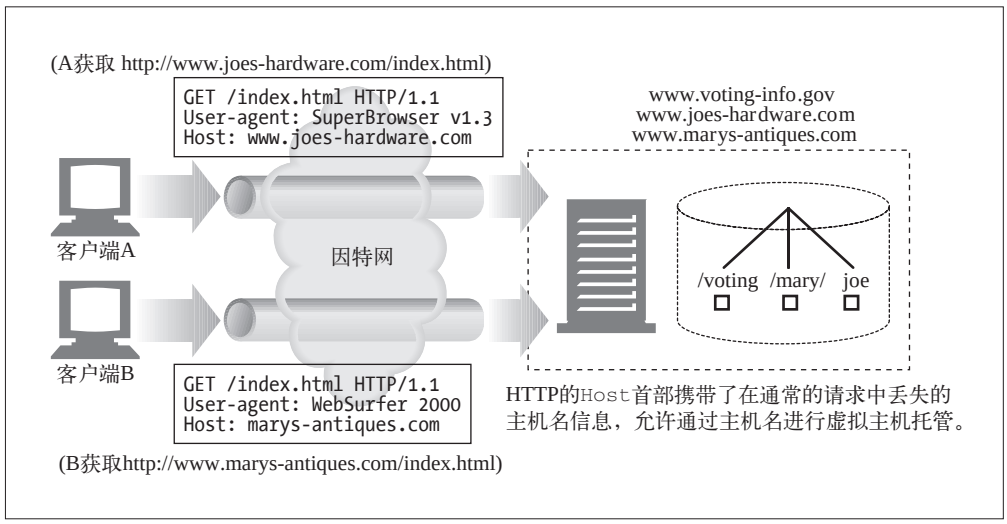
尽管虚拟 IP 的主机托管存在消耗地址的问题，但它仍然得到了广泛的运用。

4. 通过Host首部进行虚拟主机托管

为了避免过度的地址消耗和虚拟 IP 地址的限制，我们希望在虚拟站点间共享同一个 IP 地址，且仍能区分站点。但正如我们看到的那样，因为大多数浏览器只是把 URL 的路径发给服务器，关键的虚拟主机名信息被其丢掉了。

为了解决这个问题，浏览器和服务器的实现者扩展了 HTTP，把原始的主机名提供给服务器。不过，浏览器不能只发送完整的 URL，因为这会使许多只能接收路径的服务器无法工作。替代的方法是，把主机名（和端口号）放在所有请求的 Host 扩展首部中传送。

在图 18-5 中，客户端 A 和客户端 B 都发送了携带有要访问的原始主机名的 Host 首部。当服务器收到对 /index.html 的请求时，可以通过 Host 首部来判断要使用哪个资源。



417 图 18-5 用 Host 首部区分请求的虚拟主机

Host 首部最早是在 HTTP/1.0+ 中引入的，它是开发商实现的 HTTP/1.0 的扩展超集。遵循 HTTP/1.1 标准则必须支持 Host 首部。绝大多数现代浏览器和服务器都支持 Host 首部，但仍有一些客户端和服务器（以及网络机器人）不支持它。

18.2.3 HTTP/1.1的Host首部

Host 首部是 HTTP/1.1 的请求首部，定义在 RFC 2068 中。由于虚拟服务器的流行，绝大多数 HTTP 客户端（即使是不遵循 HTTP/1.1 的客户端），都实现了 Host 首部。

1. 语法与用法

Host 首部描述了所请求的资源所在的因特网主机和端口号，和原始的 URL 中得到的一样：

```
Host = "Host" ":" host [ ":" port ]
```

但要注意以下问题。

- 如果 Host 首部不包含端口，就使用地址方案中默认的端口。
- 如果 URL 中包含 IP 地址，Host 首部就应当包含同样的地址。
- 如果 URL 中包含主机名，Host 首部就必须包含同样的名字。
- 如果 URL 中包含主机名，Host 首部就不应当包含 URL 中这个主机名对应的 IP 地址，因为这样会扰乱虚拟主机托管服务器的工作，它在同一个 IP 地址上堆叠了很多虚拟站点。
- 如果 URL 中包含主机名，Host 首部就不应当包含这个主机名的其他别名，因为这样也会扰乱虚拟主机托管服务器的工作。
- 如果客户端显式地使用代理服务器，客户端就必须把*原始*服务器，而不是代理服务器的名字和端口放在 Host 首部中。以往，若干个 Web 客户端在启用客户端代理设置时，错误地把发出的 Host 首部设置成代理的主机名。这种错误行为会使代理和原始服务器都无法正常处理请求。
- Web 客户端必须在所有请求报文中包含 Host 首部。
- Web 代理必须在转发请求报文之前，添加 Host 首部。
- HTTP/1.1 的 Web 服务器必须用 400 状态码来响应所有缺少 Host 首部字段的 HTTP/1.1 请求报文。

下面是一段简单的 HTTP 请求报文，用于获取 www.joes-hardware.com 的主页，其中带有必需的 Host 首部字段：

418

```
GET http://www.joes-hardware.com/index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.51 [en] (X11; U; IRIX 6.2 IP22)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/
png, */*
Accept-Encoding: gzip
Accept-Language: en
Host: www.joes-hardware.com
```

2. 缺失Host首部

有少量在用的老式浏览器不会发送 Host 首部。如果某个虚拟主机托管服务器使用 Host 首部来判断所服务的是哪个网站，而报文中没有出现 Host 首部的话，那它可能会把用户导向某个默认的 Web 页面（例如网络服务提供商的 Web 页面），也可能返回一个错误页面建议用户升级浏览器。

3. 解释Host首部

对于没有进行虚拟主机托管，而且不允许资源随请求主机的不同而变化的原始服务器来说，可以忽略 Host 首部字段的值。但资源会随主机名的不同而变化的原始服务器，都必须在一条 HTTP/1.1 请求判断其所请求的资源时使用下列规则。

（1）如果 HTTP 请求报文中的 URL 是绝对的（也就是说，包含方案和主机部分），就忽略 Host 首部的值。
（2）如果 HTTP 请求报文中的 URL 没有主机部分，而该请求带有 Host 首部，则主机 / 端口的值就从 Host 首部中取。
（3）如果通过第（1）步或第（2）步都无法获得有效的主机，就向客户端返回 400 Bad Request 响应。

4. Host首部与代理

某些版本的浏览器发送的 Host 首部不正确，尤其是配置使用代理的时候。例如，配置使用代理时，一些老版本的 Apple 和 PointCast 客户端会错误地把代理的名字，而不是原始服务器的名字放在 Host 首部里发送。

18.3 使网站更可靠

在下面列出的这些时间段内，网站通常是无法运作的。

- 服务器宕机的时候。
- 交通拥堵：突然间很多人都要看某个特别的新闻广播或涌向某个大甩卖网店。突然的拥堵可以使 Web 服务器过载，降低其响应速度，甚至使它彻底停机。
- 网络中断或掉线。

419 本节会展示一些预判和处理这些常见问题的方法。

18.3.1 镜像的服务器集群

服务器集群是一排配置相同的 Web 服务器，互相可以替换。每个服务器上的内容可以通过镜像复制，这样当某个服务器出问题的时候，其他的可以顶上。

镜像的服务器常常组成层次化的关系。某个服务器可能充当“内容权威”——它含有原始内容（可能就是内容作者上传的那个服务器）。这个服务器称为*主原始服务器*（master origin server）。从主原始服务器接收内容的镜像服务器称为*复制原始服务器*（replica origin server）。一种简单的部署服务器集群的方法是用网络交换机把请求分发给服务器。托管在服务器上的每个网站的 IP 地址就设置为交换机的 IP 地址。

在图 18-6 显示的镜像服务器集群中，主原始服务器负责把内容发送给复制原始服务器。对集群外部来说，内容所在的 IP 地址就是交换机的 IP 地址。交换机负责把请求发送到服务器上去。

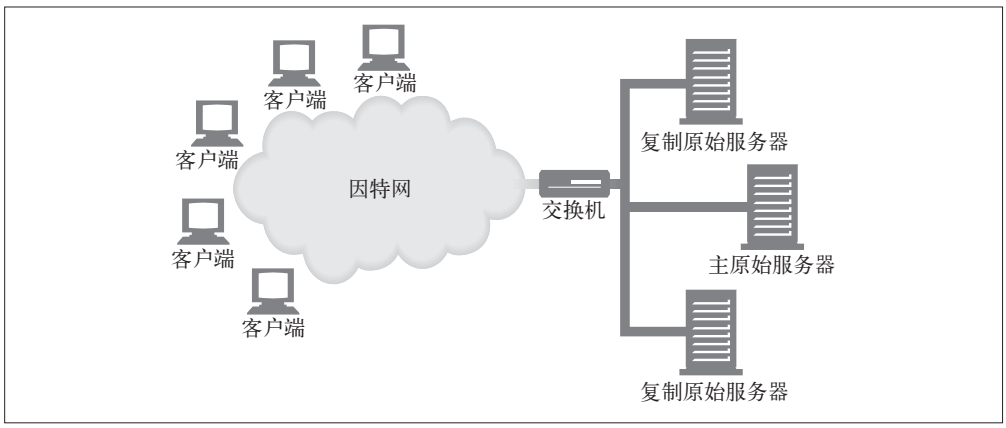


图 18-6　镜像的服务器集群

镜像 Web 服务器可以在不同的地点包含同样内容的副本。图 18-7 展示了 4 个镜像服务器，其中主服务器在芝加哥，复制服务器在纽约、迈阿密和小石城。主服务器为芝加哥地区的客户端服务，并肩负把内容传播给复制服务器的任务。

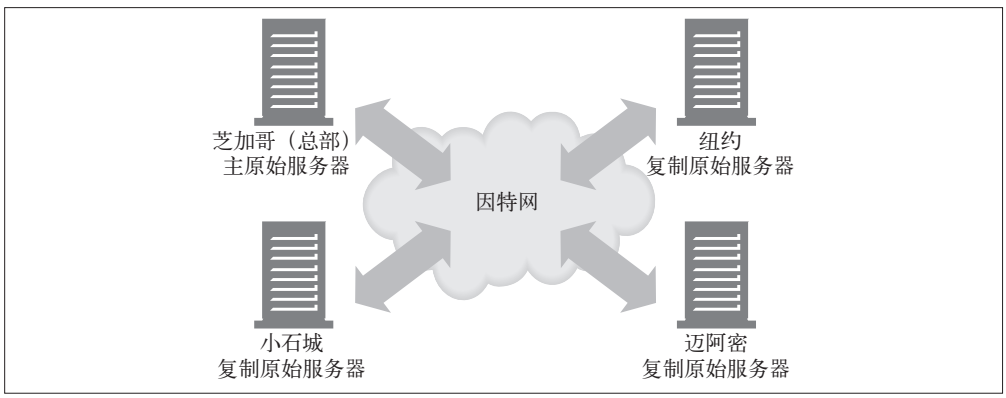


图 18-7　分散的镜像服务器

在图 18-7 的场景中，有以下两种方法把客户端的请求导向特定的服务器。

- HTTP 重定向

420 该内容的 URL 会解析到主服务器的 IP 地址，然后它会发送重定向到复制服务器。

- DNS 重定向

  该内容的 URL 会解析到 4 个 IP 地址，DNS 服务器可以选择发送给客户端的 IP 地址。

请参见第 20 章，以获取详细信息。

18.3.2 内容分发网络

简单地说，内容分发网络（CDN）就是对特定内容进行分发的专门网络。这个网络中的节点可以是 Web 服务器、反向代理或缓存。

18.3.3 CDN中的反向代理缓存

在图 18-6 和图 18-7 中，复制原始服务器可以用反向代理（也称为替代物）缓存来代替。反向代理缓存可以像镜像服务器一样接受服务器请求。它们代表原始服务器中的一个特定集合来接收服务器请求。（根据内容所在的 IP 地址的广告方式，这是有可能的，原始服务器和反向代理缓存之间通常有协作关系，到特定的原始服务器的请求就由反向代理缓存来接收。）

反向代理和镜像服务器之间的区别在于反向代理通常是需求驱动的。它们不会保存原始服务器的全部内容副本，它们只保存客户端请求的那部分内容。内容在其高速缓存中的分布情况取决于它们收到的请求，原始服务器不负责更新它们的内容。为了更容易地访问“热点”内容（就是高请求率的内容），有些反向代理具有“预取”特性，可以在用户请求之前就从服务器上载入内容。

421 CDN 中带有反向代理时，可能会由于存在代理的层次关系而增加其复杂性。

18.3.4 CDN中的代理缓存

代理缓存也可以部署在类似图 18-6 和图 18-7 的环境中。与反向代理不同，传统的代理缓存能收到发往任何 Web 服务器的请求。（在代理缓存与原始服务器之间不需要有任何工作关系或 IP 地址约定。）但是与反向代理比起来，代理缓存的内容一般都是按需驱动的，不能指望它是对原始服务器内容的精确复制。某些代理缓存也可以预先载入热点内容。

按需驱动的代理缓存可以部署在其他环境中——尤其是拦截环境，在这种情况下，2 层或 3 层设备（交换机或路由器）会拦截 Web 流量并将其发送给代理缓存（参见图 18-8）。

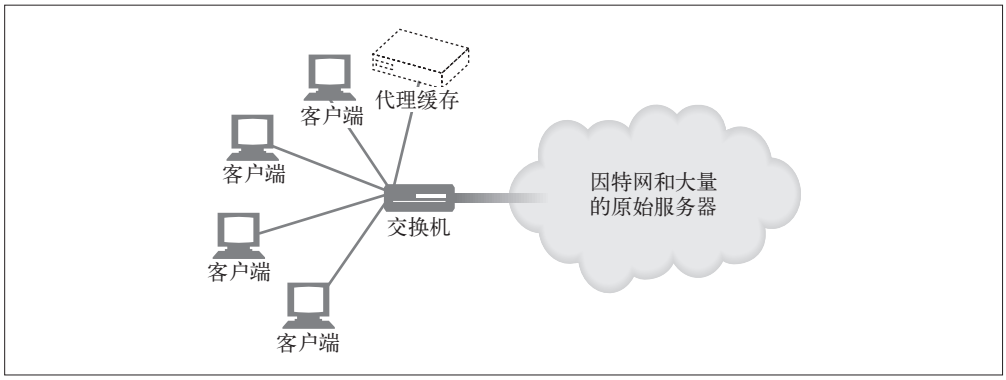


图 18-8　客户端的请求被交换机拦截并发给代理缓存

拦截环境依赖于在客户端和服务器之间设置网络的能力，这样，所有合适的 HTTP 请求才能真正发送到缓存中去。（参见第 20 章）。根据收到的请求，将内容分布在缓存中。

18.4　让网站更快

前面一节提到的很多技术也能帮助网站更快地加载。服务器集群和分布式代理缓存或反向代理服务器分散了网络流量，可以避免拥塞。分发内容使之更靠近终端用户，这样从服务器到客户端的传输时间就更短了。请求和响应穿过因特网，在客户端和服务器间传输的方式是影响资源访问速度最主要的因素。重定向方法的详细内容参见第 20 章。

加速网站访问的另一种方法是对内容进行编码以便更快地传输。比如，对内容进行压缩，但前提是接收的客户端能够把内容解压缩。请参见第 15 章了解更多细节。 422

18.5　更多信息

参阅第 3 部分以了解如何使 Web 站点安全。下面的因特网草案和文档提供了 Web 虚拟主机服务和内容分发的更多细节。

- http://www.ietf.org/rfc/rfc3040.txt
  RFC 3040，“Internet Web Replication and Caching Taxonomy”（“因特网 Web 复制和缓存分类法”），这份文档是关于 Web 复制与缓存应用术语的参考文献。

- http://search.ietf.org/internet-drafts/draft-ietf-cdi-request-routing-reqs-00.txt
  “Request-Routing Requirements for Content Internetworking”（“内容网际互连的请求路由需求”）。

- *Apache: The Definitive Guide*[1]（《Apache 权威指南》）
  Ben Laurie 和 Peter Laurie 著，O’Reilly & Associates 公司出版。这本书讲述如何
423 运行开源的 Apache Web 服务器。

注 1：本书影印版由人民邮电出版社出版。（编者注）

第19章

发布系统

怎样创建 Web 页面并放到 Web 服务器上去呢？在 Web 发展的“蛮荒”时代（比如 1995 年），可能要在文本编辑器中手工拼凑 HTML，用 FTP 手动把内容上传到 Web 服务器。这个过程很痛苦，很难与同事配合，也不是特别安全。

如今的发布工具使得创建、发布以及管理 Web 内容方便了许多。今天，用户可以交互式地编辑 Web 内容，在屏幕上看到它实际呈现的样子，轻轻点击一下就可以把内容发布到服务器，还能得到所有文件变化的通知。

许多支持远程发布内容的工具都使用了扩展的 HTTP 协议。本章将讲解以 HTTP 为基础的两种重要的 Web 内容发布技术：FrontPage 和 DAV。

19.1 FrontPage为支持发布而做的服务器扩展

FrontPage（FP）是微软公司提供的一种通用 Web 写作和发布工具包。FrontPage 的原始创意（FrontPage 1.0）是由维美尔（Vermeer）技术公司在 1994 年构思的，它是首个把网站管理和创建整合进一个统一工具的产品。微软公司 1996 年收购了维美尔公司，发行了 FrontPage 1.1。最新的版本——FrontPage 2002 版，是这条产品线上的第六代，是微软办公套件的核心组成部分。

19.1.1 FrontPage服务器扩展

作为“随处发布”战略的一部分，微软公司发布了一系列服务器端软件，称为“FrontPage 服务器扩展”，（FPSE）。这些服务器端组件和 Web 服务器集成在一起，
424 在网站和运行 FrontPage 的客户端（以及其他支持这些扩展的客户端）之间提供了必要的转接工作。

我们主要关注 FrontPage 客户端和 FPSE 之间的发布协议。该协议是对 HTTP 核心服务进行扩展而无需改变 HTTP 语义的一个设计范例。

FrontPage 的发布协议在 HTTP 的 POST 请求之上实现了一个 RPC（Remote Procedure Call，远程过程调用）层。它允许 FrontPage 客户端向服务器发送命令来更新网站上的文档、进行搜索以及在多个 Web 作者之间进行协作，等等。图 19-1 给出了这个通信过程的概貌。

Web 服务器看到以 FPSE（在非微软 IIS 服务器上就以一组 CGI 程序的方式实现）为接收地址的 POST 请求，就对其进行相应的引导。只要中间的防火墙和代理服务器都配置为允许使用 POST 方法，FrontPage 就能与服务器持续通信。

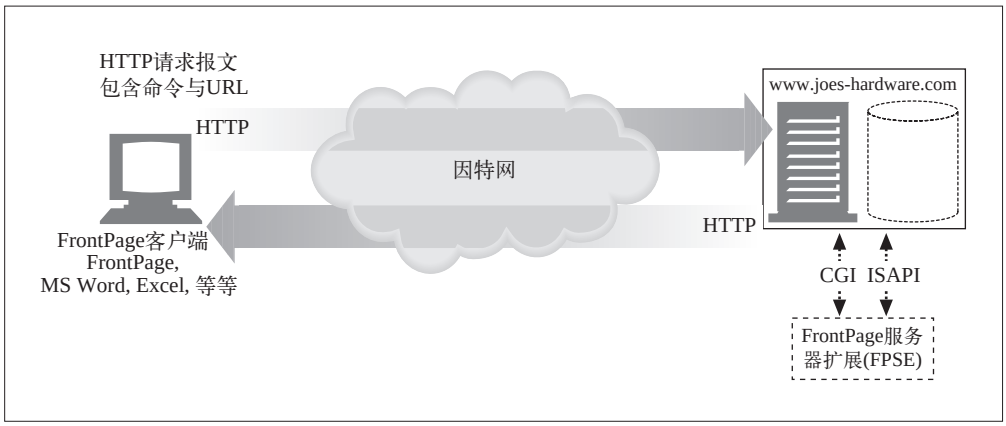


图 19-1　FrontPage 发布系统体系结构

19.1.2　FrontPage术语表

在我们深入研究 FPSE 定义的 RPC 层之前，先来了解一下其常用术语。

- *虚拟服务器*
  在同一服务器上运行的多个网站之一，每个都有唯一的域名和 IP 地址。本质上说，虚拟服务器允许在单一的 Web 服务器上托管多个网站，在浏览器看来每个网站都像是由它自己专门的 Web 服务器托管的一样。支持虚拟服务器的 Web 服务器称为*多路托管*（multi-hosting）Web 服务器。配置有多个 IP 地址的机器称为*多宿主*（multi-homed）服务器（参阅 18.2 节以获取更多信息）。

- *根 Web*
  Web 服务器默认的顶层内容目录，或者是在多路托管环境下，虚拟 Web 服务器的顶层内容目录。要访问根 Web，只需指定该服务器的 URL，而不需要指定页面名称。每个 Web 服务器只能有一个根 Web。

425

- *子 Web*
  根 Web 的已命名子目录或另一个完全由 FPSE 扩展的子 Web。子 Web 可以是完全独立的实体，能够指定自己的管理和写作权限。此外，子 Web 还能提供方法（比如搜索）的作用范围。

19.1.3　FrontPage的RPC协议

FrontPage 客户端与 FPSE 使用专用的 RPC 协议来通信。该协议构建在 HTTP 的 POST 方法之上，它把 RPC 的方法及其相关的变量嵌入在 POST 请求的主体中。

在开始处理之前，客户端需要知道服务器上目标程序（FPSE 包中能够执行这些 POST 请求的相关部分）的位置和名称。接下来它会发送一个特殊的 GET 请求（参见图 19-2）。

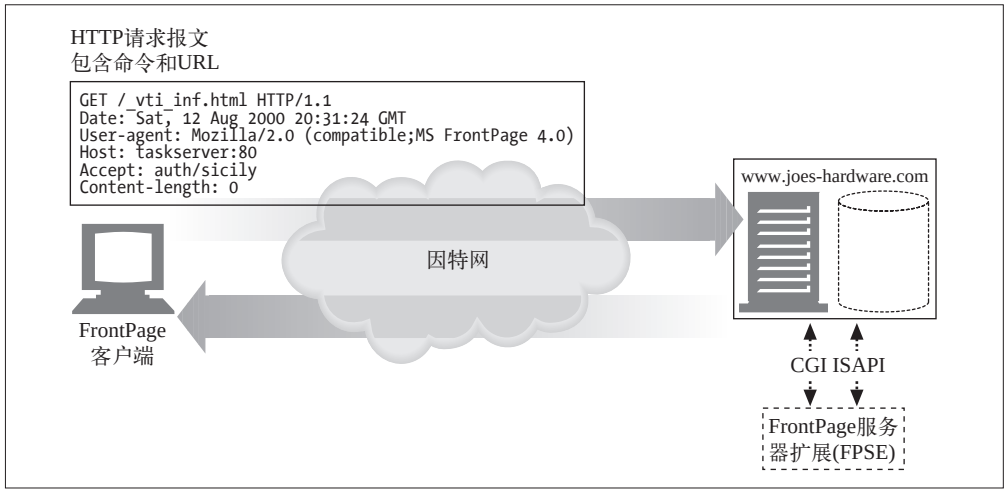


图 19-2　初始请求

得到返回的文件之后，FrontPage 客户端读取响应，寻找与 FPShtmlScriptUrl、FPAuthorScriptUrl 以及 FPAdminScriptUrl 相关的值。通常，这些值看起来是这样的：

```
FPShtmlScriptUrl="_vti_bin/_vti_rpc/shtml.dll"
FPAuthorScriptUrl="_vti_bin/_vti_aut/author.dll"
FPAdminScriptUrl="_vti_bin/_vti_adm/admin.dll"
```

FPShtmlScriptUrl 告诉客户端要执行“浏览时”命令（例如，获取 FPSE 的版本号）时应向哪里 POST 请求。

FPAuthorScriptUrl 告诉客户端要执行“写作时”命令时应向哪里 POST 请求。
426 类似地，FPAdminScriptUrl 告诉 FrontPage 向哪里发送管理操作的 POST 请求。

现在我们已知道这些程序所在的位置，可以发出请求了。

1. 请求

POST 请求的主体包含 RPC 命令，形式是 method=<command> 及任何需要的参数。例如，请求文档列表的 RPC 报文如下：

```
POST /_vti_bin/_vti_aut/author.dll HTTP/1.1
Date: Sat, 12 Aug 2000 20:32:54 GMT
```

```
User-Agent: MSFrontPage/4.0
.....................................

<BODY>
method=list+documents%3a4%2e0%2e2%2e3717&service%5fname=&listHiddenDoc
s=false&listExplorerDocs=false&listRecurse=false&listFiles=true&listFo
lders=true&listLinkInfo=true&listIncludeParent=true&listDerived=false
&listBorders=false&listChildWebs=true&initialUrl=&folderList=%5b%3bTW%
7c12+Aug+2000+20%3a33%3a04+%2d0000%5d
```

POST 方法的主体中含有发送给 FPSE 的 RPC 命令。与 CGI 程序一样，方法中的空格被编码为加号（+）字符。所有其他非字母数字的字符都被编码为 %*XX* 格式，*XX* 表示该字符的 ASCII 码。根据这种记号方式，更容易辨识的主体版本如下所示：

```
method=list+documents:4.0.1.3717
&service_name=
&listHiddenDocs=false
&listExplorerDocs=false
.....
```

其中某些元素的含义如下所述。

- `service_name`
  方法应该在该 URL 表示的网站上执行。必须是已有文件夹或者已有文件夹的下层文件夹。

- `listHiddenDocs`
  如果值为 true，就显示网站中隐藏的文档。所谓“隐藏”是指其 URL 的路径部分以“_”开头。

- `listExploreDocs`
  如果值为 true，就列出任务列表。

2. 响应

大多数 RPC 协议方法都有返回值。大多数常见的返回值都用来表示方法成功和各种错误。有些方法还有第三种类别的返回值，称为“采样返回码”。FrontPage 会对这些代码进行适当的解释，为用户提供准确的反馈。

427

继续讨论前面的例子，FPSE 处理这个 `listdocuments` 请求并返回必须的信息。示例响应如下：

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sat, 12 Aug 2000 22:49:50 GMT
Content-type: application/x-vermeer-rpc
```

```
X-FrontPage-User-Name: IUSER_MINSTAR

<html><head><title>RPC packet</title></head>
<body>
<p>method=list documents: 4.0.2.3717
<p>document_list=
<ul>
    <li>document_name=help.gif
<\ul>
```

可以从响应中看到，Web 服务器上可用文档的列表返回给了 FrontPage 客户端。在微软公司的网站上可以找到各种命令和响应的完整列表。

19.1.4 FrontPage的安全模型

任何直接访问 Web 服务器内容的发布系统都要非常注意其行为的潜在安全影响。FPSE 在极大程度上是依赖 Web 服务器来提供安全性的。

FPSE 安全模型定义了 3 种用户：管理员、作者以及浏览者，其中管理员拥有完全控制权。所有权限都是累积的，也就是说，所有的管理员都能编写和浏览 FrontPage 的网站。类似地，所有作者都有浏览权限。

对于给定的由 FPSE 扩展的网站，管理员、作者以及浏览者的列表都要定义好。所有的子 Web 可以从根 Web 继承权限，也可以自行定义。对于非 IIS 的 Web 服务器，所有的 FPSE 程序都要保存在标记为“可执行”的目录中（所有其他 CGI 程序也都有同样的限制）。Fpsrvadm，FrontPage 的服务器管理员实用工具，可以用来进行这种工作。在 IIS 服务器上，则可用 Windows 操作系统自身集成的安全模型。

在非 IIS 服务器上，Web 服务器的访问控制机制负责指定能够访问指定程序的用户。在 Apache 和 NCSA 的 Web 服务器上，访问控制文件名为 .htaccess；在 Netscape 服务器上，文件名是 .nsconfig。访问控制文件将用户、用户组以及 IP 地址与不同级别的权限关联起来：GET 是读权限，POST 是写权限，等等。例如，为了使用户在 Apache 的 Web 服务器上具有作者权限，.htaccess 文件应当允许该用户对 author.exe 进行 POST。这些访问规范文件常常是以目录为单位来定义的，这为权限定义提供了极大的灵活性。

428

在 IIS 服务器上，权限是通过给定根 Web 目录或子 Web 的根目录上的 ACL（Access Control List，访问控制列表）来制定的。当 IIS 收到请求时，首先登录，并模拟用户，接着发送请求到上述三个 DLL（Dynamic Link Library，动态链接库）之一。收到请求的 DLL 根据目标文件夹上定义的 ACL 检查所扮演用户的证书。如果检查通过，请求的操作就由扩展 DLL 来执行。否则，就向客户端发回“permission denied”

（没有权限）报文。由于 IIS 和 Windows 操作系统的安全管理紧密集成在一起，所以可以使用用户管理器进行细粒度的控制。

尽管有着精密的安全模型，启用 FPSE 还是背负了带来显著安全风险的恶名。在多数情况下，这是由于网站管理员的粗心大意引起的。然而，早期版本的 FPSE 的确有严重的安全漏洞，从而引起了这种安全风险的担忧。完整地实现严密的安全模型困难重重，也加剧了这种担忧。

19.2 WebDAV与协作写作

WebDAV（Web Distributed Authoring and Versioning，分布式写作与版本管理）为 Web 发布增添了新的内容——协作。当前，最常见的协作实践（电子邮件或分布式文件共享）显然没有什么技术含量。这种实践很不方便还容易出错，而且几乎没有过程控制。请考虑运行一个为多国用户服务的汽车制造商的多语种网站。很容易看出，它非常需要安全可靠的带有发布原语和协作原语（比如锁定和版本管理等）的健壮系统。

WebDAV（作为 RFC 2518 发表）专注于对 HTTP 进行扩展，以提供协作写作的适宜平台。目前是在 IETF 的组织下，得到了许多厂商的支持，包括 Adobe、Apple、IBM、Microsoft、Netscape、Novell、Oracle 以及 Xerox。

19.2.1 WebDAV的方法

WebDAV 定义了一些新的 HTTP 方法并修改了其他一些 HTTP 方法的操作范围。WebDAV 新增的方法如下所述。

- PROPFIND
  获取资源的属性。
- PROPPATCH
  在一个或多个资源上设定一个或多个属性。
- MKCOL
  创建集合。
- COPY
  从指定的源端把资源或者资源集合复制到指定的目的地。目的地可以在另一台机器上。

- MOVE
  从指定的源端把资源或者资源集合移动到指定的目的地。目的地可以在另一台机器上。

- LOCK
  锁定一个或多个资源。

- UNLOCK
  把先前锁定的资源解锁。

  WebDAV 修改的 HTTP 方法有 DELETE、PUT 以及 OPTIONS。本章稍后将详细讨论新方法和修改后的方法。

19.2.2 WebDAV与XML

WebDAV 的方法通常都需要在请求和响应中关联大量的信息。HTTP 通常用报文首部来交换这类信息。然而，只在首部传输必要的信息已经暴露了一些局限性，包括难以有选择地对请求中的多个资源应用首部信息、不利于表示层次结构等。

WebDAV 借助了 XML 解决这个问题，它是一种元标记语言，提供了描述结构化数据的格式。XML 为 WebDAV 提供了以下解决方案。

- 对那些描述数据处理方式的指令进行格式化的方法。
- 在服务器上对复杂的响应进行格式化的方法。
- 交换与所处理的集合和资源有关的定制信息的方法。
- 承载数据自身的灵活工具。
- 对大多数国际化问题的健壮解决方案。

习惯上会把 XML 文档里引用的方案定义保存在一个 DTD（Document Type Definition，文档类型定义）文件中。因此，试图解释 XML 文档时，可以根据其中的 DOCTYPE 定义项得到和这份 XML 文档相关的 DTD 文件名。

WebDAV 定义了一个显式的 XML 名字空间——“DAV:”。简单地说，XML 的名字空间就是元素或属性的名字的集合。名字空间限定了嵌入的名字在域内必须是唯一的，这样就可以避免名字冲突。

WebDAV 规范（也就是 RFC 2518），定义了完整的 XML 方案。预定义的方案允许
430 解析软件不必读取 DTD 文件，而是根据预定义的 XML 方案来解释。

19.2.3 WebDAV首部集

WebDAV 的确引入了一些 HTTP 首部来增强新方法的功能。本节对其进行了简要介绍，请从 RFC 2518 中获取更多信息。新的首部如下所示。

- DAV

  用于了解服务器的 WebDAV 能力。WebDAV 支持的所有资源在响应 OPTIONS 请求时都要含有此首部。更多细节参见 19.2.14 节。

```
DAV = "DAV" ":" "1" ["," "2"] ["," 1#extend]
```

- Depth

  这是一个关键元素，用于把 WebDAV 扩展到支持含有多级层次关系的资源组。参见 19.2.10 节以获取更多关于集合的详细解释。

```
Depth = "Depth" ":" ("0" | "1" | "infinity")
```

  我们来看一个简单的例子。假设有个目录 DIR_A，其中有文件 file_1.html 和 file_2.html。如果某方法设置了 `Depth:0`，此方法就只作用到目录 DIR_A 自身；如果设置了 `Depth:1`，就作用到目录 DIR_A 及其包含的文件 file_1.html 和 file_2.html。

  `Depth` 首部对 WebDAV 定义的许多方法进行了修饰。用到 `Depth` 首部的方法有：LOCK、COPY 以及 MOVE。

- Destination

  定义这个首部是用来辅助 COPY 或 MOVE 方法标识目标 URI 的。

```
Destination = "Destination" ":" absoluteURI
```

- If

  定义的唯一一个状态令牌是锁定令牌（参见 19.2.5 节）。`If` 首部定义了一组条件，如果这些条件都取值为非，请求就失败。类似 COPY 和 PUT 等方法可以在 `If` 首部中指定前置条件，使其有条件地适用。在实践中，最常见的需要满足的前置条件是先获得锁。

```
If ="If"":" (1*No-tag-list | 1*Tagged-list)
No-tag-list = List
Tagged-list = Resource 1*List
Resource = Coded-URL
List = "(" 1*(["Not"](State-token | "[" entity-tag "]")) ")"
State-token = Coded-URL
Coded-URL = "<" absoluteURI ">"
```

- Lock-Token

  UNLOCK 方法需要用这个首部指定要删除的锁。LOCK 方法的响应中也有

Lock-Token 首部，载有关于锁定令牌的必须信息。

431

```
Lock-Token ="Lock-Token"":"Coded-URL
```

- Overwrite
  用于 COPY 和 MOVE 方法，指定是否要覆盖目标。参见本章后面关于 COPY 和 MOVE 方法的详细介绍。

```
Overwrite ="Overwrite"":" ("T" |"F")
```

- Timeout
  客户端用这个请求首部指定要求锁定的超时值。参见 19.2.5 节获取更多信息。

```
TimeOut = "Timeout" ":" 1#TimeType
TimeType = ("Second-" DAVTimeOutVal | "Infinite" | Other)
DAVTimeOutVal = 1*digit
Other = "Extend" field-value
```

我们已经概述了 WebDAV 的意图及其实现，下面来仔细看看它提供的各种功能。

19.2.4 WebDAV的锁定与防止覆写

根据定义，协作要有不止一个人在给定的文档上工作。图 19-3 展示了和协作相关的固有问题。

在这个例子中，作者 A 和 B 联合编写一份规范。A 和 B 各自独立地对文档做了一些修改。A 把更新的文档上传到仓库。之后，B 也把自己的版本提交到仓库。不幸的是，由于 B 压根不知道 A 的修改，他没有把自己的版本与 A 的版本进行合并，从
432 而导致 A 的修改丢失。

为了改善这种问题，WebDAV 支持锁定的概念。但单靠锁定不能完全解决这个问题，还需要版本管理和消息传送才能提供完整的解决方案。

WebDAV 支持两种类型的锁：

- 对资源或集合的独占写锁；
- 对资源或集合的共享写锁。

*独占写锁*保证只有锁的拥有者有写权限。这种锁完全消除了潜在的冲突。*共享写锁*允许多个人在某个给定的文件上工作。这种锁定机制在多名作者对各自的活动都知晓的环境下可以很好地工作。WebDAV 通过 PROPFIND 方法提供了属性发现机制，可以判断对锁定的支持和所支持的锁定类型。

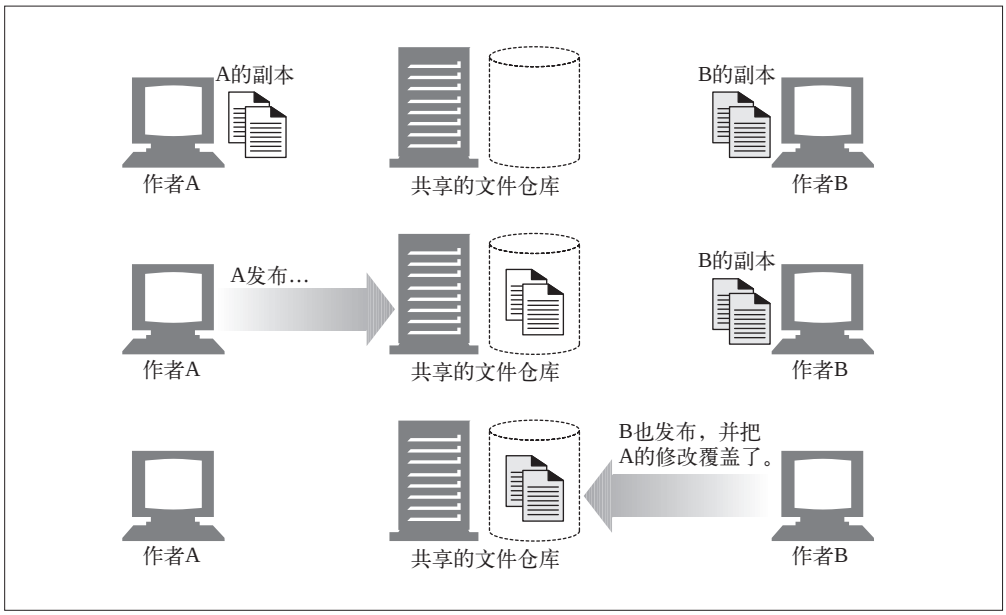


图 19-3　丢失更新问题

WebDAV 中有两个新方法支持锁定机制：LOCK 和 UNLOCK。

为了实现锁定，还需要有一种识别作者的机制。WebDAV 采用的是摘要认证（参见第 13 章）。

批准锁定时，服务器将域内唯一的令牌返回给客户端。与此相关的规范是 opaquelocktoken 锁定令牌 URI 方案。当客户端随后要执行写操作时，它连接到服务器并完成摘要认证步骤。一旦认证完成，WebDAV 客户端就发出带有锁定令牌的 PUT 请求。这样，只有正确的用户加上锁定令牌才可以完成写操作。

19.2.5　LOCK方法

WebDAV 中的一个强大特性是它能够允许单个 LOCK 请求锁定多个资源。WebDAV 的锁定不需要客户端保持与服务器的连接。

这是一个简单的 LOCK 请求示例：

433

```
LOCK /ch-publish.fm HTTP/1.1
Host: minstar
Content-Type: text/xml
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)
Content-Length: 201
```

```
<?xml version="1.0"?>
<a:lockinfo xmlns:a="DAV:">
     <a:lockscope><a:exclusive/></a:lockscope>
     <a:locktype><a:write/></a:locktype>
     <a:owner><a:href>AuthorA</a:href></a:owner>
</a:lockinfo>
```

提交的 XML 以 <lockinfo> 元素作为其基元素。在 <lockinfo> 结构中，有以下 3 种子元素。

- <locktype>
  指明锁定的类型。当前只有一种可选值，即 write。

- <lockscope>
  指明这是独占锁还是共享锁。

- <owner>
  这个字段设置为当前持有锁的人。

下面是这个 LOCK 请求的成功响应：

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 10 May 2002 20:56:18 GMT
Content-Type: text/xml
Content-Length: 419

<?xml version="1.0"?>
<a:prop xmlns:a="DAV:">
<a:lockdiscovery><a:activelock>
<a:locktype><a:write/></a:locktype>
<a:lockscope><a:exclusive/></a:lockscope>
<a:owner xmlns:a="DAV:"><a:href>AutherA</a:href></a:owner>
<a:locktoken><a:href>opaquelocktoken:*****</a:href></a:locktoken>
<a:depth>0</a:depth>
<a:timeout>Second-180</a:timeout>
</a:activelock></a:lockdiscovery>
</a:prop>
```

<lockdiscovery> 元素充当着存储锁信息的容器。嵌入在 <lockdiscovery> 元素中的子元素有 <activelock>，它持有请求发送来的信息（<locktype>、<lockscope> 以及 <owner>）。此外，<activelock> 中还含有以下子元素。

- <locktoken>
  用称为 opaquelocktoken 的 URI 方案唯一标识的锁。考虑到 HTTP 天生就是无状态的，该令牌用于在将来的请求中标识锁的所有权。

- <depth>
  它是 Depth 首部的值的副本。

- <timeout>
  指明锁的超时时间。在上面的响应中，超时值是 180 秒。

1. opaquelocktoken方案

opaquelocktoken 是设计用来在所有时间内对所有资源提供唯一令牌的方案。为了确保唯一性，WebDAV 规范规定采用 ISO-11578 中描述的 UUID 机制。 434

在实际实现的时候，有一定的回旋余地。服务器可以选择为每个 LOCK 请求生成一个 UUID，或者生成单个 UUID 并通过在结尾附加额外的字符来维护唯一性。从性能角度衡量，选择后面那种方法更好。不过，如果服务器选择实现后面那种方法，就必须保证附加的扩展部分永远不会重用。

2. XML元素<lockdiscovery>

XML 元素 <lockdiscovery> 提供了发现活跃的锁的机制。如果有人试图去给已经被锁定的文件上锁，他会收到指明当前拥有者的 XML 元素 <lockdiscovery>。<lockdiscovery> 元素列出了所有未解除的锁和相应的属性。

3. 锁的刷新和Timeout首部

为了刷新锁，客户端需要重新提交锁定请求，并把锁定令牌放在 If 首部中。返回的超时值可能和早先的超时值不同。

除了接受服务器给定的超时值，客户端也可以在 LOCK 请求中指明要求的超时值。这可以通过 Timeout 首部做到。Timeout 首部的语法允许客户端在逗号分隔的列表中描述一些选项。例如：

```
Timeout : Infinite, Second-86400
```

服务器没有义务必须满足这些选项。但是，客户端必须在 XML 元素 <timeout> 中提供锁定过期的时间。无论怎样，锁定超时只是一个指导值，服务器不一定受其约束。管理员可以手工重设，某些异常事件也可能导致服务器重设锁。客户端应当避免使锁定时间太长。

尽管有这些原语，图 19-3 中显示的“丢失更新问题”并没有得到完全解决。为了彻底解决这个问题，需要带有版本控制的协作事件系统。

19.2.6 UNLOCK方法

435 UNLOCK 方法用于解除资源上的锁。示例如下：

```
UNLOCK /ch-publish.fm HTTP/1.1
Host: minstar.inktomi.com
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)
Lock-Token:
opaquelocktoken:*********

HTTP/1.1 204 OK
Server: Microsoft-IIS/5.0
Date: Fri, 10 May 2002 20:56:18 GMT
```

与大多数资源管理请求一样，要使 UNLOCK 操作成功，WebDAV 要满足两个条件：第一，先前已经成功完成了摘要认证步骤；第二，要与在 `Lock-Token` 首部中发送的锁定令牌相匹配。

如果解锁成功，会向客户端发送 204 No Content 状态码。表 19-1 总结了 LOCK 和 UNLOCK 方法可能的状态码。

表19-1 LOCK和UNLOCK方法的状态码

状 态 码	定 义 者	方 法	效 果
200 OK	HTTP	LOCK	表明锁定成功
201 Created	HTTP	LOCK	表明通过创建该资源已成功锁定了不存在的资源
204 No Content	HTTP	UNLOCK	表明解锁成功
207 Multi-Status	WebDAV	LOCK	请求锁定多个资源。返回的资源状态码不完全一样，因此，这些状态码被封装在一个 207 响应中
403 Forbidden	HTTP	LOCK	表明客户端没有权限锁定资源
412 Precondition Failed	HTTP	LOCK	可能是随 LOCK 命令发送的 XML 中指明要满足某条件而服务器无法完成，也可能是无法强制执行锁定令牌
422 Unprocessable Property	WebDAV	LOCK	语义不适用——比如为不是集合的资源指定了非 0 的 `Depth`
423 Locked	WebDAV	LOCK	已处于锁定状态
424 Failed Dependency	WebDAV	UNLOCK	UNLOCK 指定了其他动作，并以它们的成功作为解锁的前提条件。如果无法成功完成这些有依赖关系的动作，就返回此错误码

19.2.7 属性和元数据

*属性*描述了资源的信息，包括作者的名字、修改日期、内容分级，等等。HTML 中的*元标记*的确提供了把这种信息嵌入在内容之中的机制，但很多种资源（比如所有二进制数据）都无法嵌入元数据。

像 WebDAV 这样的分布式协作系统对属性的需求就更复杂了。例如，考虑作者属性：当文档被编辑之后，应当更新这个属性以反映新的作者。WebDAV 专门把这种可动态修改的属性称为“活”属性。与之相对的是更长久的静态属性，比如 Content-Type，称为“死”属性。

为了支持查找和修改属性，WebDAV 扩展了 HTTP 以包括两个新方法：PROPFIND 和 PROPPATCH。后面几节给出了示例并讲解了相关的 XML 元素。 436

19.2.8 PROPFIND方法

PROPFIND 方法用于获取一个给定文件或一组文件（也称为“集合”）的属性。PROPFIND 支持 3 种类型的操作：

- 请求所有的属性及其值；
- 请求一组属性及其值；
- 请求所有属性的名称。

下面这个例子中，请求的是所有属性及其值：

```
PROPFIND /ch-publish.fm HTTP/1.1
Host: minstar.inktomi.com
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)
Depth: 0
Cache-Control: no-cache
Connection: Keep-Alive
Content-Length: 0
```

<propfind> 请求元素指定了从 PROPFIND 方法返回的属性。下面的列表总结了用于 PROPFIND 请求的一些 XML 元素：

- <allprop>
  要求返回所有属性的名字和值。为了请求所有的属性及其值，WebDAV 客户端可以将 XML 子元素 <allprop> 作为 <propfind> 元素的一部分发送，或提交一个没有主体的请求。

- <propname>
  指定要返回属性名字的集合。

- <prop>
  <propfind> 元素的子元素。指定需要返回值的属性。例如:<a:prop> <a:owner />..... </a:prop>。

下面是对一个 PROPFIND 请求示例的响应： 437

```
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
...........

<?xml version="1.0"?>
<a:multistatusxmlns:b="urn:uuid:********/" xmlns:c="xml:"
xmlns:a="DAV:">
<a:response>
<a:href>http://minstar/ch-publish.fm </a:href>
<a:propstat>
<a:status>HTTP/1.1 200OK</a:status>
<a:prop>
<a:getcontentlength b:dt="int">1155</a:getcontentlength>
.....................
.....................
<a:ishidden b:dt="boolean">0</a:ishidden>
<a:iscollection b:dt="boolean">0</a:iscollection>
</a:prop>
</a:propstat>
</a:response></a:multistatus>
```

在这个例子中，服务器以 207 Multi-Status 状态码进行响应。WebDAV 将 207 用在 PROPFIND 和其他几个 WebDAV 方法上，它们同时作用在多个资源上并且每个资源可能有不同的响应。

响应中的几个 XML 元素的定义如下所示。

- `<multistatus>`
  多重响应的容器。

- `<href>`
  标识资源的 URI。

- `<status>`
  包含特定请求的 HTTP 状态码。

- `<propstat>`
  将一个 `<status>` 元素和一个 `<prop>` 元素组织在一起。`<prop>` 元素可以包含给定资源的一个或多个属性名 / 值对。

在上面列出的示例响应中，响应是针对 URI：http://minstar/ch-publish.fm 的。`<propstat>` 元素内嵌了一个 `<status>` 元素和一个 `<prop>` 元素。服务器为这个 URI 回复了一个 200 OK 响应，它定义在 `<status>` 元素中。`<prop>` 元素中有若干子元素，例子中只列出了一部分。

PROPFIND 的实例应用是对目录列表的支持。考虑到 PROPFIND 请求的表达能力，

单次调用就能获取集合的整个层次结构和其中各个独立实体的所有属性。

19.2.9 PROPPATCH方法

PROPPATCH 方法为对指定资源设置或删除多个属性提供了原子化机制。原子化可以保证要么所有请求都成功，要么跟所有请求都没发出一样。

PROPPATCH 方法的 XML 基元素是 <propertyupdate>。它作为一个容器使用，容纳了需要修改的属性。XML 的 <set> 和 <remove> 元素用于描述操作。

- <set>
  指定要设置的属性值。<set> 含有一个或多个 <prop> 子元素，它们依次包含了该资源上要设置的属性的名 / 值对。如果属性已存在，其值就被覆盖。 438
- <remove>
  指定要删除的属性。与 <set> 不同的是，在 <prop> 容器中只列出了属性的名称。

下面这个小例子设置并删除了 owner 属性：

```
<d:propertyupdate xmlns:d="DAV:" xmlns:o="http://name-space/scheme/">
  <d:set>
   <d:prop>
     <o:owner>Author A</o:owner>
   </d:prop>
  </d:set>

  <d:remove>
   <d:prop>
     <o:owner/>
   </d:prop>
  </d:remove>
</d:propertyupdate>
```

对 PROPPATCH 请求的响应和对 PROPFIND 请求的响应非常像。参见 RFC 2518 以获取更多信息。

表 19-2 总结了 PROPFIND 和 PROPPATCH 方法的状态码。

表19-2 PROPFIND与PROPPATCH方法的状态码

状 态 码	定 义 者	方 法	效 果
200 OK	HTTP	PROPFIND, PROPPATCH	命令成功
207 Multi-Status	WebDAV	PROPFIND, PROPPATCH	作用于一个或多个资源（或者集合）时，每个对象的状态都被封装到一个 207 响应中。这是一种常见的成功响应
401 Unauthorized	HTTP	PROPATCH	需要授权才能完成对属性的修改操作

（续）

状 态 码	定 义 者	方 法	效 果
403 Forbidden	HTTP	PROPFIND, PROPPATCH	对于 PROPFIND 来说，客户端不允许访问该属性。对于 PROPPATCH 来说，客户端不允许修改该属性
404 Not Found	HTTP	PROPFIND	属性不存在
409 Conflict	HTTP	PROPPATCH	与修改语义冲突——例如，试图修改只读的属性
423 Locked	WebDAV	PROPPATCH	目标资源被锁定，并且没有提供锁定令牌，或者锁定令牌不匹配
507 Insufficient Storage	WebDAV	PROPPATCH	没有足够的空间登记修改的属性

19.2.10 集合与名字空间管理

集合是指对预定义的层次结构中的资源进行的逻辑或物理上的分组。集合的一个典
439 型的例子就是目录。就像文件系统中的目录一样，集合作为其他资源（也包括其他集合，和文件系统中的目录一样）的容器使用。

WebDAV 使用了 XML 的名字空间机制。与传统的名字空间不同，XML 名字空间的分区在阻止所有名字空间冲突的同时，还允许进行精确的结构控制。

WebDAV 提供了 5 种方法对名字空间进行操作：DELETE、MKCOL、COPY、MOVE 以及 PROPFIND。本章前面已经讨论过 PROPFIND 了，下面来讨论其他方法。

19.2.11 MKCOL方法

MKCOL 方法允许客户端在服务器上指定的 URL 处创建集合。乍一看，仅仅为了创建集合而定义一个新方法好像有点儿多余。在 PUT 或 POST 方法之上加以修饰看起来就是个完美的替代方案。WebDAV 协议的设计者确实考虑过这些替代方案，但最终还是选择定义一个新方法。决策背后的一些理由如下所述。

- 为了使用 PUT 或 POST 来创建集合，客户端要随请求发送一些额外的“语义黏胶”。这当然是可以做到的，但定义这种特别的东西总是乏味且易错的。
- 大多数访问控制机制都是建立在方法类型之上的——只有少数能在库中创建和删除资源。如果给其他方法过多的功能，这些访问控制机制就无法运作了。

下面是请求的例子：

```
MKCOL /publishing HTTP/1.1
```

```
Host: minstar
Content-Length: 0
Connection: Keep-Alive
```

其响应可能是：

```
HTTP/1.1 201 Created
Server: Microsoft-IIS/5.0
Date: Fri, 10 May 2002 23:20:36 GMT
Location: http://minstar/publishing/
Content-Length: 0
```

我们再考察下面几种异常情况。

- 假设集合已经存在。如果发出 MKCOL /colA 请求而 colA 已存在（也就是说有名字空间冲突），请求会失败，状态码是 405 Method Not Allowed。
- 如果没有写权限，MKCOL 请求会得到 403 Forbidden 失败状态码。
- 如果发出 MKCOL /colA/colB 这样的请求而 colA 不存在，请求会失败，状态码是 409 Conflict。

440

创建了文件或集合之后，可以用 DELETE 方法来删除。

19.2.12 DELETE方法

我们已经在第 3 章探讨过 DELETE 方法了。WebDAV 扩展了它的语义以覆盖集合。

如果需要删除一个目录，就要提供 `Depth` 首部。如果没有指定 `Depth` 首部，DELETE 方法就假定 `Depth` 首部设定为无穷大——也就是说，该目录中的所有文件和子目录都会被删除。响应中也有 `Content-Location` 首部，其值就是刚被删除的集合。下面是一个请求的示例：

```
DELETE /publishing HTTP/1.0
Host: minstar
```

其响应的示例如下：

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 14 May 2002 16:41:44 GMT
Content-Location: http://minstar/publishing/
Content-Type: text/xml
Content-Length: 0
```

删除集合时，总是有可能发生其中某个文件被其他人锁定而无法删除的情况。在这种情况下，集合自身也无法删除，服务器会以 207 Multi-Status 状态码响应。请求示例如下：

```
DELETE /publishing HTTP/1.0
Host: minstar
```

其响应的示例如下：

```
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
Content-Location: http://minstar/publishing/
..............
<?xml version="1.0"?>
<a:multistatus xmlns:a="DAV:">
<a:response>
<a:href>http://minstar/index3/ch-publish.fm</a:href>
<a:status> HTTP/1.1 423 Locked </a:status>
</a:response>
</a:multistatus>
```

在这次事务中，XML 元素 <status> 中含有状态码 423 Locked，表明资源 ch-
441 publish.fm 被别的用户锁定了。

19.2.13 COPY与MOVE方法

和 MKCOL 一样，有若干种方法可以定义新的 COPY 和 MOVE 操作方法。其中一种方式规定 COPY 方法先对源进行 GET 请求，下载资源，然后用 PUT 请求上传回服务器。可以设想，MOVE 方法也有类似的操作情况（有个额外的 DELETE 操作）。然而，这种处理过程无法很好地适应规模扩展——考虑一下在多级的集合上进行 COPY 或 MOVE 操作管理所涉及的问题吧。

COPY 和 MOVE 方法都将请求 URL 作为源，HTTP 的 Destination 首部的内容作为目标。MOVE 方法在 COPY 方法之外还要做一些工作：它把源 URL 复制到目的地，检查新创建的 URI 的完整性，再把源删除。请求示例如下：

```
{COPY,MOVE} /publishing HTTP/1.1
Destination: http://minstar/pub-new
Depth: infinity
Overwrite: T
Host: minstar
```

其响应示例如下：

```
HTTP/1.1 201 Created
Server: Microsoft-IIS/5.0
Date: Wed, 15 May 2002 18:29:53 GMT
Location: http://minstar.inktomi.com/pub-new/
Content-Type: text/xml
Content-Length: 0
```

在对集合操作时，COPY 或 MOVE 的行为受到 Depth 首部的影响。如果没有

Depth 首部，就默认其值是无穷大（就是说，默认会把源目录的整个结构进行复制或移动）。如果 Depth 设置为 0，方法就只作用于资源本身。如果我们是对集合进行复制或移动的话，在目的地就只创建和源具有相同属性的集合——集合内部的成员就不再复制或移动了。

对于 MOVE 方法，Depth 首部的值只允许为无穷大，原因显而易见。

1. Overwrite首部的效果

COPY 和 MOVE 方法也可能使用 Overwrite 首部。Overwrite 首部的值可以是 T 或 F。如果设置为 T 而且目标已存在，就在 COPY 或 MOVE 之前，对目标资源执行 Depth 值为无穷大的 DELETE 操作。如果 Overwrite 标志设置为 F 而目标资源存在，则操作会失败。

2. 对属性的COPY/MOVE

当复制集合或元素时，默认会复制其所有属性。不过，请求可以带有可选的 XML 主体来提供额外的操作信息。可以指定要使操作成功，必须成功复制所有属性；或者定义要使操作成功，必须复制哪些属性。

下面有两个特殊状况下的例子。

- 假设把 COPY 或 MOVE 作用到 CGI 程序或者其他产生内容的脚本程序的输出上。为了保持语义，如果由 CGI 脚本产生的文件被复制或移动了，WebDAV 要提供 src 和 link 这两个 XML 元素，指向产生此页面的程序的位置。
- COPY 和 MOVE 方法不一定能够复制所有的活属性。例如，我们来看一个 CGI 程序。如果从 cgi-bin 目录中把它拷贝走，可能就不会再去执行它了。WebDAV 的现有规范让 COPY 和 MOVE 实现的是“尽力而为”解决方案，复制所有的静态属性和合适的活属性。

3. 被锁定的资源与COPY/MOVE

如果资源目前正被锁定，COPY 和 MOVE 都禁止把锁移动或复制到目标上。在这两种情况下，如果要在一个自己有锁的现存集合中创建目标，所复制或移动的资源就会被加到那个锁中。请看下面的例子：

```
COPY /publishing HTTP/1.1
Destination: http://minstar/archived/publishing-old
```

假设 /publishing 和 /archived 分别处于两个不同的锁之下：lock1 和 lock2。当 COPY

操作结束时，/publishing 仍旧处于 lock1 的范围内，而由于移动到了已被 lock2 锁定的集合中，publishing-old 被加入到了 lock2 中。如果是 MOVE 操作，就只有 publishing-old 被加入 lock2。

表 19-3 列出了 MKCOL、DELETE、COPY 以及 MOVE 方法最有可能碰到的状态码。

表19-3　MKCOL、DELETE、COPY和MOVE方法的状态码

状　态　码	定　义　者	方　　法	效　　果
102 Processing	WebDAV	MOVE、COPY	如果请求花费的时间超过 20 秒，服务器就发送这个状态码防止客户端超时。通常在 COPY 或 MOVE 大的集合时可以见到
201 Created	HTTP	MKCOL、COPY、MOVE	对于 MKCOL，表示集合创建成功。对于 COPY 和 MOVE，表示资源 / 集合已经复制或移动成功
204 No Content	HTTP	DELETE、COPY、MOVE	对于 DELETE，表示标准的成功响应。对于 COPY 和 MOVE，表示资源被成功地复制或移动而覆盖了已有的实体
207 Multi-Status	WebDAV	MKCOL、COPY、MOVE	对于 MKCOL，表示常见的成功响应。对于 COPY 和 MOVE 来说，如果有与资源相关（除请求 URI 之外的）的错误，服务器就回复 207 响应，其中带有详述错误的 XML 主体
403 Forbidden	HTTP	MKCOL、COPY、MOVE	对于 MKCOL，表明服务器不允许在指定的位置创建集合。对于 COPY 和 MOVE，表明源和目的是相同的
409 Conflict	HTTP	MKCOL、COPY、MOVE	三种情况类似，都是方法试图创建集合或资源，而中间集合不存在——例如，试图创建 colA/colB，而 colA 不存在
412 Precondition Failed	HTTP	COPY、MOVE	或者是 `Overwrite` 首部设置为 `F` 而目标存在，或者是 XML 主体描述了一个特定需求（比如保持 `liveness` 属性），而 COPY 或 MOVE 方法无法保持该属性
415 Unsupported Media Type	HTTP	MKCOL	服务器不支持或不理解如何创建请求的实体类型
422 Unprocessable Entity	WebDAV	MKCOL	服务器不理解请求中发送的 XML 主体
423 Locked	WebDAV	DELETE、COPY、MOVE	源或目标资源被锁定，或者方法提供的锁定令牌不匹配
502 Bad Gateway	HTTP	COPY、MOVE	目标在不同的服务器上并且缺少权限
507 Insufficient Storage	WebDAV	MKCOL、COPY	没有足够的空闲空间创建资源

19.2.14 增强的HTTP/1.1方法

WebDAV 修改了 HTTP 中 DELETE、PUT 以及 OPTIONS 方法的语义。GET 和 HEAD 方法的语义保持不变。POST 执行的操作总是由特定的服务器实现来定义的，而 WebDAV 没有对 POST 的语义进行任何修改。我们已经在 19.2.10 节讨论过 DELETE 方法了。这里将讨论 PUT 和 OPTIONS 方法。

1. PUT方法

尽管 PUT 不是由 WebDAV 定义的，但这是作者把内容传送到共享站点上的唯一方法。我们在第 3 章中讨论过 PUT 的一般功能。WebDAV 修改了该方法以支持锁定。 444

请看下面的例子：

```
PUT /ch-publish.fm HTTP/1.1
Accept: */*
If:<http://minstar/index.htm>(<opaquelocktoken:********>)
User-Agent: DAV Client (C)
Host: minstar.inktomi.com
Connection: Keep-Alive
Cache-Control: no-cache
Content-Length: 1155
```

为了支持锁定，WebDAV 在 PUT 请求中增加了 `If` 首部。在上面的事务中，`If` 首部的语义规定，如果 `If` 首部中说明的锁定令牌与资源（在这个例子中，是 `ch-publish.fm`）上的锁相匹配，就应当执行 PUT 操作。`If` 首部还用在其他一些方法中，比如 PROPPATCH、DELETE、MOVE、LOCK 以及 UNLOCK 等。

2. OPTIONS方法

我们在第 3 章中讨论过 OPTIONS。这通常是启用了 WebDAV 的客户端发出的第一个请求。客户端可以用 OPTIONS 方法验证 WebDAV 的能力。请看一个事务，其请求如下：

```
OPTIONS /ch-publish.fm HTTP/1.1
Accept: */*
Host: minstar.inktomi.com
```

其响应如下：

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
MS-Author-Via: DAV
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE,
```

```
MKCOL,PROPFIND,PROPPATCH, LOCK, UNLOCK, SEARCH
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, COPY, MOVE,
PROPFIND,PROPPATCH, SEARCH, LOCK, UNLOCK
```

在对 OPTIONS 方法的响应中有一些有趣的首部。下面的介绍略微打乱了一下顺序。

- DAV 首部携带了 DAV 遵从级别的信息。有下面两类遵从。

  第 1 类遵从
  要求服务器遵从 RFC2518 每节中的所有 MUST 需求。如果资源只能达到第 1 类遵从，就要在 DAV 首部中发送 1。

  第 2 类遵从
  满足所有第 1 类的需求，并增加对 LOCK 方法的支持。除了 LOCK 方法之外，第 2 类遵从还要求支持 Timeout 和 Lock-Token 首部以及 <supportedlock> 和 <lockdiscovery> 这两个 XML 元素。在 DAV 首部中的值 2 表明第 2 类遵从。

  在上面的例子中，DAV 首部表明这两类遵从都满足。

- Public 首部列出了这个特定的服务器支持的全部方法。
- Allow 首部通常包括 Public 首部所列方法的一个子集。它只列出了对这个特定资源（ch-publish.fm）有效的方法。
- DASL 首部说明了在 SEARCH 方法中使用的查询语法的类型。在这个例子中，就是 sql。更多关于 DASL 首部的细节参见 http://www.webdav.org。

19.2.15 WebDAV中的版本管理

这可能有点儿讽刺，尽管 DAV 这个名字中有 V，但版本管理这个特性却不是开始就有的。在多个作者的协作环境中，版本管理是至关重要的。实际上，为了完全解决更新丢失的问题（图 19-3 中有展示），锁定和版本化都是必不可少的。和版本管理相关的一些常见特征包括保存和访问以前的文档版本的能力、管理变更历史以及与变更相关的注解以详细说明变更过程的能力。

在 RFC 3253 中为 WebDAV 加入了版本管理功能。

19.2.16 WebDAV的未来发展

WebDAV 现在已经获得了良好的支持。可以工作的客户端实现包括 IE 5.x 及以上版本，Windows 系统的文件管理器以及微软的办公软件。在服务器端，可用的实现包括 IIS5.x 及以上版本，Apache 的 mod_dav 以及很多其他的系统。Windows XP 和 Mac OS 10.x 都提供了对 WebDAV 的原生支持。因此，为这些操作系统编写的任何

应用程序天生都具备使用 WebDAV 的能力。

19.3 更多信息

参考下列网址以获取更多信息。

- http://officeupdate.microsoft.com/frontpage/wpp/serk/
  微软 FrontPage 2000 服务器扩展资源包。

- http://www.ietf.org/rfc/rfc2518.txt?number=2518
  “HTTP Extensions for Distributed Authoring—WEBDAV”（“HTTP 的分布式写作扩展—WEBDAV”），作者是 Y. Goland、J. Whitehead、A. Faizi、S. Carter 以及 D. Jensen。 446

- http://www.ietf.org/rfc/rfc3253.txt?number=3253
  “Versioning Extensions to WebDAV”（“对 WebDAV 进行版本管理扩展”），作者是 G. Clemm、J. Amsden、T. Ellison、C. Kaler 和 J. Whitehead。

- http://www.ics.uci.edu/pub/ietf/Webdav/intro/Webdav_intro.pdf
  “WEBDAV: IETF Standard for Collaborative Authoring on the Web”（“WEBDAV：在 Web 上进行协同写作的 IETF 标准”），作者是 J. Whitehead 和 M. Wiggins。

- http://www.ics.uci.edu/~ejw/http-future/whitehead/http_pos_paper.html
  “Lessons from WebDAV for the Next Generation Web Infrastructure”（“在下一代 Web 的基础架构中借鉴 WebDAV”），作者是 J. Whitehead。

- http://www.microsoft.com/msj/0699/dav/davtop.htm
  “Distributed Authoring and Versioning Extensions for HTTP Enable Team Authoring”（“HTTP 中为支持团队写作而做的分布式写作和版本管理扩展”），作者是 L. Braginski 和 M.Powell。

- http://www.webdav.org/dasl/protocol/draft-dasl-protocol-00.html
  “DAV Searching & Locating”（“DAV 中的搜索和定位”），作者是 S. Reddy、D. Lowry、S. Reddy、R. Henderson、J. Davis 以及 A. Babich。 447

第20章

重定向与负载均衡

HTTP 并不是独自运行在网上的。很多协议都会在 HTTP 报文的传输过程中对其数据进行管理。HTTP 只关心旅程的端点（发送者和接收者），但在包含有镜像服务器、Web 代理和缓存的网络世界中，HTTP 报文的目的地不一定是直接可达的。

本章介绍重定向技术，涉及网络工具、重定向技术细节以及判定 HTTP 报文最终目的地的协议。重定向技术通常可以用来确定报文是否终结于某个代理、缓存或服务器集群中某台特定的服务器。重定向技术可以将报文发送到客户端没有显式请求的地方去。

本章，我们会学习下列重定向技术，它们是如何工作的以及它们的负载均衡能力如何（如果有的话）：

- HTTP 重定向；
- DNS 重定向；
- 任播路由；
- 策略路由；
- IP MAC 转发；
- IP 地址转发；
- WCCP（Web 缓存协调协议）；
- ICP（缓存间通信协议）；
- HTCP（超文本缓存协议）；
- NECP（网元控制协议）；
- CARP（缓存阵列路由协议）；
- WPAD（Web 代理自动发现协议）。

448

20.1 为什么要重定向

由于 HTTP 应用程序总是要做下列 3 件事情，所以在现代网络中重定向是普遍存在的：

- 可靠地执行 HTTP 事务；
- 最小化时延；
- 节约网络带宽。

出于这些原因，Web 内容通常分布在很多地方。这么做是出于可靠性的考虑。这样，如果一个位置出问题了，还有其他的可用；如果客户端能去访问较近的资源，就可以更快地收到所请求的内容，以降低响应时间；将目标服务器分散，还可以减少网络拥塞。可以将重定向当作一组有助于找到“最佳”分布式内容的技术。

由于重定向和负载均衡是共存的，所以本章也涵盖了负载均衡的话题。大多数重定向部署都包含了某些形式的负载均衡。也就是说，它们可以将输入报文的负载分摊到一组服务器中去。反之，因为输入报文一定会在分担负荷的服务器之间进行某种分布，所以任意形式的负载均衡中都包含了重定向。

20.2 重定向到何地

从客户端向目标发送 HTTP 请求，目标对其进行处理的角度来看，服务器、代理、缓存和网关对客户端来说都是服务器。很多重定向技术都可用于服务器、代理、缓存和网关，因为它们具有共同的，与服务器类似的特征。其他一些重定向技术是专门为特定类型的端点设计的，没有通用性。本章稍后的小节会介绍一些通用及专用的重定向技术。

Web 服务器会根据每个 IP 来处理请求。将请求分摊到复制的服务器中去，就意味着应该把对某特定 URL 的每条请求都发送到最佳的 Web 服务器上去（最靠近客户端的、或负载最轻的或采用其他优化策略选择的服务器）。重定向到某台服务器就像将所有需要给汽车加油的司机都送到最近的加油站去一样。

代理希望根据每个协议来处理请求。在理想情况下，某个代理附近的所有 HTTP 流量都应该通过这个代理传输。比如，如果某代理缓存靠近各种不同的客户端，那么理想情况下，所有请求都应流经这个代理缓存，因为代理缓存上会存储常用的文档，可以直接提供，从而避免通过更长、更昂贵的路径连接到原始服务器。重定向到代理就像从一条主要通路（无论它通往何处）上将流量分流到一条本地快捷路径上去一样。

449

20.3 重定向协议概览

重定向的目标是尽快地将 HTTP 报文发送到可用的 Web 服务器上去。在穿过因特网的路径上，HTTP 报文传输的方向会受到 HTTP 应用程序和报文经由的路由设备的影响，参见以下示例。

- 配置创建客户端报文的浏览器应用程序，使其将报文发送给代理服务器。
- DNS 解析程序会选择用于报文寻址的 IP 地址。对不同物理地域的不同客户端来说，这个 IP 地址可能不同。
- 报文经过网络传输时，会被划分为一些带有地址的分组；交换机和路由器会检查分组中的 TCP/IP 地址，并据此来确定分组的发送路线。
- Web 服务器可以通过 HTTP 重定向将请求反弹给不同的 Web 服务器。

浏览器配置、DNS、TCP/IP 路由以及 HTTP 都提供了重定向报文的机制。注意，有些方法，比如浏览器配置，只有在将流量重定向到代理的时候才有意义，而其他一些方法（比如 DNS 重定向），则可用于将流量发送给任意服务器。[1]

表 20-1 总结了用来将报文重定向到服务器的重定向方法，本章稍后逐一讨论每种方法。

表20-1　通用的重定向方法

机　制	工作方式	重新路由的基础	局　限　性
HTTP 重定向	最初，HTTP 请求先到第一台 Web 服务器，这台服务器会选择一台“最佳”的 Web 服务器为其提供内容。第一台 Web 服务器会向客户端发送一条到指定服务器的 HTTP 重定向。客户端会将请求重新发送到选中的服务器上	选择最短路径时可用的选项很多，包括轮转（round-robin）负载均衡和最小化时延等	可能会很慢——每个事务都包含了附加的重定向步骤。而且，第一台服务器一定要能够处理请求负载
DNS 重定向	DNS 服务器决定在 URL 的主机名中返回多个 IP 地址中的哪一个	选择最短路径时可用的选项很多，包括轮转（round-robin）负载均衡和最小化时延等	需要配置 DNS 服务器
任播寻址	几台服务器使用相同的 IP 地址。每台服务器都会伪装成一个骨干路由器。其他路由器会将共享 IP 地址分组发送给最近的服务器（认为它们将分组发送给最近的路由器）	路由器有内建的最短路径路由功能	需要拥有 / 配置路由器。有地址冲突的风险。如果路由变化了，与已建立的 TCP 连接相关的分组会被发送到其他的服务器，可能会使 TCP 连接断裂
IP MAC 转发	交换机或路由器这样的网元会读取分组的目的地址。如果应该将分组重定向，交换机会将服务器或代理的目标 MAC 地址赋予分组	节省带宽，提高 QOS。负载均衡	服务器或代理的跳距必须是 1
IP 地址转发	第四层交换机会评估分组的目的端口并将重定向分组的 IP 地址改成代理或镜像服务器的 IP 地址	节省带宽，提高 QOS。负载均衡	服务器或代理可能看不到真正的客户端 IP 地址

注 1：DNS 也不能随便重定向，目标 IP 地址的服务器上没有托管所需的虚拟主机的话，就和浏览器代理配错一样，毫无意义。（译者注）

表 20-2 总结了将报文重定向到代理服务器的重定向方法。

表20-2 代理与缓存重定向技术

机　制	工作方式	重新路由的基础	局　限　性
显式浏览器配置	配置 Web 浏览器，使其将 HTTP 报文发送给附近的一个代理，通常是缓存。可以由终端用户或管理浏览器的服务进行配置	节省带宽，提高 QoS。负载均衡	取决于配置浏览器的能力
代理自动配置（PAC）	Web 浏览器从配置服务器中解析出 PAC 文件。这个 PAC 文件会告诉浏览器为每个 URL 使用什么代理	节省带宽，提高 QoS。负载均衡	必须配置浏览器，使其去查询配置服务器
Web Proxy 代理自动发现协议（WPAD）	Web 浏览器向配置服务器查询一个 PAC 文件的 URL。与单独使用 PAC 不同，不需要将浏览器配置为使用特定的配置服务器	配置服务器，将 URL 建立在客户端 HTTP 请求首部提供的信息之上。负载均衡	只有部分浏览器支持 WPAD
Web 缓存协调协议（WCCP）	路由器会评估一个分组的目的地址，并用代理或镜像服务器的 IP 地址将重定向分组封装起来。可以与很多现有路由器共同工作。可以将分组封装起来，这样客户端的 IP 地址就不会丢失了	节省带宽，提高 QoS。负载均衡	必须使用支持 WCCP 的路由器。有些拓扑结构方面的限制
因特网缓存协议（ICP）	代理缓存会在一组兄弟代理缓存中查询所请求的内容。还支持缓存的分层结构	从兄弟代理或父代理缓存中获取内容比从原始服务器中获取更快	请求内容时只使用了 URL，所以会降低缓存命中率
缓存分组路由协议（CARP）	一种代理缓存散列协议。允许缓存将请求转发给一个父缓存。与 ICP 不同的是，高速缓存上的内容是不相交的，这组缓存会像一个大型缓存那样工作	从附近的对等高速缓存中获取内容要比从原始服务器上获取快	CARP 无法支持兄弟关系。所有 CARP 客户端都必须在配置上达成一致；否则，不同的客户端就会向不同的父代理缓存发送相同的 URI，降低命中率
超文本缓存协议（HTCP）	参与的代理缓存可以向一组兄弟缓存查询所请求的内容。支持 HTTP 1.0 和 1.1 首部，以便精细地调整缓存查询	从兄弟代理或父代理缓存中获取内容比从原始服务器上获取快	

20.4 通用的重定向方法

本节我们会深入介绍服务器和代理常用的各种重定向方法。可以通过这些技术将流量重定向到不同的（可能更优的）服务器，或者通过代理来转发流量。具体来说，我们会介绍 HTTP 重定向、DNS 重定向、任播寻址、IP MAC 转发以及 IP 地址转发。

20.4.1 HTTP重定向

Web 服务器可以将短的重定向报文发回给客户端，告诉他们去其他地方试试。有些 Web 站点会将 HTTP 重定向作为一种简单的负载均衡形式来使用。处理重定向的服务器（重定向服务器）找到可用的负载最小的内容服务器，并将浏览器重定向到那台服务器上去。对广泛分布的 Web 站点来说，确定“最佳”的可用服务器会更复杂一些，不仅要考虑到服务器的负载，还要考虑到浏览器和服务器之间的因特网距离。与其他一些形式的重定向相比，HTTP 重定向的优点之一就是重定向服务器知道客户端的 IP 地址；理论上来讲，它可以做出更合理的选择。

下面是 HTTP 重定向的工作过程。在图 20-1a 中，Alice 向 www.joes-hardware.com 发送了一条请求：

```
GET /hammers.html HTTP/1.0
Host: www.joes-hardware.com
User-Agent: Mozilla/4.51 [en] (X11; U; IRIX 6.2 IP22)
```

在图 20-1b 中，服务器没有回送带有 HTTP 状态码 200 的 Web 页面主体，而是回送了一个带有状态码 302 的重定向报文：

```
HTTP/1.0 302 Redirect
Server: Stronghold/2.4.2 Apache/1.3.6
Location: http://161.58.228.45/hammers.html
```

现在，在图 20-1c 中，浏览器会用重定向 URL 重新发送请求，这次会发送给主机 161.58.228.45：

```
GET /hammers.html HTTP/1.0
Host: 161.58.228.45
User-Agent: Mozilla/4.51 [en] (X11; U; IRIX 6.2 IP22)
```

另一个客户端可能会被重定向到另一台服务器上去。在图 20-1d-f 中，Bob 的请求会
452 被重定向到 161.58.228.46。

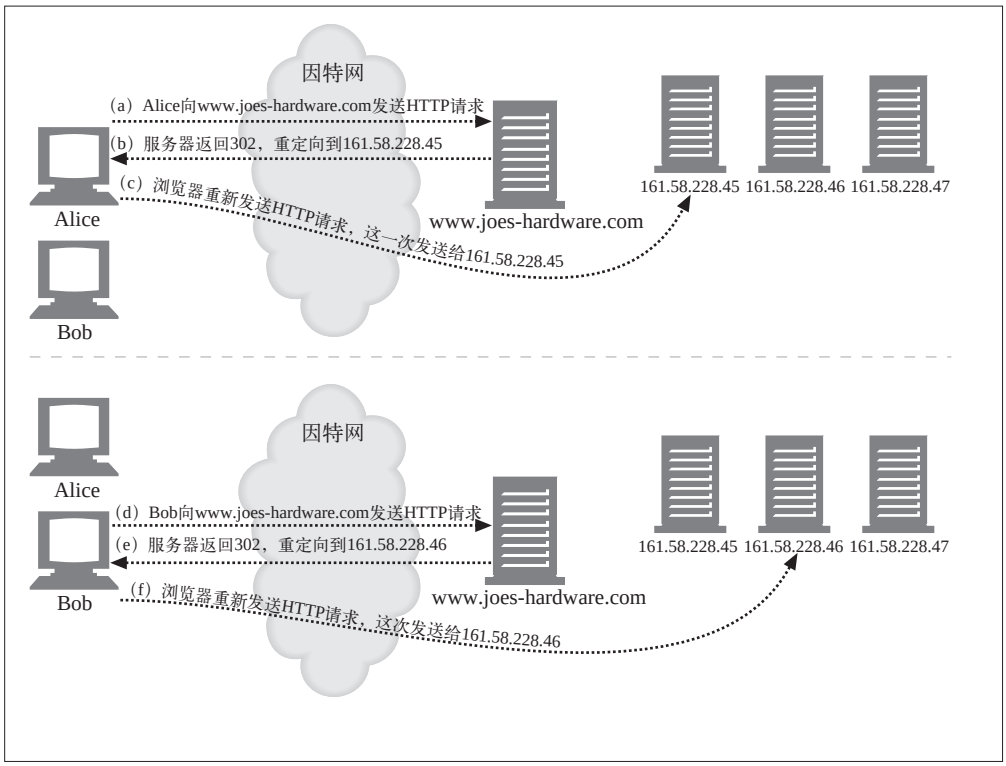


图 20-1　HTTP 重定向

HTTP 重定向可以在服务器间导引请求，但它有以下几个缺点。

- 需要原始服务器进行大量处理来判断要重定向到哪台服务器上去。有时，发布重定向所需的处理量几乎与提供页面本身所需的处理量一样。
- 增加了用户时延，因为访问页面时要进行两次往返。
- 如果重定向服务器出故障，站点就会瘫痪。

由于存在这些弱点，HTTP 重定向通常都会与其他一种或多种重定向技术结合使用。

20.4.2　DNS重定向

每次客户端试图访问 Joe 的五金商店的网站时，都必须将域名 www.joes-hardware.com 解析为 IP 地址。DNS 解析程序可能是客户端自己的操作系统，可能是客户端网络中的一台 DNS 服务器，或者是一台远距离的 DNS 服务器。DNS 允许将几个 IP 地址关联到一个域中，可以配置 DNS 解析程序，或对其进行编程，以返回可变的 IP 地址。解析程序返回 IP 地址时所基于的原则可以很简单（轮转），也可以很复杂（比如查看几台服务器上的负载，并返回负载最轻的服务器的 IP 地址）。

453

在图 20-2 中，Joe 为 www.joes-hardware.com 运行了 4 台服务器。DNS 服务器要决定为 www.joes-hardware.com 返回 4 个 IP 地址中的哪一个。最简单的 DNS 决策算法就是轮转。

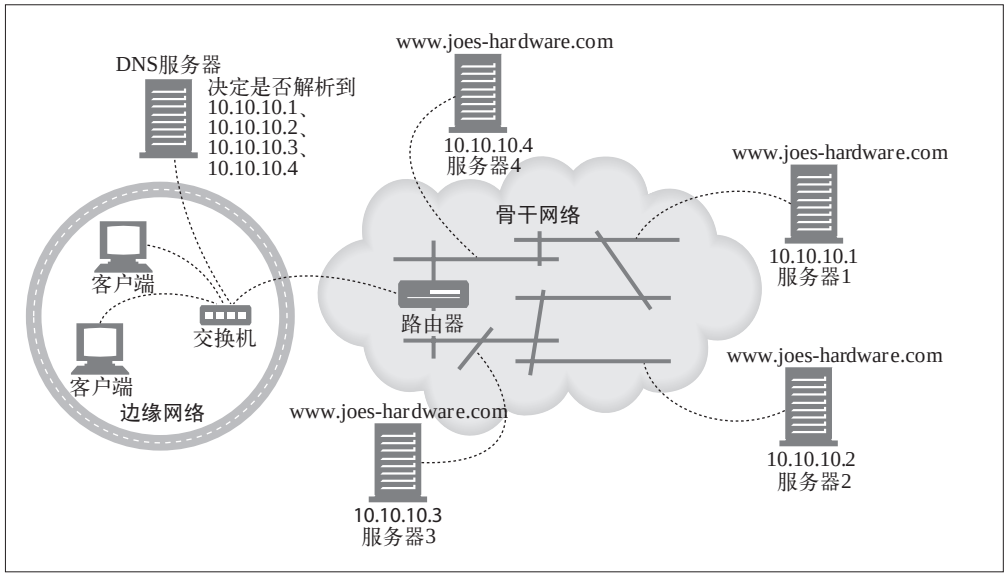


图 20-2　基于 DNS 的重定向

整个 DNS 解析过程的完整介绍，请参见本章末尾列出的 DNS 参考资料。

1. DNS轮转

最常见的重定向技术之一也是最简单的重定向技术之一。DNS 轮转使用了 DNS 主机名解析中的一项特性，在 Web 服务器集群中平衡负载。这是一种单纯的负载均衡策略，没有考虑任何与客户端和服务器的相对位置，或者服务器当前负载有关的因素。

我们来看看 CNN.com 实际上都做了些什么。我们在 2000 年 5 月初，用 Unix 中的工具 nslookup 来查找与 CNN.com 相关的 IP 地址。例 20-1 给出了结果。[2]

例 20-1　www.cnn.com 的 IP 地址

```
% nslookup www.cnn.com
Name: cnn.com
Addresses: 207.25.71.5, 207.25.71.6, 207.25.71.7, 207.25.71.8
207.25.71.9, 207.25.71.12, 207.25.71.20, 207.25.71.22, 207.25.71.23
207.25.71.24, 207.25.71.25, 207.25.71.26, 207.25.71.27, 207.25.71.28
```

454

注 2：DNS 的结果是在 2000 年 5 月 7 日，从北加州解析出来的。有些特定值可能会随时间发生变化，有些 DNS 系统还会根据客户端的位置返回不同的值。

```
207.25.71.29, 207.25.71.30, 207.25.71.82, 207.25.71.199, 207.25.71.245
207.25.71.246
Aliases: www.cnn.com
```

网站 www.cnn.com 实际上是 20 个不同的 IP 地址组成的集群。每个 IP 地址通常都意味着一台不同的物理服务器。

2. 多个地址及轮转地址的循环

大多数 DNS 客户端只会使用多地址集中的第一个地址。为了均衡负载，大多数 DNS 服务器都会在每次完成查询之后对地址进行轮转。这种地址轮转通常称作 DNS 轮转。

例如，对 www.cnn.com 进行三次连续的 DNS 查找可能会返回例 20-2 给出的那种 IP 地址轮转列表。

例 20-2　轮转 DNS 地址列表

```
% nslookup www.cnn.com
Name: cnn.com
Addresses: 207.25.71.5, 207.25.71.6, 207.25.71.7, 207.25.71.8
207.25.71.9, 207.25.71.12, 207.25.71.20, 207.25.71.22, 207.25.71.23
207.25.71.24, 207.25.71.25, 207.25.71.26, 207.25.71.27, 207.25.71.28
207.25.71.29, 207.25.71.30, 207.25.71.82, 207.25.71.199, 207.25.71.245
207.25.71.246

% nslookup www.cnn.com
Name: cnn.com
Addresses: 207.25.71.6, 207.25.71.7, 207.25.71.8, 207.25.71.9
207.25.71.12, 207.25.71.20, 207.25.71.22, 207.25.71.23, 207.25.71.24
207.25.71.25, 207.25.71.26, 207.25.71.27, 207.25.71.28, 207.25.71.29
207.25.71.30, 207.25.71.82, 207.25.71.199, 207.25.71.245, 207.25.71.246
207.25.71.5

% nslookup www.cnn.com
Name: cnn.com
Addresses: 207.25.71.7, 207.25.71.8, 207.25.71.9, 207.25.71.12
207.25.71.20, 207.25.71.22, 207.25.71.23, 207.25.71.24, 207.25.71.25
207.25.71.26, 207.25.71.27, 207.25.71.28, 207.25.71.29, 207.25.71.30
207.25.71.82, 207.25.71.199, 207.25.71.245, 207.25.71.246, 207.25.71.5
207.25.71.6
```

在例 20-2 中：

- 第一次 DNS 查找时的第一个地址为 207.25.71.5；
- 第二次 DNS 查找时的第一个地址为 207.25.71.6；
- 第三次 DNS 查找时的第一个地址为 207.25.71.7。

3. 用来平衡负载的DNS轮转

由于大多数 DNS 客户端只使用第一个地址，所以 DNS 轮转可以在多台服务器间提供负载均衡。如果 DNS 没有对地址进行轮转，大部分客户端就总是会将负载发送给第一台服务器。

图 20-3 说明了 DNS 轮转循环是如何平衡负载的。

- Alice 试图连接 www.cnn.com 时，会用 DNS 查找 IP 地址，得到 207.25.71.5 作为第一个 IP 地址。在图 20-3c 中，Alice 连接到 Web 服务器 207.25.71.5。
- Bob 随后试图连接 www.cnn.com 时，也会用 DNS 查找 IP 地址，但由于地址列表在 Alice 上次请求的基础上轮转了一个位置，所以他会得到一个不同的结果。Bob 得到 207.25.71.6 作为第一个 IP 地址，在图 20-3f 中它连接到了这台服务器上。

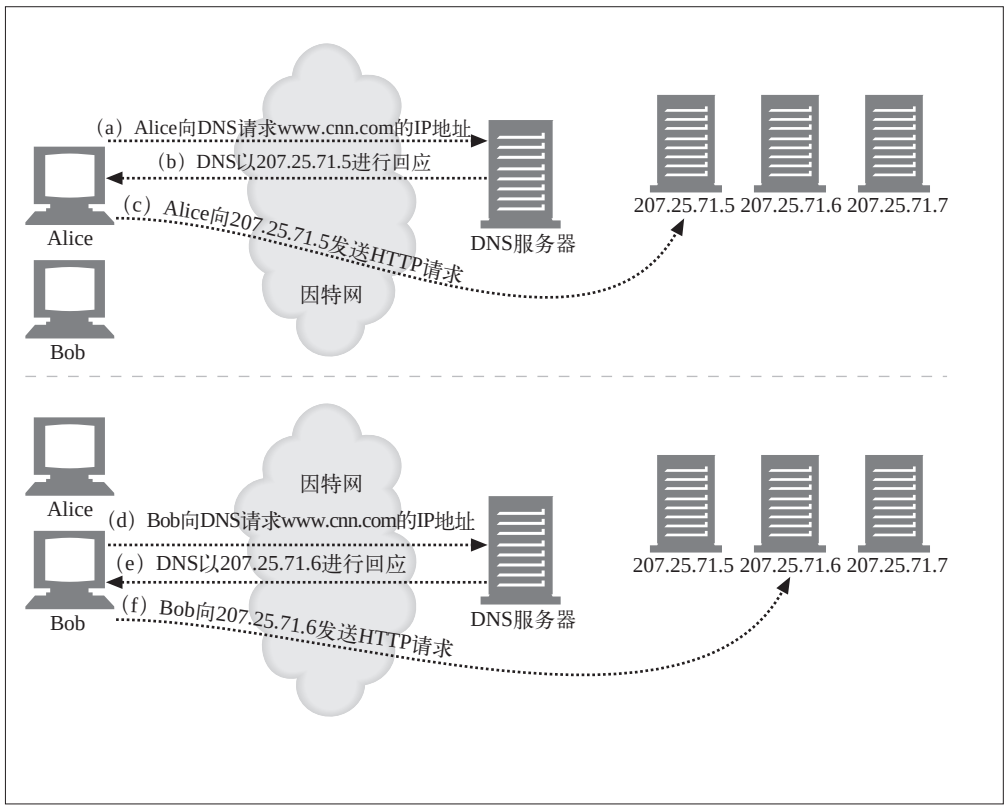


图 20-3　DNS 轮转在服务器集群的各台服务器之间进行负载均衡

4. DNS缓存带来的影响

DNS 对服务器的每次查询都会得到不同的服务器地址序列，所以 DNS 地址轮转会将负载分摊。但是这种负载均衡并不完美，因为 DNS 查找的结果可能会被记住，并被各种应用程序、操作系统和一些简易的子 DNS 服务器重用。很多 Web 浏览器都 456
会对主机进行 DNS 查找，然后一次次地使用相同的地址，以减少 DNS 查找的开销，而且有些服务器也更愿意保持与同一台客户端的联系。另外，很多操作系统都会自动进行 DNS 查找，并将结果缓存，但并不会对地址进行轮转。因此，DNS 轮转通常都不会平衡单个客户端的负载——一个客户端通常会在很长时间内连接到一台服务器上。

尽管 DNS 没有对单个客户端的事务进行跨服务器副本的处理，但在分散多个客户端的总负荷方面它做得相当好。只要有大量具有相同需求的客户端，就可以将负载合理地分散到各个服务器上去。

5. 其他基于DNS的重定向算法

我们已经讨论了 DNS 是如何对每条请求进行地址列表轮转的。但是，有些增强的 DNS 服务器会使用其他一些技术来选择地址的顺序。

- *负载均衡算法*
  有些 DNS 服务器会跟踪 Web 服务器上的负载，将负载最轻的 Web 服务器放在列表的最前面。

- *邻接路由算法*
  Web 服务器集群在地理上分散时，DNS 服务器会尝试着将用户导向最近的 Web 服务器。

- *故障屏蔽算法*
  DNS 服务器可以监视网络的状况，并将请求绕过出现服务中断或其他故障的地方。

通常，运行复杂服务器跟踪算法的 DNS 服务器就是在内容提供者控制之下的一个权威服务器（参见图 20-4）。

有一些分布式主机服务会使用这个 DNS 重定向模型。对于那些要查找附近服务器的服务来说，这个模型的一个缺点就是 , 权威 DNS 服务器只能用本地 DNS 服务器的 IP 地址，而不能用客户端的 IP 地址来做决定。

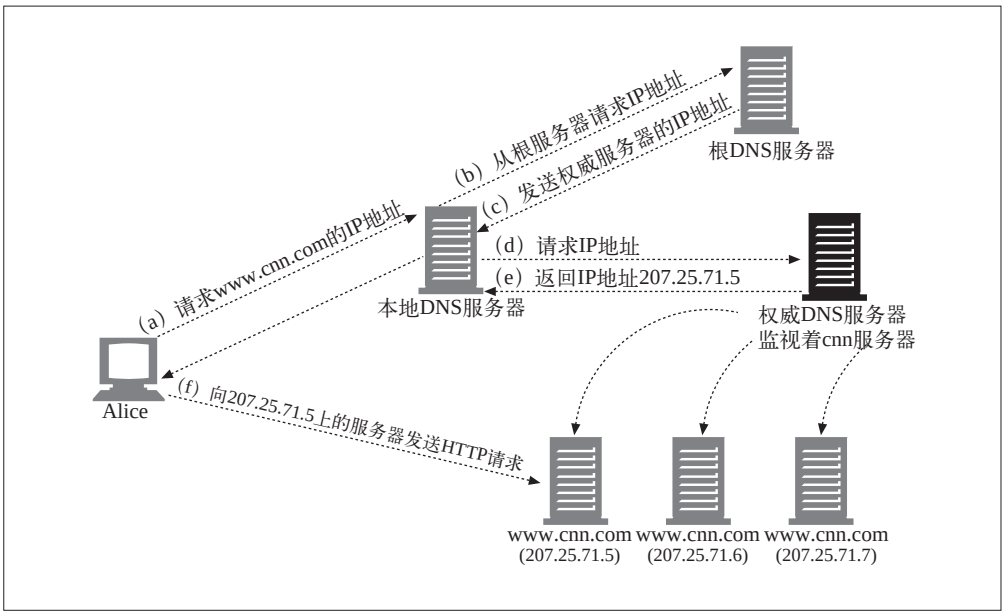


图 20-4　涉及权威服务器的 DNS 请求

20.4.3　任播寻址

在任播寻址中，几个地理上分散的 Web 服务器拥有完全相同的 IP 地址，而且会通过骨干路由器的“最短路径”路由功能将客户端的请求发送给离它最近的服务器。要使这种方法工作，每台服务器都要向邻近的骨干路由器广告，表明自己是一台路由器。Web 服务器会通过路由器通信协议与其邻近的骨干路由器通信。骨干路由器收到发送给任播地址的分组时，会（像平常一样）寻找接受那个 IP 地址的最近的“路由器”。由于服务器是将自己作为那个地址的路由器广告出去的，所以骨干路由器会将分组发送给服务器。

457

在图 20-5 中，三台服务器为同一个 IP 地址 10.10.10.1 服务。洛杉矶（LA）服务器将此地址广告给 LA 路由器，纽约（NY）服务器同样将此地址广告给 NY 路由器，以此类推。服务器会通过路由器协议与路由器进行通信。路由器会将目标为 10.10.10.1 的客户端请求自动地转发到广告这个地址的最近的服务器上去。在图 20-5 中，对 IP 地址 10.10.10.1 的请求会被转发给服务器 3。

任播寻址仍然是项实验性技术。要使用分布式任播技术，服务器就必须“使用路由器语言”，而且路由器必须能够处理可能出现的地址冲突，因为因特网地址基本上都是假定一台服务器只有一个地址的。（如果没有正确地实现，可能会造成很严重的“路由泄露”问题。）分布式任播是一种新兴技术，可以为那些自己控制骨干网络的

内容提供商提供一种解决方案。

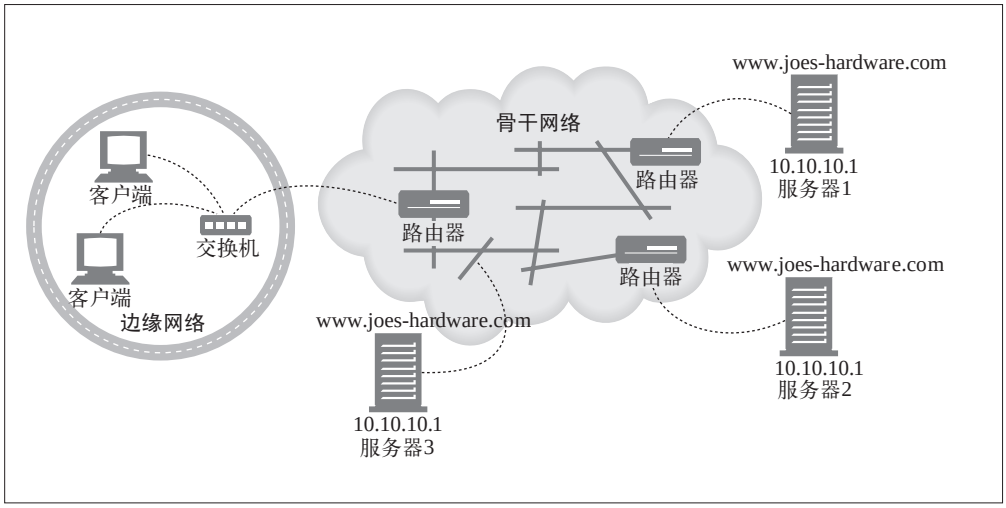


图 20-5　分布式任播寻址

20.4.4　IP MAC转发

在以太网中，HTTP 报文都是以携带地址的数据分组的形式发送的。每个分组都有一个第四层地址，由源 IP 地址、目的 IP 地址以及 TCP 端口号组成，它是第四层设备所关注的地址。每个分组还有一个第二层地址，MAC（Media Access Control，媒体访问控制）地址，这是第二层设备（通常是交换机和 Hub）所关注的地址。第二层设备的任务是接收具有特定输入 MAC 地址的分组，然后将其转发到特定的输出 MAC 地址上去。

比如，图 20-6 中交换机的程序会将来自 MAC 地址 MAC3 的所有流量都发送到 MAC 地址 MAC4 上去。

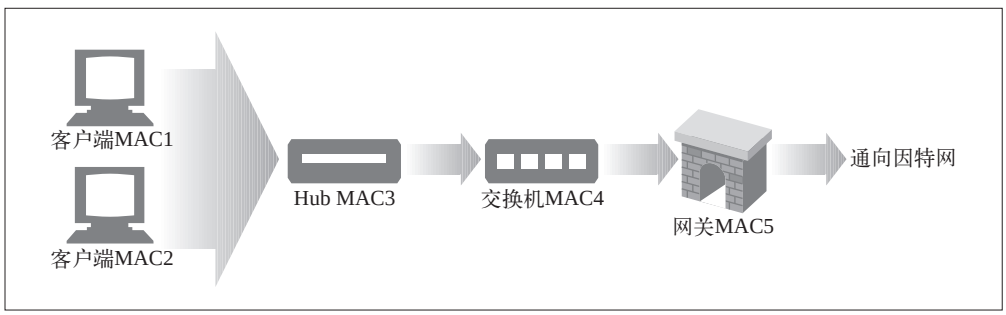


图 20-6　第二层交换机会将客户端请求发送给一个网关

第四层交换机能够检测出第四层地址（IP 地址和 TCP 端口号），并据此来选择路由。比如，一台第四层交换机可以将所有目的为端口 80 的 Web 流量都发送到某个代理上去。在图 20-7 中，编写交换机程序，将 MAC3 上所有端口 80 的流量都转发到 MAC6（代理缓存）上去。MAC3 上所有其他流量都会被转发到 MAC5 上去。

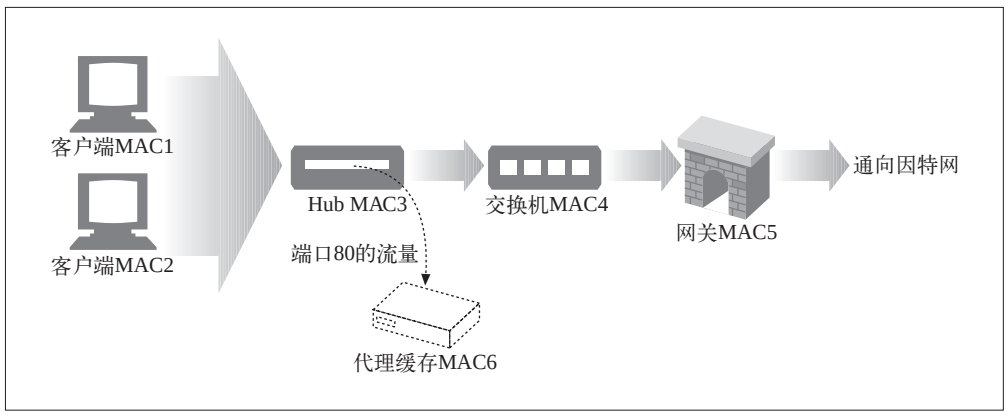


图 20-7　通过第四层交换机进行 MAC 转发

通常，如果缓存中有所请求的 HTTP 内容，而且是新鲜的，那么就由代理缓存来提供内容；否则，代理缓存就会代表客户端向此内容的原始服务器发送一条 HTTP 请求。交换机会将端口 80 的请求从代理（MAC6）发送给因特网网关（MAC5）。

支持 MAC 转发的第四层交换机通常会将请求转发给几个代理缓存，并在它们之间
459 平衡负载。类似地，也可以将 HTTP 流量转发给备用 HTTP 服务器。

因为 MAC 地址转发只是点对点的，所以服务器或代理只能位于离交换机一跳远的地方。

20.4.5　IP地址转发

在 IP 地址转发中，交换机或其他第四层设备会检测输入分组中的 TCP/IP 地址，并通过修改目的 IP 地址（不是目的 MAC 地址），对分组进行相应的转发。与 MAC 转发相比，这么做的优点是目标服务器不需要位于一跳远的地方；只需要位于交换机的上游就行了，而且通常第三层的端到端因特网路由都会将分组传送到正确的地方。这种类型的转发也被称为 NAT（Network Address Translation，网络地址转换）。

但还有一个问题，就是对称路由。从客户端接受输入 TCP 连接的交换机管理着连接；交换机必须通过那条 TCP 连接将响应回送给客户端。这样，所有来自目标服务器或代理的响应都必须返回给交换机（参见图 20-8）。

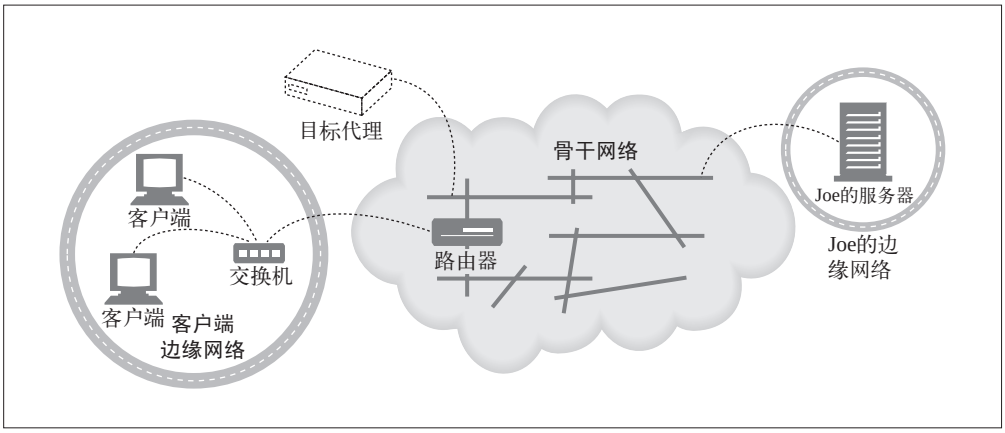


图 20-8　对代理缓存或镜像 Web 服务器进行 IP 地址转发的交换机

460

有以下两种方式可以控制响应的返回路径。

- 将分组的源 IP 地址改成交换机的 IP 地址。通过这种方式，无论交换机和服务器之间采用何种网络配置，响应分组都会被发送给交换机。这种方式被称为完全 NAT（full NAT），其中的 IP 转发设备会对目的 IP 地址和源 IP 地址都进行转换。图 20-9 显示了对一条 TCP/IP 数据报进行完全 NAT 的效果。这样做的缺点是服务器不知道客户端的 IP 地址，那种需要认证和计费的 Web 服务器无法获知客户端的 IP 地址。
- 如果源 IP 地址仍然是客户端的 IP 地址，就要确保（从硬件的角度来看）没有从服务器到客户端的直接路由（绕过交换机的）。这种方式有时被称为半 NAT（half NAT）。这种方法的优点是服务器知道客户端的 IP 地址，但缺点是要对客户端和服务器之间的整个网络都有某种程度的控制。

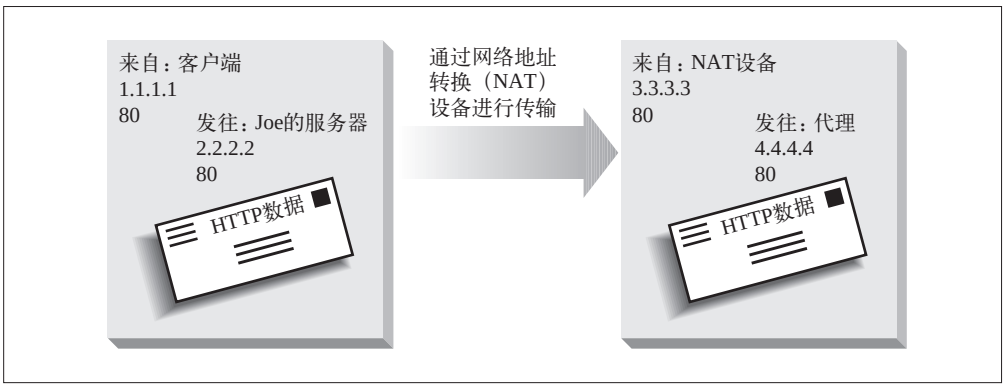


图 20-9　一条 TCP/IP 数据报的完全 NAT

20.4.6 网元控制协议

NECP（Network Element Control Protocol，网元控制协议）允许网元（NE，路由器和交换机等负责转发 IP 分组的设备）与服务器元素（SE，Web 服务器和代理缓存等提供应用层请求的设备）进行交互。NECP 并未显式提供对负载均衡的支持；它只是为 SE 提供了一种发送负载均衡信息给 NE 的方式，这样 NE 就可以在它认为合适的情况下进行负载均衡了。与 WCCP 一样，NECP 也提供了几种转发分组的方式：MAC 转发、GRE 封装和 NAT。

NECP 支持例外。SE 可以决定它不能为某些特定的源 IP 地址提供服务，并将这些地址发送给 NE。然后，NE 可以将来自这些 IP 地址的请求转发给原始服务器。

报文

461 表 20-3 描述了 NECP 报文。

表20-3 NECP报文

报　文	发 送 者	含　义
NECP_NOOP		不操作——什么都不做
NECP_INIT	SE	SE 初始化了与 NE 的通信。SE 打开与 NE 的 TCP 连接后，将此报文发送给它。SE 必须知道要连接哪个 NE 端口
NECP_INIT_ACK	NE	确认 NECP_INIT
NECP_KEEPALIVE	NE 或 SE	询问对等实体是否仍然活跃
NECP_KEEPALIVE_ACK	NE 或 SE	对 keep-alive 报文的应答
NECP_START	SE	SE 说：“我在这里，做好接受网络流量的准备了。”可以指定端口
NECP_START_ACK	NE	确认 NECP_START
NECP_STOP	SE	SE 告诉 NE“停止向我发送流量。”
NECP_STOP_ACK	NE	NE 确认 NECP_STOP 操作
NECP_EXCEPTION_ADD	SE	SE 说要向 NE 列表中添加一个或多个例外。例外可以基于源 IP 地址、目的 IP 地址、（IP 之上的）协议，或者端口
NECP_EXCEPTION_ADD_ACK	NE	对 NECP_EXCEPTION_ADD 进行证实
NECP_EXCEPTION_DEL	SE	请求 NE 从列表中删除一个或多个例外
NECP_EXCEPTION_DEL_ACK	NE	对 NECP_EXCEPTION_DEL 进行证实
NECP_EXCEPTION_RESET	SE	请求 NE 删除整个例外列表
NECP_EXCEPTION_RESET_ACK	NE	对 NECP_EXCEPTION_RESET 进行证实
NECP_EXCEPTION_QUERY	SE	查询 NE 的整个例外列表
NECP_EXCEPTION_RESP	NE	响应例外查询

20.5 代理的重定向方法

到目前为止，我们已经讨论过通用的重定向方法了。（出于潜在的安全考虑）内容也可能需要通过各种代理来访问，或者网络中可能有一个客户端可利用的代理缓存。（因为获取已缓存的内容很可能要比直接连接到原始服务器快得多。）

但 Web 浏览器客户端怎么才会知道要连接到某个代理上去呢？可以用 3 种方法来判断：显式的浏览器配置，动态自动配置以及透明拦截。我们会在本节中讨论这 3 种技术。

代理可以顺次将客户端请求重定向到另一个代理上去。比如，没有缓存此内容的代理缓存可能会选择将客户端重定向到另一个代理缓存。这样一来，响应就会来自与客户端请求资源的地址不同的另外一个地址，所以，我们还会讨论几种用于对等代理 – 缓存重定向的协议：ICP、CARP 和 HTCP。

462

20.5.1 显式浏览器配置

大多数浏览器都可以配置为从代理服务器上获取内容——浏览器中有一个下拉菜单，用户可以在这个菜单中输入代理的名字或 IP 地址以及端口号。然后浏览器的所有请求都可以发送给这个代理。有些服务提供商不允许用户配置普通浏览器来使用代理，它们会要求用户下载事先配置好的浏览器。这些浏览器知道所要使用的代理的地址。

显式浏览器配置有以下两个主要的缺点。

- 配置为使用代理的浏览器，即使在代理无法响应的情况下，也不会去联系原始服务器。如果代理崩溃了，或者没有正确配置浏览器，用户就会遇到连接方面的问题。
- 对网络架构进行修改，并将这些修改通知给所有的终端用户都是很困难的。如果服务提供商要添加更多的代理服务器，或者使其中一些退出服务，用户都要修改浏览器代理设置。

20.5.2 代理自动配置

显式配置浏览器使其联系特定的代理，这样会限制网络架构方面的变动，因为它是靠用户来介入并重新配置浏览器的。自动配置方式可以动态配置浏览器，连接到正确的代理服务器，以解决这个问题。这种方法已经实现了，被称为代理自动配置（PAC）协议。PAC 是网景公司定义的，网景公司的 Navigator 和微软的 Inernet Explorer 浏览器都支持此协议。

PAC 的基本思想是让浏览器去获取一个称为 PAC 的特殊文件，这个文件说明了每个 URL 所关联的代理。必须配置浏览器，为这个 PAC 文件关联一个特定的服务器。这样，浏览器每次重启的时候都可以获取这个 PAC 文件了。

PAC 文件是个 JavaScript 文件，其中必须定义函数：

```
function FindProxyForURL(url, host)
```

如下所示，浏览器要为请求的每条 URL 调用这个函数：

```
return_value = FindProxyForURL(url_of_request, host_in_url);
```

其返回值为一个字符串，用来说明浏览器应该到哪里请求这个 URL。返回值可以是所关联的代理名称列表（比如，PROXY proxy1.domain.com，PROXY proxy2.domain.com），或者是字符串 "DIRECT"，这个字符串说明浏览器应该绕开所有的代理，直接连接原始服务器。

图 20-10 给出了浏览器对 PAC 文件的请求以及响应此请求的操作顺序。在本例中，服务器回送了带有 JavaScript 程序的 PAC 文件。JavaScript 程序中有一个 FindProxyForURL 函数，用来告知浏览器，如果所请求的 URL 的主机位于 netscape.com 域中，就直接与原始服务器联系，所有其他请求都连接到 proxy1.joes-cache.com。浏览器会为它所请求的每个 URL 调用这个函数，并根据此函数返回的结果进行连接。

463

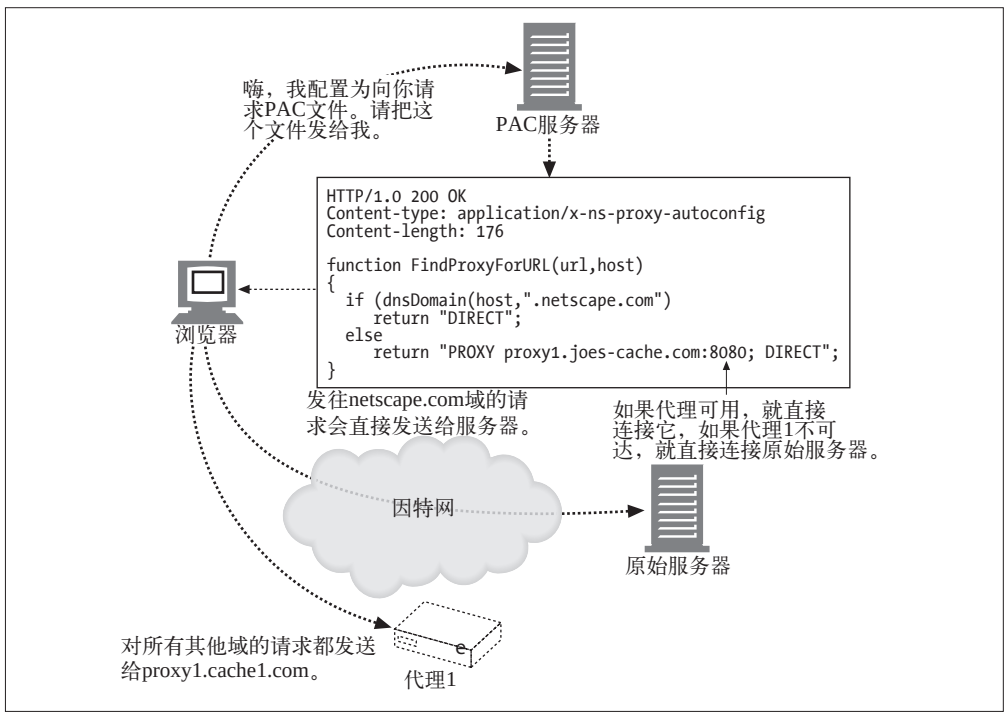


图 20-10　代理自动配置

PAC 协议是相当强大的：JavaScript 程序可以请求浏览器根据大量与主机名相关的参数来选择代理，比如 DNS 地址和子网，甚至星期几或具体时间。只要服务器中的 PAC 文件保持更新，能反映代理位置的变化，PAC 就允许浏览器根据网络结构的变化自动与合适的代理进行联系。PAC 存在的主要问题是必须要对浏览器进行配置，让它知道要从哪个服务器获取 PAC 文件，因此它就是一个全自动配置的系统。下一节讨论的 WPAD 解决了这个问题。

就像那些预配置浏览器一样，现在一些主要的 ISP 都在使用 PAC。

20.5.3 Web代理自动发现协议

WPAD（Web 代理自动发现协议）的目标是在不要求终端用户手工配置代理设置，而且不依赖透明流量拦截的情况下，为 Web 浏览器提供一种发现并使用附近代理的方式。由于可供选择的发现协议有很多，而且不同浏览器的代理使用配置也存在差异，因此定义 Web 代理自动发现协议时，普通的问题会被复杂化。

464

本节包含了一个经过缩略，且重新组织过的 WPAD 因特网草案版本。现在，这个草案是作为 IETF 的 Web 中间人工作组的一部分开发的。

1. PAC文件自动发现

WPAD 允许 HTTP 客户端定位一个 PAC 文件，并使用这个 PAC 文件找到适当的代理服务器的名字。WPAD 不能直接确定代理服务器的名字，因为这样就无法使用 PAC 文件提供的附加功能了（负载均衡，请求路由到一组服务器上去，故障时自动转移到备用代理服务器等）。

如图 20-11 所示，WPAD 协议发现了 PAC 文件 URL，这个 URL 也被称为配置 URL（CURL）。PAC 文件执行了一个 JavaScript 程序，这个程序会返回合适的代理服务器地址。

实现 WPAD 协议的 HTTP 客户端：

- 用 WPAD 找到 PAC 文件的 CURL；
- 根据这个 CURL 获取 PAC 文件（又名配置文件或 CFILE）；
- 执行 PAC 文件来确定代理服务器；
- 向 PAC 文件返回的那个代理服务器发送 HTTP 请求。

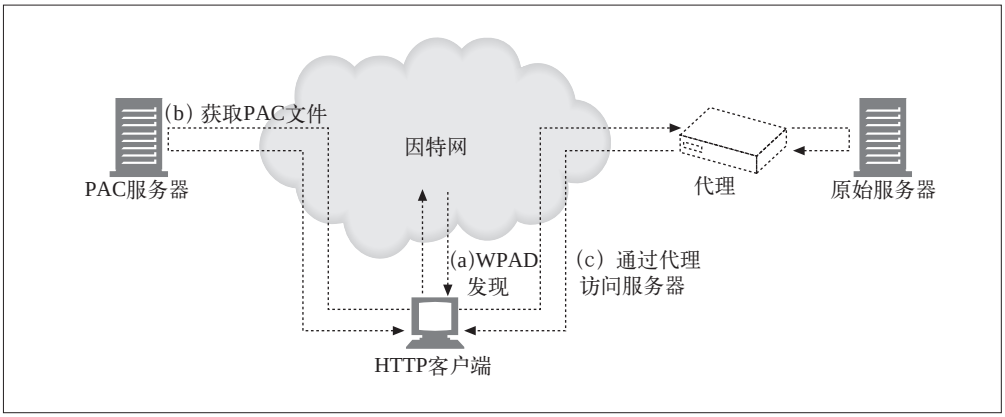


图 20-11　WPAD 确定了 PAC URL，这个 PAC 文件确定了代理服务器

2. WPAD算法

WPAD 使用了一系列资源发现技术来确定适当的 PAC 文件 CURL。并不是所有的
465 组织都可以使用所有技术的，所以 WPAD 指定了多种发现技术。在成功获得 CURL 之前，WPAD 客户端会一个个地尝试每种技术。

当前的 WPAD 规范按序定义了下列技术：

- DHCP（动态主机配置协议）；
- SLP（服务定位协议）；
- DNS 知名主机名；
- DNS SRV 记录；
- DNS TXT 记录中提供的服务 URL。

在这 5 种机制中，要求 WPAD 客户端必须支持 DHCP 和 DNS 知名主机名技术。我们会在后继小节中提供更多的细节。

WPAD 客户端会按顺序用上面提供的发现机制发送一系列资源发现请求。客户端只会尝试它们所支持的机制。只要某次发现尝试成功了，客户端就会用得到的信息来构建 PAC CURL。

如果从那个 CURL 上成功获取到 PAC 文件，这个过程就结束了。如果没有，客户端就从它在预定义的资源发现请求系列里中断的地方开始恢复。如果尝试了所有的发现机制之后，都没有获取到 PAC 文件，WPAD 协议就失败了，客户端会配置为不使用代理服务器。

客户端首先会尝试 DHCP，然后是 SLP。如果没有获取到 PAC 文件，客户端会继续

执行那些基于 DNS 的机制。

客户端会在 DNS SRV、知名主机名和 DNS TXT 记录等方法中循环多次。每次都使 DNS 查询的 QNAME 变得越来越不具体。通过这种方式，客户端就可以定位出尽可能具体的配置信息，但也可能会转而使用一些不太具体的信息。每次 DNS 查找都会在 QNAME 前加上 wpad，用以说明请求的资源类型。

考虑主机名为 johns-desktop.development.foo.com 的客户端。下面是一个完整的 WPAD 客户端会执行的发现尝试顺序：

- DHCP；
- SLP；
- 用 QNAME=wpad.development.foo.com 进行 DNS A 查找；
- 用 QNAME=wpad.development.foo.com 进行 DNS SRV 查找；
- 用 QNAME=wpad.development.foo.com 进行 DNS TXT 查找；
- 用 QNAME=wpad.foo.com 进行 DNS A 查找；
- 用 QNAME=wpad.foo.com 进行 DNS SRV 查找；
- 用 QNAME=wpad.foo.com 进行 DNS TXT 查找。

466

说明整个操作过程的详细伪代码请参见 WPAD 规范。后面的小节将讨论两种必备机制——DHCP 和 DNS A 查找。有关 CURL 发现方法的其他详细内容参见 WPAD 规范。

3. 用DHCP进行CURL发现

要使用这种机制，就必须将 CURL 存储在 WPAD 客户端可以查询的 DHCP 服务器上。WPAD 客户端可以通过向 DHCP 服务器发送 DHCP 查询来获取 CURL。（如果 DHCP 服务器中配置了这种信息，）就可以在 DHCP 可选代码 252 中获取 CURL。所有 WPAD 客户端实现都必须支持 DHCP。RFC 2131 详细介绍了 DHCP 协议。现存的 DHCP 选项列表参见 RFC 2132。

如果 WPAD 客户端已经在其初始化过程中执行了 DHCP 查询，DHCP 服务器可能就已经提供了那个值。如果无法通过客户端 OS API 获得这个值，客户端就向 DHCP 服务器发送一条 DHCPINFORM 报文，以获取这个值。

WPAD 的 DHCP 可选代码 252 为 STRING 类型，可以是任意长度。这个字符串中包含了一个指向适当 PAC 文件的 URL。比如：

```
"http://server.domain/proxyconfig.pac"
```

4. DNS A记录查找

要让这种机制工作，就必须将合适的代理服务器的IP地址存储在WPAD客户端可以查询的DNS服务器上。WPAD客户端会向DNS服务器发送一个A记录查询，以获取CURL。成功查询的结果中会包含合适的代理服务器的IP地址。

WPAD客户端实现必须支持这种机制。这应该是很简单的，因为它只要求基本的DNS A记录查找。用知名DNS别名进行资源发现的详细过程请参见RFC 2219。对WPAD来说，规范使用了“wpad”的“知名别名”来进行Web代理自动发现。

客户端执行了下列DNS查找：

```
QNAME=wpad.TGTDOM., QCLASS=IN, QTYPE=A
```

成功的查找中包含了IP地址，WPAD客户端根据这个地址构建CURL。

5. 获取PAC文件

只要创建了候选的CURL，WPAD客户端通常都会向CURL发送一条GET请求。发出请求时，WPAD客户端必须要发送一些带有适当CFILE格式信息的`Accept`首
467 部，这些CFILE格式都是它们所能处理的。比如：

```
Accept: application/x-ns-proxy-autoconfig
```

而且，如果CURL的结果是要进行重定向，客户端就必须跟随这些重定向到其最终目的地。

6. 何时执行WPAD

至少要在出现以下情况的时候进行Web代理自动发现。

- 在Web客户端启动的时候——WPAD只在第一个实例启动的时候执行。后面的实例会继承这种设置。
- 只要有来自网络栈的通知，就说明客户端主机的IP地址改变了。

哪个选项在其环境中有意义，Web客户端就可以选择哪个。而且，客户端还必须根据HTTP的过期时间，为之前下载的PAC文件的过期时间尝试一个发现周期。PAC文件过期时，客户端遵循过期时间，重新运行WPAD过程是很重要的。

如果PAC文件没有提供替换方案，在当前配置的代理失效的情况下，客户端还可以选择重新运行WPAD过程。

只要客户端决定使当前的PAC文件失效，就必须重新运行整个WPAD协议，以确

保它会发现当前正确的 CURL。具体来说，就是协议不能有条件地获取 PAC 文件的 `If-Modified-Since`。

WPAD 协议广播与 / 或多播通信可能需要大量的网络环回时间。WPAD 协议的激活频率不应该高于上面指定的频率（比如在每次获取 URL 时进行一次）。

7. WPAD欺骗

WPAD 的 IE 5 实现允许 Web 客户端在没有用户干预的情况下，自动检测代理设置。WPAD 使用的算法会在全称域名前加上主机名“wpad”，并会逐渐删除子域名，直到它找到能够响应主机名的 WPAD 服务器，或到达第三级域名。比如，域 a.b.microsoft.com 中的 Web 客户端会先查询 wpad.a.b.microsoft、wpad.b.microsoft.com，然后再查询 wpad.microsoft.com。

这样会暴露出一个安全漏洞，因为在国际应用（及其他特定的配置）中，第三级域名可能是不可信的。恶意用户可以建立一个 WPAD 服务器，并提供他选中的代理配置命令。后继（5.01 及以后）的 IE 版本修正了这个问题。

8. 超时

WPAD 会经过多个级别的发现，客户端必须确保每个阶段都有时限保证。可能的情况下，将每个阶段都限制在 10 秒以内是比较合理的，但实现者可能会选择其他更适合其网络特性的值。比如，运行在无线网络上的设备实现，由于带宽较低或时延较长，可能就会使用更大的时限。

468

9. 管理者的考虑

管理者至少应该在其环境中配置 DHCP 或 DNS A 记录查找方式中的一种，因为只有这两种方式是所有兼容客户端都必须实现的。除此之外，通过配置环境使其支持搜索列表中顺序靠前的机制，可以缩短客户端的启动时间。

使用这种协议结构的主要动力之一是支持客户端定位附近的代理服务器。在很多环境中，都会有多个代理服务器（工作组、公司网关、ISP、骨干网等）。

在 WPAD 框架结构中，可以在很多地方确定代理服务器是否“邻近”。

- 不同子网的 DHCP 服务器会返回不同的答案。还可以根据客户端的 `cipaddr` 字段或客户端标识符选项作出决定。
- 可以对 DNS 服务器进行配置，使其为不同的域名后缀（比如，QNAME wpad.marketing.bigcorp.com 和 wpad.development.bigcorp.com）返回不同的 SRV/A/TXT 资源记录（RR）。

- 处理 CURL 请求的 Web 服务器会根据 User-Agent 首部、Accept 首部、客户端 IP 地址 / 子网 / 主机名、附近代理服务器的拓扑分布等作出决定。可能由处理 CURL 的 CGI 可执行文件进行这种处理。如前所述，甚至可能是某个处理 CURL 请求的代理服务器来作出这些决定。
- PAC 文件的表达能力可能足以在客户端运行时从一组候选的代理服务器中进行选择。CARP 就是在此基础上实现缓存阵列的。PAC 文件可以计算出到一组候选代理服务器的网络距离（或其他合理的度量方式），并选择“最近”或“响应最积极”的服务器，这并不是什么不可思议的事情。

20.6 缓存重定向方法

我们已经讨论过一些将流量重定向到通用服务器的技术，以及一些将流量导向代理或网关的专用技术了。这一节会介绍一些更复杂的、用于缓存代理服务器的重定向技术。这些技术要尽量做到可靠、高效且能感知内容——这样可以将请求分配到可
469 能包含特定内容的位置上去，因此比前面讨论过的那些协议更复杂。

WCCP重定向

Cisco 系统公司开发的 WCCP 可以使路由器将 Web 流量重定向到代理缓存中去。WCCP 负责路由器和缓存服务器之间的通信，这样路由器就可以对缓存进行验证（确保它们已启动且正在运行），在缓存之间进行负载均衡，并将特定类型的流量发送给特定的缓存了。WCCP 版本 2（WCCP2）是个开放的协议。这里我们会探讨 WCCP2。

1. WCCP重定向是怎样工作的

下面是 WCCP 重定向在 HTTP 上工作过程的概述（WCCP 对其他协议的重定向过程也是类似的）。

- 启动包含了一些支持 WCCP 的路由器和缓存的网络，这些路由器和缓存之间可以相互通信。
- 一组路由器及其目标缓存构成一个 WCCP 服务组。服务组的配置说明了要将何种流量发往何处、流量是如何发送的以及如何在服务组的缓存之间进行负载均衡。
- 如果服务组配置为重定向 HTTP 流量，服务组中的路由器就会将 HTTP 请求发送给服务组中的缓存。
- HTTP 请求抵达服务组中的路由器时，路由器会（根据对请求 IP 地址的散列，或者“掩码 / 值”的配对策略）选择服务组中的某个缓存为请求提供服务。

- 路由器向缓存发送请求分组，可以用缓存的 IP 地址来封装分组，也可以通过 IP MAC 转发来实现。
- 如果缓存无法为请求提供服务，就将分组返回给路由器进行普通的转发。
- 服务组中的成员会互相交换心跳报文，不断验证对方的可用性。

2. WCCP2报文

WCCP2 报文有 4 种，如表 20-4 所示。

表20-4　WCCP2报文

报　文　名	发　送　者	所承载的信息
WCCP2_HERE_I_AM	从缓存发送给路由器	这些报文告诉路由器，缓存可以接收流量。报文中包含了该缓存的服务组的所有信息。只要有缓存加入了服务组，它就会将这些报文发送给组里所有的路由器。可以通过这些报文与发送 WCCP2_I_SEE_YOU 报文的路由器进行沟通
WCCP2_I_SEE_YOU	从路由器发送给缓存	这些报文是对 WCCP2_HERE_I_AM 报文的响应。可以通过这些报文对分组转发方式、分配方法（哪个是指定的缓存）、分组返回方法和安全性进行沟通
WCCP2_REDIRECT_ASSIGN	从指定的缓存发送给路由器	这些报文为负载均衡分配任务，它们可以为散列表负载均衡发送桶信息，或为“掩码 / 值”负载均衡发送“掩码 / 值”组对信息
WCCP2_REMOVAL_QUERY	从路由器发送给在 2.5 × HERE_I_AM_T 秒内没有发送过 WCCP2_HERE_I_AM 报文的缓存	如果路由器没有周期性地收到 WCCP2_HERE_I_AM 报文，路由器就会发送此报文，以查看是否应该将这个缓存从服务组中删除掉。来自缓存的正确响应为三条完全相同的 WCCP2_HERE_I_AM 报文，中间间隔 HERE_I_AM_T/10 秒

470

WCCP2_HERE_I_AM 的报文格式为：

```
WCCP Message Header
Security Info Component
Service Info Component
Web-cache Identity Info Component
Web-cache View Info Component
Capability Info Component （可选）
Command Extension Component （可选）
```

WCCP2_I_SEE_YOU 的报文格式为：

```
WCCP Message Header
Security Info Component
```

```
Service Info Component
Router Identity Info Component
Router View Info Component
Capability Info Component （可选）
Command Extension Component （可选）
```

WCCP2_REDIRECT_ASSIGN 的报文格式为：

```
WCCP Message Header
Security Info Component
Service Info Component
Assignment Info Component, or Alternate Assignment Component
```

WCCP2_REMOVAL_QUERY 的报文格式为：

```
WCCP Message Header
Security Info Component
Service Info Component
Router Query Info Component
```

3. 报文组件

每条 WCCP2 报文都由一个首部和一些组件构成。WCCP 首部信息包含报文类型（Here I Am、I See You、Assignment 或 Removal Query）、WCCP 版本和报文长度（不包括首部的长度）。

每个组件都以一个描述组件类型和长度的 4 字节首部开始。组件长度不包括组件首
471 部的长度。报文组件如表 20-5 所述。

表20-5　WCCP2报文组件

组　件	描　述
安全信息	包含安全选项和安全实现。安全选项可以是： WCCP2_NO_SECURITY (0) WCCP2_MD5_SECURITY (1) 如果选项是不安全的（NO），安全实现字段就不存在。如果选项为 MD5，安全实现字段就是一个包含了报文校验和、服务组密码的 16 字节的字段。密码不能超过 8 个字节
服务信息	描述服务组。服务类型 ID 可以使用两个值： WCCP2_SERVICE_STANDARD (0) WCCP2_SERVICE_DYNAMIC (1) 如果服务类型是标准的（STANDARD），服务就是完全由服务 ID 定义的知名服务。HTTP 就是个知名服务。如果服务类型是动态的（DYNAMIC），就由下列设置来定义服务：优先级、协议、服务标记（决定是否散列）以及端口
路由器身份信息	包含路由器的 IP 地址和 ID，并（通过 IP 地址）列出了路由器想要通信的所有 Web 缓存

（续）

组　　件	描　　述
Web 缓存身份信息	包含了 Web 高速缓存的 IP 地址和重定向散列表映射
路由器视图信息	包含了路由器的服务组视图（路由器和缓存的身份）
Web 缓存视图信息	包含了 Web 缓存的服务组视图
分配信息	显示了如何将 Web 缓存分配到特定的散列桶中去
路由器查询信息	包含了路由器的 IP 地址、所要查询的 Web 缓存的地址以及服务组中最近从 Web 缓存中收到 Here I Am 报文的路由器 ID
能力信息	那些需要广告所支持的分组转发方式、负载均衡和分组返回方式的路由器会使用此信息。那些要让路由器知道它希望使用哪种方法的 Web 缓存也会使用此信息
替换分配	包含了负载均衡的散列表分配信息
分配图	包含了服务组的“掩码 / 值”设置元素
命令扩展	Web 缓存用它来告诉路由器它们正在关机。路由器用它来确认缓存是否关闭

4. 服务组

服务组（service group）由一组支持 WCCP 的路由器和缓存组成，它们之间可以交换 WCCP 报文。路由器会向服务组中的缓存发送 Web 流量。服务组的配置确定了如何将流量分配到服务组的缓存中去。路由器和缓存会在 Here I Am 和 I See You 报文中交换服务组的配置信息。

5. GRE分组封装

支持 WCCP 的路由器会用服务器的 IP 地址将 HTTP 分组封装起来，将其重定向到特定的服务器上去。分组封装中还包含了 IP 首部的 proto 字段，用来说明通用路由器封装（GRE）。proto 字段的存在告诉接收代理，它有一个封装的分组。分组被封装起来，客户端的 IP 地址就不会丢失了。图 20-12 显示了 GRE 分组的封装过程。 472

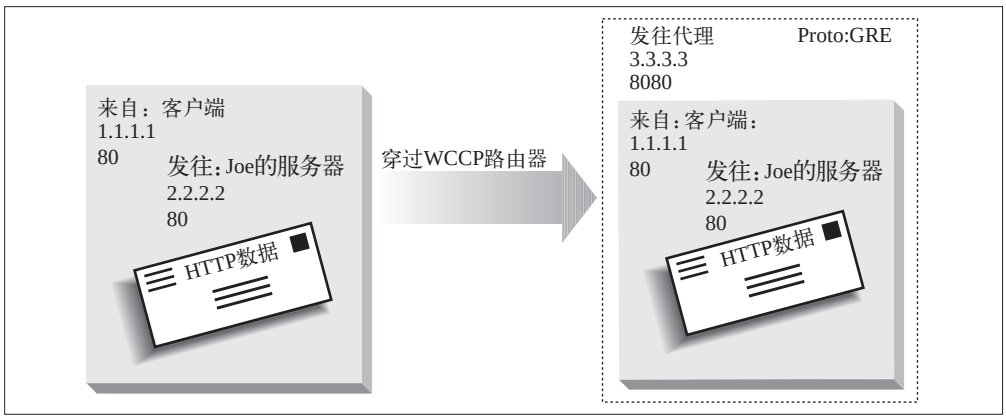


图 20-12　WCCP 路由器如何修改 HTTP 分组的目的 IP 地址

6. WCCP的负载均衡

除了路由功能之外，WCCP 路由器还可以在几个接收服务器之间进行负载均衡。WCCP 路由器及其接收服务器会交换心跳报文（heartbeat message），以便相互通知自己处于启动运行状态。如果某特定接收服务器停止发送心跳报文，WCCP 路由器就会将请求流量直接发送到因特网上，而不会将其重定向给那个节点。节点重新提供服务时，WCCP 路由器会再次开始接收心跳报文，并继续向节点发送请求流量。

20.7 因特网缓存协议

ICP（因特网缓存协议）允许缓存在其兄弟缓存中查找命中内容。如果某个缓存中没有 HTTP 报文所请求的内容，它可以查明内容是否在附近的兄弟缓存中，如果在，就从那里获取内容，以避免查询原始服务器而带来的更多开销。可以把 ICP 当作一个缓存集群协议。HTTP 请求报文的最终目的地可以通过一系列的 ICP 查询确定，从这个角度来说，它就是一个重定向协议。

ICP 是一个对象发现协议。它会同时去询问附近的多个缓存，看看它们的缓存中是否有特定的 URL。附近的缓存如果有那个 URL 的话，就会返回一个简短的报文 HIT，如果没有，就返回 MISS。然后，缓存就可以打开一条到拥有此对象的邻居缓存的 HTTP 连接了。

ICP 是很简单直接的。ICP 报文是一个以网络字节序表示的 32 位封装结构，这样更
473 便于进行解析。为了提高效率，可以由 UDP 数据报承载其报文。UDP 是一种不可靠的因特网协议，说明在传输的过程中数据可能会被破坏，因此使用 ICP 的程序要具有超时功能，以检测丢失的数据报。

下面简要描述一下 ICP 报文中的部分信息。

- Opcode（操作码）
  Opcode 是个 8 位的二进制值，用以描述 ICP 报文的含义。基本的 opcode 包括 ICP_OP_QUERY 请求报文和 ICP_OP_HIT 和 ICP_OP_MISS 响应报文。
- 版本
  8 位的版本号描述了 ICP 协议的版本编号。Squid 使用的 ICP 版本记录在 RFC 2186 第 2 版中。
- 报文长度
  以字节为单位的 ICP 报文总长。因为只有 16 位，所以 ICP 报文的长度不能超过 16 383 字节。URL 通常都小于 16 KB，如果超过这个长度，很多 Web 应用程序就无法处理它了。

- 请求编号
  支持 ICP 的缓存会用请求编号来记录多个同时发起的请求和响应。ICP 应答报文数必须与触发应答的 ICP 请求报文数相同。

- 选项
  32 位的 ICP 选项字段是个包含了若干标记的位矢量，这些标记可用来修改 ICP 的行为。ICPv2 定义了两个标记，这两个标记都会修改 ICP_OP_QUERY 请求。ICP_FLAG_HIT_OBJ 标记用来启动或禁止在 ICP 响应中返回文档数据。ICP_FLAG_SRC_RTT 标记请求由兄弟缓存测量的、到原始服务器的环回时间的估计值。

- 可选数据
  保留了 32 位的可选数据用于可选特性。ICPv2 使用了可选数据的低 16 位来装载从兄弟缓存到原始服务器的可选环回时间的估计值。

- 发送端主机地址
  承载了报文发送端 32 位 IP 地址的著名字段。实际中并未使用。

- 净荷
  净荷内容的变化取决于报文的类型。对 ICP_OP_QUERY 来说，净荷是一个 4 字节的原始请求端主机地址，后面跟着一个由 NUL 结尾的 URL。对 ICP_OP_HIT_OBJ 来说，净荷是一个由 NUL 结尾的 URL，后面跟着一个 16 位的对象长度，接着是对象数据。

  更多有关 ICP 的信息，请参见 RFC 2186 和 RFC 2187。从美国应用网络研究国家实验室的网站上（http://www.nlanr.net/Squid/）也可以获得一些很棒的有关 ICP 和对等实体的参考文献。

474

20.8 缓存阵列路由协议

代理服务器通过拦截来自单个用户的请求，提供所请求 Web 对象的缓存副本，极大地降低了发往因特网的流量。但随着用户数的增加，大量流量可能会使代理服务器自身超载。

对此问题的一种解决方案就是使用多个代理服务器将负载分散到一组服务器上。CARP（缓存阵列路由协议）是微软公司和网景公司提出的一个标准，通过这个协议来管理一组代理服务器，使这组代理服务器对用户来说就像一个逻辑缓存一样。

CARP 是 ICP 的一个替代品。CARP 和 ICP 都允许管理者通过使用多个代理服务器来提高性能。本节讨论了 CARP 与 ICP 的区别，用 CARP 代替 ICP 的优缺点以及

CARP 协议实现上的一些技术细节。

ICP 中出现缓存未命中时，代理服务器会用 ICP 报文格式来查询附近的缓存，以确定 Web 对象是否存在。附近的缓存会以 HIT 或 MISS 进行响应，请求代理服务器会用这些响应来选择能够获取到对象的最适当的位置。如果 ICP 代理服务器是以层次结构排列的，未命中的查询会被提交给其父代理。图 20-13 以图形方式显示了如何通过 ICP 来解决命中和未命中的问题。

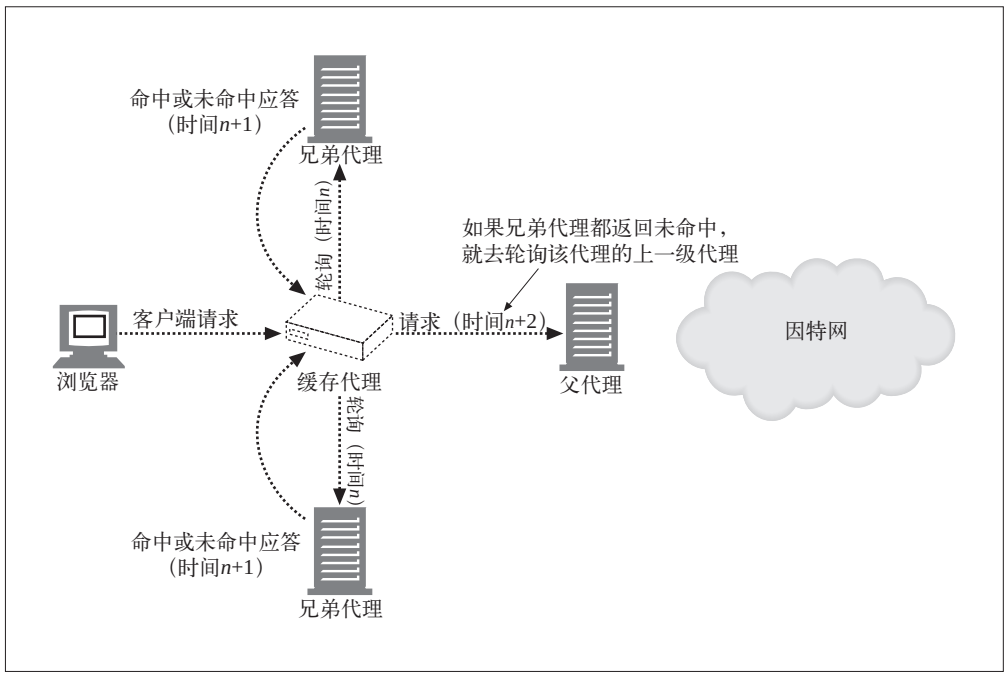


图 20-13　ICP 查询

475 注意，通过 ICP 协议连接起来的每个代理服务器都是将内容进行了冗余镜像的独立缓存服务器，这就说明在不同的代理服务器之间复制 Web 对象条目是可行的。相反，用 CARP 连接起来的一组服务器会被当作一个大型的服务器，其中每个组件服务器都只包含全部缓存文档中的一部分。通过对某个 Web 对象的 URL 应用散列函数，CARP 就可以将此对象映射到特定的代理服务器上去。每个 Web 对象都有一个唯一的家，所以我们可以通过单次查找确定对象的位置，而无须去查询集合中配置的每个代理服务器。图 20-14 总结了 CARP 重定向的方式。

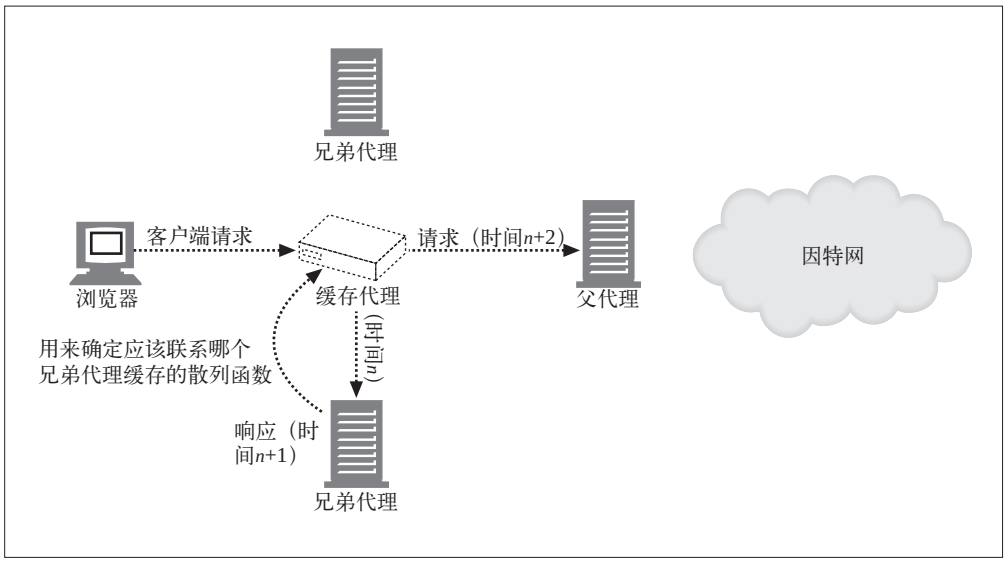


图 20-14　CARP 重定向

如图 20-14 所示，作为客户端和代理服务器中间人的缓存代理可以在各个代理服务器之间分配负载，但这项功能也可以由客户端自身提供。可以配置 Internet Explorer 和网景公司的 Navigator 这样的商用浏览器，以插件的形式计算散列函数，来确定应该把请求发送给哪个代理服务器。

CARP 对代理服务器做出的确定性解析说明它无须向所有邻居发送查询，这也就意味着这种方法所需发送的缓存间报文会比较少。随着越来越多的代理服务器添加到配置系统中来，缓存系统集群的规模会变得相当大。但 CARP 的一个缺点就是，如果某个代理服务器不可用了，就要重新修改散列表以反映这种变化，而且必须重新配置现存代理服务器上的内容。如果代理服务器经常崩溃的话，这么做的开销可能会很高。相反，ICP 代理服务器中存在的冗余内容就表示它不需要重新配置。另一个潜在的问题是，由于 CARP 是个新协议，CARP 集群中可能不会包含那些现存的、只运行 ICP 协议的代理服务器。

476

介绍了 CARP 和 ICP 间的区别之后，我们来详述一下 CARP。CARP 重定向方法要完成下列任务。

- 保存一个参与 CARP 的代理服务器列表。周期性地查询这些代理服务器，看看它们是否仍然活跃。
- 为每个参与的代理服务器计算一个散列函数。散列函数的返回值要考虑此代理所能处理的负载量。

- 定义一个独立的散列函数，这个函数会根据所请求 Web 对象的 URL 返回一个数字。
- 将 URL 散列函数的结果代入代理服务器的散列函数，得到一个数字阵列。这些数字中的最大值决定了要为这个 URL 使用的代理服务器。由于算出来的值是确定的，所以对同一个 Web 对象的后继请求会被转发给同一台代理服务器。

以上 4 项任务可以由浏览器、插件执行，也可以在一个中间服务器上计算。

为每个代理服务器集群创建一个表，表中列出了集群中的所有服务器。表中的每个条目都应该包含全局参数的相关的信息，比如，负载因子、生存时间（TTL）、倒计数值和应该以何频率查询成员之类的全局参数。负载因子说明机器可以处理多少负载，这取决于那台机器的 CPU 速度和硬盘容量。可以通过 RPC 接口对此表进行远程维护。只要表中的字段被 RPC 修改了，就可以使其对下游的客户端和代理可见，或将其发布给它们。这项发布工作是在 HTTP 中进行的，这样，所有的客户端或代理服务器就都可以在不引入另一种代理间协议的基础上消化表格信息了。客户端和代理服务器只用了一个知名 URL 来获取这张表。

所使用的散列函数必须能够确保 Web 对象在参与的代理服务器间是统计分布的。应该用代理服务器的负载因子来确定分配给那台代理的 Web 对象的统计概率。

总之，CARP 协议允许将一组代理服务器看成单个的集群缓存，而不是（像 ICP 中那样的）一组相互合作但又相互独立的缓存服务器。确定的请求解析路径会在一跳
477 内找到某个特定的 Web 对象的家。这样会降低 ICP 在一组代理服务器中查找 Web 对象时常会产生的代理间流量。CARP 还可以避免在不同的代理服务器上存储 Web 对象的多个副本的问题，这样做的优点是缓存系统集群的 Web 对象存储容量较大，缺点是任意一个代理的故障都要改写现存代理的部分缓存内容。

20.9 超文本缓存协议

之前我们讨论了 ICP，这个协议允许代理缓存向兄弟缓存查询文件是否存在。但设计 ICP 时考虑的是 HTTP/0.9 协议，因此，向兄弟缓存查询资源是否存在时，只允许缓存发送 URL。HTTP 版本 1.0 和 1.1 引入了很多新的请求首部，这些首部可以和 URL 一起用来确定文件是否匹配。因此，只在请求中发送 URL 可能无法得到精确的响应。

HTCP（超文本缓存协议）允许兄弟缓存之间通过 URL 和所有的请求及响应首部来相互查询文档是否存在，以降低错误命中的可能。而且 HTCP 允许兄弟缓存监视或请求在对方的缓存中添加或删除所选中的文档，并修改对方已缓存文档的缓存策略。

图 20-13 说明了一个 ICP 事务，此图也可以用来说明 HTCP 事务，后者是另一个对象发现协议。如果附近的缓存中有这个文档，发起请求的缓存可以打开一条到此缓存的 HTTP 连接，以获取那个文档的副本。ICP 和 HTCP 事务之间的区别体现在请求和响应细节上。

HTCP 报文的结构如图 20-15 所示。首部中包含了报文的长度和报文版本。数据部分开始是数据长度，包含了 opcode、响应代码、一些标记及 ID，最后是实际的数据。可选的认证部分跟在 Data 小节的后面。

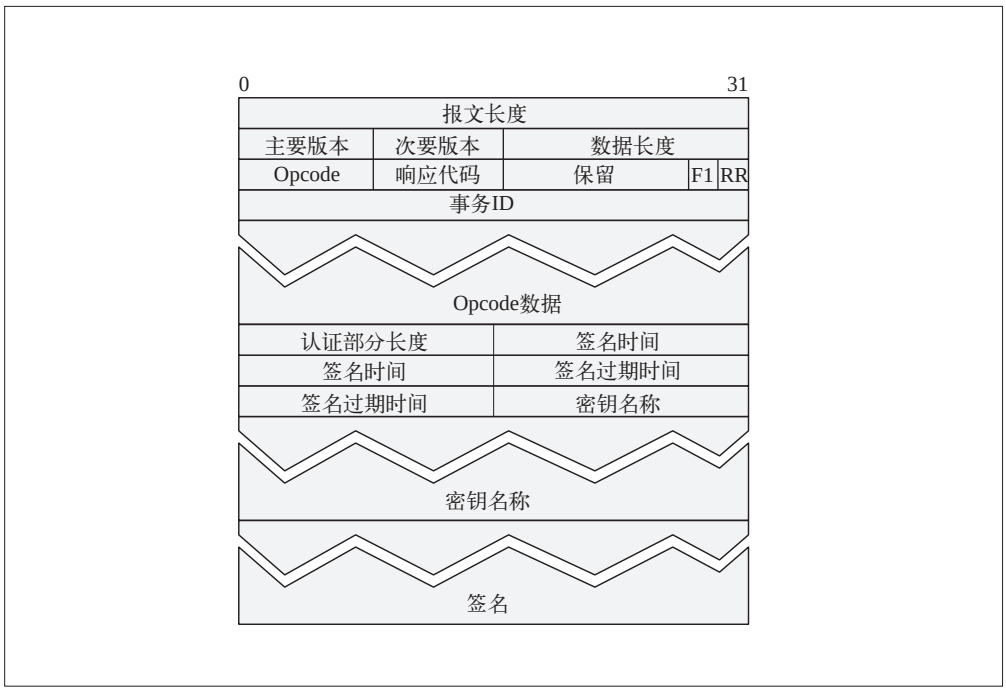


图 20-15　HTCP 报文格式

报文字段的详细内容如下所述。

- 首部

  Header 部分包含 32 位的报文长度，8 位的主要协议版本和 8 位的次要协议版本。报文长度包含所有首部、数据和认证部分的长度。

- 数据

  Data 部分包含了 HTCP 报文，结构如图 20-15 所示。数据组件如表 20-6 所示。 478

表20-6　HTCP数据组件

组　件	描　述
数据长度	16 位的 Data 部分字节数，包含 Length 字段自身的长度
Opcode	HTCP 事务的 4 位操作代码。表 20-7 列出了 Opcode 的完整内容
响应代码	说明事务成功或失败的 4 位键值。可能的值有： 0——没有进行认证，但需要进行认证； 1——需要进行认证，但没有得到满足； 2——未实现的 Opcode； 3——不支持主要版本； 4——不支持次要版本； 5——不合适、不允许或非预期的 Opcode。
F1	F1 是重载的——如果报文是一条请求，F1 就是请求端设置的 1 位标记，说明需要响应（F1=1）；如果报文是一条响应，F1 就是一个 1 位标记，用来说明应该将响应作为对整条报文的响应来解释（F1=1），还是将其作为对 Opcode 数据字段的响应来解释（F1=0）
RR	用来说明报文是请求（RR=0）还是响应（RR=1）的 1 位标记
事务 ID	32 位的值，与请求端的网络地址组合在一起可以唯一地标识 HTCP 事务
Opcode 数据	Opcode 数据与 Opcode 有关。参见表 20-7

表 20-7 列出了 HTCP Opcode 代码及其相应的数据类型。

表20-7　HTCP Opcode

Opcode	值	描　述	响应代码	Opcode数据
NOP	0	本质上是一个 ping 操作	总是 0	无
TST	1		如果有实体，就为 0，如果没有提供实体，就为 1	在请求中包含 URL 和请求首部，在响应中只包含响应首部
MON	2		接受就为 0，拒绝就为 1	
SET	3	SET 报文允许缓存请求修改缓存策略。可以用于 SET 报文的首部参见表 20-9	接受就为 0，忽略就为 1	
CLR	4		如果曾经有过，但现在没有了，就为 0；如果曾经有过，而且现在还有，就为 1；如果从未有过，就为 2	

20.9.1　HTCP认证

HTCP 报文的认证部分是可选的。其结构如图 20-15 所示，表 20-8 列出了它的认证组件。

表20-8　HTCP认证组件

组　件	描　述
认证部分长度	16 位的报文认证部分字节数，包含了长度字段自身的长度
签名时间	32 位数，表示从格林尼治标准时间 1970 年 1 月 1 日 00:00:00 开始，到产生签名的时间之间的秒数
签名过期时间	32 位数，表示从格林尼治标准时间 1970 年 1 月 1 日 00:00:00 开始，到签名过期时所经历的秒数
密钥名称	用来表示共享密钥名称的字符串。密钥字段有两个部分：用来说明后面那个字符串长度的 16 位的字节数，后面跟着的字符串是未经解释的字节流
签名	HMAC-MD5 摘要，它是 B 值为 64（表示源 IP 地址和目的 IP 地址及端口）、报文的主要及次要 HTCP 版本、签名时间和签名过期值，完整的 HTCP 数据以及密钥的摘要。签名也包含两个部分：16 位长的字符串字节数，后面跟着这个字符串

20.9.2　设置缓存策略

SET 报文允许缓存请求对已缓存文档的缓存策略进行修改。表 20-9 中给出了可以在 SET 报文中使用的首部。

480

表20-9　修改缓存策略的缓存首部列表

首　部	描　述
`Cache-Vary`	请求端已经知道内容会随一组首部的变化而变化，这组首部与响应 `Vary` 首部中的那一组不同。这个首部会覆盖响应的 `Vary` 首部
`Cache-Location`	可能有此对象副本的代理缓存的列表
`Cache-Policy`	关于此对象的缓存策略，请求端已经了解到的比响应首部中指定的更详细。可能的值包括：`no-cache`，说明响应是不可缓存的，但可以在多个同时发起请求的请求端之间共享；`no-share`，说明对象是不可共享的；`no-cache-cookie`，说明内容可能会随 cookie 而发生变化，不推荐缓存
`Cache-Flags`	请求端修改了对象的缓存策略，可能要对它进行特别的处理，不一定要根据其实际的策略进行处理
`Cache-Expiry`	发送端了解到的文档实际过期时间
`Cache-MD5`	请求端计算出来的对象的 MD5 校验和，可能与 `Content-MD5` 首部的值有所不同，也可能在对象没有 `Content-MD5` 首部的情况下提供
`Cache-to-Origin`	请求端测量的到原始服务器的往返时间。此首部值的格式为 *< 原始服务器名称或 IP 地址 >< 以秒为单位的平均往返时间 >< 采样数 >< 请求端和原始服务器之间的路由器跳数 >*

HTCP 允许通过查询报文将请求和响应首部发送给兄弟缓存，这样可以降低缓存查询中的错误命中率。通过进一步允许在兄弟缓存间交换策略信息，HTCP 还可以提高兄弟缓存之间的合作能力。

20.10 更多信息

更多信息可参考下列文献。

- *DNS and Bind*[3]（《DNS 与 BIND》）
  Cricket Liu、Paul Albitz 和 Mike Loukides 著，O'Reilly & Associates 公司出版。
- http://www.wrec.org/Drafts/draft-cooper-Webi-wpad-00.txt
  “Web Proxy Auto-Discovery Protocol”（“Web 代理自动发现协议”）。
- http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html
  “Navigator Proxy Auto-Config File Format”（“Navigator 代理自动配置文件格式”）。
- http://www.ietf.org/rfc/rfc2186.txt
  IETF RFC 2186，“Intercache Communication Protocol (ICP) Version2”（“缓存间的通信协议，版本 2”），D. Wessels 和 K. Claffy 编写。
- http://icp.ircache.net/carp.txt
  “Cache Array Routing Protocol v1.0”（“缓存阵列路由协议 v1.0”）。
- http://www.ietf.org/rfc/rfc2756.txt
  IETF RFC 2756，“Hyper Text Caching Protocol(HTCP/0.0)”（“超文本缓存协议（HTCP/0.0）”），P. Vixie 和 D. Wessels 编写。
- http://www.ietf.org/internet-drafts/draft-wilson-wrec-wccp-v2-00.txt
  “Web Cache Communication Protocol v2.0”（“Web 缓存通信协议 v2.0”），M. Cieslak、D. Forster、G. Tiwana 和 R. Wilson 编写。
- http://www.ietf.org/rfc/rfc2131.txt?number=2131
  “Dynamic Host Configuration Protocol”（“动态主机配置协议”）。
- http://www.ietf.org/rfc/rfc2132.txt?number=2132
  “DHCP Options and BOOTP Vendor Extensions”（“DHCP 选项与 BOOTP 厂商扩展”）。
- http://www.ietf.org/rfc/rfc2608.txt?number=2608
  “Service Location Protocol,Version2”（“服务定位协议，版本 2”）。
- http://www.ietf.org/rfc/rfc2219.txt?number=2219
  “Use of DNS Aliases for Network Services”（“为网络服务使用 DNS 别名”）。

481

482

注 3：本书中译本已由中国电力出版社出版。（编者注）

第21章

日志记录与使用情况跟踪

几乎所有的服务器和代理都会记录下它们所处理的 HTTP 事务摘要。这么做出于一系列的原因：跟踪使用情况、安全性、计费、错误检测，等等。本章简要介绍了日志记录，研究了通常会记录 HTTP 事务哪些方面的信息以及一些常见日志格式中所包含的内容。

21.1 记录内容

大多数情况下，日志的记录出于两种原因：查找服务器或代理中存在的问题（比如，哪些请求失败了），或者是生成 Web 站点访问方式的统计信息。统计数据对市场营销、计费和容量规划（比如，决定是否需要增加服务器或带宽）都非常有用。

可以把一个 HTTP 事务中所有的首部都记录下来，但对每天要处理数百万个事务的服务器和代理来说，这些数据的体积超大，很快就会失控。不应该记录实际上你并不感兴趣，甚至从来都不会去看一眼的数据。

通常，只记录事务的基本信息就行了。通常会记录下来的几个字段示例为：

- HTTP 方法；
- 客户端和服务器的 HTTP 版本；
- 所请求资源的 URL；
- 响应的 HTTP 状态码；
- 请求和响应报文的尺寸（包含所有的实体主体部分）；
- 事务开始时的时间戳；
- `Referer` 首部和 `User-Agent` 首部的值。

483

HTTP 方法和 URL 说明了请求试图做些什么——比如，GET 某个资源或 POST 某个定单。可以用 URL 来记录 Web 站点上页面的受欢迎程度。

版本字符串给出了与客户端和服务器有关的一些提示，在客户端和服务器之间出现一些比较奇怪或非预期的交互动作时，它会非常有用。比如，如果请求的失败率高于预期，那版本信息指向的可能是一个无法与服务器进行交互的新版浏览器。

HTTP 状态码说明了请求的执行状况：是否成功执行，认证请求是否失败，资源是否找到等（HTTP 状态码列表参见 3.2.2 节）。

请求 / 响应的大小和时间戳主要用于记账；就是记录流入、流出或流经应用程序的字节有多少。还可用时间戳将观察到的问题与当时发起的一些请求关联起来。

21.2 日志格式

目前已有一些日志格式的标准了。本节我们会讨论一些最常见的日志格式。大部分商用和开源的 HTTP 应用程序都支持以一种或多种常用格式进行日志记录。很多这样的应用程序都支持管理者配置日志格式，创建自定义的格式。

应用程序支持管理者使用这些更标准的格式的主要好处之一就在于，可以充分利用那些已构建好的工具处理这些日志，并产生基本的统计信息。有很多开源包和商用包都可用来压缩日志，以进行汇报。使用标准格式，应用程序及其管理员就都可以利用这些包了。

21.2.1 常见日志格式

现在，最常见的日志格式之一就是*常用日志格式*。这种日志格式最初由 NCSA 定义，很多服务器在默认情况下都会使用这种日志格式。可以将大部分商用及开源服务器配置为使用这种格式，有很多商用及免费工具都可辅助解析常用日志格式的文件。表 21-1 按序列出了常用日志格式中的字段。

表21-1 常用日志格式字段

字 段	描 述
remotehost	请求端机器的主机名或 IP 地址（如果没有配置服务器去执行反向 DNS 或无法查找请求端的主机名，就使用 IP 地址）
username	如果执行了 ident 查找，就是请求端已认证的用户名[a]
auth-username	如果进行了认证，就是请求端已认证的用户名
timestamp	请求的日期和时间
request-line	精确的 HTTP 请求行文本，GET /index.html HTTP/1.1
response-code	响应中返回的 HTTP 状态码
response-size	响应主体中的 `Content-Length`，如果响应中没有返回主体，就记录 0

a：RFC 931 描述了在此认证中使用的 ident 查找。ident 协议是在第 5 章介绍的。

484

例 21-1 列出了几个常见日志格式条目。

例 21-1 常见日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00 -0400] "GET / HTTP/1.0" 200 1024
http-guide.com - dg [03/Oct/1999:14:16:32 -0400] "GET / HTTP/1.0" 200 477
http-guide.com - dg [03/Oct/1999:14:16:32 -0400] "GET /foo HTTP/1.0" 404 0
```

在这些例子中，字段的分配如下所示。

字段1	条目1	条目2	条目2
remotehost	209.1.32.44	http-guide.com	http-guide.com
username	< 空 >	< 空 >	< 空 >
auth-username	< 空 >	dg	dg
timestamp	03/Oct/1999:14:16:00 -0400	03/Oct/1999:14:16:32 -0400	03/Oct/1999:14:16:32 -0400
request-line	GET / HTTP/1.0	GET / HTTP/1.0	GET /foo HTTP/1.0
response-code	200	200	404
response-size	1024	477	0

注意，remotehost 字段可以是 http-guide.com 那样的主机名，也可以是 209.1.32.44 这样的 IP 地址。

第二个（username）和第三个（auth-username）字段之间的破折号说明字段为空。这说明要么是没有进行 ident 查找（第二个字段为空），要么是没有进行认证（第三个字段为空）。

21.2.2 组合日志格式

另一种常用日志格式为组合日志格式（Combined Log Format），例如 Apache 服务器就支持这种格式。组合日志格式与常用日志格式很类似。实际上，它就是常用日志格式的精确镜像，只是添加了（表 21-2 中列出的）两个字段。User-Agent 字段用于说明是哪个 HTTP 客户端应用程序在发起已被记录的请求，而 Referer 字段则提供了更多与请求端在何处找到这个 URL 的有关信息。

485

表21-2 新加的组合日志格式字段

字 段	描 述
Referer	Referer 首部的内容
User-Agent	User-Agent 首部的内容

例 21-2 给出了一个组合日志格式的条目。

例 21-2 组合日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00 -0400] "GET / HTTP/1.0" 200 1024
"http://www.joes-hardware.com/"  "5.0: Mozilla/4.0 (compatible; MSIE
5.0; Windows 98)"
```

在例 21-2 中，Referer 字段和 User-Agent 字段的值如下所示。

字 段	值
Referer	http://www.joes-hardware.com/
User-Agent	5.0: Mozilla/4.0（兼容的，MSIE 5.0，Windows 98）

例 21-2 的组合日志格式条目示例中的前七个字段和常用日志格式中的完全一样（参见例 21-1 中的第一个条目）。两个新字段 Referer 和 User-Agent 附加在日志条目的末尾。

21.2.3 网景扩展日志格式

网景进入商用 HTTP 应用程序领域时，为其服务器定义了很多其他 HTTP 应用程序开发者已接纳的日志格式。网景的格式是基于 NCSA 的常用日志格式的，但它们扩展了该格式，以支持与代理和 Web 缓存这样的 HTTP 应用程序相关的字段。

网景扩展日志格式的前 7 个字段与常用日志格式中的那些字段完全相同（参见表 21-1）。表 21-3 按序列出了网景扩展日志格式引入的新字段。

表21-3 网景扩展日志格式新加的字段

字 段	描 述
proxy-response-code	如果事务处理经过了某个代理，就是从服务器传往代理的 HTTP 响应码
proxy-response-size	如果事务处理经过了某个代理，就是发送给代理的服务器响应实体的 `Content-Length`
client-request-size	发给代理的客户端请求的所有主体或实体的 `Content-Length`
proxy-request-size	如果事务处理经过了某个代理，就是代理发往服务器的请求的所有主体或实体的 `Content-Length`
client-request-hdr-size	以字节为单位的客户端请求首部的长度
proxy-response-hdr-size	如果事务处理经过了某个代理，就是以字节为单位的、发送给请求端的代理响应首部的长度
proxy-request-hdr-size	如果事务处理经过了某个代理，就是以字节为单位的、发送给服务器的代理请求首部的长度
server-response-hdr-size	以字节为单位的，服务器响应首部的长度
proxy-timestamp	如果事务处理经过了某个代理，就是请求和响应经过代理传输所经过的时间（单位为秒）

486

例 21-3 给出了一个网景扩展日志格式的条目。

例 21-3 网景扩展日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00-0400] "GET / HTTP/1.0" 200 1024
200 1024 0 0 215 260
279 254 3
```

在这个例子中，扩展字段的值如下所示。

字　段	值
proxy-response-code	200
proxy-response-size	1024
client-request-size	0
proxy-request-size	0
client-request-hdr-size	215
proxy-response-hdr-size	260
proxy-request-hdr-size	279
server-response-hdr-size	254
proxy-timestamp	3

例 21-3 中网景扩展日志格式条目示例中的前 7 个字段是常用日志格式条目示例的镜像（参见例 21-1 中的第一个条目）。

21.2.4　网景扩展2日志格式

另一种网景日志格式，网景扩展 2 日志格式采用了扩展日志格式，并添加了一些与 HTTP 代理和 Web 缓存应用程序有关的附加信息。这些附加字段有助于更好地描绘 HTTP 客户端和 HTTP 代理应用程序间的交互图景。

网景扩展 2 日志格式是基于网景扩展日志格式的，初始字段与表 21-3 中列出的字段
487 完全相同（它也是表 21-1 中常用日志格式字段的扩展）。

表 21-4 按序列出了网景扩展 2 日志格式新加的字段。

表21-4　附加的网景扩展2日志格式字段

字　段	描　述
route	代理用来向客户端发送请求的路径（参见表 21-5）
client-finish-status-code	客户端完成状态码。说明了发送给代理的客户端请求是成功完成（FIN）了，还是被打断了（INTR）
proxy-finish-status-code	代理完成状态码。说明代理发送给服务器的请求是成功完成（FIN）了，还是被打断了（INTR）
cache-result-code	缓存结果代码；说明缓存是如何响应请求的[a]

a：表 21-7 列出了网景的缓存结果代码。

例 21-4 给出了一个网景扩展 2 日志格式的条目。

例 21-4　网景扩展 2 日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00-0400] "GET / HTTP/1.0" 200 1024
200 1024 0 0 215 260
```

```
279 254 3 DIRECT FIN FIN WRITTEN
```

这个例子中扩展字段的值如下所示。

字　段	值
route	DIRECT
client-finish-status-code	FIN
proxy-finish-status-code	FIN
cache-result-code	WRITTEN

例 21-4 的网景扩展 2 日志格式条目中的前 16 个字段就是网景扩展日志格式示例条目的镜像（参见例 21-3）。

表 21-5 列出了有效的网景路由代码。

表21-5　网景路由代码

值	描　述
DIRECT	资源是直接从服务器上获取的
PROXY(host:port)	资源是通过代理“host:port”获取的
SOCKS(socks:port)	资源是通过 SOCKS 服务器“host:port”获取的

表 21-6 列出了有效的网景完成代码。

表21-6　网景完成状态码

值	描　述
-	请求未开始
FIN	请求成功完成
INTR	请求被客户端中断或被代理 / 服务器终止
TIMEOUT	请求被代理 / 服务器超时

表 21-7 列出了有效的网景缓存代码。[1]

表21-7　网景缓存代码

代　码	描　述
-	资源不可缓存
WRITTEN	资源被写入了缓存
REFRESHED	资源被缓存，且被刷新了
NO-CHECK	返回已缓存的资源，未进行新鲜性检查

注 1：表 21-7 列出了网景缓存结果代码。

（续）

代　码	描　述
UP-TO-DATE	返回已缓存的资源，进行了新鲜性检查
HOST-NOT-AVAILABLE	返回已缓存的资源。由于远程服务器不可用，所以未进行新鲜性检查
CL-MISMATCH	未将资源写入缓存。由于 Content-Length 与资源尺寸不匹配，放弃了写操作
ERROR	因为出错，资源未被写入缓存。比如，出现了超时，或客户端放弃了此事务

与很多其他 HTTP 应用程序一样，网景应用程序也有其他的日志格式，包括一种灵活日志格式和一种管理者输出自定义日志字段的方式。这些格式给予管理者更大的控制权，并可以选择在日志中报告 HTTP 事务处理的哪些部分（首部、状态、尺寸等），以自定义其日志。

由于很难预测管理者希望从其日志中获取哪些信息，才添加了管理者配置自定义格式的能力。很多其他的代理和服务器都有发布自定义日志的能力。

21.2.5　Squid代理日志格式

Squid 代理缓存（http://www.squid-cache.org）是 Web 上一个很古老的部分。其起源可以回溯到一个早期的 Web 代理缓存项目（ftp://ftp.cs.colorado.edu/pub/techreports/schwartz/Harvest.Conf.ps.Z）。Squid 是开源社团多年来扩展增强的一个开源项目。有很多工具可以用来辅助管理 Squid 应用程序，包括一些有助于处理、审核及开发其日志的工具。很多后继代理缓存都为自己的日志使用了 Squid 格式，这样才能更好地利用这些工具。

489

Squid 日志条目的格式相当简单。表 21-8 总结了该日志格式的字段。

表21-8　Squid日志格式字段

字　段	描　述
timestamp	请求到达时的时间戳，是从格林尼治标准时间 1970 年 1 月 1 日开始的秒数
time-elapsed	请求和响应通过代理传输所经历的时间（以毫秒为单位）
host-ip	客户端（请求端）主机的 IP 地址
result-code/status	result 字段是 Squid 类型的，用来说明在此请求过程中代理采取了什么动作；[a] code 字段是代理发送给客户端的 HTTP 响应代码
size	代理响应客户端的字节长度，包括 HTTP 响应首部和主体
method	客户端请求的 HTTP 方法

a：表 21-9 列出了各种结果代码及其含义。

（续）

字　段	描　述
url	客户端请求中的 URL[b]
rfc931-ident[c]	客户端经过认证的用户名 [d]
hierarchy/from	与网景格式中的 route 字段一样，hierarchy 字段说明了代理向客户端发送请求时经由的路径。[e]from 字段说明了代理用来发起请求的服务器的名称
content-type	代理响应实体的 `Content-Type`

b：回顾第 2 章，代理通常会记录下整条请求 URL，这样如果 URL 中有用户和密码组件，代理就会在不经意间记录下此信息。

c：rfc931-ident、hierarchy/from 和 content-type 字段是在 Squid 1.1 中添加的。早期版本中都没有这些字段。

d：RFC 931 描述了这种认证方式中使用的 ident 查找。

e：http://squid.nlanr.net/Doc/FAQ/FAQ-6.html#ss6.6 列出了所有有效的 Squid hierarchy 代码。

例 21-5 给出了一个 Squid 日志格式条目的例子。

例 21-5　Squid 日志格式

```
99823414 3001 209.1.32.44 TCP_MISS/200 4087 GET http://www.joes-
hardware.com - DIRECT/
proxy.com text/html
```

这些字段的值如下所示。

字　段	值
timestamp	99823414
time-elapsed	3001
host-ip	209.1.32.44
action-code	TCP_MISS
status	200
size	4087
method	GET
URL	http://www.joes-hardware.com
RFC 931 ident	-
hierarchy	DIRECT[a]
from	proxy.com
content-type	text/html

a：DIRECT Squid hierarchy 值与网景日志格式中的 DIRECT route 值一样。

表 21-9 列出了各种 Squid 结果代码。[2]

注 2：这些行为代码中有几个通常是处理 Squid 代理缓存的内部行为，因此其他使用了 Squid 日志格式的代理并没有使用全部的代码。

表21-9 Squid结果代码

行 为	描 述
TCP_HIT	资源的有效副本是由缓存提供的
TCP_MISS	资源不在缓存中
TCP_REFRESH_HIT	资源在缓存中，但需要进行新鲜性检查。代理与服务器再次验证了资源，发现缓存中的副本确实还是新鲜的
TCP_REF_FAIL_HIT	资源在缓存中，但需要进行新鲜性检查。但再验证失败了（可能是代理无法连接到服务器），因此返回的是“过期”的资源
TCP_REFRESH_MISS	资源在缓存中，但需要进行新鲜性检查。在与服务器进行验证的时候，代理得知缓存中的资源过期了，并收到一个新的副本
TCP_CLIENT_REFRESH_MISS	请求端发送了一个 `Pragma: no-cache`，或类似的 `Cache-Control` 指令，命令代理必须去获取资源
TCP_IMS_HIT	请求端发布了一个条件请求，对资源的已缓存副本进行验证
TCP_SWAPFAIL_MISS	代理认为资源位于缓存中，但出于某些原因无法访问该资源
TCP_NEGATIVE_HIT	返回已缓存的响应，但响应是否定的已缓存响应。Squid 支持对资源错误信息的缓存（比如，缓存 404 Not Found 响应）。这样，如果有多条对某无效资源的请求都经过这个代理缓存，就可以由这个代理缓存提供错误信息
TCP_MEM_HIT	资源的有效副本是由缓存提供的，资源位于代理缓存的内存中（与必须访问磁盘才能获取已缓存资源的方式相反）
TCP_DENIED	对此资源的请求被否决了，可能是请求端没有请求此资源的权利
TCP_OFFLINE_HIT	所请求的资源是在离线状态下从缓存中解析出来的。Squid（或另一个使用此格式的代理）处于离线模式时，资源是未经验证的
UDP_*	UDP_* 代码说明请求是通过到代理的 UDP 接口收到的。HTTP 通常会使用 TCP 传输协议，因此这些请求使用的都不是 HTTP 协议 [a]
UDP_HIT	资源的有效副本是由缓存提供的
UDP_MISS	资源不在缓存中
UDP_DENIED	对此资源的请求被否决了，可能是由于请求端没有请求此资源的权限
UDP_INVALID	代理收到的请求是无效的
UDP_MISS_NOFETCH	Squid 在特定的操作模式下，或在缓存常见错误的缓存中使用。会返回缓存未命中，而且也没有获得资源
NONE	有时与错误信息一起记录
TCP_CLIENT_REFRESH	参见 TCP_CLIENT_REFRESH_MISS
TCP_SWAPFAIL	参见 TCP_SWAPFAIL_MISS
UDP_RELOADING	参见 UDP_MISS_NOFETCH

a：Squid 有自己的用于发起这些请求的协议：ICP。这是缓存到缓存的请求所使用的协议。更多信息请参见 http://www.squid-cache.org。

21.3 命中率测量

原始服务器通常会出于计费的目的保留详细的日志记录。内容提供者需要知道URL的受访频率，广告商需要知道广告的出现频率，网站作者需要知道所编写的内容的受欢迎程度。客户端直接访问Web服务器时，日志记录可以很好地跟踪这些信息。

但是，缓存服务器位于客户端和服务器之间，用于防止服务器同时处理大量访问请求[3]（这正是缓存的目的）。缓存要处理很多HTTP请求，并在不访问原始服务器的情况下满足它们的请求，服务器中没有客户端访问其内容的记录，导致日志文件中出现遗漏。

由于日志数据会遗失，所以，内容提供者会对其最重要的页面进行*缓存清除*（cache bust）。缓存清除是指内容提供者有意将某些内容设置为无法缓存，这样，所有对此内容的请求都会被导向原始服务器。[4]于是，原始服务器就可以记录下访问情况了。不使用缓存可能会生成更好的日志，但会减缓原始服务器和网络的请求速度，并增加其负荷。

由于代理缓存（及一些客户端）都会保留自己的日志，所以如果服务器能够访问这些日志（或者至少有一种粗略的方式可以判断代理缓存会以怎样的频率提供其内容），就可以避免使用缓存清除。命中率测量协议是对HTTP的一种扩展，它为这个问题提供了一种解决方案。命中率测量协议要求缓存周期性地向原始服务器汇报缓存访问的统计数据。

492

RFC 2227详细定义了命中率测量协议。本节将详细介绍此协议。

21.3.1 概述

命中率测量协议定义了一种HTTP扩展，它提供了一些基本的功能，缓存和服务器可以实现这些功能来共享访问信息，规范已缓存资源的可使用次数。

缓存给日志访问带来了问题，命中率测量并不是这个问题的完整解决方案，但它确实提供了一种基本方式，以获取服务器希望跟踪的度量值。命中率测量协议并没有（而且可能永远都不会）得到广泛的实现或应用。也就是说，在维护缓存性能增益的同时，像命中率测量这样的合作方案会给出一些提供精确访问统计信息的承诺。希望这会推动命中率测量协议的实现，而不是把内容标记为不可缓存的。

注3：回想一下，几乎每个浏览器都会有一个缓存。

注4：第7章说明了怎样将HTTP响应标记为不可缓存的。

21.3.2 Meter首部

命中率测量扩展建议使用新增加的首部 Meter，缓存和服务器可以通过它在相互间传输与用法和报告有关的指令，这与用来进行缓存指令交换的 Cache-Control 首部很类似。

表 21-10 列出了定义的各种指令和谁可以在 Meter 首部传输这些指令。

表21-10 命中率测量指令

指令	缩写	执行者	描述
will-report-and-limit	w	缓存	缓存可以报告使用情况并遵循服务器指定的所有使用限制
wont-report	x	缓存	缓存可以遵循使用限制，但不报告使用情况
wont-limit	y	缓存	缓存可以报告使用情况但不会限制使用
count	c	缓存	报告指令，以 uses/reuses 整数的形式说明。比如：count=2/4 [a]
max-uses	u	服务器	允许服务器指定某响应可被缓存使用的最大次数。比如：max-uses=100
max-reuses	r	服务器	允许服务器指定某响应可被缓存重用的最大次数。比如：max-reuses=100
do-report	d	服务器	服务器要求代理发送使用报告
dont-report	e	服务器	服务器不要求使用汇报
timeout	t	服务器	允许服务器指定对某资源进行计量的超时时间。缓存应该在指定的超时时间，或在此时间之前发送报告，允许有 1 分钟的误差。超时是以分钟为单位的。比如：timeout=60
wont-ask	n	服务器	服务器不需要任何计量信息

a：命中率测量定义了一个 use，用一个响应来满足请求，还定义了一个 reuse，对客户端请求进行再验证。

493

图 21-1 给出了一个执行中的命中率测量实例。事务的第一部分就是客户端和代理缓存之间一个普通的 HTTP 事务，但在代理请求中，要注意有插入的 Meter 首部和来自服务器的响应。这里，代理正在通知服务器它可以进行命中率测量，作为回应，服务器则请求代理报告它的命中次数。

从客户端的角度来看，请求正常结束了，代理开始代表服务器跟踪该请求资源的命中次数。稍后，代理会尝试与服务器再次验证资源。代理会在发送给服务器的条件请求中嵌入它跟踪记录的计量信息。

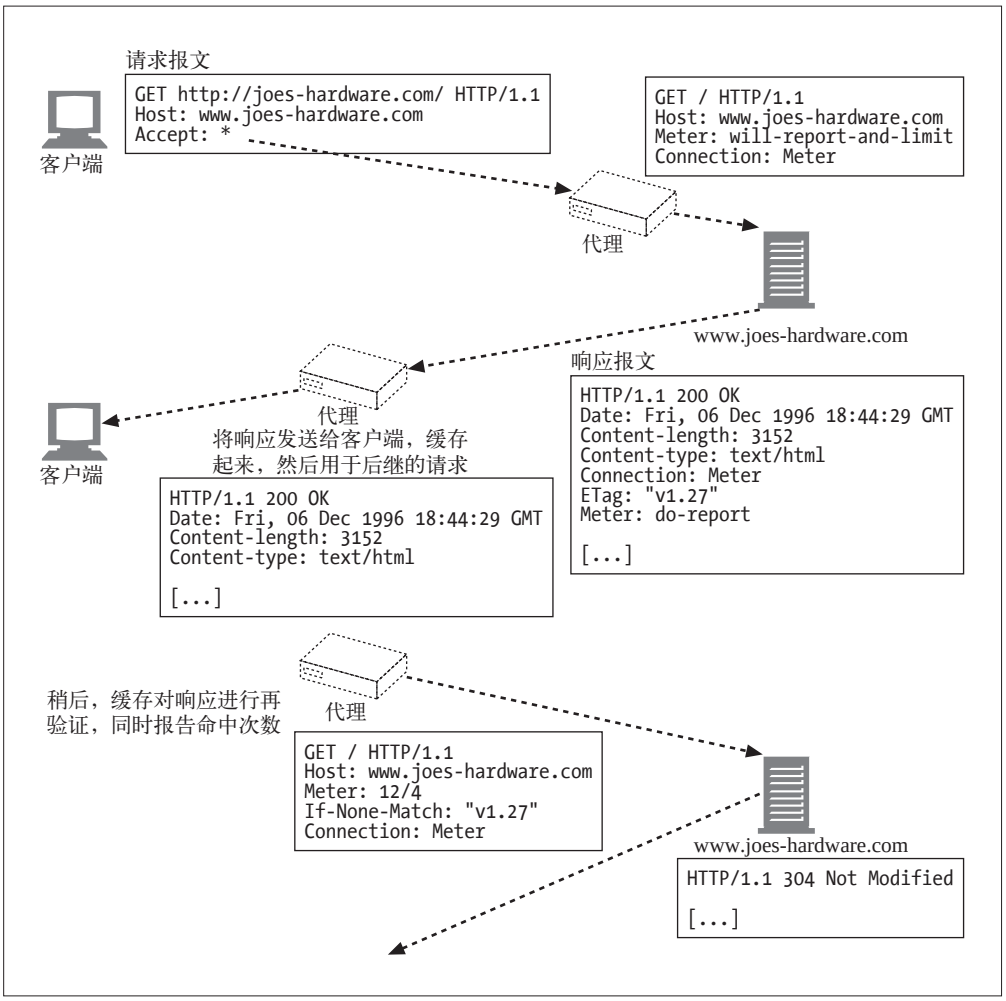


图 21-1　命中率测量示例

494

21.4　关于隐私的考虑

日志记录实际上就是服务器和代理执行的一项管理功能，所以整个操作对用户来说都是透明的。通常，用户甚至都不清楚他们的 HTTP 事务已被记录——实际上，很多用户可能甚至都不知道他们访问 Web 上的内容时是在使用 HTTP 协议。

Web 应用程序的开发者和管理者要清楚跟踪用户的 HTTP 事务可能带来的影响。他可以根据获取的信息收集很多有关用户的情况。很显然，这些信息可以用于不良目的——歧视、骚扰、勒索等。进行日志记录的 Web 服务器和代理一定要注意保护其终端用户的隐私。

有些情况下，比如在工作环境中，跟踪某用户的使用情况以确保他没有偷懒是可行的，但管理员也应该将监视大家事务处理的事情公之于众。

简单来说，日志记录对管理者和开发者来说都是很有用的工具。只是要清楚在没有获得用户许可，或在其不知情的情况下，使用记录其行为的日志可能会存在侵犯隐私的问题。

21.5 更多信息

更多有关日志记录的信息，请参见以下资源。

- http://httpd.apache.org/docs/logs.html
  “Apache HTTP Server: Log Files”（“Apache HTTP 服务器：日志文件”）。Apache HTTP 服务器项目网站。
- http://www.squid-cache.org/Doc/FAQ/FAQ-6.html
  “Squid Log Files”（“Squid 日志文件”）。Squid 代理缓存网站。
- http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format
  “Logging Control in W3C httpd”（“W3C httpd 中的日志记录控制”）。
- http://www.w3.org/TR/WD-logfile.html
  “Extended Log File Format”（“扩展日志文件格式”）。
- http://www.ietf.org/rfc/rfc2227.txt
  RFC 2227，J. Mogul 和 P. Leach 编写的“Simple Hit-Metering and Usage-Limiting for HTTP”（“简单的 HTTP 命中率测量和使用限制”）。

495

第六部分

附　录

本书附录集中包含了一些有用的参考表格、背景信息以及关于 HTTP 结构和实现各种主题的指南。

- 附录 A　URI 方案
- 附录 B　HTTP 状态码
- 附录 C　HTTP 首部参考
- 附录 D　MIME 类型
- 附录 E　Base-64 编码
- 附录 F　摘要认证
- 附录 G　语言标记
- 附录 H　MIME 字符集注册表

496
~
498

附录A

URI方案

已定义的 URI 方案有很多，但常用的并不多。一般来说，有相关的 RFC 对其解释说明的 URI 方案更常用一些，但确实也有少数由主导软件公司（特别是 Netscape 和 Microsoft）开发，但未被正式发布的方案得到了广泛应用。

W3C 维护了一个 URI 方案列表，可以通过以下地址访问：

http://www.w3.org/Addressing/schemes.html

IANA 也维护了一个 URL 方案的列表，网址是：

http://www.iana.org/assignments/uri-schemes

表 A-1 介绍了部分已经提出的和正在使用的方案。注意，表中大约有 90 个方案，其中很多都没有得到广泛应用，而且有些已被废弃。

表A-1　在W3C注册的URI方案

方　案	描　述	RFC
about	研究浏览器各方面特性的 Netscape 方案。比如说，使用 about 自身的效果就跟选择 Navigator 的 Help 菜单中的 About Communicator 一样，about:cache 显示的就是磁盘缓存的统计数据，about:plugins 显示的是与已配置的插件有关的信息。其他浏览器，比如微软的 Internet Explorer，也使用了这个方案	
acap	应用程序配置访问协议	2244
afp	用于使用 AFP（Apple Filing Protocol，苹果文件协议）提供的文件共享服务，是作为已过期的 IETF draft-ietf-svrloc-afp-service-01.txt 的一部分定义的	
afs	保留，以备 Andrew 文件系统将来使用	
callto	初始化微软的 NetMeeting 会议的会话，比如：callto:ws3.joes-hardware.com/joe@joes-hardware.com	
chttp	Real 网络公司定义的 CHTTP 缓存协议。RealPlayer 没有缓存 HTTP 传输的所有条目。作为一种替代方式，可以在文件的 URL 中使用 chttp:// 代替 http:// 来说明要缓存的文件。RealPlayer 在 SMIL 文件中读到 CHTTP URL 时，首先会查看磁盘的缓存中是否有该文件。如果没有此文件，就通过 HTTP 请求此文件，并将其存储在自己的缓存中	
cid	在电子邮件中通过 [MIME] 传送 Web 页面及其相关图片时，需要一个 URL 方案允许 HTML 引用报文中所含图片或其他数据 Content-ID URL，即 cid，就用于这个目的	2392 2111
clsid	允许引用微软的 OLE/COM 类。用于向 Web 页面中插入活动对象	

（续）

方　案	描　述	RFC
data	允许将一些小的常量数据条目作为“即时”数据包含在内。这个 URL 会将 text/plain 字符串 A brief note 编码为：data:A%20brief%20note	2397
date	支持日期的方案建议，比如 date:1999-03-04T20:42:08	
dav	为了确保基于此规范的互操作的正确性，IANA 必须保留以 DAV: 和 opaquelocktoken: 开头的 URI 名字空间，供这个规范和它的修订版本以及相关的 WebDAV 规范使用	2518
dns	供 REBOL 软件使用。 参见 http://www.rebol.com/users/valurl.html	
eid	外部 ID 方案提供了一种机制，本地应用程序可以通过这种机制引用通过其他非 URL 方案获取的数据。这个方案试图提供一种通用的转义机制，以便那些无法提出自己方案的专业应用程序访问信息。这个 URI 的使用方式存在争议。参见 http://www.ics.uci.edu/pub/ietf/uri/draft-finseth-url-00.txt	
fax	方案 fax 描述了一条连接，此连接连至可处理电传的终端（传真机）	2806
file	在特定主机上标识出可访问的文件。其中可以包含主机名，但这个方案的特殊性在于它没有为此类文件指定因特网协议或访问方式；这样，它在主机间网络协议中的效用就会受到限制	1738
finger	finger URL 的格式如下：*finger://host*[:*port*][/<*request*>]。 <request> 必须与 RFC 1288 的请求格式一致。参见 http://www.ics.uci.edu/pub/ietf/uri/draft-ietf-uri-url-finger-03.txt	
freenet	获取中迅互联分布式信息系统中信息所用的 URI。参见 http://freenet.sourceforge.net	
ftp	文件传输协议方案	1738
gopher	古老的 gopher 协议	1738
gsm-sms	用于 GSM 移动电话短信业务的 URI。	
h323，h324	多媒体会议的 URI 方案。 参见 http://www.ics.uci.edu/pub/ietf/uri/draft-cordell-sg16-conv-url-00.txt	
hdl	Handle 是个被广泛应用的系统，用于分配、管理数字对象和因特网上的其他资源，并将其解析成名为 handles 的永久标识符。可以将 handle 作为 URN 使用。参见 http://www.handle.net	
hnews	HNEWS 是 NNTP 新闻协议的一个 HTTP 隧道变体。hnews URL 语法设计与当前常用的新闻 URL 方案兼容。参见 http://www.ics.uci.edu/pub/ietf/uri/draft-stockwell-hnews-url-00.txt	
http	HTTP 协议。更多信息请参见本书	2616

（续）

方　案	描　述	RFC
https	SSL 上的 HTTP。 参见 http://sitesearch.netscape.com/eng/ssl3/draft302.txt	
iioploc	CORBA 扩展。可互操作的名字服务定义了一种 URL 格式的对象引用——iioploc，可以将其输入一个程序中，以获取包括名字服务在内的已定义远程服务。比如，以下 iioploc 标识符： iioploc://www.omg.org/NameService 就表示运行在 IP 地址与域名 www.omg.org 相对应的机器上的 CORBA 名字服务。参见 http://www.omg.org	
ilu	ILU（Inter-Language Unification，跨语言统一）系统是一个多语言对象接口系统。ILU 提供的对象接口隐藏了不同语言、不同地址空间和不同操作系统之间的实现差异。可以通过 ILU，用经过良好说明的语言无关接口来构建多语言的面向对象库（类库）。还可以将其用于实现分布式系统。 参见 ftp://parcftp.parc.xerox.com/pub/ilu/ilu.html	
imap	IMAP URL 方案用于分配 IMAP 服务器、邮箱、报文、MIME 主体 [MIME]，在因特网主机上搜索可以通过 IMAP 协议访问的程序	2192
IOR	CORBA 的互操作对象引用。 请参见 http://www.omg.org	
irc	irc URL 方案用于表示 IRC（Internet Relay Chat，因特网中继聊天）服务器或者 IRC 服务器上的独立实体（信道或人）。 参见 http://www.w3.org/Addressing/draft-mirashi-url-irc-01.txt	
isbn	建议用于 ISBN 书籍参考的方案。 参见 http://lists.w3.org/Archives/Public/www-talk/1991NovDec/0008.html	
java	用来识别 Java 类	
javascript	网景的浏览器会处理 javascript URL，如果冒号（：）后面有表达式，则计算它的值，只要表达式字符串的值不是未定义的，就会加载包含了这个值的页面	
jdbc	用于 Java SQL API	
ldap	允许因特网客户端直接访问 LDAP 协议	2255
lid	本地标识符方案。 参见 draft-blackketter-lid-00	
lifn	UTK 开发的批量文件分发分布式存储系统所使用的 lifn（location-independent file name，位置无关文件名）	
livescript	JavaScript 的曾用名	

（续）

方　案	描　述	RFC
lrq	参见 h323	
mailto	URL 方案 mailto 用于访问单个用户或服务的因特网邮件地址	2368
mailserver	1994 ~ 1995 年的老建议，支持将整条报文都编码到一个 URL 中去，这样（比如说）URL 可以自动向邮件服务器发送订阅邮件列表的电子邮件了	
md5	MD5 是一种密码校验和	
mid	mid 方案用电子邮件报文的 message-id（一部分）来引用一个特定的报文	2392 2111
mocha	参见 javascript	
modem	modem 方案描述了一条连接，连接到能够处理输入数据呼叫的终端上去	2806
mms、mmst、mmsu	MMS（Microsoft Media Server，微软媒体服务器）以流方式传送 ASF（Active Streaming Format，活动流格式）文件时使用的方案。强制使用 UDP 传输时，使用 mmsu 方案。强制使用 TCP 传输时，使用 mmst 方案	
news	news URL 方案指的是 USENET 新闻中的新闻组，或独立的文章。 news URL 使用下列两种格式之一： *news:<newsgroup-name>* 或 *news:<message-id>*	1738 1036
nfs	指的是 NFS 服务器上的文件和目录	2224
nntp	另一种引用 news 文章的方法，指定 NNTP 服务器上的 news 文章时很有用。nntp URL 看起来如下所示： nntp://*<host>*:*<port>*/*<newsgroup-name>*/*<article-num>* 注意，尽管 nntp URL 为文章资源指定了唯一的位置信息，但现在因特网上大部分 NNTP 服务器都配置为只允许从本地客户端访问，nntp URL 因此就无法指定全球可访问的资源了。因此，URL 的 news 格式更常作为识别新闻性文章的一种方式	1738 977
opaquelocktoken	以 URI 形式表示的 WebDAV 锁定令牌，用于标识特定锁的。每个成功的 LOCK 操作都会在响应主体的 lockdiscovery 特性中返回锁定令牌，也可以通过资源的锁发现操作找到它。参见 RFC 2518	
path	path 方案定义了一个统一的层次化命名空间。在这个空间中，path URN 就是由一些组件和可选的不透明字符串组成的序列。 参见 http://www.hypernews.org/~liberte/www/path.html	
phone	在“电话的 URL”中使用。在 RFC 2806 中被 tel: 取代	
pop	POP URL 指定了一个 POP 电子邮件服务器，一个可选的端口号、认证机制、认证 ID 和 / 或授权 ID	2384

（续）

方　案	描　述	RFC
pnm	Real 网络公司的流媒体协议	
pop3	POP3 URL 方案允许 URL 指定一个 POP3 服务器，允许其他协议使用通用的“用于邮件访问的 URL”取代对 POP3 的显式引用。在已过期的 draft-earhart-url-pop3-00.txt 中定义	
printer	用于服务定位标准的抽象 URL。 参见 draft-ietf-srvloc-printer-scheme-02.txt	
prospero	通过 prospero 目录服务访问的名字资源	1738
res	微软的方案，指定了一个要从某模块中获取的资源。包含一个字符串或数字资源类型和一个字符串或数字 ID	
rtsp	实时流协议，是 Real 网络公司现代流媒体控制协议的基础协议	2326
rvp	RVP 集合点协议的 URL，用于在某计算机网络上发布用户到来的通知。参见 draft-calsyn-rvp-01	
rwhois	RWhois 是在 RFC 1714 和 RFC 2167 中定义的因特网目录访问协议。RWhois URL 将 rwhois 的直接访问权赋予了客户端。 参见 http://www.rwhois.net/rwhois/docs/	
rx	一种结构，允许远程图像应用程序在 Web 页面中显示数据。 参见 http://www.w3.org/People/danield/papers/mobgui/	
sdp	sdp（session description protocol，会话描述协议）URL。参见 RFC 2327	
service	service 方案可以为任意网络服务提供访问信息。这些 URL 为基于客户端的网络软件提供了一种可扩展的框架，以获取使用网络服务所需的配置信息	2609
sip	sip* 族方案用于建立使用 sip（session initiation protocol，会话发起协议）的多媒体会议	2543
shttp	S-HTTP 是 HTTP 的超集，用于保护 HTTP 连接，它提供了大量机制用以实现保密性、认证功能和完整性。S-HTTP 没有被广泛采用，主要是被 HTTPS（经过 SSL 加密的 HTTP）取代了。 参见 http://www.homeport.org/~adam/shttp.html	
snews	经 SSL 加密的 news	
STANF	用于可靠网络文件名的老建议。与 URN 有关。 参见 http://Web3.w3.org/Addressing/#STANF	
t120	参见 h323	
tel	通过电话网打电话的 URL	2806
telephone	用于 tel 的早期草案中	

（续）

方　案	描　述	RFC
telnet	指定了可能会被 Telnet 协议访问的交互式业务。Telent URL 的格式如下所示： *telnet://<user>:<password>@<host>:<port>/*	1738
tip	支持 TIP 原子化的因特网事务处理	2371 2372
tn3270	根据 ftp://ftp.isi.edu/in-notes/iana/assignments/url-schemes 保留	
tv	TV URL 命名了一个特定的电视广播信道	2838
uuid	UUID（通用唯一标识符）不包含与位置有关的信息。也称为 GUID（全球唯一标识符）。由一个 128 位的唯一 ID 组成。它和 URN 一样，不会随时间发生变化。要使用无法或不应该依赖于特定的物理根名字空间（比如一个 DNS 名称）的通用标识符时，UUID URI 是很有用的。 参见 draft-kindel-uuid-uri-00.txt	
urn	持久的、与位置无关的 URN	2141
vemmi	允许 VEMMI（万用多媒体接口）客户端软件和 VEMMI 终端连接 VEMMI 兼容的服务。VEMMI 是一种在线多媒体应用服务的国际标准	2122
videotex	允许 videotex 客户端软件或终端连接与 ITU-T 和 ETSI videotex 标准兼容的 videotex 服务。 参见 http://www.ics.uci.edu/pub/ietf/uri/draft-mavrakis-videotex-url-spec-01.txt	
view-source	网景的 Navigator 的源码查看器。这些 view-source URL 可以显示用 JavaScript 生成的 HTML	
wais	广域信息服务——一种早期搜索引擎形式	1738
whois++	WHOIS++ 简单因特网目录协议的 URL。 参见 http://martinh.net/wip/whois-url.txt	1835
whodp	WhoDP（Widely Hosted Object Data Protocol，广泛托管对象数据协议）用于沟通大量动态、可重定位对象的当前位置和状态传递。WhoDP 程序通过“订阅”定位对象，接收与某对象有关的信息，并“发布”这些信息，控制此对象的位置和可见状态	
z39.50r，z39.50s	Z39.50 会话与检索 URL。Z39.50 是一种信息检索协议，不能很好地适用于主要为获取无状态数据而设计的检索模式。它将通用的用户查询设计为面向会话的多步任务，服务器在继续处理任务之前，会向客户端请求额外的参数，因此每一步任务都可能被临时挂起	2056

503

504

附录B

HTTP状态码

附录 B 是 HTTP 状态码及其含义的快速参考。

B.1 状态码分类

HTTP 状态码分为 5 类，如表 B-1 所示。

表B-1 状态码分类

总体范围	已定义范围	类 别
100~199	100~101	信息
200~299	200~206	成功
300~399	300~305	重定向
400~499	400~415	客户端错误
500~599	500~505	服务器错误

B.2 状态码

表 B-2 是 HTTP/1.1 规范定义的所有状态码的快速参考，表中概述了每种状态码及其含义。3.4 节曾详细地介绍了这些状态码及其用法。

表B-2 状态码

状态码	原因短语	含 义
100	Continue（继续）	收到了请求的起始部分，客户端应该继续请求
101	Switching Protocols（切换协议）	服务器正根据客户端的指示将协议切换成 Update 首部列出的协议
200	OK	服务器已成功处理请求
201	Created（已创建）	对那些要服务器创建对象的请求来说，资源已创建完毕
202	Accepted（已接受）	请求已接受，但服务器尚未处理
203	Non-Authoritative Information（非权威信息）	服务器已将事务成功处理，只是实体首部包含的信息不是来自原始服务器，而是来自资源的副本
204	No Content（没有内容）	响应报文包含一些首部和一个状态行，但不包含实体的主体内容
205	Reset Content（重置内容）	另一个主要用于浏览器的代码。意思是浏览器应该重置当前页面上所有的 HTML 表单
206	Partial Content（部分内容）	部分请求成功

（续）

状态码	原因短语	含　义
300	Multiple Choices（多项选择）	客户端请求了实际指向多个资源的 URL。这个代码是和一个选项列表一起返回的，然后用户就可以选择他希望使用的选项了
301	Moved Permanently（永久搬离）	请求的 URL 已移走。响应中应该包含一个 Location URL，说明资源现在所处的位置
302	Found（已找到）	与状态码 301 类似，但这里的搬离是临时的。客户端应该用 `Location` 首部给出的 URL 对资源进行临时定位
303	See Other（参见其他）	告诉客户端应该用另一个 URL 获取资源。这个新的 URL 位于响应报文的 `Location` 首部
304	Not Modified（未修改）	客户端可以通过它们所包含的请求首部发起条件请求。这个代码说明资源未发生过变化
305	Use Proxy（使用代理）	必须通过代理访问资源，代理的位置是在 `Location` 首部中给出的
306	（未用）	这个状态码当前并未使用
307	Temporary Redirect（临时重定向）	和状态码 301 类似。但客户端应该用 `Location` 首部给出的 URL 对资源进行临时定位
400	Bad request（坏请求）	告诉客户端它发送了一条异常请求
401	Unauthorized（未授权）	与适当的首部一起返回，在客户端获得资源访问权之前，请它进行身份认证
402	Payment Required（要求付款）	当前此状态码并未使用，是为未来使用预留的
403	Forbidden（禁止）	服务器拒绝了请求
404	Not Found（未找到）	服务器无法找到所请求的 URL
405	Method Not Allowed（不允许使用的方法）	请求中有一个所请求的 URI 不支持的方法。响应中应该包含一个 `Allow` 首部，以告知客户端所请求的资源支持使用哪些方法
406	Not Acceptable（无法接受）	客户端可以指定一些参数来说明希望接受哪些类型的实体。服务器没有资源与客户端可接受的 URL 相匹配时可使用此代码
407	Proxy Authentication Required（ 要求进行代理认证）	和状态码 401 类似，但用于需要进行资源认证的代理服务器
408	Request Timeout（请求超时）	如果客户端完成其请求时花费的时间太长，服务器可以回送这个状态码并关闭连接
409	Conflict（冲突）	发出的请求在资源上造成了一些冲突

（续）

状态码	原因短语	含　义
410	Gone（消失了）	除了服务器曾持有这些资源之外，与状态码 404 类似
411	Length Required（要求长度指示）	服务器要求在请求报文中包含 Content-Length 首部时会使用这个代码。发起的请求中若没有 Content-Length 首部，服务器是不会接受此资源请求的
412	Precondition Failed（先决条件失败）	如果客户端发起了一个条件请求，如果服务器无法满足其中的某个条件，就返回这个响应码
413	Request Entity Too Large（请求实体太大）	客户端发送的实体主体部分比服务器能够或者希望处理的要大
414	Request URI Too Long（请求 URI 太长）	客户端发送的请求所携带的请求 URL 超过了服务器能够或者希望处理的长度
415	Unsupported Media Type（不支持的媒体类型）	服务器无法理解或不支持客户端所发送的实体的内容类型
416	Requested Range Not Satisfiable（所请求的范围未得到满足）	请求报文请求的是某范围内的指定资源，但那个范围无效，或者未得到满足
417	Expectation Failed（无法满足期望）	请求的 Expect 首部包含了一个预期内容，但服务器无法满足
500	Internal Server Error（内部服务器错误）	服务器遇到了一个错误，使其无法为请求提供服务
501	Not Implemented（未实现）	服务器无法满足客户端请求的某个功能
502	Bad Gateway（网关故障）	作为代理或网关使用的服务器遇到了来自响应链中上游的无效响应
503	Service Unavailable（未提供此服务）	服务器目前无法为请求提供服务，但过一段时间就可以恢复服务
504	Gateway Timeout（网关超时）	与状态码 408 类似，但是响应来自网关或代理，此网关或代理在等待另一台服务器的响应时出现了超时
505	HTTP Version Not Supported（不支持的 HTTP 版本）	服务器收到的请求是以它不支持或不愿支持的协议版本表示的

507

附录C

HTTP首部参考

回想起第一个 HTTP 版本——版本 0.9，还是挺有趣的，因为它没有定义任何首部。尽管这样肯定存在弊端，但不得不为其简洁的优雅而啧啧称奇。

好吧，回到现实中来。现在有很多的 HTTP 首部，有一些是规范中定义的，还有一些是对规范的扩展。本附录提供了一些有关这些正式首部和扩展首部的背景知识，你还可以将其作为本书各种首部的索引使用，说明了这些首部的概念和特性是在正文的什么地方讨论的。这些首部大部分都很简单、直接，是它们之间或者与 HTTP 其他特性之间的交互使得事情变得比较复杂。本附录为所列首部提供了一些背景知识，并指导用户参阅书中详细讨论的对应章节。

本附录列出的首部是从 HTTP 规范、相关文档和我们自己使用 HTTP 报文和因特网上各种服务器和客户端的经验中提取出来的。

这个列表远远称不上完备。Web 中还有很多其他的扩展首部，更别说私有内部网络中使用的那些首部了。尽管如此，我们已经使这个表尽可能地完整了。当前的 HTTP/1.1 规范和官方首部及其规范描述参见 RFC 2616。

Accept

客户端用 Accept 首部来通知服务器可以接受哪些媒体类型。Accept 首部字段的值是客户端可以使用的媒体类型列表。如果 Web 浏览器无法显示 Web 上所有的多媒体对象类型，就可以在请求中包含 Accept 首部，这样浏览器就不会去下载它无法使用的视频或其他对象类型了。

为了防止服务器有多种版本的媒体类型，还可以在 Accept 首部字段中包含一个质量值（q 值）列表，用以告知服务器它优选哪种媒体类型。有关内容协商和 q 值的
508 完整讨论参见第 17 章。

类型 请求首部

注释 “*”是个特殊值用来通配媒体类型。比如，“*/*”表示所有类型，“image/*”表示所有的图片类型。

举例

```
Accept: text/*, image/*
Accept: text/*, image/gif, image/jpeg; q=1
```

Accept-Charset

客户端用 Accept-Charset 首部来通知服务器它可以接受哪些字符集或哪些是优选字符集。这个请求首部的值是个字符集列表和所列字符集可能的质量值。当服务器上有以多种可接受字符集表示的文档时，可以通过质量值告知服务器哪个字符集是优选的。有关内容协商和 q 值的完整讨论参见第 17 章。

类型 请求首部

注释 与 Accept 首部一样，“*”是个特殊字符。如果有“*”，就表示除了显式地用值设置的字符集之外的所有字符集。如果没有“*”，那么值字段中没有设置的所有字符集的 q 值都默认为零，这不包括字符集 iso-latin-1，它的默认值为 1。

基本语法
```
Accept-Charset: 1# ((charset | "*") [";" "q" "=" qvalue])
```

举例
```
Accept-Charset: iso-latin-1
```

Accept-Encoding

客户端用 Accept-Encoding 首部来告知服务器它可以接受哪些编码方式。如果服务器所持有的内容是经过编码的（可能是压缩过的），这个请求首部可以告诉服务器客户端是否会接受它。第 17 章探讨了 Accept-Encoding 首部。

类型 请求首部

基本语法
```
Accept-Encoding: 1# ((content-coding | "*") [";" "q" "=" qvalue])
```

举例
```
Accept-Encoding:[1]
Accept-Encoding:gzip
Accept-Encoding: compress;q=0.5,
gzip;q=1
```

509

Accept-Language

和其他 Accept 首部一样，客户端可以通过 Accept-Language 请求首部通知服务器可接受或优选哪些语言（比如，内容所使用的自然语言）。第 17 章详尽介绍了 Accept-Language 首部。

类型 请求首部

基本语法
```
Accept-Language: 1# (language-range [";" "q" "=" qvalue])
language-range = ((1*8ALPHA * ("-" 1*8ALPHA)) | "*")
```

举例
```
Accept-Language: en
Accept-Language: en;q=0.7, en-gb;q=0.5
```

注 1：这并不是印刷错误。它指的是身份编码——也就是未编码的内容。如果提供了空的 Accept-Encoding 首部，就说明只能接受未编码的内容。

Accept-Ranges

Accept-Ranges 首部与其他 Accept 首部不同——它是服务器使用的一种响应首部，用来告知客户端它们是否接受请求资源的某个范围。如果这个首部有赋值的话，这个值就说明了服务器允许对指定资源的哪个范围类型进行访问。

客户端可以在没有收到这个首部的情况下，对某资源发起范围请求。如果服务器不支持对那个资源的范围请求，可以以适当的状态码进行响应[2]，将 Accept-Ranges 的值设置为 none。服务器可以为普通请求发送 none 值，这样客户端以后就不会发起范围请求了。

第 17 章完整介绍了 Accept-Ranges 首部。

类型 响应首部
基本语法 `Accept-Ranges: 1# range-unit | none`
举例 `Accept-Ranges: none`
`Accept-Ranges: bytes`

Age

Age 首部可以告诉接收端响应已产生了多长时间。对于原始服务器是在多久之前产生的响应或是在多久之前向原始服务器再次验证响应而言，这是发送端所做的最好的猜测。首部的值是发送端所做的猜测，以秒为单位递增。更多有关 Age 首部的内容参见第 7 章。

类型 响应首部
510 **注释** HTTP/1.1 缓存必须在发送的每条响应中都包含一个 Age 首部。
基本语法 `Age: delta-seconds`
举例 `Age: 60`

Allow

Allow 首部用于通知客户端可以对特定资源使用哪些 HTTP 方法。

类型 响应首部
注释 发送 405 Method Not Allowed 响应的 HTTP/1.1 服务器必须包含 Allow 首部。[3]
基本语法 `Allow: #Method`

注 2：比如，状态码 416（参见 3.4.4 节）。
注 3：更多有关状态码 405 的内容参见 3.4 节。

举例 `Allow: GET, HEAD`

Authorization

`Authorization` 首部是由客户端发送的，用来向服务器回应自己的身体验证信息。客户端收到来自服务器的 401 Authentication Required 响应后，要在其请求中包含这个首部。这个首部的值取决于所使用的认证方案。有关 `Authorization` 首部的详细讨论参见第 14 章。

类型 请求首部

基本语法 `Authorization: authentication-scheme #authentication-param`

举例 `Authorization: Basic YnJpYW4tdG90dHk6T3ch`

Cache-Control

`Cache-Control` 首部用于传输对象的缓存信息。这个首部是 HTTP/1.1 引入的比较复杂的首部之一。它的值是一个缓存指令，给出了与某个对象可缓存性有关的缓存特有指令。

第 7 章简要介绍了缓存，还说明了与这个首部有关的特定细节。

类型 通用首部

举例 `Cache-Control: no-cache`

Client-ip

`Client-ip` 首部是一些比较老的客户端和代理使用的扩展首部，用来传输运行客户端程序的计算机 IP 地址。

类型 扩展请求首部

注释 实现者应该了解这个首部的值所提供的信息是不安全的。

基本语法 `Client-ip: ip-address`

举例 `Client-ip: 209.1.33.49`

Connection

`Connection` 首部是个多少有点儿过载了的首部，它可能会把你搞晕。这个首部用于扩展了 keep-alive 连接的 HTTP/1.0 客户端，keep-alive 连接用于控制信息。[4] 在

注 4：更多有关 keep-alive 和持久连接的内容参见第 4 章。

HTTP/1.1 中，能识别出大部分较老的语义，但这个首部被赋予了新的功能。

在 HTTP/1.1 中，Connection 首部的值是一个标记列表，这些标记对应各种首部名称。应用程序收到带有 Connection 首部的 HTTP/1.1 报文后，应该对列表进行解析，并删除报文中所有在 Connection 首部列表中出现过的首部。它主要用于有代理网络环境，这样服务器或其他代理就可以指定不应传递的逐跳首部了。

close 是一个典型的标记值。这个标记意味着响应结束之后，连接会被关闭。不支持持久连接的 HTTP/1.1 应用程序要在所有请求和响应中插入带有 close 标记的 Connection 首部。

类型 通用首部

注释 虽然 RFC 2616 没有专门声明将 keep-alive 作为连接标记使用，有些（包括那些将 HTTP/1.1 作为版本号发送的）浏览器还是会在发起请求时使用它。

基本语法 `Connection: 1# (connection-token)`

512 **举例** `Connection: close`

Content-Base

服务器可以通过 Content-Base 首部为响应主体部分中要解析的 URL 指定一个基础 URL。[5]Content-Base 首部的值是一个绝对 URL，可以用来解析在实体内找到的相对 URL。

类型 实体首部

注释 RFC 2616 中没有定义这个首部。它是早期在 RFC 2068 中定义的，RFC 2068 是一个较早的 HTTP/1.1 规范草案，已经从官方规范中删除了。

基本语法 `Content-Base: absoluteURL`

举例 `Content-Base: http://www.joes-hardware.com/`

Content-Encoding

Content-Encoding 首部用于说明是否对某对象进行过编码。通过对内容进行编码，服务器可以在发送响应之前将其进行压缩。Content-Encoding 首部的值可以告诉客户端，服务器对对象执行过哪种或哪些类型的编码。有了这个信息，客户端就可以对报文进行解码了。

注 5：更多有关基础 URL 的信息参见 2.3 节。

有时服务器会对某个实体进行多种编码，在这种情况下，必须按照执行的顺序将编码列出来。

类型 实体首部
基本语法 `Content-Encoding: 1# content-coding`
举例 `Content-Encoding: gzip`
`Content-Encoding: compress, gzip`

Content-Language

Content-Language 首部用来告诉想要理解对象的客户端，应该理解哪种自然语言。比如说，一篇用法语编写的文档就应该有一个表示法语的 Content-Language 值。如果在响应中没有提供这个值，对象就是提供给所有用户的。首部值中有多种语言就说明对象适用于使用所列各种语言的用户。

这里需要说明的是，这个首部的值可能只表示了此对象目标用户的自然语言，而不是对象中包含的所有或者任意一种语言。而且，此首部并不局限于文本或书面数据对象；图像、视频和其他媒体类型也可以用其目标用户的自然语言来标识。

513

类型 实体首部
基本语法 `Content-Language: 1# language-tag`
举例 `Content-Language: en`
`Content-Language: en, fr`

Content-Length

Content-Length 首部说明实体主体部分的长度或尺寸。如果对 HEAD HTTP 请求的响应报文中有这个首部，此首部的值就表示如果发送的的话，实体主体部分的长度（实际上并不发送主体）。

类型 实体首部
基本语法 `Content-Length: 1*DIGIT`
举例 `Content-Length: 2417`

Content-Location

Content-Location 首部包含在一个 HTTP 报文中，给出了与报文的实体部分相对应的 URL。对可能有多个 URL 的对象来说，响应报文中可以包含一个 Content-Location 首部，说明用来产生响应的对象的 URL。Content-Location 可以与所请求的 URL 不同。服务器通常会用它将客户端导向或重定向到一个新 URL 上去。

如果 URL 是相对的，就应该相对于 Content-Base 首部加以解释。如果没有提供 Content-Base 首部，就应该使用请求中的 URL。

类型 实体首部
基本语法 Content-Location: (absoluteURL | relativeURL)
举例 Content-Location: http://www.joes-hardware.com/index.html

Content-MD5

514 Content-MD5 首部是服务器用来对报文主体进行报文完整性检查的。只有原始服务器或发起请求的客户端可以在报文中插入 Content-MD5 首部。首部值是（可能需要编码的）报文主体的 MD5 摘要。[6]

通过这个首部的值可以端到端地检查数据，在检查传输过程中是否对数据进行了无意的修改时非常有用。不应该将其用于安全目的。

RFC 1864 更详细地定义了这个首部。

类型 实体首部
注释 根据 RFC 1864 的定义，MD5 摘要值是一个 Base-64（参见附录 E）或 128 位的 MD5 摘要。
基本语法 Content-MD5: md5-digest
举例 Content-MD5: Q2h1Y2sgSW51ZwDIAXR5IQ==

Content-Range

请求传输某范围内的文档时，产生的结果由 Content-Range 首部给出。它提供了请求实体所在的原始实体内的位置（范围），还给出了整个实体的长度。

如果值为“*”，而不是整个实体的长度，就意味着发送响应时，长度未知。

更多有关 Content-Range 的内容请参见第 15 章。

类型 实体首部
注释 以 206 Partial Content 响应码进行响应的服务器，不能包含将“*”作为长度使用的 Content-Range 首部。
举例 Content-Range: bytes 500-999 / 5400

注 6：MD5 摘要是在 RFC 1864 中定义的。

Content-Type

Content-Type 首部说明了报文中对象的媒体类型。

类型 实体首部

基本语法 `Content-Type: media-type`

举例 `Content-Type: text/html; charset=iso-latin-1`

Cookie

Cookie 首部是用于客户端识别和跟踪的扩展首部。第 11 章详细讨论了 Cookie 首部及其用法（还请参见 Set-Cookie）。

类型 扩展请求首部

举例 `Cookie: ink=IUOK164y59BC708378908CFF89OE5573998A115`

Cookie2

Cookie2 首部是用于客户端识别和跟踪的扩展首部。Cookie2 用于识别请求发起者能够理解哪种类型的 Cookie。在 RFC 2965 中对其进行了更加详细的定义。

第 11 章详细地讨论了 Cookie2 首部及其用法。

类型 扩展请求首部

举例 `Cookie2: $version="1"`

Date

Date 首部给出了报文创建的日期和时间。服务器响应中要包含这个首部，因为缓存在评估响应的新鲜度时，要用到这个服务器认定的报文创建时间和日期。对客户端来说，这个首部是可选的，但包含这个首部会更好。

类型 通用首部

基本语法 `Date: HTTP-date`

举例 `Date: Tue, 3 Oct 1997 02:15:31 GMT`

HTTP 有几种特定的日期格式。这种格式是在 RFC 822 中定义的，这是 HTTP/1.1 报文的优选格式。但在早期的 HTTP 规范中，没有明确说明日期的格式，因此服务器和客户端的实现者使用了一些其他格式，为了解决这些遗留问题仍然需要支持这些格式。你可能会碰到 RFC 850 中说明的那些日期格式，asctime() 系统调用产生的日期格式。下面是用这些格式所表示的上述日期：

```
Date: Tuesday, 03-Oct-97 02:15:31 GMT RFC 850 format
Date: Tue Oct 3 02:15:31 1997 asctime( ) format
```

大家都不喜欢 asctime() 格式，因为它表示的是本地时间，而且没有说明时区（比如，GMT）。总的来说，日期首部应该是 GMT 时间；但健壮的应用程序在处理
516 日期时，应该能够处理那些没有指定时区，或者包含了非 GMT 时间的 Date 值。

ETag

ETag 首部为报文中包含的实体提供了*实体标记*。实体标记实际上就是一种标识资源的方式。

第 15 章曾探讨过实体标记及其与资源之间的关系。

类型 实体首部

基本语法
```
ETag: entity-tag
```

举例
```
ETag: "11e92a-457b-31345aa"
ETag: W/"11e92a-457b-3134b5aa"
```

Expect

客户端通过 Expect 首部告知服务器它们需求某种行为。现在此首部与响应码 100 Continue 紧密相关（参见 3.4.1 节）。

如果服务器无法理解 Expect 首部的值，就应该以状态码 417 Expectation Failed 进行响应。

类型 请求首部

基本语法
```
Expect: 1# ("100-continue" | expectation-extension)
```

举例
```
Expect: 100-continue
```

Expires

Expires 首部给出了响应失效的日期和时间。这样，像浏览器这样的客户端就可以缓存一份副本，在这个时间到期之前，不用去询问服务器它是否有效了。

第 7 章曾讨论过 Expires 首部的用法——尤其是，它是如何与缓存关联，怎样与原始服务器进行响应再验证的。

类型 实体首部

基本语法
```
Expires: HTTP-date
```

举例
```
Expires: Thu, 03 Oct 1997 17:15:00 GMT
```

From

From 首部说明请求来自何方。其格式就是（RFC 1123 规定的）客户端用户的有效电子邮件地址。

使用 / 填充这个首部存在潜在的隐私问题。客户端实现者在请求报文中包含这个首部之前，应该通知用户，请他们作出选择。有些人会去收集不请自来的邮件报文中携带的电子邮件地址，这可能造成潜在的滥用。因此，未做声明就将此首部广播出去的实现者一定会非常懊悔，他们不得不向愤怒的用户说抱歉。

类型 请求首部

基本语法 `From: mailbox`

举例 `From: slurp@inktomi.com`

Host

客户端通过 Host 首部为服务器提供客户端想要访问的那台机器的因特网主机名和端口号。主机名和端口号来自客户端所请求的 URL。

只要服务器能够在同一台机器（即，在同一个 IP 地址）上提供多个不同的主机名，服务器就可以通过 Host 首部，根据主机名来区分不同的相对 URL。

类型 请求首部

注释 HTTP/1.1 客户端必须在所有请求中包含 Host 首部。所有的 HTTP/1.1 服务器都必须以 400 Bad Request 状态码去响应没有提供 Host 首部的客户端。

基本语法 `Host: host [":" port]`

举例 `Host: www.hotbot.com:80`

If-Modified-Since

If-Modified-Since 请求首部用来发起条件请求。客户端可以用 GET 方法去请求服务器上的资源，而响应则取决于客户端上次请求此资源之后，该资源是否被修改过。

如果对象未被修改过，服务器会回送一条 304 Not Modified 响应，而不会回送此资源。如果对象被修改过，服务器就会像对待非条件 GET 请求一样进行响应。第 7 章详细地讨论了条件请求。

类型 请求首部

基本语法 `If-Modified-Since: HTTP-date`

举例 `If-Modified-Since: Thu, 03 Oct 1997 17:15:00 GMT`

If-Match

与 If-Modified-Since 首部类似，If-Match 首部也可以用于发起条件请求。If-Match 请求使用的是实体标记，而不是日期。服务器将对比 If-Match 首部的实体标记与资源当前的实体标记，如果标记匹配，就将对象返回。

服务器应该用 If-Match 值“*”与某资源拥有的所有实体标记进行匹配。除非服务器上没有这个资源了，否则“*”总会与实体标记相匹配。

类型 请求首部
基本语法 `If-Match: ("*" | 1# entity-tag)`
举例 `If-Match: "11e92a-457b-31345aa"`

If-None-Match

与所有 If 首部一样，If-None-Match 首部可以用于发起条件请求。客户端为服务器提供一个实体标记列表，服务器将这些标记与它拥有的资源实体标记进行比较，只在都不匹配的时候才将资源返回。

这样缓存就可以只在资源已被修改的情况下才更新。通过 If-None-Match 首部，缓存可以用一条请求使它拥有的实体失效，同时在响应中接收新的实体。第 7 章曾讨论过条件请求。

类型 请求首部
基本语法 `If-None-Match: ("*" | 1# entity-tag)`
举例 `If-None-Match: "11e92a-457b-31345aa"`

If-Range

与所有 If 首部一样，If-Range 首部可以用于发起条件请求。应用程序拥有某范围内资源的副本，它要对范围进行再验证，如果范围无效的话，要获取新的资源，在这种情况下会使用这个首部。第 7 章详细讨论了条件请求。

类型 请求首部
519 **基本语法** `If-Range: (HTTP-date | entity-tag)`
举例
```
If-Range: Tue, 3 Oct 1997 02:15:31 GMT
If-Range: "11e92a-457b-3134b5aa"
```

If-Unmodified-Since

If-Unmodified-Since 和 If-Modified-Since 首部是一对“双胞胎”。在请求

中包含此首部就可以发起条件请求。服务器应该去查看首部的日期值，只有在从该首部提供的日期之后，对象都未被修改过，才会返回对象。第 7 章详细介绍了条件请求。

类型 请求首部
基本语法 `If-Unmodified-Since: HTTP-date`
举例 `If-Unmodified-Since: Thu, 03 Oct 1997 17:15:00 GMT`

Last-Modified

`Last-Modified` 首部试图提供这个实体最后一次被修改的相关信息。这个值可以说明很多事情。比如，资源通常都是一台服务器上的文件，因此 `Last-Modified` 值可能就是服务器的文件系统所提供的最后修改时间。另一方面，对于那些动态创建的资源（比如，由脚本创建的资源），`Last-Modified` 值可能就是创建响应的时间。

服务器要注意，`Last-Modified` 时间不应该是未来的时间。如果它比 `Date` 首部中要发送的值还迟，HTTP/1.1 服务器就会将 `Last-Modified` 时间重置。

类型 实体首部
基本语法 `Last-Modified: HTTP-date`
举例 `Last-Modified: Thu, 03 Oct 1997 17:15:00 GMT`

Location

服务器可以通过 `Location` 首部将客户端导向某个资源的地址，这个资源可能在客户端最后一次请求之后被移动过，也可能是在对请求的响应中创建的。

类型 响应首部
基本语法 `Location: absoluteURL`
举例 `Location: http://www.hotbot.com`

520

Max-Forwards

这个首部只能和 TRACE 方法一同使用，以指定请求所经过的代理或其他中间节点的最大数目。它的值是个整数。所有收到带此首部的 TRACE 请求的应用程序，在将请求转发出去之前都要将这个值减 1。

如果应用程序收到请求时，这个首部的值为零，就要向请求回应一条 200 OK 响应，并在实体的主体部分包含原始请求。如果 TRACE 请求中没有 `Max-Forwards` 首部，就假定没有转发最大次数的限制。

其他 HTTP 方法都应该忽略这个首部。更多有关 TRACE 方法的信息参见 3.3 节。

类型 请求首部
基本语法 `Max-Forwards: 1*DIGIT`
举例 `Max-Forwards: 5`

MIME-Version

MIME 是 HTTP 的近亲。尽管两者存在根本区别，但有些 HTTP 服务器确实构造了一些在 MIME 规范下同样有效的报文。在这种情况下，服务器可以提供 MIME 版本的首部。

尽管 HTTP/1.0 规范中提到过这个首部，但它从未写入官方规范。很多比较老的服务器会发送带有这个首部的报文，但这些报文通常都不是有效的 MIME 报文，这样会让人觉得这个首部令人迷惑且不可信。

类型 扩展的通用首部
基本语法 `MIME-Version: DIGIT "." DIGIT`
举例 `MIME-Version: 1.0`

Pragma

Pragma 首部用于随报文传送一些指令。这些指令几乎可以包含任何内容，但通常会用这些指令来控制缓存的行为。Pragma 首部的目标可以是接收这条报文的所有应用程序，因此代理和网关一定不能将其删除。

最常见的 Pragma 形式——`Pragma: no-cache` 是一个请求首部，通过它可以迫使缓存在有新鲜副本可用的情况下，向原始服务器请求文档或对其进行再验证。用户
521 点击重新加载 / 刷新按钮时，浏览器就会发出这个首部。很多服务器会将 `Pragma: no-cache` 作为响应首部发送（和 `Cache-Control:no-cache` 等价）。尽管这个首部得到了广泛的使用，但从技术上来说，并没有定义过其行为，不是所有的应用程序都支持 Pragma 响应首部。

第 7 章探讨了 Pragma 首部以及 HTTP/1.0 应用程序如何通过它来控制缓存。

类型 请求首部
基本语法 `Pragma: 1# pragma-directive`[7]
举例 `Pragma: no-cache`

注 7：规范中定义的唯一的一个 Pragma 指令就是 no-cache，但我们可能会碰到其他作为规范扩展而定义的 Pragma 首部。

Proxy-Authenticate

Proxy-Authenticate 首部的功能类似于 WWW-Authenticate 首部。代理会这个首部来质询发送请求的应用程序，要求其对自身进行认证。第 14 章详细讨论了这个质询 / 响应过程和 HTTP 的其他安全机制。

如果一台 HTTP/1.1 代理服务器发送了一条 407 Proxy Authentication Required 响应，就必须包含 Proxy-Authenticate 首部。

代理和网关在解释所有的 Proxy 首部时要特别小心。通常它们都是逐跳首部，只适用于当前的连接。比如，Proxy-Authenticate 首部会要求对当前的连接进行认证。

类型 响应首部
基本语法 `Proxy-Authenticate: challenge`
举例 `Proxy-Authenticate: Basic realm="Super Secret Corporate FinancialDocuments"`

Proxy-Authorization

Proxy-Authorization 首部的功能与 Proxy-Authorization 首部类似。客户端应用程序可以用它来响应 Proxy-Authenticate 质询。更多有关质询 / 响应安全机制工作原理的内容参见第 14 章。

类型 请求首部
基本语法 `Proxy-Authorization: credentials`
举例 `Proxy-Authorization: Basic YnJpYW4tdG90dHk6T3ch`

Proxy-Connection

Proxy-Connection 首部的语义与 HTTP/1.0 Connection 首部类似。在客户端和代理之间可以用它来指定与连接（主要是 keep-alive 连接）有关的选项。[8] 它并不是一个标准的首部，标准委员会把它当作一个临时首部。但它得到了浏览器和代理的广泛使用。

浏览器实现者创建了 Proxy-Connection 首部，来解决客户端发送的 HTTP/1.0 Connection 首部被哑代理盲转发的问题。收到被盲转发的 Connection 首部的服务器会将客户端连接的功能与代理连接的功能混淆起来。

注 8：更多有关 keep-alive 和持久连接的内容参见第 14 章。

客户端知道要经过代理传输时，就会发送 Proxy-Connection 首部，而不是 Connection 首部。服务器如果无法识别 Proxy-Connection 首部，就会将其忽略，这样，对首部进行盲转发的哑代理就不会带来任何问题了。

如果在从客户端到服务器的路径上有多个代理，这种解决方法就会有问题。如果第一个代理将首部盲转发给第二个能够理解它的代理，那么第二个代理就会像服务器看到 Connection 首部一样，无法理解。

这是 HTTP 工作组的解决方案存在的问题——他们将其作为一种黑客工具，可以解决单个代理的问题，但无法解决更大的问题。尽管如此，这种方式确实能够处理一些比较常见的情况，而且由于网景的 Navigator 和微软的 Internet Explorer 的较老版本都实现了这个首部，因而代理的实现者也需要对其进行处理，更多信息参见第 4 章。

类型 通用首部
基本语法 `Proxy-Connection: 1# (connection-token)`
举例 `Proxy-Connection: close`

Public

服务器可以用 Public 首部告知客户端它支持哪些方法。今后客户端发起的请求就可以使用这些方法了。代理收到服务器发出的带有 Public 首部的响应时，要特别小心。这个首部说明的是服务器支持的方法，而不是代理的，因此代理在将响应发送给客户端之前，要对首部的方法列表加以编辑，或者将此首部删除。

523 **类型** 响应首部
注释 RFC 2616 中没有定义这个首部。它是之前在 HTTP/1.1 规范的早期草案 RFC 2068 中定义的，而官方规范已经将其删除了。
基本语法 `Public: 1# HTTP-method`
举例 `Public: OPTIONS, GET, HEAD, TRACE, POST`

Range

在请求某实体的部分内容中会用到 Range 首部。它的值说明了报文所包含实体的范围。

请求某范围内的文档可以更有效地对大型对象发出请求（分段对其发出请求），或者更有效地从传输错误中恢复（允许客户端请求没有完成的那部分资源）。第 15 章详细说明了范围请求和能实现范围请求的首部。

类型 实体首部

举例 `Range: bytes=500-1500`

Referer

在客户端请求中插入 `Referer` 首部，可以使服务器知道客户端是从哪里获得其请求的 URL。这是一种对服务器有益的自愿行为，这样服务器就可以更好地记录请求，或执行其他任务了。Referer 的拼写错误要回溯到 HTTP 的早期，令世界各地以英语为母语的文字编辑们万分沮丧。

浏览器所做的工作相当简单。如果在主页 A 上点击一个链接，进入主页 B，浏览器就会在请求中插入一个带有值 A 的 `Referer` 首部。只有在你点击链接的时候，浏览器才会插入 `Referer` 首部；自己输入的 URL 中不会包含 `Referer` 首部。

因为有些页面是私有的，所以这个首部会有隐私问题。尽管有些只是毫无根据的猜想，但 Web 服务器及其管理者确实可以通过这个首部看到你来自何方，这样他们就能更好地追踪你的浏览行为了。因此，HTTP/1.1 规范建议应用程序编写者让用户来选择是否传输这个首部。

类型 请求首部

基本语法 `Referer: (absoluteURL | relativeURL)`

举例 `Referer: http://www.inktomi.com/index.html`

524

Retry-After

服务器可以用 `Retry-After` 首部告知客户端什么时候重新发送某资源的请求。这个首部可以与 503 Service Unavailable（服务不可用）状态码配合使用，给出客户端可以重试其请求的具体日期和时间（或者秒数）。

服务器还可以在将客户端重定向到资源时，通过这个首部通知客户端在对重定向的资源发送请求之前需要等待的时间。[9] 对那些正在创建动态资源的服务器来说，这个首部是非常有用的，服务器可以通过它将客户端重定向到新创建的资源，并给出了资源创建所需的时间。

类型 响应首部

基本语法 `Retry-After: (HTTP-date | delta-seconds)`

举例
```
Retry-After: Tue, 3 Oct 1997 02:15:31 GMT
Retry-After: 120
```

注 9：更多有关服务器重定向响应的信息参见表 3-8。

Server

Server 首部与 User-Agent 首部类似。它为服务器提供了一种向客户端标识自己的方式。它的值就是服务器名字和一个可选的服务器注释。

Server 首部是用来识别服务器软件的，而且包含了与软件有关的附加注释，所以其格式比较随意。如果编写的软件与服务器标识自己的方式有关，就应该测试服务器软件，看看它会发回什么内容，因为这些标记会随软件及其发布版本的不同而有所不同。

像 User-Agent 首部一样，如果较老的代理或网关在 Server 首部中插入了相当于 Via 首部的内容，千万不要感到吃惊。

类型 响应首部
基本语法 `Server: 1* (product | comment)`
举例
```
Server: Microsoft-Internet-Information-Server/1.0
Server: Websitepro/1.1f (s/n wpo-07d0)
Server: apache/1.2b6 via proxy gateway CERN-HTTPD/3.0 libwww/2.13
```

Set-Cookie

525 Set-Cookie 首部是 Cookie 首部的搭档。第 11 章介绍了这个首部的用法。

类型 扩展响应首部
基本语法 `Set-Cookie: command`
举例
```
Set-Cookie: lastorder=00183; path=/orders
Set-Cookie: private_id=519; secure
```

Set-Cookie2

Set-Cookie2 首部是对 Set-Cookie 首部的扩展。第 11 章详细了探讨了这个首部的用法。

类型 扩展响应首部
基本语法 `Set-Cookie2: command`
举例
```
Set-Cookie2: ID="29046"; Domain=".joes-hardware.com"
Set-Cookie2: color=blue
```

TE

TE 首部的名字起得不太好（本应该将其命名为 Accept-Transfer-Encoding），

它的功能与 Accept-Encoding 首部类似，但它是用于传输编码的。TE 首部还可以用来说明客户端能否处理位于分块编码的响应拖挂中的首部。更多有关 TE 首部、分块编码和拖挂的内容参见第 15 章。

类型 请求首部

注释 如果这个值为空，就只接受分块传输编码。特定标记“trailers”说明分块响应中可以接受 Trailer 首部。

基本语法
```
TE: # (transfer-codings)
transfer-codings= "trailers" | (transfer-extension [accept-params])
```

举例
```
TE:
TE: chunked
```

Trailer

Trailer 首部用于说明报文拖挂中提供了哪些首部。第 15 章详细说明了分块编码和拖挂。

类型 通用首部

基本语法
```
Trailer: 1#field-name
```

举例
```
Trailer: Content-Length
```

Title

Title 首部不像人们所期望的那样，会给出实体标题的规范化首部。这个首部是早期 HTTP/1.0 扩展的一部分，主要用于 HTML 页面，这些 HTML 页面有着服务器可以使用的明确的标题标记。但即使不是大部分，也有很多 Web 媒体类型没有便捷的标题解析手段，这个标题的用处有限。因此，尽管网络上一些比较老的服务器仍然在忠实地发送这个首部，但它从未成为官方规范。

类型 响应首部

注释 RFC 2616 中没有定义 Title 首部。最早是在 HTTP/1.0 草案（http://www.w3.org/Protocols/HTTP/HTTP2.html）中定义的，但之后就从官方规范中删除了。

基本语法
```
Title: document-title
```

举例
```
Title: CNN Interactive
```

Transfer-Encoding

如果要通过某些编码来安全地传送 HTTP 报文主体，报文中就要包含 Transfer-

Encoding 首部。它的值是一个对报文主体执行过的编码的列表。如果进行了多种编码，就将其按序排列。

Transfer-Encoding 首部与 Content-Encoding 首部不同，因为服务器或其他中间应用程序是通过执行 Transfer-Encoding 对要传输的报文进行编码的。

第 15 章介绍过传输编码。

类型 通用首部
基本语法 `Transfer-Encoding: 1# transfer-coding`
举例 `Transfer-Encoding: chunked`

UA-(CPU, Disp, OS, Color, Pixels)

这些 User-Agent 首部是非标准的，现在也不常见了。它们提供了客户端机器的相
527 关信息，以便服务器更好地进行内容选择。比如，如果服务器知道用户机器只有一个 8 位彩色显示器，服务器就可以选择适合那类显示器的图片了。

有些首部给出了与客户端相关的信息，不使用这些首部就无法获知这些信息。所有这样的首部都有一些安全方面的隐患（更多信息参见第 14 章）。

类型 扩展请求首部
注释 RFC 2616 没有定义这些首部，而且不推荐使用这些首部。
基本语法

```
"UA" "-" ("CPU" | "Disp" | "OS" | "Color" | "Pixels")
":" machine-value
machine-value = (cpu | screensize | os-name |display-
color-depth)
```

举例

```
UA-CPU: ×86              客户端机器的 CPU
UA-Disp: 640, 480, 8     客户端显示器的尺寸和色彩深度
UA-OS: Windows 95        客户端机器的操作系统
UA-Color: color8         客户端显示器的色彩深度
UA-Pixels: 640×480       客户端显示器的尺寸
```

Upgrade

Upgrade 首部为报文发送者提供了一种手段，使其指定另一种可能完全不同协议并将此意愿向外广播。比如，HTTP/1.1 客户端可以向服务器发送一条 HTTP/1.0 请求，其中包含了值为“HTTP/1.1”的 Update 首部，这样客户端就可以测试一下服务器是否也使用 HTTP/1.1 了。

如果服务器也可以使用 HTTP/1.1，就可以发送一条适当的响应，让客户端知道可以使用新的协议。这样就提供了一种切换使用其他协议的有效方式。现在大部分服务器都只兼容 HTTP/1.0，通过这种策略，在判定服务器确实能够使用 HTTP/1.1 之前，客户端就不会用很多的 HTTP/1.1 首部骚扰服务器了。

服务器发送 101 Switching Protocols 响应时，必须包含这个首部。

类型 通用首部
基本语法 `Upgrade: 1# protocol`
举例 `Upgrade: HTTP/2.0`

User-Agent

客户端应用程序用 `User-Agent` 首部来标识其类型，与服务器的 `Server` 首部类似。它的值就是应用程序的名称，可能还会有一个描述性注释。

528

这个首部的格式比较随意。它的值会随客户端应用程序和发布版本的不同而有所不同。有时这个首部甚至会包含一些有关客户端机器的信息。

与 `Server` 首部一样，如果较老的代理或网关应用程序在 `User-Agent` 首部中插入了与 `Via` 首部等效的内容，请不要感到惊奇。

类型 请求首部
基本语法 `User-Agent: 1* (product | comment)`
举例 `User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)`

Vary

服务器通过 `Vary` 首部来通知客户端，在服务器端的协商中会使用哪些来自客户端请求的首部。[10] 它的值是一个首部列表，服务器会去查看这些首部，以确定将什么内容作为响应发回给客户端。

根据客户端 Web 浏览器特性来发送特定 HTML 页面的服务器就是一例。为某个 URL 发送这些特定页面的服务器会包含一个 `Vary` 首部，以说明它是查看了请求的 `User-Agent` 首部之后，才决定发送什么内容作为响应的。

代理缓存也会使用 `Vary` 首部。更多有关 `Vary` 首部与已缓存的 HTTP 响应关联方式的信息参见第 7 章。

注 10：更多与内容协商有关的内容参见第 17 章。

类型 响应首部
基本语法 `Vary: ("*" | 1# field-name)`
举例 `Vary: User-Agent`

Via

`Via` 首部用于在报文经过代理和网关时对其进行跟踪。这是一个信息首部，通过它可以看出哪些应用程序在对请求和响应进行处理。

报文在向客户端或服务器传输的途中经过某个 HTTP 应用程序时，这个应用程序可以通过 `Via` 首部对通过它传输的报文进行标记。这是个 HTTP/1.1 首部，而很多较老的应用程序会在请求和响应的 `User-Agent` 或 `Server` 首部插入类似 `Via` 的字符串。

如果报文是通过多个中间应用程序传输的，那么每个应用程序都会向其 `Via` 字符串
529 中附加一些内容。必须通过 HTTP/1.1 代理和网关来插入 `Via` 首部。

类型 通用首部
基本语法 `Via: 1# (received-protocol received-by [comment])`[11]
举例 `Via: 1.1 joes-hardware.com ( Joes-Server/1.0)`

上面这个例子说明报文是通过运行在机器 joes-hardware.com 上的 Joes 的服务器软件 1.0 版传输的。`Via` 首部的格式应该如下所示：

```
HTTP-Version machine-hostname (Application-Name-Version)
```

Warning

`Warning` 首部可以给出更多与请求过程中所发生情况有关的信息。它为服务器提供了一种手段，可以发送除状态码或原因短语之外的其他信息。HTTP/1.1 规范中定义了以下几种警告代码。

- 101 响应过时了

当知道一条响应报文已过期时（比如，原始服务器无法进行再验证时），就必须包含这条警告信息。

- 111 再验证失败

如果缓存试图与原始服务器进行响应再验证，但由于缓存无法抵达原始服务器造成了再验证失败，那就必须在发给客户端的响应中包含这条警告信息。

注 11：完整的 `Via` 头部语法参见 HTTP/1.1 规范。

- 112 断开连接操作

通知性警告信息。如果缓存到网络的连接被删除了就应该使用此警告信息。

- 113 试探性过期

如果新鲜性试探过期时间大于 24 小时，而且返回的响应使用期大于 24 小时，缓存中就必须包含这条警告信息。

- 199 杂项警告

收到这条警告的系统不能使用任何自动响应。报文中可能，而且很可能应该包含一个主体，其中携带了为用户提供的额外信息。

- 214 使用了转换

如果中间应用程序执行了任何会改变响应内容编码的转换，就必须由任意一个中间应用程序（比如代理）来添加这条警告。

- 299 持久杂项警告

接收这条警告的系统不能进行任何自动的回应。错误中可能包含一个主体部分，它为用户提供了更多的信息。

类型 响应首部
基本语法 `Warning: 1# warning-value`
举例 `Warning: 113`

530

WWW-Authenticate

`WWW-Authenticate` 首部用于 401 Unauthorized 响应，向客户端发布一个质询认证方案。第 14 章深入讨论了 `WWW-Authenticate` 首部及其在 HTTP 基本质询 / 响应认证系统中的使用方法。

类型 响应首部
基本语法 `WWW-Authenticate: 1# challenge`
举例 `WWW-Authenticate: Basic realm="Your Private Travel Profile"`

X-Cache

X 开头的都是扩展首部。Squid 用 `X-Cache` 首部来通知客户端某个资源是否可用。

类型 扩展响应首部
举例 `X-Cache: HIT`

X-Forwarded-For

很多代理服务器（比如，Squid）会用这个首部来说明某条请求都被转发给了谁。与前面提到的 `Client-ip` 首部类似，这个请求首部说明了请求是从哪个地址发出的。

类型 扩展请求首部
基本语法 `X-Forwarded-For: addr`
举例 `X-Forwarded-For: 64.95.76.161`

X-Pad

这个首部用来解决某些浏览器中与响应首部长度有关的 bug。它在响应报文的首部
531 填充了一些字节，以解决这个 bug。

类型 扩展通用首部
基本语法 `X-Pad: pad-text`
举例 `X-Pad: bogosity`

X-Serial-Number

`X-Serial-Number` 首部是个扩展首部。某些较老的 HTTP 应用程序会用它向 HTTP 报文中插入许可软件的序列号。

它基本上已经没什么用处了，它只是作为 X 开头的首部的一个示例列在这里。

类型 扩展通用首部
基本语法 `X-Serial-Number: serialno`
532 **举例** `X-Serial-Number: 010014056`

附录D

MIME类型

MIME 媒体类型（简称为 MIME 类型）是描述报文实体主体内容的一些标准化名称（比如，text/html、image/jpeg）。本附录说明了 MIME 类型是怎样工作的、如何注册新的名字以及到哪里去查找更多的相关信息。

本附录还包含了 10 张便捷表格，详细描述了从全球众多资源中搜集来的数百种 MIME 类型。这可能是有史以来最详细的 MIME 类型列表。希望这些表格会对你有所帮助。

此附录的主要内容有：

- 在 D.1 节列出主要的参考资料；
- 在 D.2 节解释 MIME 类型的结构；
- 在 D.3 节说明了如何注册 MIME 类型；
- 附加的 10 张表格让你更便捷地查找 MIME 类型。

本附录包含了下列 MIME 类型表：

- application/*——表 D-3
- audio/*——表 D-4
- chemical/*——表 D-5
- image/*——表 D-6
- message/*——表 D-7
- model/*——表 D-8
- multipart/*——表 D-9
- text/*——表 D-10
- video/*——表 D-11
- 其他——表 D-12

533

D.1 背景知识

MIME 类型最初是为多媒体电子邮件而开发的，但目前在 HTTP 和其他几种需要描述数据对象格式和用途的协议中也使用了 MIME 类型。

MIME 主要由下列 5 份文档定义。

- RFC 2045，“MIME: Format of Internet Message Bodies”（“MIME：因特网报文主体的格式”）
  描述了 MIME 报文结构的概况，并介绍了 HTTP 借用的 `Content-Type` 首部。
- RFC 2046，“MIME: Media Types”（“MIME：媒体类型”）

介绍了 MIME 类型及其结构。

- RFC 2047，“MIME: Message Header Extensions for Non-ASCII Text”（“MIME：非 ASCII 文本的报文首部扩展”）
  定义了一些在首部包含非 ASCII 字符的方式。
- RFC 2048，“MIME: Registration Procedures”（“MIME：注册过程”）
  定义了如何向因特网号码分配机构（“Internet Assigned Numbers Authority，IANA”）注册 MIME 值。
- RFC 2049，“MIME: Conformance Criteria and Examples”（“MIME：一致性标准及实例”）
  详细介绍了一致性规则，并提供了一些实例。

根据 HTTP 的目标，我们最感兴趣的文档是 RFC 2046 和 RFC 2048。

D.2 MIME类型结构

每种 MIME 媒体类型都包含类型、子类型和可选参数的列表。类型和子类型由一个斜杠分隔，如果有可选参数的话，则以分号开始。在 HTTP 中，MIME 媒体类型被广泛用于 `Content-Type` 和 `Accept` 首部。下面是几个例子：

```
Content-Type: video/quicktime
Content-Type: text/html; charset="iso-8859-6"
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
Accept: image/gif
```

D.2.1 离散类型

MIME 类型可以直接用于描述对象类型，也可以用于描述其他对象类型的集合或类型包。如果直接用 MIME 类型来描述某个对象类型，它就是一种离散类型（discrete type）。其中包括文本文件、视频和应用程序特有的文件格式。

D.2.2 复合类型

如果 MIME 类型描述的是其他内容的集合或封装包，这种 MIME 类型就被称为复合类型（composite type）。复合类型描述的是封装包的格式。将封装包打开时，其中包含的每个对象都会有其各自的类型。

534

D.2.3 多部分类型

多部分媒体类型是复合类型。多部分对象包含多个组件类型。下面是一个多部分 / 混合内容实例，每个组件都有自己的 MIME 类型：

```
Content-Type: multipart/mixed; boundary=unique-boundary-1

--unique-boundary-1
Content-type: text/plain; charset=US-ASCII

Hi there, I'm some boring ASCII text...

--unique-boundary-1
Content-Type: multipart/parallel; boundary=unique-boundary-2

--unique-boundary-2
Content-Type: audio/basic

    ... 8000 Hz single-channel mu-law-format
        audio data goes here ...

--unique-boundary-2
Content-Type: image/jpeg

    ... image data goes here ...

--unique-boundary-2--

--unique-boundary-1
Content-type: text/enriched

This is <bold><italic>enriched.</italic></bold>
<smaller>as defined in RFC 1896</smaller>

Isn't it <bigger><bigger>cool?</bigger></bigger>

--unique-boundary-1
Content-Type: message/rfc822

From: (mailbox in US-ASCII)
To: (address in US-ASCII)
Subject: (subject in US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

    ... Additional text in ISO-8859-1 goes here ...

--unique-boundary-1--
```


D.2.4 语法

如前所述，MIME 类型由主类型、子类型和可选参数的列表组成。

主类型可以是预定义类型、IETF 定义的扩展标记，或者（以“x-”开头的）实验性标记。表 D-1 列出了一些常见的主类型。

表D-1　常见的主MIME类型

类　型	描　述
application	应用程序特有的内容格式（离散类型）
audio	音频格式（离散类型）
chemical	化学数据集（离散 IETF 扩展类型）
image	图片格式（离散类型）
message	报文格式（复合类型）
model	三维模型格式（离散 IETF 扩展类型）
multipart	多部分对象集合（复合类型）
text	文本格式（离散类型）
video	视频电影格式（离散类型）

子类型可以是主类型（比如，“text/text”）、IANA 注册的子类型，或者是（以“x-”开头的）实验性扩展标记。

类型和子类型都是由 US-ASCII 字符的一个子集构成的。空格和某些保留分组以及标点符号称为“tspecials”，它们是控制字符，不能用于类型和子类型名。

RFC 2046 定义的语法如下所示：

```
TYPE := "application" | "audio" | "image" | "message" | "multipart" |
        "text" | "video" | IETF-TOKEN | X-TOKEN
SUBTYPE := IANA-SUBTOKEN | IETF-TOKEN | X-TOKEN

IETF-TOKEN := <extension token with RFC and registered with IANA>
IANA-SUBTOKEN := <extension token registered with IANA>
X-TOKEN := <"X-" or "x-" prefix, followed by any token>

PARAMETER := TOKEN "=" VALUE
VALUE := TOKEN / QUOTED-STRING
TOKEN := 1*<any (US-ASCII) CHAR except SPACE, CTLs, or TSPECIALS>
TSPECIALS := "(" | ")" | "<" | ">" | "@" |
             "," | ";" | ":" | "\" | <"> |
             "/" | "[" | "]" | "?" | "="
```

536

D.3　在IANA注册MIME类型

RFC 2048 描述了 MIME 媒体类型的注册过程。使用注册过程的目的不仅能简化注册新媒体类型的过程，而且还能提供一些完整性检测，以确保新类型是经过深思熟虑后提出的。

D.3.1 注册树

MIME 类型标记被分成了 4 类，称为“注册树”，每一类都有自己的注册规则。表 D-2 描述了这 4 棵树——IETF、厂商、个人和实验性的树。

表D-2 4种MIME媒体类型注册树

注册树	举例	描述
IETF	text/html（HTML 文本）	IETF 树用于那些对因特网来说具有普遍意义的类型。新的 IETF 树媒体类型要由因特网工程指导组（Internet Engineering Steering Group，IESG） 审批，并且要有一个附属的标准追踪 RFC。 IETF 树类型标记中没有句点（.）
厂商（vnd.）	image/vnd.fpx （柯达的 FlashPix 图片 ）	厂商树用于可商用产品的媒体类型。鼓励大家进行新厂商类型的公开评审，但并不强制这么做。 厂商树类型以 vnd. 开头
个人 / 不重要的（prs.）	image/prs.btif （美国国家银行使用的内部检查管理格式）	私有、个人或不重要的媒体类型可以注册在个人树中。这些媒体类型不应该进行商业化发布。 个人数类型以 prs. 开头
实验性（x- 或 x.）	application/x-tar （Unix 的 tar 档案）	实验树用于未注册或实验性的媒体类型。由于注册新的厂商或个人媒体类型的过程相对简单，所以不应该广泛地发布使用 x- 类型的软件。 实验树类型以 x. 或 x- 开头

D.3.2 注册过程

关于 MIME 媒体类型的注册过程请仔细阅读 RFC 2048。

基本的注册过程并不是官方的标准过程，只是一个管理过程，目的是在尽短时间内，就能通过 IANA 对新类型的完整性检查，将其记录在注册表中。这个过程遵循下列步骤。

（1）将媒体类型提交给 IANA 审阅。

向 ietf-types@iana.org 邮件列表发送一封媒体类型注册建议书，审阅期为两周。发布公告，征求公众对名字、互操作性和安全问题的反馈意见。在注册完成之前，都可以使用 RFC 2045 中指定的前缀 x-。

537

（2）IESG 审批（仅对 IETF 树而言）

如果要将媒体类型注册到 IETF 树中去，就必须提交给 IESG 审批，而且必须有一个附加的标准追踪 RFC。

（3）IANA 注册

只要媒体类型达到了审批要求，作者就可以通过例 D-1 中的电子邮件模板向 IANA 提交注册请求，并将注册信息发送到 ietf-types@iana.org。IANA 会注册媒体类型，在 http://www.isi.edu/in-notes/iana/assignments/media-types/ 上向公众提供该媒体类型应用程序。

D3.3 注册规则

只有在响应 IESG 批准某指定注册申请的通知时，IANA 才能在 IETF 树中注册媒体类型。

只要满足下面这些条件，IANA 就会自动注册厂商和个人类型，不需要进行任何正式的审查。

（1）媒体类型一定要像实际的媒体格式一样工作。像传输编码或字符集那样工作的类型是不能注册为媒体类型的。

（2）所有媒体类型都要有适当的类型和子类型名。所有类型名都要由标准追踪 RFC 定义。所有子类型名都必须是唯一的，必须与那类名称的 MIME 语法相符，而且必须包含恰当的树前缀。

（3）个人树类型必须提供格式规范或指向格式规范的指针。

（4）不要忽略安全问题。所有开发因特网软件的人都要为防范安全漏洞作出贡献。

D.3.4 注册模板

实际的 IANA 注册是通过电子邮件完成的。可以用例 D-1 中显示的模板来完成注册表格，并将其发送到 ietf-types@iana.org。[1]

例 D-1 IANA 的 MIME 注册电子邮件模板

```
To: ietf-types@iana.org
Subject: Registration of MIME media type XXX/YYY

MIME media type name:
```

538

注 1：这个格式没有很严格的结构，稍做调整并不会影响信息阅读，但由机器处理则比较困难。这就是很难找到一份可读的、经过良好组织的 MIME 类型小结的原因之一，这也是我们在本附录末尾创建那些表格的原因。

```
MIME subtype name:

Required parameters:

Optional parameters:

Encoding considerations:

Security considerations:

Interoperability considerations:

Published specification:

Applications which use this media type:

Additional information:

    Magic number(s):
    File extension(s):
    Macintosh File Type Code(s):

Person & email address to contact for further information:

Intended usage:

(One of COMMON, LIMITED USE or OBSOLETE)

Author/Change controller:

(Any other information that the author deems interesting may be added
below this line.)
```

D.3.5 MIME媒体类型注册

可以通过IANA的网站（http://www.iana.org）访问那些已提交的表格。编写本书时，实际存储MIME媒体类型的数据库位于http://www.isi.edu/in-notes/iana/assignments/media-types/的ISI Web服务器上。

媒体类型存储在一棵目录树中，以主类型和子类型进行结构划分，每种媒体类型都有一个叶子文件。每个文件中都包含了电子邮件提交信息。但是，每个人所完成的注册模板都略有不同，因此，不同人提交的信息质量和格式都有所不同。（在本附录的表格中，我们试着填补了注册用户忽略的部分。）

D.4 MIME类型表

本节在10张表格中对数百种MIME类型进行了总结。每张表格都列出了一个特定主类型（image、text等）中的MIME媒体类型。

539

这些信息是从很多地方搜集来的，包括 IANA 的媒体类型注册登记处、Apache 的 mime.types 文件，以及各种各样的因特网 Web 页面。我们花了好多天来提炼数据，填补漏洞，并在其中包含了交叉参考文献中的描述性总结，使数据的可用性更高。

这可能是有史以来最详细的 MIME 类型清单列表了。希望对你有所帮助！

D.4.1 application/*

表 D-3 描述了很多应用程序特有的 MIME 媒体类型。

表D-3 Application MIME类型

MIME类型	描　述	扩　展	联系方式与参考文献
application/activemessage	支持主动邮件群件系统		*Readings in Groupware and Computer-Supported CooperativeWork*, “Active Mail: A Framework for Integrated Groupware Applications”, Ronald M. Baecker, Morgan Kaufmann 出版社, ISBN1558602410
application/andrew-inset	支持用安德鲁工具集创建多媒体内容	ez	*Multimedia Applications Development with the Andrew Toolkit*,Nathaniel S. Borenstein, Prentice Hall 出版社, AS-IN 0130366331 nsb@bellcore.com
application/applefile	在允许对非特定用户数据进行一般性访问的同时，允许对带有 Apple/Macintosh 特有信息的数据进行基于 MIME 的传输		RFC 1740
application/atomicmail	ATOMICMAIL 是贝尔通信研究所的一个实验型研究项目，设计用来在电子邮件报文中包含一些在阅读邮件时会执行的程序。对 safe-tcl 的支持会很快使 ATOMICMAIL 过时		“ATOMICMAIL Language Reference Manual”, Nathaniel S. Borenstein, 贝尔科技备忘录, TM ARH-018429
application/batch-SMTP	定义了一个 MIME 内容类型，可以用隧道方式通过任意能够进行 MIME 传输的路径实现 ESMTP 邮件事务		RFC 2442

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/beep+xml	支持名为BEEP的交互协议。BEEP支持在对等实体之间对MIME报文进行同步且独立的交互，这些报文通常都是XML结构的文本		RFC 3080
application/cals-1840	支持对美国国防部数字数据进行MIME电子邮件交换，早期这些数据是MILSTD-1840定义的由磁带实现数据交换的		RFC1895
application/commonground	Common Ground是个电子文档交换及发布程序，允许用户创建一些文档，这些文档无需在系统中创建任何应用程序或字体，所有人都可以查看、搜索和打印		Nick Gault No Hands软件 ngault@nohands.com
application/cybercash	支持通过CyberCash协议进行信用卡支付。用户开始付费时，商家会向客户发送一条报文，作为MIME类型application/cybercash的报文主体使用		RFC 1898
application/dca-rft	IBM文档内容结构		“IBM Document Content Architec-ture/Revisable Form Text Reference”，文档号SC23-0758-1，国际商用机器公司（International Business Machines，IBM）
application/dec-dx	DEC的文档传输格式		“Digital Document Transmission (DX) Technical Notebook”，文档号EJ29141-86，数字设备公司（DEC，Digital Equipment Corporation）
application/dvcs	支持DVCS（Data Validation and Certification Server，数据验证及证书服务器）所使用的协议，该服务器在公共密钥安全架构中作为第三方受信站点使用的		RFC 3029
application/EDI-Consent	支持通过EDI（electronic data interchange，电子数据交换）用非标准的规范进行双边贸易		http://www.isi.edu/in-notes/iana/assignments/media-types/application/EDI-Consent

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/EDI-X12	支持通过EDI用ASC X12 EDI规范进行双边贸易		http://www.isi.edu/in-notes/iana/assignments/media-types/application/EDI-X12
application/EDIFACT	支持通过EDI用EDIFACT规范进行双边贸易		http://www.isi.edu/in-notes/iana/assignments/media-types/application/EDIFACT
application/eshop	未知		Steve Katz 系统结构商店 steve_katz@eshop.com
application/font-tdpfr	定义了包含一组字形的PFR（Portable Font Resource，便携式字体资源），每个字形都与一个字符码相关		RFC 3073
application/http	用于封装由一条或多条HTTP请求或响应报文（不混合）构成的管道		RFC 2616
application/hyperstudio	支持HyperStudio教学超媒体文件的传输	stk	http://www.hyperstudio.com
application/iges	CAD模型交换的常用格式		“ANS/US PRO/IPO-100”U.S. Product Data Association2722 Merrilee Drive, Suite 200 Fairfax, VA 22031-4499
application/index application/index.cmd application/index.obj application/index.response application/index.vnd	支持CIP（Common Indexing Protocol，公共索引协议）。CIP是对Whois++目录服务的发展，用于在服务器间传送索引信息，以便通过分布式数据库系统来重定向或复制请求		RFC 2652、RFC 2651、RFC1913和RFC1914
application/iotp	支持在HTTP上发送IOTP（Internet Open Trading Protocol，因特网开放贸易协议）报文		RFC 2935

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/ipp	支持在 HTTP 上使用 IPP（Internet Printing Protocol，因特网打印协议）		RFC 2910
application/mac-binhex40	将 8 位字节的字符串编码为 7 位字节的字符串，这样对某些应用程序来说更安全一些（没有 6 位的 Base-64 编码安全）	hqx	RFC 1341
application/mac-compactpro	来自 Apache 的 mime.types	cpt	
application/macwriteii	Claris MacWrite II		
application/marc	MARC 对象是机读编目记录——书目及相关信息的表示和通信标准	mrc	RFC 2220
application/mathematica application/mathematica-old	支持 Mathematica 和 MathReader 数值分析软件	nb、ma、mb	*The Mathematica Book*，Stephen Wolfram，剑桥大学出版社，ISBN 0521643147
application/msword	微软的 Word MIME 类型	doc	
application/news-message-id			RFC 822（报文 ID）、RFC1036（新闻应用程序）以及 RFC977（NNTP）
application/news-transmission	允许通过电子邮件或其他传输方式传送新闻文章		RFC 1036
application/ocsp-request	支持 OCSP（Online Certificate Status Protocol，在线证书状态协议），此协议提供了一种无需本地证书撤销列表即可查看数字证书有效性的方法	orq	RFC 2560
application/ocsp-response	同上	ors	RFC 2560
application/octet-stream	未分类的二进制数据	bin、dms、lha、lzh、exe、class	RFC 1341

542

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/oda	根 据 ODA（Office Document Architecture，办公文档结构）标准， 用 ODIF（Office Document Interchange Format，办公文档交互格式）表示法对信息进行编码。Content-Type 行也应该像： Content-Type:application/oda;profile=Q112 这样指定用来说明 DAP（document application profile，文档应用外观）的属性 / 值对	oda	RFC 1341 ISO 8613；“Information Processing: Text and Office System; Office Document Architecture (ODA) and Interchange Format (ODIF)”，第 1-8 部分，1989 年
application/parityfec	RTP 数据流的前向纠错奇偶码		RFC 3009
application/pdf	Adobe PDF 文件	pdf	参见 *Portable Document Format Reference Manual*，Adobe System, Inc. Addison Wesley，ISBN0201626284
application/pgp-encrypted	PGP 加密数据		RFC 2015
application/pgp-keys	PGP 公共密钥块		RFC 2015
application/pgp-signature	PGP 加密签名		RFC 2015
application/pkcs10	公共密钥加密系统 #10——传输 PKCS #10 证书请求时，主体类型必须用 application/pkcs10	p10	RFC 2311
application/pkcs7-mime	公共密钥加密系统 #7——这种类型用于传送包括 envelopedData 和 signedData 在内的几种类型的 PKCS #7 对象	p7m	RFC 2311
application/pkcs7-signature	公共密钥加密系统 #7——这种类型总是包含一个 signedData 类型的 PKCS #7 对象	p7s	RFC 2311
application/pkix-cert	传输 X.509 证书	cer	RFC 2585
application/pkix-crl	传输 X.509 证书撤销列表	crl	RFC 2585

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/pkixcmp	X.509 公共密钥基础设施证书管理协议所使用的报文格式	pki	RFC 2510
application/postscript	Adobe PostScript 图像文件（程序）	ai、ps、eps	RFC 2046
application/prs.alvestrand.titrax-sheet	Harald T. Alvestrand 的“TimeTracker”程序		http://domen.uninett.no/~hta/titrax/
application/prs.cww	Windows 下的 CU-Writer	cw、cww	Somchai Prasitjutrakul 博士 somchaip@chulkn.car.chula.ac.th
application/prs.nprend	未知	rnd、rct	John M. Doggett jdoggett@tiac.net
application/remote-printing	包含了远程打印时用于打印机封面的元信息		RFC 1486 Marshall T. Rose mrose@dbc.mtview.ca.us
application/riscos	Acorn 的 RISC OS 二进制文件		RISC OSProgrammer’s Reference Manuals，AcornComputers,Ltd.，ISBN1852501103
application/sdp	SDP 用于描述各种多媒体会话，目的是进行会话声明、会话邀请以及其他形式的多媒体会话初始化		RFC 2327 Henning Schulzrinne hgs@cs.columbia.edu
application/set-payment application/set-payment-initiation application/set-registration application/set-registration-initiation	支持 SET 安全电子交易支付协议		http://www.visa.com http://www.mastercard.com
application/sgml-open-catalog	用于支持 SGML Open TR9401: 1995“实体管理”规范的系统		SGML Open 910 Beaver Grade Road，#3008 Coraopolis，PA 15109 info@sgmlopen.org
application/sieve	Sieve 邮件过滤脚本		RFC 3028

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/slate	BBN/Slate 文档格式是作为 BBN/Slate 产品标准文档集的一部分发布的		BBN/Slate Product Mgr BBN Systems and Technologies 10 Moulton Street Cambridge, MA 02138
application/smil	SMIL（同步多媒体集成语言，Synchronized Multimedia Integration Language）将一组独立的多媒体对象整合成了一种同步多媒体表示形式	smi、smil	http://www.w3.org/AudioVideo/
application/tve-trigger	支持在增强型电视接收器里嵌入 URL		“SMPTE: Declarative Data Essence, Content Level 1”，由电影电视工程师协会生成。 http://www.smpte.org
application/vemmi	增强型 videotex 标准		RFC 2122
application/vnd.3M.Post-it-Notes	由 Post-it® 因特网设计者须知因特网控制 / 插件使用	pwn	http://www.3M.com/psnotes/
application/vnd.accpac.simply.aso	Simply Accounting 软件的 7.0 及以上版本。这种类型的文件与 Open Financial Exchange 版本 1.02 的规范一致	aso	http://www.ofx.net
application/vnd.accpac.simply.imp	Simply Accounting 软件的 7.0 及以上版本使用，用于输入它自己的数据	imp	http://www.ofx.net
application/vnd.acucobol	ACUCOBOL-GT Runtime		Dovid Lubin dovid@acucobol.com
application/vnd.aether.imp	支持在 AOL 即时通信、Yahoo! Messenger 或 MSN Messenger 这样的即时消息服务，和运行在无线设备上的特殊即时通信客户端软件集之间进行高效的即时消息通信		有许可证就可以从 Aether 系统中获得 IMP（Wireless Instant Messaging Protocol，无线即时通信协议）规范
application/vnd.anser-Web-certificate-issue-initiation	Web 浏览器装载 ANSER-WEB-Terminal Client 程序的触发器	cii	Hiroyoshi Mori mori@mm.rd.nttdata.co.jp

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.anser-Web-funds-transfer-initiation	同上	fti	同上
application/vnd.audiograph	AudioGraph	aep	Horia Cristian H.C.Slusanschi@massey.ac.nz
application/vnd.bmi	CADAM 系统的 BMI 图像格式	bmi	Tadashi Gotoh tgotoh@cadamsystems.co.jp
application/vnd.businessobjects	BusinessObjects4.0 及以上版本	rep	
application/vnd.canon-cpdl application/vnd.canon-lips	支持佳能公司的办公图像产品		Shin Muto shinmuto@pure.cpdc.canon.co.jp
application/vnd.claymore	Claymore.exe	cla	Ray Simpson ray@cnation.com
application/vnd.commerce-battelle	支持一种通用的智能卡卡载信息定界机制，可用于数字商务、身份识别、认证以及基于智能卡的持证人信息交互	ica、icf、icd、icc、ic0、ic1、ic2、ic3、ic4、ic5、ic6、ic7、ic8	David C. Applebaum applebau@131.167.52.15
application/vnd.commonspace	允许通过基于 MIME 的进程正确传输 CommonSpace ™文档。CommonSpace 是由 HoughtonMifflin 公司的 Sixth Floor Media 发布的	csp、cst	Ravinder Chandhok chandhok@within.com
application/vnd.contact.cmsg	用于 CONTACT 软件的 CIM DATABASE	cdbcmsg	Frank Patz fp@contact.de http://www.contact.de
application/vnd.cosmocaller	允许从 Web 站点下载包含了连接参数的文件，调用 CosmoCaller 应用程序对参数进行解释，并初始化与 CosmoCallACD 服务器的连接	cmc	Steve Dellutri sdellutri@cosmocom.com
application/vnd.ctc-posml	Continuum 技术公司的 PosML	pml	Bayard Kohlhepp bayardk@ctcexchange.com

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.cups-postscript application/vnd.cups-raster application/vnd.cups-raw	支持 UNIX 通用打印系统的服务器和客户端		http://www.cups.org
application/vnd.cybank	Cybank 数据特有的数据类型		Nor Helmee B. Abd. Halim helmee@cybank.net http://www.cybank.net
application/vnd.dna	DNA 的目标是便捷地将所有 32 位的 Windows 应用程序都转换成 Web 程序	dna	Meredith Searcy msearcy@newmoon.com
application/vnd.dpgraph	由 DpGraph 2000 和 MathWare Cyclone 使用	dpg、mwc dpgraph	David Parker http://www.davidparker.com
application/vnd.dxr	PSI 技术公司的 Digital Xpress Reports	dxr	Michael Duffy miked@psiaustin.com
application/vnd.ecdis-update	支持 ECDIS 应用程序		http://www.sevencs.com
application/vnd.ecowin.chart application/vnd.ecowin.filerequest application/vnd.ecowin.fileupdate application/vnd.ecowin.series application/vnd.ecowin.seriesrequest application/vnd.ecowin.seriesupdate	EcoWin	mag	Thomas Olsson thomas@vinga.se
application/vnd.enliven	支持 Enliven 交互式多媒体的传输	nml	Paul Santinelli psantinelli@narrative.com
application/vnd.epson.esf	精工爱普生 QUASS 流播放器的特有内容	esf	Shoji Hoshina Hoshina.Shoji@exc.epson.co.jp

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.epson.msf	精工爱普生 QUASS 流播放器的特有内容	msf	同上
application/vnd.epson.quickanime	精工爱普生 QuickAnime 播放器的特有内容	qam	Yu Gu guyu@rd.oda.epson.co.jp
application/vnd.epson.salt	精工爱普生 SimpleAnimeLite 播放器的特有内容	slt	Yasuhito Nagatomo naga@rd.oda.epson.co.jp
application/vnd.epson.ssf	精工爱普生 QUASS 流播放器的特有内容	ssf	Shoji Hoshina Hoshina.Shoji@exc.epson.co.jp
application/vnd.ericsson.quickcall	电话倍增器快速呼叫	qcall、qca	Paul Tidwell paul.tidwell@ericsson.com http://www.ericsson.com
application/vnd.eudora.data	Eudora 4.3 及以上版本		Pete Resnick presnick@qualcomm.com
application/vnd.fdf	Adobe 表单数据格式		Forms Data Format（表单数据格式），Technical Note 5173，Adobe Systems
application/vnd.ffsns	用于与 FirstFloor 的 Smart Delivery 进行通信的应用程序		Mary Holstege holstege@firstfloor.com
application/vnd.FloGraphIt	NpGraphIt	gph	
application/vnd.framemaker	Adobe FrameMaker 文件	fm、mif、book	http://www.adobe.com
application/vnd.fsc.Weblaunch	支持 Friendly 软件公司的高尔夫模拟软件	fsc	Derek Smith derek@friendlysoftware.com
application/vnd.fujitsu.oasys application/vnd.fujitsu.oasys2	支持富士通的 OASYS 软件	oas	Nobukazu Togashi togashi@ai.cs.fujitsu.co.jp
application/vnd.fujitsu.oasys2	支持富士通的 OASYS V2 软件	oa2	同上
application/vnd.fujitsu.oasys3	支持富士通的 OASYS V5 软件	oa3	Seiji Okudaira okudaira@candy.paso.fujitsu.co.jp
application/vnd.fujitsu.oasysgp	支持富士通的 OASYS GraphPro 软件	fg5	Masahiko Sugimoto sugimoto@sz.sel.fujitsu.co.jp

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.fujitsu.oasysprs	支持富士通的 OASYS 展示软件	bh2	Masumi Ogita ogita@oa.tfl.fujitsu.co.jp
application/vnd.fujixerox.ddd	支持富士施乐的 EDMICS 2000 和 DocuFile	ddd	Masanori Onda Masanori.Onda@fujixerox.co.jp
application/vnd.fujixerox.docuworks	支持富士施乐的 DocuWorks Desk 及 DocuWorks Viewer 软件	xdw	Yasuo Taguchi yasuo.taguchi@fujixerox.co.jp
application/vnd.fujixerox.docuworks.binder	支持富士施乐的 DocuWorks Desk 及 DocuWorks Viewer 软件	xbd	同上
application/vnd.fut-misnet	未知		Jaan Pruulmann jaan@fut.ee
application/vnd.grafeq	让 GrafEq 的用户通过 Web 和电子邮件交换 GrafEq 文档	gqf、gqs	http://www.peda.com
application/vnd.groove-account	Groove 是为小群组交互实现了虚拟空间的对等实体通信系统	gac	Todd Joseph todd_joseph@groove.net
application/vnd.groove-identity-message	同上	gim	同上
application/vnd.groove-injector	同上	grv	同上
application/vnd.groove-tool-message	同上	gtm	同上
application/vnd.groove-tool-template	同上	tpl	同上
application/vnd.groove-vcard	同上	vcg	同上
application/vnd.hhe.lesson-player	支持 LessonPlayer 和 Presentation-Editor 软件	les	Randy Jones Harcourt E-Learning randy_jones@archipelago.com
application/vnd.hp-HPGL	HPGL 文件		*TheHP-GL/2 and HP RTL Referemce Guide*，Addison Wesley，ISBN 0201310147

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.hp-hpid	支持惠普的即时传输软件	hpi、hpid	http://www.instant-delivery.com
application/vnd.hp-hps	支持惠普的 WebPrintSmart 软件	hps	http://www.hp.com/go/Webprintsmart_mimetype_specs/
application/vnd.hp-PCL application/vnd.hp-PCLXL	PCL 打印机文件	pcl	“PCL-PJL Technical Reference Manual Documentation Package”，HP Part No. 5012-0330
application/vnd.httphone	IP 系统上的 HTTPhone 异步语音		Franck LeFevre franck@k1info.com
application/vnd.hzn-3d-crossword	用于对 Horizon，A Glimpse of Tomorrow 公司的字谜游戏进行加密	x3d	James Minnis james_minnis@glimpse-of-tomorrow.com
application/vnd.ibm.afplinedata	PSF（Print Services Facility，打印服务设施），ACIF（AFP Conversion and Indexing Facility，AFP 转换与索引设施）		Roger Buis buis@us.ibm.com
application/vnd.ibm.MiniPay	MiniPay 认证和支付软件	mpy	Amir Herzberg amirh@vnet.ibm.com
application/vnd.ibm.modcap	Mixed Object Document Content	list3820、listafp、afp、pseg3820	Reinhard Hohensee rhohensee@vnet.ibm.com “MixedObject Document Content Architecture Reference”，IBM publication，SC31-6802
application/vnd.informix-visionary	Informix Visionary	vis	Christopher Gales christopher.gales@informix.com
application/vnd.intercon.formnet	支持 Intercon Associates FormNet 软件	xpw、xpx	Thomas A. Gurak assoc@intercon.roc.servtech.com
application/vnd.intertrust.digibox application/vnd.intertrust.nncp	支持用于安全电子商务和数字权限管理的 InterTrust 结构		InterTrust Technologies 460 Oakmead Parkway Sunnyvale, CA 94086 USA info@intertrust.com http://www.intertrust.com

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.intu.qbo	仅用于 QuickBooks 6.0（加拿大）	qbo	Greg Scratchley greg_scratchley@intuit.com Open Financial Exchange 规范中讨论的文件的格式，可从 http://www.ofx.net 上获得
application/vnd.intu.qfx	仅用于 Quicken 99 及后继版本	qfx	同上
application/vnd.is-xpr	搜信公司的 Express	xpr	Satish Natarajan satish@infoseek.com
application/vnd.japannet-directory-service application/vnd.japannet-jpnstore-wakeup application/vnd.japannet-payment-wakeup application/vnd.japannet-registration application/vnd.japannet-registration-wakeup application/vnd.japannet-setstore-wakeup application/vnd.japannet-verification application/vnd.japannet-verification-wakeup	支持 三菱电机公司的 JapanNet 安全、认证和支付软件		Jun Yoshitake yositake@iss.isl.melco.co.jp
application/vnd.koan	在 SSEYO Koan Netscape Plugin 这样的辅助程序的帮助下，在因特网上自动重放 Koan 音乐文件	skp、skd skm、skt	Peter Cole pcole@sseyod.demon.co.uk
application/vnd.lotus-1-2-3	Lotus-1-2- 和 Lotus Approach	123、wk1 wk3、wk4	Paul Wattenberger Paul_Wattenberger@lotus.com

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.lotus-approach	Lotus Approach	apr、vew	同上
application/vnd.lotus-freelance	Lotus Freelance	prz、pre	同上
application/vnd.lotus-notes	Lotus Notes	nsf、ntf、ndl、ns4 ns3、ns2 nsh、nsg	Michael Laramie laramiem@btv.ibm.com
application/vnd.lotus-organizer	Lotus Organizer	or3、or2、org	Paul Wattenberger Paul_Wattenberger@lotus.com
application/vnd.lotus-screencam	Lotus ScreenCam	scm	同上
application/vnd.lotus-wordpro	Lotus WordPro	lwp、sam	同上
application/vnd.mcd	Micro CADAM 公司的 CAD 软件	mcd	Tadashi Gotoh tgotoh@cadamsystems.co.jp http://www.cadamsystems.co.jp
application/vnd.mediastation.cdkey	支持 Media Station 公司的 CDKey 远程 CDROM 通信协议	cdkey	Henry Flurry henryf@mediastation.com
application/vnd.meridian-slingshot	Meridian 数据公司的 Slingshot		Eric Wedel Meridian Data, Inc. 5615 Scotts Valley Drive Scotts Valley, CA 95066 ewedel@meridian-data.com
application/vnd.mif	FrameMaker 交换格式	mif	ftp://ftp.frame.com/pub/techsup/techinfo/dos/mif4.zip Mike Wexler Adobe Systems, Inc333 W. San Carlos St. San Jose, CA 95110 USA mwexler@adobe.com
application/vnd.minisoft-hp3000-save	NetMail 3000 保存格式		Minisoft，Inc support@minisoft.com ftp://ftp.3k.com/DOC/ms92-save-format.txt

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.mitsubishi.misty-guard.trustWeb	支持三菱电机公司的 TrustWeb 软件		Manabu Tanaka mtana@iss.isl.melco.co.jp
application/vnd.Mobius.DAF	支持 Mobius Managenect System 软件	daf	Celso Rodriguez crodrigu@mobius.com Greg Chrzczon gchrzczo@mobius.com
application/vnd.Mobius.DIS	同上	dis	同上
application/vnd.Mobius.MBK	同上	mbk	同上
application/vnd.Mobius.MQY	同上	mqy	同上
application/vnd.Mobius.MSL	同上	msl	同上
application/vnd.Mobius.PLC	同上	plc	同上
application/vnd.Mobius.TXF	同上	txf	同上
application/vnd.motorola.flexsuite	FLEXsuite ™是无线报文协议的集合。无线报文服务提供商的网关、无线 OS 和应用程序都使用此类型		Mark Patton Motorola Personal Networks Group fmp014@email.mot.com 在适当许可协议的支持下，可以从摩托罗拉获得 FLEXsuite ™规范
application/vnd.motorola.flexsuite.adsi	FLEXsuite ™是无线报文协议的集合。这种类型为各种数据加密解决方案提供了一种无线友好格式		同上
application/vnd.motorola.flexsuite.fis	FLEXsuite ™是无线报文协议的集合。这种类型是向无线设备高效传输结构化信息（比如新闻、股票、天气）的无线友好格式		同上
application/vnd.motorola.flexsuite.gotap	FLEXsuite ™是无线报文协议的集合 这种类型为通过空中报文进行无线设备属性编程提供了一种通用的无线友好格式		同上

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.motorola.flexsuite.kmr	FLEXsuite ™是无线报文协议的集合。这种类型为加密密钥管理提供了一种无线友好格式		同上
application/vnd.motorola.flexsuite.ttc	FLEXsuite ™是无线报文协议的集合。这种类型支持在通过标记文本压缩进行高效文本传输时使用的一种无线友好格式		同上
application/vnd.motorola.flexsuite.wem	FLEXsuite ™是无线报文协议的集合。这种类型为向无线设备传输因特网电子邮件提供了无线友好格式		同上
application/vnd.mozilla.xul+xml	支持 Mozilla 的因特网应用程序集	xul	Dan Rosen2 dr@netscape.com
application/vnd.ms-artgalry	支持微软的 Art Gallery	cil	deansl@microsoft.com
application/vnd.ms-asf	ASF 是一种多媒体文件格式，要通过网络以流的形式传输其内容以支持分布式多媒体应用程序。ASF 内容中可包含任意媒体类型的任意组合（比如音频、视频、图片、URL、HTML 内容、MIDI、二维和三维模式、脚本以及各种类型的对象）	asf	Eric Fleischman ericf@microsoft.com http://www.microsoft.com/mind/0997/netshow/netshow.asp
application/vnd.ms-excel	微软的 Excel 电子表格	xls	Sukvinder S. Gill sukvg@microsoft.com
application/vnd.ms-lrm	微软的特有格式	lrm	Eric Ledoux ericle@microsoft.com
application/vnd.ms-powerpoint	微软的 PowerPoint 演示文稿	ppt	Sukvinder S. Gill sukvg@microsoft.com
application/vnd.ms-project	微软的项目文件	mpp	同上
application/vnd.ms-tnef	标识通常只能由 MAPI 应用程序处理的附件。这种类型是 Rich Text 本和 Icon 信息等富 MAPI 属性的封装格式，如果不把这些信息封装起来，就会在报文传输中降级		同上

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.ms-works	微软的 Works 软件		同上
application/vnd.mseq	MSEQ 是一种适用于无线设备的压缩多媒体格式	mseq	Gwenael Le Bodic Gwenael.le_bodic@alcatel.fr http://www.3gpp.org
application/vnd.msign	由那些实现了 msign 协议的应用程序使用，要求移动设备提供签名		Malte Borcherding Malte.Borcherding@brokat.com
application/vnd.music-niff	NIFF 音乐文件		Cindy Grande 72723.1272@compuserve.com ftp://blackbox.cartah.washington.edu/pub/NIFF/NIFF6A.TXT
application/vnd.musician	RenaiScience 公司设计并开发的 MUSICIAN 乐谱语言 / 编码系统	mus	Robert G. Adams gadams@renaiscience.com
application/vnd.netfpx	用于多分辨率图片信息的动态检索，就像惠普公司的 Imaging for Internet 软件一样	fpx	Andy Mutz andy_mutz@hp.com
application/vnd.noblenet-directory	支持 RogueWave 购买的 NobleNet Directory 软件	nnd	http://www.noblenet.com
application/vnd.noblenet-sealer	支持 RogueWave 购买的 NobleNet Sealer 软件	nns	http://www.noblenet.com
application/vnd.noblenet-Web	支持 RogueWave 购买的 NobleNet Web 软件	nnw	http://www.noblenet.com
application/vnd.novadigm.EDM	支持 Novadigm 的 RADIA 和 EDM 产品	edm	Phil Burgard pburgard@novadigm.com
application/vnd.novadigm.EDX	同上	edx	同上
application/vnd.novadigm.EXT	同上	ext	同上
application/vnd.osa.netdeploy	支持 Open Software Associates 的 netDeploy 应用程序部署软件	ndc	Steve Klos stevek@osa.com http://www.osa.com
application/vnd.palm	PalmOS 系统软件和应用程序使用——这种新的类型 application/vnd.palm，取代了老的类型 application/x-pilot	prc、pdb、pqa、oprc	Gavin Peacock gpeacock@palm.com

552

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.pg.format	宝洁公司特有的标准报告系统	str	April Gandert TN152 Procter & Gamble Way Cincinnati, Ohio 45202 (513) 983-4249
application/vnd.pg.osasli	宝洁公司特有的标准报告系统	ei6	同上
application/vnd.powerbuilder6 application/vnd.powerbuilder6-s application/vnd.powerbuilder7 application/vnd.powerbuilder7-s application/vnd.powerbuilder75 application/vnd.powerbuilder75-s	仅在赛贝斯公司的 PowerBuilder 发行版本 6、7 和 7.5 运行时环境中使用，包括不安全和安全两种版本	pbd	Reed Shilts reed.shilts@sybase.com
application/vnd.previewsystems.box	Preview Systems 公司的 ZipLock/VBox 产品	box、vbox	Roman Smolgovsky romans@previewsystems.com http://www.previewsystems.com
application/vnd.publishare-delta-tree	在 Capella 计算机的 PubliShare 运行时环境中使用	qps	Oren Ben-Kiki publishare-delta-tree@capella.co.il
application/vnd.rapid	Emultek 的快速封装应用程序	zrp	Itay Szekely etay@emultek.co.il
application/vnd.s3sms	将 Sonera SmartTrust 产品的传输机制整合进因特网架构		Lauri Tarkkala Lauri.Tarkkala@sonera.com http://www.smarttrust.com
application/vnd.seemail	支持对 SeeMail 文件的传输。SeeMail 是一种应用程序，它可捕获视频和音频，并通过按位压缩方式将这两个部分压缩存档到一个文件	see	Steven Webb steve@wynde.com http://www.realmediainc.com
application/vnd.shana.informed.formdata	Shana 电子表格数据格式	ifm	Guy Selzler Shana Corporation gselzler@shana.com
application/vnd.shana.informed.formtemp	Shana 电子表格数据格式	itp	同上

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.shana.informed.interchange	Shana 电子表格数据格式	iif、iifl	同上
application/vnd.shana.informed.package	Shana 电子表格数据格式	ipk、ipkg	同上
application/vnd.street-stream	Street 技术公司特有的格式		Glenn Levitt Street Technologies streetd1@ix.netcom.com
application/vnd.svd	Dateware 电子公司的 SVD 文件		Scott Becker dataware@compumedia.com
application/vnd.swiftview-ics	支持 SwiftView®		Randy Prakken tech@ndg.com http://www.ndg.com/svm.htm
application/vnd.triscape.mxs	支持 Triscape Map 浏览器	mxs	Steven Simonoff scs@triscape.com
application/vnd.trueapp	True BASIC 文件	tra	J. Scott Hepler scott@truebasic.com
application/vnd.truedoc	Bitstream 公司特有的格式		Brad Chase brad_chase@bitstream.com
application/vnd.ufdl	UWI 的 UFDL 文件	ufdl、ufd、frm	Dave Manning dmanning@uwi.com http://www.uwi.com/
application/vnd.uplanet.alert application/vnd.uplanet.alert-wbxml application/vnd.uplanet.bearer-choi-wbxml application/vnd.uplanet.bearer-choice application/vnd.uplanet.cacheop application/vnd.uplanet.cacheop-wbxml	Unwired Planet（现在的 Openwave）公司为移动设备提供的微浏览器 UP 所使用的格式		iana-registrar@uplanet.com http://www.openwave.com

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.uplanet.channel application/vnd.uplanet.channel-wbxml application/vnd.uplanet.list application/vnd.uplanet.list-wbxml application/vnd.uplanet.listcmd application/vnd.uplanet.listcmd-wbxml application/vnd.uplanet.signal			
application/vnd.vcx	VirtualCatalog	vcx	Taisuke Sugimoto sugimototi@noanet.nttdata.co.jp
application/vnd.vectorworks	VectorWorks 的图像文件	mcd	Paul C. Pharr pharr@diehlgraphsoft.com
application/vnd.vidsoft.vidconference	VidConference 格式	vsc	Robert Hess hess@vidsoft.de
application/vnd.visio	Visio 文件	vsd、vst、vsw、vss	Troy Sandal troys@visio.com
application/vnd.vividence.scriptfile	Vividence 文件	vsf、vtd、vd	Mark Risher markr@vividence.com
application/vnd.wap.sic	WAP Service Indication 格式	sic、wbxml	WAP Forum Ltd http://www.wapforum.org
application/vnd.wap.slc	WAP Service Loading 格式。与 Service Loading 规范相符的所有内容都可以在 http://www.wapforum.org 上找到	slc、wbxml	同上
application/vnd.wap.wbxml	无线设备的 WAP WBXML 二进制 XML 格式	wbxml	同上 “WAP BinaryXMLContent Format——WBXML vevsion1.1”

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/vnd.wap.wmlc	无线设备的 WAP WML 格式	wmlc、wbxml	同上
application/vnd.wap.wmlscriptc	WAP WMLScript 格式	wmlsc	同上
application/vnd.Webturbo	WebTurbo 格式	wtb	Yaser Rehem Sapient Corporation yrehem@sapient.com
application/vnd.wrq-hp3000-labelled	支持 HP3000 格式		support@wrq.com support@3k.com
application/vnd.wt.stf	支持 Worldtalk 软件	stf	Bill Wohler wohler@worldtalk.com
application/vnd.xara	CorelXARA 保存的 Xara 文件，这是 Xara 有限公司编写（并由 Corel 销售的）的面向对象的矢量图像包	xar	David Matthewman david@xara.com http://www.xara.com
application/vnd.xfdl	UWI 的 XFDL 文件	xfdl、xfd、frm	Dave Manning dmanning@uwi.com http://www.uwi.com
application/vnd.yellowriver-custom-menu	支持 Yellow River CustomMenu 插件，这个插件提供了自定义的浏览器下拉菜单	cmp	yellowriversw@yahoo.com
application/whoispp-query	在 MIME 中定义了 Whois++ 协议的查询方式		RFC 2957
application/whoispp-response	在 MIME 中定义了 Whois++ 协议的响应方式		RFC 2958
application/wita	Wang（王安）信息传输结构		文档号 715-0050A，Wang Laboratories（王安实验室） campbell@redsox.bsw.com
application/wordperfect5.1	WordPerfect 文档		
application/x400-bp	承载了所有没有注册 IANA 映射的 X.400 主体部分		RFC 1494
application/x-bcpio	老版的二进制 CPIO 归档格式	bcpio	

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/x-cdlink	允许在 Web 页面内整合 CD-ROM 媒体	vcd	http://www.cdlink.com
application/x-chess-pgm	来自 Apache mime.types	pgn	
application/x-compress	Unix 压缩的二进制数据	z	
application/x-cpio	CPIO 归档文件	cpio	
application/x-csh	CSH 脚本	csh	
application/x-director	Macromedia 的 Director 档案	dcr、dir、dxr	
application/x-dvi	TeX DVI 文件	dvi	
application/x-futuresplash	来自 Apache 的 mime.types	spl	
application/x-gtar	GNU 的 tar 归档文件	gtar	
application/x-gzip	GZIP 压缩数据	gz	
application/x-hdf	来自 Apache 的 mime.types	hdf	
application/x-javascript	JavaScript 文件	js	
application/x-koan	在 SSEYO Koan Netscape Plugin 辅助应用程序的帮助下，支持在因特网上自动重放 Koan 音乐文件	skp、skd skt、skm	
application/x-latex	LaTeX 文件	latex	
application/x-netcdf	NETCDF 文件	nc、cdf	
application/x-sh	SH 脚本	sh	
application/x-shar	SHAR 归档文件	shar	

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
application/x-shockware-flash	Macromedia 的 Flash 文件	swf	
application/x-stuffit	Stufflt 归档文件	sit	
application/x-sv4cpio	Unix SysV R4 CPIO 归档文件	sv4cpio	
application/x-sv4crc	Unix SysV R4 CPIO 带 CRC 校验的归档文件	sv4crc	
application/x-tar	TAR 归档文件	tar	
application/x-tcl	TCL 脚本	tcl	
application/x-tex	TeX 文件	tex	
application/x-texinfo	TeX 信息文件	texinfo、texi	
application/x-troff	TROFF 文件	t、tr、roff	
application/x-troff-man	TROFF Unix 帮助手册	man	
application/x-troff-me	TROFF+me 文件	me	
application/x-troff-ms	TROFF+ms 文件	ms	
application/x-ustar	扩展 tar 交互格式	ustar	参见 IEEE 1003.1(1990) 规范
application/x-wais-source	WAIS 源文件结构	Src	
application/xml	可扩展标记语言格式文件（如果希望浏览器等浏览工具将文件作为纯文本处理，就使用 text/xml）	xml、dtd	RFC 2376
application/zip	PKWARE zip 归档文件	zip	

D.4.2 audio/*

表 D-4 总结了音频内容类型。

表D-4 “Audio”MIME类型

MIME类型	描　　述	扩　　展	联系方式与参考文献
audio/32kadpcm	32 kHz ADPCM 音频编码		RFC 2421
audio/basic	8kHz 单声道 8 位 ISDN u-lawPCM 编码的音频	au、snd	RFC 1341
audio/G.772.1	G.722.1 将 50Hz-7kHz 的音频信号压缩为 24kbit/s 或 32kbit/s。可用于语音、音乐和其他类型的音频		RFC 3047
audio/L16	Audio/L16 基于 RFC 1890 中描述的 L16。L16 表示以 16 位带符号表示法表示的非压缩音频数据		RFC 2586
audio/MP4A-LATM	MPEG-4 音频		RFC 3016
audio/midi	MIDI 音乐文件	mid、midi、kar	
audio/mpeg	MPEG 编码音频文件	mpga、mp2、mp3	RFC 3003
audio/parityfec	RTP 音频的奇偶前向纠错码		RFC 3009
audio/prs.sid	科摩多公司的 64 SID 音频文件	sid、psid	http://www.geocities.com/SiliconValley/Lakes/5147/sidplay/docs.html#fileformats
audio/telephoneevent	逻辑电话事件		RFC 2833
audio/tone	电话声音模式		RFC 2833
audio/vnd.cns.anp1	支持 Comverse 网络系统公司 Access NP 网络服务平台上的声音和统一报文应用程序特性		Ann McLaughlin Comverse 网络系统 amclaughlin@comversens.com
audio/vnd.cns.inf1	支持 Comverse 网络系统公司 TRILOGUE Infinity 网络服务平台上的声音和统一报文应用程序特性		同上

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
audio/vnd.digitalwinds	Digital Winds 音乐是封装在很小的封包（<3K）内的永不终结、可再生的交互式 MIDI 音乐	eol	Armands Strazds armands.strazds@medienhaus-bremen.de
audio/vnd.everad.plj	EverAD 音频编码的特有格式	plj	Tomer Weisberg tomer@everad.com
audio/vnd.lucent.voice	包含朗讯科技的 Intuity ™ AUDIX® Multimedia Messaging System 和 Lucent Voice Player 在内的声音报文	lvp	Frederick Block rickblock@lucent.com http://www.lucent.com/lvp/
audio/vnd.nortel.vbk	北电网络公司的声音块音频编码特有的格式	vbk	Glenn Parsons Glenn.Parsons@NortelNetworks.com
audio/vnd.nuera.ecelp4800	纽亚通信的音频和语音编码所特有的格式，用于纽亚的 VoIP 网关、终端、应用程序服务器，还作为各种主机平台和 OS 的媒体服务使用	ecelp4800	Michael Fox mfox@nuera.com
audio/vnd.nuera.ecelp7470	同上	ecelp7470	同上
audio/vnd.nuera.ecelp9600	同上	ecelp9600	同上
audio/vnd.octel.sbc	朗讯科技的 Sierra ™、Overture ™和 IMA ™ 平台上的声音报文使用的平均值为 18kbps 的可变速率编码		Jeff Bouis jbouis@lucent.com
audio/vnd.qcelp	高通的音频编码	qcp	Andy Dejaco adejaco@qualcomm.com
audio/vnd.rhetorex.32kadpcm	在朗讯科技的 CallPerformer ™、Unified Messenger ™这样的声音报文产品和其他产品中使用的 32kbps Rhetorex ™ ADPCM 音频编码		Jeff Bouis jbouis@lucent.com
audio/vnd.vmx.cvsd	在包含朗讯科技的 Overture200 ™、Overture 300 ™和 VMX 300 ™产品线在内的声音报文产品中使用的音频编码		同上

（续）

MIME类型	描　　述	扩　　展	联系方式与参考文献
audio/x-aiff	AIFF 音频文件格式	aif, aiff, aifc	
audio/x-pn-realaudio	Real Networks（前身是 Progressive Networks）公司的 RealAudio 元文件格式	ram, rm	
audio/x-pn-realaudio-plugin	来自 Apache 的 mime.types	rpm	
audio/x-realaudio	Real Networks（前身是 Progressive Networks）公司的 RealAudio 音频格式	ra	
audio/x-wav	WAV 音频文件	wav	

D.4.3　chemical/*

表 D-5 中很多信息都是从 Chemical MIME 主页上获得的（http://www.ch.ic.ac.uk/chemime/）。

表D-5　“Chemical”MIME类型

MIME类型	描　　述	扩展	联系方式和参考文献
chemical/x-alchemy	Alchemy 格式	alc	http://www.camsoft.com
chemical/x-cache-csf		csf	
chemical/x-cactvs-binary	CACTVS 二进制格式	cbin	http://cactvs.cit.nih.gov
chemical/x-cactvs-ascii	CACTVS ASCII 格式	cascii	http://cactvs.cit.nih.gov
chemical/x-cactvs-table	CACTVS 表格式	ctab	http://cactvs.cit.nih.gov
chemical/x-cdx	ChemDraw eXchange 文件	cdx	http://www.camsoft.com
chemical/x-cerius	MSI Cerius II 格式	cer	http://www.msi.com
chemical/x-chemdraw	ChemDraw 文件	chm	http://www.camsoft.com
chemical/x-cif	Crystallographic Interchange Format	cif	http://www.bernstein-plus-sons.com/software/rasmol/ http://ndbserver.rutgers.edu/NDB/mmcif/examples/index.html
chemical/x-mmcif	高分子 CIF	mcif	同上
chemical/x-chem3d	Chem3D 格式	c3d	http://www.camsoft.com
chemical/x-cmdf	CrystalMaker 数据格式	cmdf	http://www.crystalmaker.co.uk

（续）

MIME类型	描　　述	扩展	联系方式和参考文献
chemical/x-compass	Takahashi 的 Compass 程序	cpa	
chemical/x-crossfire	Crossfire 文件	bsd	
chemical/x-cml	化学标记语言	cml	http://www.xml-cml.org
chemical/x-csml	化学风格标记语言	csml、csm	http://www.mdli.com
chemical/x-ctx	Gasteiger 小组的 CTX 文件格式	ctx	
chemical/x-cxf		cxf	
chemical/x-daylight-smiles	Smiles 格式	smi	http://www.daylight.com/dayhtml/smiles/index.html
chemical/x-embl-dl-nucleotide	EMBL 核苷酸格式	emb	http://mercury.ebi.ac.uk
chemical/x-galactic-spc	光谱和色谱数据的 SPC 格式	spc	http://www.galactic.com/galactic/Data/spcvue.htm
chemical/x-gamess-input	GAMESS 输入格式	inp、gam	http://www.msg.ameslab.gov/GAMESS/Graphics/MacMolPlt.shtml
chemical/x-gaussian-input	Gaussian Input 格式	gau	http://www.mdli.com
chemical/x-gaussian-checkpoint	Gaussian Checkpoint 格式	fch、fchk	http://products.camsoft.com
chemical/x-gaussian-cube	Gaussian Cube（Wavefunction 波函数）格式	cub	http://www.mdli.com
chemical/x-gcg8-sequence		gcg	
chemical/x-genbank	ToGenBank 格式	gen	
chemical/x-isostar	分子间相互作用的 IsoStar Library	istr、ist	http://www.ccdc.cam.ac.uk
chemical/x-jcamp-dx	JCAMP Spectroscopic Data Exchange 格式	jdx、dx	http://www.mdli.com
chemical/x-jjc-review-surface	Re_View3 Orbital Contour 文件	rv3	http://www.brunel.ac.uk/depts/chem/ch241s/re_view/rv3.htm
chemical/x-jjc-review-xyz	Re_View3Animation 文件	xyb	http://www.brunel.ac.uk/depts/chem/ch241s/re_view/rv3.htm
chemical/x-jjc-review-vib	Re_View3 Vibration（振动）文件	rv2、vib	http://www.brunel.ac.uk/depts/chem/ch241s/re_view/rv3.htm

559

（续）

MIME类型	描　　述	扩展	联系方式和参考文献
chemical/x-kinemage	Kinetic（Protein Structure, 蛋白质结构）图片	kin	http://www.faseb.org/protein/kinemages/MageSoftware.html
chemical/x-macmolecule	高分子文件格式	mcm	
chemical/x-macromodel-input	宏模型分子力学	mmd、mmod	http://www.columbia.edu/cu/chemistry/
chemical/x-mdl-molfile	MDL Molfile	mol	http://www.mdli.com
chemical/x-mdl-rdfile	Reaction（反应）数据文件	rd	http://www.mdli.com
chemical/x-mdl-rxnfile	MDL Reaction 格式	rxn	http://www.mdli.com
chemical/x-mdl-sdfile	MDL Structure 数据文件	sd	http://www.mdli.com
chemical/x-mdl-tgf	MDL Transportable Graphics 格式	tgf	http://www.mdli.com
chemical/x-mif		mif	
chemical/x-mol2	SYBYL 分子的便携式表示形式	mol2	http://www.tripos.com
chemical/x-molconn-Z	Molconn-Z 格式	b	http://www.eslc.vabiotech.com/molconn/molconnz.html
chemical/x-mopac-input	MOPAC Input 格式	mop	http://www.mdli.com
chemical/x-mopac-graph	MOPAC Graph 格式	gpt	http://products.camsoft.com
chemical/x-ncbi-asn1		asn（老格式）	
chemical/x-ncbi-asn1-binary		val	
chemical/x-pdb	蛋白质数据库 pdb	pdb	http://www.mdli.com
chemical/x-swissprot	SWISS-PROT 蛋白质序列数据库	sw	http://www.expasy.ch/spdbv/text/download.htm
chemical/x-vamas-iso14976	材料及标准的凡尔赛协定	vms	http://www.acolyte.co.uk/JISO/
chemical/x-vmd	视觉分子动力学软件	vmd	http://www.ks.uiuc.edu/Research/vmd/
chemical/x-xtel	Xtelplot 文件格式	xtel	http://www.recipnet.indiana.edu/graphics/xtelplot/xtelplot.htm
chemical/x-xyz	Co-ordinate Animation 格式	xyz	http://www.mdli.com

D.4.4 image/*

表 D-6 总结了电子邮件和 HTTP 经常交换的一些图片类型。

表D-6 “Image”MIME类型

MIME类型	描　　述	扩　　展	联系方式和参考文献
image/bmp	Windows 的 BMP 图片格式	bmp	
image/cgm	CGM（Computer Graphics Metafile，计算机图形元文件）是便携式存储和传输二维插图的国际标准		Alan Francis A.H.Francis@open.ac.uk 参见 ISO 8632:1992，IS 8632:1992 Amendment1（1994）和 IS 8632:1992 Amendment2（1995）
image/g3fax	G3 传真字节流		RFC 1494
image/gif	Compuserve GIF 图片	gif	RFC 1341
image/ief		ief	RFC 1314
image/jpeg	JPEG 图片	jpeg、jpg、jpe、jfif	JPEG 草案标准 ISO 10918-1 CD
image/naplps	NAPLPS（北美表示层协议语法，North American Presentation Layer Protocol Syntax）图片		ANSI X3.110-1983 CSA T500-1983
image/png	PNG（便携式网络图像）图片	png	因特网草案 draft-boutell-png-spec-04.txt，“Png Specification Version1.0”
image/prs.btif	美国国家银行为支票及其他应用程序的 BTIF 图片视图所使用的格式	btif、btf	Arthur Rubin arthurr@crt.com
image/prs.pti	PTI 编码图片	pti	Juern Laun juern.laun@gmx.de http://server.hvzgymn.wn.schule-bw.de/pti/
image/tiff	TIFF 图片	tiff、tif	RFC 2302
image/vnd.cns.inf2	支持 Comverse 网络系统公司 TRILOGUE Infinity 网络服务器平台上可用的应用程序特性		Ann McLaughlin Comverse 网络系统公司 amclaughlin@comversens.com
image/vnd.dxf	DXF 矢量 CAD 文件	dxf	

561

（续）

MIME类型	描　　述	扩　　展	联系方式和参考文献
image/vnd.fastbidsheet	FastBid Sheet 中包含一个用来表示工程或建筑图纸的光栅或矢量图片	fbs	Scott Becker scottb@bxwa.com
image/vnd.fpx	柯达的 FlashPix 图片	fpx	Chris Wing format_change_request@kodak.com http://www.kodak.com
image/vnd.fst	FAST 搜索及传输的图片格式	fst	Arild Fuldseth Arild.Fuldseth@fast.no
image/vnd.fujixerox.edmics-mmr	富士施乐的 EDMICS MMR 图片格式	mmr	Masanori Onda Masanori.Onda@fujixerox.co.jp
image/vnd.fujixerox.edmics-rlc	富士施乐的 EDMICS RLC 图片格式	rlc	同上
image/vnd.mix	MIX 文件包含了流中用来表示图片及相关信息的二进制数据。由微软的 PhotDraw 和 PictureIt 软件使用		Saveen Reddy2 saveenr@microsoft.com
image/vnd.net-fpx	柯达的 FlashPix 图片		Chris Wing format_change_request@kodak.com http://www.kodak.com
image/vnd.wap.wbmp	来自 Apache 的 mime.types	wbmp	
image/vnd.xiff	Pagis 软件所用的扩展图片格式	xif	Steve Martin smartin@xis.xerox.com
image/x-cmu-raster	来自 Apache 的 mime.types	ras	
image/x-portable-anymap	PBM 通用图片	pnm	Jeff Poskanzer http://www.acme.com/software/pbmplus/
image/x-portable-bitmap	PBM 位图图片	pbm	同上
image/x-portable-graymap	PBM 灰度图片	pgm	同上
image/x-portable-pixmap	PBM 彩色图片	ppm	同上
image/x-rgb	硅图的 RGB 图片	rgb	
image/x-xbitmap	X-Window System 的位图图片	xbm	
image/x-xpixmap	X-Window System 的彩色图片	xpm	
image/x-xwindowdump	X-Window System 的屏幕截取图片	xwd	

562

D.4.5 message/*

报文是用来（通过电子邮件、HTTP 或其他传输协议）交互数据对象的复合类型。表 D-7 描述了常见的 MIME 报文类型。

表D-7 “Message”MIME类型

MIME类型	描　　述	扩　　展	联系方式及参考文献
message/delivery-status			
message/disposition-notification			RFC 2298
message/external-body			RFC 1341
message/http			RFC 2616
message/news	定义了一种通过电子邮件传输可读新闻文章的方式——新闻首部的语义超出了 RFC 822 定义的范围，所以仅使用 message/rfc822 是不够的		RFC 1036
message/partial	允许对那些太大，无法直接通过电子邮件传输的主体部分进行分段传输		RFC 1341
message/rfc822	完整的电子邮件报文		RFC 1341
message/s-http	安全 HTTP 报文，是对 SSL 上 HTTP 的替代方式		RFC 2660

D.4.6 model/*

MIME 类型 model 是 IETF 注册的扩展类型。它表示的是物理世界的数学模型，用于计算机辅助设计和三维图像。表 D-8 描述了部分 model 格式。

表D-8 “Model”MIME类型

MIME类型	描　　述	扩　　展	联系方式及参考文献
model/iges	IGES（Initial Graphics Exchange Specification，初始图像交换规范）定义了一种中立数据格式，通过它可以在 CAD 系统之间进行信息的数字化交换	igs、iges	RFC 2077
model/mesh		msh、mesh、silo	RFC 2077

（续）

MIME类型	描　　述	扩　　展	联系方式及参考文献
model/vnd.dwf	DWF CAD 文件	dwf	Jason Pratt jason.pratt@autodesk.com
model/vnd.flatland.3dml	支持 Flatland 产品的 3DML 模型	3dml、3dm	Michael Powers pow@flatland.com http://www.flatland.com
model/vnd.gdl model/vnd.gs-gdl	GDL（Geometric Description Language，几何描述语言）是 Graphisoft 为 ArchiCAD 设计的参数化对象定义语言	gdl、gsm、win、dor、lmp、rsm、msm、ism	Attila Babits ababits@graphisoft.hu http://www.graphisoft.com
model/vnd.gtw	Gen-Trix 模型	gtw	Yutaka Ozaki yutaka_ozaki@gen.co.jp
model/vnd.mts	Virtue 的 MTS 模型格式	mts	Boris Rabinovitch boris@virtue3d.com
model/vnd.parasolid.transmit.binary	二进制 Parasolid 建模文件	x_b	http://www.ugsolutions.com/products/parasolid/
model/vnd.parasolid.transmit.text	文本 Parasolid 建模文件	x_t	http://www.ugsolutions.com/products/parasolid/
model/vnd.vtu	Virtue 的 VTU 模型格式	vtu	Boris Rabinovitch boris@virtue3d.com
model/vrml	虚拟现实标记语言格式的文件	wrl、vrml	RFC 2077

563

D.4.7　multipart/*

MIME 类型 multipart 是包含有其他对象的组合对象。子类型描述了多部分封装的实现，以及如何处理组件。表 D-9 总结了 Multipart 媒体类型。

表D-9　“Multipart”MIME类型

MIME类型	描　　述	扩　　展	联系方式及参考文献
multipart/alternative	内容包括一个可替换表达方式列表，每种方式都有自己的 Content-Type。客户端可以选择其支持得最好的组件		RFC 1341
multipart/appledouble	苹果 Macintosh 文件包含了“资源分支”和其他用来描述实际文件内容的桌面数据。这个由多部分组成的内容在一个部分中发送 Apple 元数据，并在另一个部分中发送实际内容		http://www.isi.edu/in-notes/iana/assignments/media-types/multipart/appledouble

（续）

MIME类型	描　述	扩　展	联系方式及参考文献
multipart/byteranges	HTTP 报文中包含了多个范围的内容时，要放在 multipart/byteranges 对象中传输。这个媒体类型包含由 MIME 边界分隔的两个或多个部分，每个部分都有自己的 Content-Type 和 Content-Range 字段		RFC 2068
multipart/digest	包含一组以易读形式表示的个人电子邮件报文		RFC 1341
multipart/encrypted	用两个部分来支持密码加密的内容。第一部分包含了解密第二主体部分的数据所必须的控制信息，它是根据协议参数值标记的。第二部分包含了 application/octet-stream 类型的加密数据		RFC 1847
multipart/form-data	根据用户填表的结果将一组值封装起来		RFC 2388
multipart/header-set	从任意描述性元数据中将用户数据分离出来		http://www.isi.edu/in-notes/iana/assignments/media-types/multipart/header-set
multipart/mixed	一组对象		RFC 1341
multipart/parallel	语法与 multipart/mixed 相同，但在能够使用它的系统中，要同时提供所有的部分		RFC 1341
multipart/related	供包含了几个相互关联的主体部分的复合对象使用。主体各部分之间的关系将其与其他对象类型区分开来。这些关系通常是由引用其他组件的对象组件内部链接表示的		RFC 2387
multipart/report	为各种类型的电子邮件报告定义了一种通用的容器类型		RFC 1892
multipart/signed	使用两个部分来支持经过密码签名的内容。第一部分是包含了其 MIME 首部的内容；第二部分包含了验证数字签名所需的信息		RFC 1847
multipart/voice-message	提供了一种机制，可将声音报文封装到一个标记为 VPIM v2 兼容的容器中		RFC 2421 和 RFC 2423

564

D.4.8 text/*

565 Text 媒体类型包含了字符和潜在的格式化信息。表 D-10 总结了 Text MIME 类型。

表D-10 “Text”MIME类型

MIME类型	描　　述	扩　　展	联系方式和参考文献
text/calendar	支持 iCalendar 日历和日程标准		RFC 2445
text/css	层级样式表	css	RFC 2318
text/directory	装载 LDAP 这样的目录数据库中的记录数据		RFC 2425
text/enriched	简单的格式化文本，支持字体、颜色和间距方式。使用类 SGML 的标记来开始或结束格式化工作		RFC 1896
text/html	HTML 文件	html、htm	RFC 2854
text/parityfec	RTP 流中文本流的前向纠错码		RFC 3009
text/plain	普通的文本	asc、txt	
text/prs.lines.tag	支持用于电子邮件注册的标记表格	tag、dsc	John Lines john@paladin.demon.co.uk http://www.paladin.demon.co.uk/tag-types/
text/rfc822-headers	就像邮件发送失败报告那样，用来绑定一组电子邮件首部		RFC 1892
text/richtext	富文本的较老形式。参见 text/enriched	rtx	RFC 1341
text/rtf	RTF（Rich Text Format，富文本格式）是一种在应用程序之间的传输所使用的，对格式化文本和图片的编码方式。这种格式得到了 MS-DOS、Windows、OS/2 和 Macintosh 平台上各种字处理程序的广泛支持	rtf	
text/sgml	SGML 标记文件	sgml、sgm	RFC 1874
text/t140	支持同步 RTP 多媒体中使用的标准 T.140 文本		RFC 2793
text/tab-separated-values	TSV 是在数据库和电子表格以及文字处理软件之间进行数据交换的通用方法。它由很多行构成，每行都有一些由 tab 字符分隔的字段	tsv	http://www.isi.edu/in-notes/iana/assignments/media-types/text/tab-separated-values
text/uri-list	URN 解析程序和所有需要进行大量 URI 列表通信的其他应用程序使用的简单的、经过注释的 URL 和 URN 列表	uris、uri	RFC 2483

566

（续）

MIME类型	描　　述	扩　　展	联系方式和参考文献
text/vnd.abc	ABC 文件是乐谱的可读格式	abc	http://www.gre.ac.uk/~c.walshaw/abc/ http://home1.swipnet.se/~w-11382/abcbnf.htm
text/vnd.curl	提供了一组由 CURL 运行时插件解释的内容定义语言	curl	Tim Hodge thodge@curl.com
text/vnd.DMClientScript	Dream Seeker 客户端应用程序会访问一些非 HTTP 站点（比如 BYOND、IRC 或 telnet），CommonDM Client Script 文件被当作指向这些站点的超链接使用	dms	Dan Bradley dan@dantom.com http://www.byond.com/code/ref/
text/vnd.fly	Fly 是个文本预处理器，它会用简单的语法来创建数据库和 Web 页面之间的接口	fly	John-Mark Gurney jmg@flyidea.com http://www.flyidea.com
text/vnd.fmi.flexstor	用于 SUVDAMA 和 UVRAPPF 项目	flx	http://www.ozone.fmi.fi/SUVDAMA/ http://www.ozone.fmi.fi/UVRAPPF/
text/vnd.in3d.3dml	用于 In3D 播放器	3dml、3dm	Michael Powers powers@insideout.net
text/vnd.in3d.spot	用于 In3D 播放器	spot、spo	同上
text/vnd.IPTC.NewsML	IPTC 国际新闻电信委员会指定的 NewsML 格式	xml	David Allen m_director_iptc@dial.pipex.com http://www.iptc.org
text/vnd.IPTC.NITF	IPTC 指定的 NITF 格式	xml	同上 http://www.nitf.org
text/vnd.latex-z	支持包含 Z 表示法的 LaTeX 文档。Z 表示法（读作“zed”）是基于 Zermelo-Fraenkel 集合论和一阶谓词逻辑的，有助于描述计算机系统		http://www.comlab.ox.ac.uk/archive/z/
text/vnd.motorola.reflex	提供了一种从 ReFLEX ™无线设备提交简单文本的通用的方式		Mark Patton fmp014@email.mot.com 有许可协议的情况下，可以从摩托罗拉得到（Enabling Protocol specification）的 FLEXsuite ™部分

（续）

MIME类型	描　　述	扩　　展	联系方式和参考文献
text/vnd.ms-mediapackage	这种类型由微软应用程序 MStore.exe 和 7 storDB.exe 处理	mpf	Jan Nelson jann@microsoft.com
text/vnd.wap.si	SI（Service Indication，业务指示符）对象中包含了用于描述事件的报文和用于描述从何处装载相应服务的 URI	si、xml	WAP Forum Ltd http://www.wapforum.org
text/vnd.wap.sl	SL（Service Loading，服务装载）内容类型提供了一种向移动客户端的用户代理传送 URI 的手段。客户端自身会自动地装载 URI 所指向的内容，在适当的时候且没有用户干预的情况下，在指定的用户代理上执行它	sl、xml	同上
text/vnd.wap.wml	WML（Wireless Markup Language，无线标记语言）是一种基于 XML 的标记语言，定义了包含移动电话和寻呼机在内的窄带设备的内容和用户接口	wml	同上
text/vnd.wap.wmlscript	WMLScript 是一种用于无线设备的 JavaScript 的拓展	wmls	同上
text/x-setext	来自 Apache 的 mime.types	etx	
text/xml	Extensible Markup 语言格式文件（如果下载时希望浏览器将文件保存下来就使用 application/xml）	xml	RFC 2376

D.4.9　video/*

表 D-11 列出了一些常见的视频电影格式。注意，有些视频格式被划分为 application 类型。

表D-11　“Video”MIME类型

MIME类型	描　　述	扩　　展	联系方式和参考文献
video/MP4V-ES	RTP 承载的 MPEG-4 视频负载		RFC 3016
video/mpeg	根据 ISO 11172 CD MPEG 标准编码的视频	mpeg、mpg、mpe	RFC 1341
video/parityfec	RTP 流上承载的数据所使用的前向纠错视频格式		RFC 3009
video/pointer	传输演示指针的位置信息		RFC 2862
video/quicktime	苹果 Quicktime 的视频格式	qt、mov	http://www.apple.com

（续）

MIME类型	描　　述	扩　　展	联系方式和参考文献
video/vnd.fvt	FAST Search & Transfer 公司的视频格式	fvt	Arild Fuldseth Arild.Fuldseth@fast.no
video/vnd.motorola.videovideo/vnd.motorola.videop	摩托罗拉 ISG 产品的特有格式		Tom McGinty Motorola ISG tmcginty@dma.isg.mot
video/vnd.mpegurl	这种媒体类型由一系列 MPEG 视频文件的 URL 构成	mxu	Heiko Recktenwald uzs106@uni-bonn.de “Power and Responsibility: Conversations with Contributors”, Guy van Belle 等, LMJ 9 (1999), 127-133, 129 (MIT Press)
video/vnd.nokia.interleaved-multimedia	在诺基亚 9210 Communicator 视频播放器及相关工具中使用	nim	Petteri Kangaslampi petteri.kangaslampi@nokia.com
video/x-msvideo	微软的 AVI 电影	avi	http://www.microsoft.com
video/x-sgi-movie	硅图公司的电影格式	movie	http://www.sgi.com

568

D.4.10　实验类型

主类型集支持大多数内容类型。表 D-12 列出了一种实验类型，用于在某些 Web 服务器上配置的会议软件。

表D-12　扩展MIME类型

MIME类型	描　　述	扩　　展	联系方式和参考文献
x-conference/x-cooltalk	网景公司的协作工具	ice	

569

附录E

Base-64编码

HTTP 将 Base-64 编码用于基本认证及摘要认证，在几种 HTTP 扩展中也使用了该编码。本附录解释了 Base-64 编码，提供了转换表和指向 Perl 软件的指针，可以帮助你在 HTTP 软件中正确使用 Base-64 编码。

E.1 Base-64编码保证了二进制数据的安全

Base-64 编码可以将任意一组字节转换成较长的常见文本字符序列，从而可以合法地作为首部字段值。Base-64 编码将用户输入或二进制数据，打包成一种安全格式，将其作为 HTTP 首部字段的值发送出去，而无须担心其中包含会破坏 HTTP 分析程序的冒号、换行符或二进制值。

Base-64 编码是作为 MIME 多媒体电子邮件标准的一部分开发的，这样 MIME 就可以在不同的合法电子邮件网关之间传输富文本和任意的二进制数据了。[1]Base-64 编码与将二进制数据文本化表示的 uuencode 和 BinHex 标准在本质上很类似，但空间效率更高。MIME RFC 2045 的第 6.8 节详细介绍了 Base-64 算法。

E.2 8位到6位

Base-64 编码将一个 8 位字节序列拆散为 6 位的片段，并为每个 6 位的片段分配一个字符，这个字符是 Base-64 字母表中的 64 个字符之一。这 64 个输出字符都是很
570 常见的，可以安全地放在 HTTP 首部字段中。这 64 个字符中包含大小写字母、数字、+ 和 /，还使用了特殊字符 =。表 E-1 显示了 Base-64 的字母表。

注意，由于 Base-64 编码用了 8 位字符来表示信息中的 6 个位，所以 Base-64 编码字符串大约比原始值扩大了 33%。

表E-1 Base-64字母表

0	A	8	I	16	Q	24	Y	32	g	40	o	48	w	56	4
1	B	9	J	17	R	25	Z	33	h	41	p	49	x	57	5
2	C	10	K	18	S	26	a	34	i	42	q	50	y	58	6
3	D	11	L	19	T	27	b	35	j	43	r	51	z	59	7
4	E	12	M	20	U	28	c	36	k	44	s	52	0	60	8
5	F	13	N	21	V	29	d	37	l	45	t	53	1	61	9
6	G	14	O	22	W	30	e	38	m	46	u	54	2	62	+
7	H	15	P	23	X	31	f	39	n	47	v	55	3	63	/

注 1：有些邮件网关会悄悄地去除 ASCII 值在 0 ～ 31 之间的“非打印”字符。其他程序会将一些字节作为流量控制字符或其他特殊控制字符来解释，或将回车符转换成换行符之类的字符。有些程序在收到带有值大于 127 的国际字符时会出现致命的错误，因为其软件不是“8 位干净”（8-bitclean）的。

图 E-1 是一个简单的 Base-64 编码实例。在这里，三个字符组成的输入值“Ow!”是 Base-64 编码的，得到的是 4 个字符的 Base-64 编码值“T3ch”。它是按以下方式工作的。

（1）字符串“Ow!”被拆分成 3 个 8 位的字节（0x4F、0x77、0x21）。
（2）这 3 个字节构成了一个 24 位的二进制值 010011110111011100100001。
（3）这些位被划分为一些 6 位的序列 010011、110111、01110、100001。
（4）每个 6 位值都表示了从 0 ～ 63 之间的一个数字，对应 Base-64 字母表中 64 个字符之一。得到的 Base-64 编码字符串是个 4 字符的字符串“T3ch”，然后就可以通过线路将这个字符串作为“安全的”8 位字符传送出去，因为只用了一些移植性最好的字符（字母、数字等）。

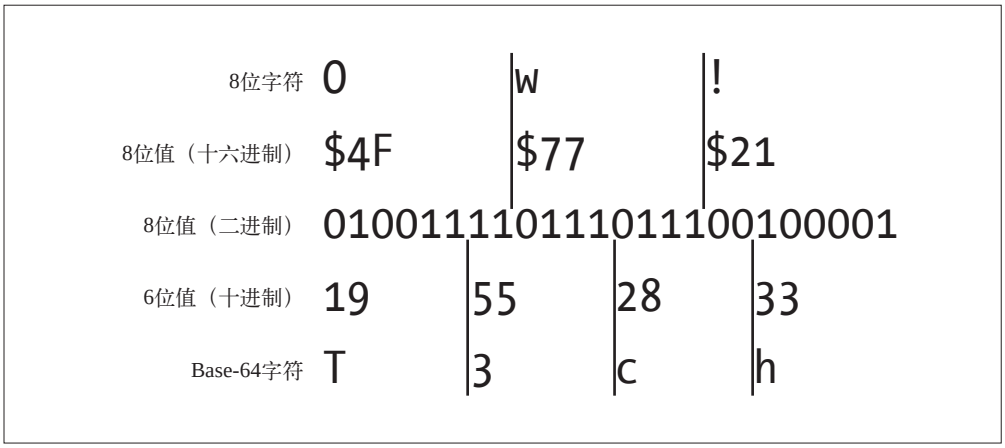


图 E-1　Base-64 编码实例

E.3　Base-64填充

Base-64 编码收到一个 8 位字节序列，将这个二进制序列流划分成 6 位的块。二进制序列有时不能正好平均地分成 6 位的块，在这种情况下，就在序列末尾填充零位，使二进制序列的长度成为 24 的倍数（6 和 8 的最小公倍数）。

对已填充的二进制串进行编码时，任何完全填充（不包含原始数据中的位）的 6 位组都由特殊的第 65 个符号“=”表示。如果 6 位组是部分填充的，就将填充位设置为 0。

表 E-2 显示了一些填充实例。初始输入字符串“a:a”为 3 字节（24 位）。24 是 6 和 8 的倍数，因此无需填充，得到的 Base-64 编码字符串为“YTph”。

表E-2　Base-64填充实例

输入数据	二进制序列（填充位以x表示）	已编码数据
a:a	011000 010011 101001 100001	YTph
a:aa	011000 010011 101001 100001 011000 01xxxx xxxxxx xxxxxx	YTphYQ==
a:aaa	011000 010011 101001 100001 011000 010110 0001xx xxxxxx	YTphYWE=
a:aaaa	011000 010011 101001 100001 011000 010110 000101 100001	YTphYWFh

然而，再增加一个字符，输入字符串会变成 32 位长。而 6 和 8 的下一个公倍数是 48，因此要添加 16 位的填充码。填充的前 4 位是与数据位混合在一起的。得到的 6 位组 01xxxx，会被当作 010000、十进制中的 16，或者 Base-64 编码的 Q 来处理。剩下的两个 6 位组都是填充码，用“=”表示。

E.4　Perl实现

MIME::Base64 是 Perl 中的 Base-64 编 / 解码模块。可以在 http://www.perldoc.com/perl5.6.1/lib/MIME/Base64.html 上看到有关这个模块的内容。

572 可以用 MIME::Base64 encode_base64 和 decode_base64 方法对字符串进行编解码：

```
use MIME::Base64;

$encoded = encode_base64('Aladdin:open sesame');
$decoded = decode_base64($encoded);
```

E.5　更多信息

更多有关 Base-64 编码的信息，参见以下信息。

http://www.ietf.org/rfc/rfc2045.txt
RFC 2045 的第 6.8 节，“MIME Part 1: Format of Internet Message Bodies,”（MIME 的第一部分：因特网报文主体的格式），是 Base-64 编码的官方规范。

http://www.perldoc.com/perl5.6.1/lib/MIME/Base64.html
这个 Web 站点提供了对 Base-64 字符串进行编 / 解码的 MIME::Base64 Perl 模块的
573 文档。

附录F

摘要认证

本附录包含了实现 HTTP 摘要认证功能所需的支撑数据和源代码。

F.1 摘要WWW-Authenticate指令

表 F-1 根据 RFC 2617 中的描述，对 WWW-Authenticate 指令进行了说明。与往常一样，最新细节请参见官方规范。

表F-1 （来自RFC 2617的）摘要WWW-Authenticate首部指令

指　令	描　述
realm	显示给用户的字符串，这样用户就可以知道该使用哪个用户名和密码了。这个字符串中至少应该包含执行认证功能的主机名字，此外可能还会说明可能拥有访问权的用户的集合。例如，registered_users@gotham.news.com
nonce	服务器特有的数据字符串，每次产生一个 401 响应时都应该生成一个唯一的数据字符串。建议这个字符串为 Base-64 或十六进制数据。需要特别说明的是，由于此字符串是放在首部行中作为引用字符串传送的，所以不允许使用双引号。 nonce 的内容是与实现有关的。实现的质量取决于选择是否合适。比如，可以将 nonce 构造成以下内容的 Base-64 编码： `time-stamp H(time-stamp ":" ETag ":" private-key)` 其中 time-stamp 是服务器生成的时间或其他不重复的数值，ETag 是与所请求实体有关的 HTTP ETag 首部的值，private-key 是只有服务器知道的数据。使用这种形式的 nonce，服务器会在收到客户端的 Authentication 首部之后重新对散列部分进行计算，如果与该首部的 nonce 不符，或者时间戳的值不够近，就可以拒绝请求。通过这种方式，服务器可以限制 nonce 的有效时间。包含 ETag 可以防止对资源更新版本的重放请求。（注意：在 nonce 中包含客户端的 IP 地址，看起来好像为服务器提供了限制最初获得此 nonce 的客户端重用 nonce 的能力，但这样会破坏代理集群，来自单个用户的请求通常都会经过集群中不同的代理进行传输。而且，IP 地址欺骗也不是很难。） 实现可以选择不接受以前用过的 nonce，或以前用过的摘要，以防止重放攻击，或者选择为 POST 或 PUT 请求使用一次性 nonce 或摘要，为 GET 请求使用时间戳
domain	一个引用的、由空格分隔的 URI 列表（如 RFC 2396，“Uniform Resource Identifiers: GenericSyntax”所述），这些 URI 定义了保护空间。如果 URI 是个 abs_path，它就是相对于受访服务器的典型根 URL 的。这个列表中的绝对 URI 所指的服务器可能不是受访服务器。 客户端可以用这个列表来判定应该将同样的认证信息发送给哪个 URI 集：可以假定所有以此列表中的 URI 作为前缀的 URI（在将两者都转换为绝对 URI 之后）都位于同一个保护空间内。 如果省略了这条指令，或者其值为空，客户端就应该假定保护空间中包含了响应服务器上的所有 URI。 这条指令在 Proxy-Authenticate 首部是无意义的，此时，保护空间总是包括整个代理；如果提供了这条指令，也应该将其忽略

574

（续）

指　令	描　述
opaque	一个由服务器指定的数据串，应该由客户端不经修改地放在后继请求的 Authorization 首部中返回，这些后继请求应使用同一保护空间内的 URI。建议这个字符串采用 Base-64 或十六进制的数据
stale	一个标志，用来说明由于 nonce 值太过陈旧，前一条来自客户端的请求被拒绝了。如果 stale 为 TRUE（不区分大小写），客户端可能希望以新加密的响应重试请求，而不用再次提示用户输入新的用户名和密码。只有在服务器收到一条 nonce 无效，但摘要有效的请求（说明客户端知道正确的用户名 / 密码）时，才应该将 stale 设置为 TRUE。如果 stale 为 FALSE，或者除 TRUE 之外的其他值，或者没有提供 stale 指令，用户名和 / 或密码就是无效的，需要获取新的值

algorithm	一个字符串，说明了一对儿用来生成摘要和校验码的算法。如果没有提供这个字符串，就假定它为“MD5”。如果不识别此算法，就忽略这种质询（如果有多个算法的话，就使用另外一个）。
	在这份文档中，用“KD(secret,data)”来表示用密码“secret”对数据“data”使用摘要算法得到的字符串，而对数据“data”使用校验和算法得到的字符串则表示为“H(data)”。表示法“unq(X)”表示引用字符串“X”的值（不包含左右两边的引号）。
	对 MD5 和 MD5-sess 算法来说：
	`H(data) = MD5(data)`
	`HD(secret, data) = H(concat(secret, ":", data))`
	也就是说，摘要就是将密码的 MD5 与冒号和数据连接在一起。MD5-sess 算法目的是支持使用高效的第三方认证服务器

qop	这条指令是可选的，只是为了与 RFC 2069[6] 后向兼容才保留的。所有与此版本的摘要方案兼容的实现都应该使用它。 如果提供了这条指令，它就是由一个或多个标记构成的引用字符串，用来说明服务器所支持的“安全保障”值。值 auth 说明要进行认证，值 auth-int 说明要进行具有完整性保护的认证。一定要忽略那些不识别的选项
<extension>	未来可以通过这条指令进行扩展。要忽略所有不认识的指令

F.2　摘要Authorization指令

表 F-2 根据 RFC 2617 的描述，对每条摘要 Authorization 指令都进行了说明。最新的细节请参见官方规范。

575

表F-2　（来自RFC 2617的）摘要Authorization首部指令

指　令	描　述
username	指定域中的用户名
realm	在 WWW-Authenticate 首部中传送给客户端的域

（续）

指　令	描　述
nonce	在 WWW-Authenticate 首部中传送给客户端的那个与服务器相同的 nonce
uri	来自请求行请求 URI 中的 URI。由于代理可以在传输中修改请求行，所以会出现重复。进行正确的摘要验证计算可能需要原始 URI
response	这个就是实际的摘要——摘要认证的重点！响应是一个由 32 个十六进制数字组成的字符串，由沟通好的摘要算法生成，用来证明用户知道这个密码
algorithm	一个字符串，说明了用来生成摘要和校验和的一对儿算法。如果未提供，就将其假定为 MD5
opaque	服务器在 WWW-Authenticate 首部指定的数据串，应该由客户端不经修改地在后继请求的 WWW-Authenticate 首部中返回，这个请求应使用同一保护空间内的 URI
cnonce	如果发送了 qop 指令，就一定要使用这条指令，如果服务器没有在 WWW-Authenticate 首部字段中发送 qop 指令，就一定不能使用这条指令。 cnonce 值是客户端提供的不透明引用字符串值，客户端和服务器都用它来避免选择明文攻击，以提供双向认证以及一些报文一致性检查。 参见本附录稍后介绍的响应摘要和请求摘要计算
qop	说明客户端对报文应用的“安全保障”是什么。如果提供了这条指令，它的值就必须是服务器在 WWW-Authenticate 首部中说明的、它支持的可选值之一。这些值会影响请求摘要的计算方式。 它只是个单独的标记，而不是 WWW-Authenticate 中那样的可选值引用列表。 这条指令是可选的，以保持对 RFC 2069 最小实现的后向兼容，但如果服务器说明它是通过在 WWW-Authenticate 首部字段中提供 qop 指令来支持 qop 的，就应该使用这条指令
nc	如果发送了 qop 指令，就一定要指定这条指令，如果服务器没有在 WWW-Authenticate 首部字段中发送 qop 指令，就一定不能使用这条指令。 其值是个十六进制值，表示客户端已经发送的包含 nounce 值的请求次数（包括当前请求在内）。比如，作为对指定 nonce 值的响应发送的第一条请求中，客户端会发送 nc="00000001"。 这条指令的目的是允许服务器通过维护自己保存的此计数值副本来发现请求重放——如果看到了两次相同的 nc 值，就说明请求是重放的
<extension>	未来可以通过这条指令进行扩展。所有不认识的指令都要忽略掉

F.3　摘要Authentication-Info指令

表 F-3 根据 RFC 2617 的描述，对每条 Authentication-Info 指令都进行了说明。
576 最新的细节请参见官方规范。

表F-3 （来自RFC 2617的）摘要Authentication-Info首部指令

指　令	描　述
nextnonce	nextnonce 指令的值是服务器希望客户端为未来的认证响应使用的 nonce。服务器可能会发送带有 nextnonce 字段的 Authentication-Info 首部，作为实现一次性 nonce 或修改 nonce 的手段。如果提供了 nextnonce 字段，客户端在为下一条请求构建 Authorization 首部时就应该使用它。客户端如果没能做到，就会收到来自服务器的 "stale=TRUE" 认证请求。 服务器实现应该仔细地考虑使用这种机制带来的性能影响。如果每条响应都包含了必须在服务器接收的下一条请求中使用的 nextnounce 指令，就不可能使用管道化请求了。应该考虑在性能和安全之间进行一些平衡，允许在有限的时间内使用老的 nonce 值，以实现请求的管道化。使用 nounce 计数可以在不影响管道化的情况下，维护一个新的服务器 nonce 的大部分安全优势
qop	说明了服务器应用到响应上的“安全保障”选项。值 auth 说明要进行认证，值 auth-int 说明要进行带有完整性保护的认证。服务器在响应中使用的 qop 指令值应该与客户端在相应请求中发送的值相同

rspauth	response auth 指令中的可选响应摘要支持双向认证——服务器证明了它知道用户的密码，而且通过 qop="auth-int"，它还为响应提供了有限的完整性保护。除了当 qop="auth" 或者没有在 Authorization 首部为请求指定 qop 的情况，response-digest 值的计算方式与 Authorization 首部的 request-digest 类似，A2 为：
	`A2 = ":" digest-uri-value`
	当 qop="auth-int" 时，A2 为：
	`A2 = ":" digest-uri-value ":" H(entity-body)`
	其中 digest-uri-value 是请求的 Authorization 首部中 uri 指令的值。cnonce 和 nc 值一定要与此报文所响应的客户端请求中的相应值相同。如果指定了 qop="auth " 或者 qop="auth-int"，就必须提供 rspauth 指令

cnonce	cnonce 值一定要与此报文所响应的客户端请求中的相应值一样。如果指定了 qop="auth " 或 qop="auth-int"，就必须提供 cnonce 指令
nc	nc 值一定要与此报文所响应的客户端请求中的相应值一样。如果指定了 qop="auth " 或 qop="auth-int"，就必须提供 nc 指令
<extension>	未来可以通过这条指令进行扩展。所有不识别的指令都要忽略掉

F.4 参考代码

下列代码实现了 RFC 2617 中 H(A1)、H(A2)、request-digest 和 response-digest 的计算。它使用了 RFC 1321 中的 MD5 实现。

F.4.1 文件digcalc.h

```
#define HASHLEN 16
typedef char HASH[HASHLEN];
#define HASHHEXLEN 32
typedef char HASHHEX[HASHHEXLEN+1];
#define IN
#define OUT
/* calculate H(A1) as per HTTP Digest spec */
void DigestCalcHA1(
    IN char * pszAlg,
    IN char * pszUserName,
    IN char * pszRealm,
    IN char * pszPassword,
    IN char * pszNonce,
    IN char * pszCNonce,
    OUT HASHHEX SessionKey
    );

/* calculate request-digest/response-digest as per HTTP Digest spec */
void DigestCalcResponse(
    IN HASHHEX HA1, /* H(A1) */
    IN char * pszNonce, /* nonce from server */
    IN char * pszNonceCount, /* 8 hex digits */
    IN char * pszCNonce, /* client nonce */
    IN char * pszQop, /* qop-value: "", "auth", "auth-int" */
    IN char * pszMethod, /* method from the request */
    IN char * pszDigestUri, /* requested URL */
    IN HASHHEX HEntity, /* H(entity body) if qop="auth-int" */
    OUT HASHHEX Response /* request-digest or response-digest */
    );
```

577

F.4.2 文件“digcalc.c”

```
#include <global.h>
#include <md5.h>
#include <string.h>
#include "digcalc.h"

void CvtHex(
    IN HASH Bin,
    OUT HASHHEX Hex
    )
    {

unsigned short i;
unsigned char j;
for (i = 0; i < HASHLEN; i++) {
    j = (Bin[i] >> 4) & 0xf;
    if (j <= 9)
        Hex[i*2] = (j + '0');
    else
        Hex[i*2] = (j + 'a' - 10);
    j = Bin[i] & 0xf;
```

```
    if (j <= 9)
        Hex[i*2+1] = (j + '0');
    else
        Hex[i*2+1] = (j + 'a' - 10);
    };
    Hex[HASHHEXLEN] = '\0';
    };

/* calculate H(A1) as per spec */
void DigestCalcHA1(
    IN char * pszAlg,
    IN char * pszUserName,
    IN char * pszRealm,
    IN char * pszPassword,
    IN char * pszNonce,
    IN char * pszCNonce,
    OUT HASHHEX SessionKey
    )
{
    MD5_CTX Md5Ctx;
    HASH HA1;
    MD5Init(&Md5Ctx);
    MD5Update(&Md5Ctx, pszUserName, strlen(pszUserName));
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszRealm, strlen(pszRealm));
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszPassword, strlen(pszPassword));
    MD5Final(HA1, &Md5Ctx);
    if (stricmp(pszAlg, "md5-sess") == 0) {
        MD5Init(&Md5Ctx);
        MD5Update(&Md5Ctx, HA1, HASHLEN);
        MD5Update(&Md5Ctx, ":", 1);
        MD5Update(&Md5Ctx, pszNonce, strlen(pszNonce));
        MD5Update(&Md5Ctx, ":", 1);
        MD5Update(&Md5Ctx, pszCNonce, strlen(pszCNonce));
        MD5Final(HA1, &Md5Ctx);
    };
    CvtHex(HA1, SessionKey);
};
/* calculate request-digest/response-digest as per HTTP Digest spec */
void DigestCalcResponse(
    IN HASHHEX HA1, /* H(A1) */
    IN char * pszNonce, /* nonce from server */
    IN char * pszNonceCount, /* 8 hex digits */
    IN char * pszCNonce, /* client nonce */
    IN char * pszQop, /* qop-value: "", "auth", "auth-int" */
    IN char * pszMethod, /* method from the request */
    IN char * pszDigestUri, /* requested URL */
    IN HASHHEX HEntity, /* H(entity body) if qop= "auth-int" */
    OUT HASHHEX Response /* request-digest or response-digest */
    )
{
    MD5_CTX Md5Ctx;
    HASH HA2;
    HASH RespHash;
    HASHHEX HA2Hex;
    // calculate H(A2)
```

```
    MD5Init(&Md5Ctx);
    MD5Update(&Md5Ctx, pszMethod, strlen(pszMethod));
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszDigestUri, strlen(pszDigestUri));
    if (stricmp(pszQop, "auth-int") == 0) {
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, HEntity, HASHHEXLEN);
};
MD5Final(HA2, &Md5Ctx);
CvtHex(HA2, HA2Hex);
// calculate response
MD5Init(&Md5Ctx);
MD5Update(&Md5Ctx, HA1, HASHHEXLEN);
MD5Update(&Md5Ctx, ":", 1);
MD5Update(&Md5Ctx, pszNonce, strlen(pszNonce));
MD5Update(&Md5Ctx, ":", 1);
if (*pszQop) {
       MD5Update(&Md5Ctx, pszNonceCount, strlen(pszNonceCount));
       MD5Update(&Md5Ctx, ":", 1);
       MD5Update(&Md5Ctx, pszCNonce, strlen(pszCNonce));
       MD5Update(&Md5Ctx, ":", 1);
       MD5Update(&Md5Ctx, pszQop, strlen(pszQop));
       MD5Update(&Md5Ctx, ":", 1);
    };
    MD5Update(&Md5Ctx, HA2Hex, HASHHEXLEN);
    MD5Final(RespHash, &Md5Ctx);
    CvtHex(RespHash, Response);
};
```

F.4.3 文件digtest.c

```
#include <stdio.h>
#include "digcalc.h"

void main(int argc, char ** argv) {
    char * pszNonce = "dcd98b7102dd2f0e8b11d0f600bfb0c093";
    char * pszCNonce = "0a4f113b";
    char * pszUser = "Mufasa";
    char * pszRealm = "testrealm@host.com";
    char * pszPass = "Circle Of Life";
    char * pszAlg = "md5";
    char szNonceCount[9] = "00000001";
    char * pszMethod = "GET";
    char * pszQop = "auth";
    char * pszURI = "/dir/index.html";
    HASHHEX HA1;
    HASHHEX HA2 = "";
    HASHHEX Response;
    DigestCalcHA1(pszAlg, pszUser, pszRealm, pszPass,
        pszNonce, pszCNonce, HA1);
        DigestCalcResponse(HA1, pszNonce, szNonceCount, pszCNonce, pszQop,
        pszMethod, pszURI, HA2, Response);
    printf("Response = %s\n", Response);
};
```

579
≀
580

附录G

语言标记

语言标记是一些短小的标准字符串，用来命名所使用的语言——比如，fr（法语）和 en-GB（英式英语）。每个标记都由一个或多个称为子标记的部分组成，中间由连字符分隔。16.4 节曾详细介绍了语言标记。

本附录总结了语言标记的规则、标准化的标记以及注册信息。包含下列参考资料：

- G.1 节总结了第一个（主）子标记所用规则；
- G.2 节总结了第二个子标记所用规则；
- 表 G-1 显示了已在 IANA 注册的语言标记；
- 表 G-2 列出了 ISO 639 语言代码；
- 表 G-3 列出了 ISO 3166 国家代码。

G.1 第一个子标记所用规则

如果第一个子标记为：

- 2 个字符长，是 ISO 639[1] 和 639-1 标准的语言代码；
- 3 个字符长，是 ISO 639-2[2] 标准中列出的语言代码；
- 字母“i”，语言标记就是在 IANA 中显式注册过的语言标记；
- 字母“x”，语言标记就是私有的、非标准的扩展子标记。

581 表 G-2 总结了 ISO 639 和 639-2 的名称。

G.2 第二个子标记所用规则

如果第二个子标记为：

- 2 个字符长，就是 ISO 3166[3] 定义的国家 / 地区码；
- 3 ～ 8 个字符长，是已在 IANA 注册过的语言标记；
- 1 个字符长，就是非法的。

表 G-3 总结了 ISO 3166 国家代码。

注 1：参见 ISO 标准 639，“Codes for the representation of names of languages”。

注 2：参见 ISO 639-2，“Codes for the representation of names of languages—Part 2: Alpha-3 code”（语言名表示码——第 2 部分：Alpha-3 代码）。

注 3：ISO 3166 将国家代码 AA、QM-QZ、XA-XZ 和 ZZ 保留作为用户分配的代码。不能用它们来构造语言标记。

G.3 IANA已注册的语言标记

表G-1 语言标记

IANA语言标记	描 述
i-bnn	布农语
i-default	默认语言背景
i-hak	客家语
i-klingon	克林根语
i-lux	卢森堡语
i-mingo	明戈语
i-navajo	纳瓦霍语
i-pwn	排湾语
i-tao	越南语
i-tay	泰雅语
i-tsu	邹族语
no-bok	挪威“书面语言”
no-nyn	挪威“新挪威语”
zh-gan	赣语
zh-guoyu	普通话或标准汉语
zh-hakka	客家话
zh-min	闽语、福州话、福建话、厦门话或台湾话
zh-wuu	上海话或吴语
zh-xiang	湘语或湖南话
zh-yue	粤语

582

G.4 ISO 639语言代码

表G-2 ISO 639和ISO 639-2语言代码

语 言	ISO 639	ISO 639–2
阿布哈西亚语	ab	abk
亚齐语		ace
阿乔利语		ach
阿当梅语		ada
阿法尔语	aa	aar

（续）

语　言	ISO 639	ISO 639–2
阿弗里希利语		afh
南非语	af	afr
其他亚非语系		afa
阿坎语		aka
阿卡德语		akk
阿尔巴尼亚语	sq	alb/sqi
阿留申语		ale
阿尔冈昆语系		alg
其他阿尔泰语系		tut
阿姆哈拉语	am	amh
阿帕切语		apa
阿拉伯语	ar	ara
阿拉米语		arc
阿拉帕霍语		arp
阿劳卡尼亚语		arn
阿拉瓦克语		arw
亚美尼亚语	hy	arm/hye
其他人工语言		art
阿萨姆语	as	asm
阿撒帕斯坎巴斯卡语系		ath
其他南岛语系		map
阿瓦尔语		ava
阿维斯陀语		ave
阿瓦德语		awa
艾马拉语	ay	aym
阿塞拜疆语	az	aze
阿兹台克语		nah
巴厘语		ban
其他波罗地语系语言		bat
俾路支语		bal
班巴拉语		bam

583

（续）

语　　言	ISO 639	ISO 639–2
巴米累克语		bai
班达语		bad
其他班图语系语言		bnt
巴萨语		bas
巴什基尔语	ba	bak
巴斯克语	eu	baq/eus
贝雅语		bej
本巴语		bem
孟加拉语	bn	ben
其他帕帕尔语		ber
博杰普尔语		bho
比哈尔语	bh	bih
比科尔语		bik
比尼语		bin
比斯拉马语	bi	bis
布拉吉语		bra
布尔吞语	be	bre
布吉语		bug
保加利亚语	bg	bul
布里亚特语		bua
缅甸语	my	bur/mya
白俄罗斯语	be	bel
卡多语		cad
巴勒比语		car
加泰罗尼亚语	ca	cat
其他高加索语系语言		cau
宿务语		ceb
其他凯尔特语系语言		cel
其他中美洲印第安语系语言		cai
查加台语		chg
查莫罗语		cha

（续）

语　言	ISO 639	ISO 639–2
车臣语		che
切罗基语		chr
584 夏延语		chy
契布卡语		chb
汉语	zh	chi/zho
契努克混合语		chn
乔克托语		cho
古教会斯拉夫语		chu
楚瓦什语		chv
科普特语		cop
康沃尔语		cor
科西嘉语	co	cos
克里语		cre
克里克语		mus
其他克里奥尔混合语		crp
其他英语克里奥尔混合语		cpe
其他法语克里奥尔混合语		cpf
其他葡萄牙语克里奥尔混合语		cpp
其他库什特语系语言		cus
克罗地亚语	hr	
捷克语	cs	ces/cze
达科他语		dak
丹麦语	da	dan
特拉华语		del
丁卡语		din
迪维希语		div
多格拉语		doi
其他德拉维语系语言		dra
杜亚拉语		dua
荷兰语	nl	dut/nla

（续）

语　言	ISO 639	ISO 639–2
中古荷兰语（约 1050 年～ 1350 年）		dum
迪乌拉语		dyu
宗喀语	dz	dzo
艾菲克语		efi
古埃及语		egy
埃克丘克语		eka
埃兰语		elx
英语	en	eng
中古英语（约 1100 年～ 1500 年）		enm
古英语（约 450 年～ 1100 年）		ang
其他爱斯基摩语系语言		esk
世界语	eo	epo
爱沙尼亚语	et	est
爱威语		ewe
艾旺多语		ewo
芳语		fan
芳蒂语		fat
法罗语	fo	fao
斐济语	fj	fij
芬兰语	fi	fin
其他芬兰 – 乌戈尔语系语言		fiu
丰族语		fon
法语	fr	fra/fre
中古法语（约 1400 年～ 1600 年）		frm
古法语（约 842 年～ 1400 年）		fro
弗里西亚语	fy	fry
富拉语		ful
加语		gaa
盖尔语（苏格兰）		gae/gdh
加利西亚语	gl	glg
干达语		lug

（续）

语　言	ISO 639	ISO 639–2
卡约语		gay
古兹语		gez
格鲁吉亚语	ka	geo/kat
德语	de	deu/ger
中古高地德语（约 1050 年～ 1500 年）		gmh
古高地德语（约 750 年～ 1050 年）		goh
其他德语语系语言		gem
吉尔伯特语		gil
贡德语		gon
哥特语		got
格列博语		grb
古希腊语（至 1453 年）		grc
现代希腊语（1453 年至今）	el	ell/gre
格陵兰语	kl	kal
瓜拉尼语	gn	grn
古吉拉特语	gu	guj
海达语		hai
豪萨语	ha	hau
夏威夷语		haw
希伯来语	he	heb
赫雷罗语		her
希利盖农语		hil
喜马偕尔语		him
印地语	hi	hin
希里莫图语		hmo
匈牙利语	hu	hun
胡帕语		hup
伊班语		iba
冰岛语	is	ice/isl
伊博语		ibo

（续）

语　　言	ISO 639	ISO 639–2
伊乔语		ijo
伊洛卡诺语		ilo
其他印度 - 雅利安语系		inc
其他印欧语系语言		ine
印度尼西亚语	id	ind
国际语	ia	ina
西方国际语	ie	ine
因纽特语	iu	iku
伊努皮亚克语	ik	ipk
其他伊朗语系语言		ira
爱尔兰语	ga	gai/iri
古爱尔兰语（至 900 年）		sga
中古爱尔兰语（900 年～ 1200 年）		mga
伊洛魁语系语言		iro
意大利语	it	ita
日语	ja	jpn
爪哇语	jv/jw	jav/jaw
犹太阿拉伯语		jrb
犹太波斯语		jpr
卡布列语		kab
景颇语		kac
坎巴语		kam
卡纳达语	kn	kan
卡努里语		kau
卡拉－卡尔帕克语		kaa
克伦语		kar
克什米尔语	ks	kas
卡威语		kaw
哈萨克语	kk	kaz
卡西语		kha
高棉语	km	khm

（续）

语　言	ISO 639	ISO 639–2
其他科依桑语系语言		khi
和阗语		kho
基库尤语		kik
卢旺达语	rw	kin
柯尔克孜语	ky	kir
科米语		kom
刚果语		kon
孔卡尼语		kok
韩语	ko	kor
克佩勒语		kpe
克鲁语		kro
宽亚玛语		kua
库梅克语		kum
库尔德语	ku	kur
库鲁克语		kru
克萨语		kus
库特内语		kut
拉迪诺语		lad
拉亨达语		lah
兰巴语		lam
奥克语（公元前 1500 年）	oc	oci
老挝语	lo	lao
拉丁语	la	lat
拉脱维亚语	lv	lav
卢森堡语		ltz
列兹基语		lez
林加拉语	ln	lin
立陶宛语	lt	lit
洛齐语		loz
卢巴－卡丹加语		lub
卢伊塞诺语		lui

588

（续）

语　言	ISO 639	ISO 639–2
隆达语		lun
卢奥语（肯尼亚与坦桑比亚）		luo
马其顿语	mk	mac/mak
马都拉语		mad
摩揭陀语		mag
迈蒂利语		mai
旺加锡语		mak
马达加斯语	mg	mlg
马来语	ms	may/msa
马拉雅拉姆语		mal
马耳他语	ml	mlt
曼丁哥语		man
曼尼普尔语		mni
马诺博诸语		mno
曼岛语		max
毛利语	mi	mao/mri
马拉地语	mr	mar
马里语		chm
马绍尔语		mah
马尔瓦利语		mwr
马萨伊语		mas
玛雅语		myn
门德语		men
密克玛克语		mic
米南卡保语		min
多种其他语言		mis
莫霍克语		moh
摩尔达维亚语	mo	mol
其他孟高棉语族		mkh
芒戈语		lol
蒙古语	mn	mon

（续）

语　　言	ISO 639	ISO 639–2
默希语		mos
多语种		mul
蒙达语族		mun
瑙鲁语	na	nau
纳瓦霍语		nav
北恩德贝勒语		nde
南恩德贝勒语		nbl
恩敦加语		ndo
尼泊尔语	ne	nep
尼瓦里语		new
其他尼日尔 – 科尔多凡语系语言		nic
其他尼罗 – 撒哈拉语系		ssa
纽埃语		niu
古诺尔斯语		non
其他北美印第安语族语言		nai
挪威语	no	nor
挪威布克莫尔语		nno
努比亚语族		nub
尼扬韦齐语		nym
尼扬贾语		nya
尼扬科勒语		nyn
尼奥罗语		nyo
恩济马语		nzi
奥吉布瓦语		oji
奥利亚语	or	ori
奥罗莫语	om	orm
奥萨格语		osa
奥塞梯语		oss
奥托米安语系		oto
钵罗维语		pal
帕劳语		pau

（续）

语　　言	ISO 639	ISO 639–2
巴利语		pli
邦板牙语		pam
邦阿西楠语		pag
旁遮普语	pa	pan
帕皮亚门托语		pap
其他巴布亚 - 澳洲语系语言		paa
波斯语	fa	fas/per
古波斯语（大约公元前 600 年～ 400 年）		peo
腓尼基语		phn
波兰语	pl	pol
波纳佩语		pon
葡萄牙语	pt	por
普拉克里特语族		pra
古普罗旺斯语（至 1500 年）		pro
普什图语	ps	pus
克丘亚语	qu	que
里托罗曼斯语	rm	roh
拉贾斯坦语		raj
拉罗汤加语		rar
其他罗曼语系语言		roa
罗马尼亚语	ro	ron/rum
罗姆语		rom
基隆迪语	rn	run
俄语	ru	rus
萨利什语系		sal
萨马利亚阿拉米语		sam
萨米语系语言		smi
萨摩亚语	sm	smo
桑达韦语		sad
桑戈语	sg	sag

（续）

语　言	ISO 639	ISO 639–2
梵语	sa	san
萨丁尼亚语		srd
苏格兰语		sco
塞尔库普语		sel
其他闪米特语系语言		sem
塞尔维亚语	sr	
塞尔维亚 - 克罗地亚语	sh	scr
塞雷尔语		srr
掸语		shn
修纳语	sn	sna
锡达莫语		sid
西克西卡语		bla
信德语	sd	snd
僧伽罗语	si	sin
其他汉藏语系语言		sit
苏语系语言		sio
其他斯拉夫语族		sla
西斯旺语	ss	ssw
斯洛伐克语	sk	slk/slo
斯洛文尼亚语	sl	slv
粟特语		sog
索马里语	so	som
桑海语		son
索布语系语言		wen
北索托语		nso
南索托语	st	sot
其他南美印第安语族语言		sai
西班牙语	es	esl/spa
苏库马语		suk
苏美尔语		sux
苏丹语	su	sun

（续）

语　言	ISO 639	ISO 639–2
苏苏语		sus
斯瓦西里语	sw	swa
斯沃兹语		ssw
瑞典语	sv	sve/swe
叙利亚语		syr
塔加洛语	tl	tgl
塔希提语		tah
塔吉克语	tg	tgk
塔马舍克语		tmh
泰米尔语	ta	tam
塔塔尔语	tt	tat
泰卢固语	te	tel
特列纳语		ter
泰语	th	tha
藏语	bo	bod/tib
提格雷语		tig
提格里尼亚语	ti	tir
滕内语		tem
蒂夫语		tiv
特林吉特语		tli
汤加语（尼亚萨）	to	tog
东加语（汤加岛）		ton
特鲁克语		tru
钦西安语		tsi
宗加语	ts	tso
茨瓦纳语	tn	tsn
奇图姆布卡语		tum
土耳其语	tr	tur
奥斯曼土耳其语（约 1500 年～ 1928 年）		ota
土库曼语	tk	tuk
图瓦语		tyv

592

（续）

语　　言	ISO 639	ISO 639–2
特威语	tw	twi
乌加里特语		uga
维吾尔语	ug	uig
乌克兰语	uk	ukr
翁本杜语		umb
未确定的语言		und
乌尔都语	ur	urd
乌兹别克语	uz	uzb
瓦伊语		vai
文达语		ven
越南语	vi	vie
沃拉普克语	vo	vol
沃提克语		vot
瓦卡什语系语言		wak
瓦拉莫语		wal
瓦赖语		war
瓦肖语		was
威尔士语	cy	cym/wel
沃洛夫语	wo	wol
科萨语	xh	xho
雅库特语		sah
瑶语		yao
雅浦语		yap
依地语	yi	yid
约鲁巴语	yo	yor
萨波特克语		zap
哲纳加语		zen
壮语	za	zha
祖鲁语	zu	zul
祖尼语		zun

G.5 ISO 3166国家代码

表G-3 ISO 3166国家代码

国　家	代　码
阿富汗	AF
阿尔巴尼亚	AL
阿尔及利亚	DZ
美属萨摩亚	AS
安道尔	AD
安哥拉	AO
安圭拉	AI
南极洲	AQ
安提瓜和巴布达	AG
阿根廷	AR
亚美尼亚	AM
阿鲁巴	AW
澳大利亚	AU
奥地利	AT
阿塞拜疆	AZ
巴哈马	BS
巴林	BH
孟加拉	BD
巴巴多斯	BB
白俄罗斯	BY
比利时	BE
伯利兹	BZ
贝宁	BJ
百慕大	BM
不丹	BT
玻利维亚	BO
波黑	BA
博茨瓦纳	BW
布韦岛	BV

（续）

国　家	代　码
巴西	BR
英属印度洋领地	IO
文莱达鲁萨兰国	BN
保加利亚	BG
布基纳法索	BF
布隆迪	BI
柬埔寨	KH
喀麦隆	CM
加拿大	CA
佛得角	CV
开曼群岛	KY
中非共和国	CF
乍得	TD
智利	CL
中国	CN
圣诞岛	CX
科科斯（基林群岛）	CC
哥伦比亚	CO
科摩罗	KM
刚果	CG
刚果（金共和国）	CD
库克群岛	CK
哥斯达黎加	CR
科特迪瓦	CI
克罗地亚	HR
古巴	CU
塞浦路斯	CY
捷克	CZ
丹麦	DK
吉布提	DJ
多米尼克	DM

595

（续）

国　家	代　码
多米尼加共和国	DO
东帝汶	TP
厄瓜多尔	EC
埃及	EG
萨尔瓦多	SV
赤道几内亚	GQ
厄立特里亚	ER
爱沙尼亚	EE
埃塞俄比亚	ET
福克兰群岛（马尔维纳斯）	FK
法罗群岛	FO
斐济群岛	FJ
芬兰	FI
法国	FR
法属圭亚那	GF
法属波利尼西亚	PF
法国南部领地	TF
加蓬	GA
冈比亚	GM
格鲁吉亚	GE
德国	DE
加纳	GH
直布罗陀	GI
希腊	GR
格陵兰	GL
格林纳达	GD
瓜德罗普	GP
关岛	GU
危地马拉	GT
几内亚	GN
几内亚比绍	GW

（续）

国　家	代　码
圭亚那	GY
596 海地	HT
赫德岛和麦克唐纳群岛	HM
梵蒂冈	VA
洪都拉斯	HN
匈牙利	HU
冰岛	IS
印度	IN
印度尼西亚	ID
伊朗（伊斯兰共和国）	IR
伊拉克	IQ
爱尔兰	IE
以色列	IL
意大利	IT
牙买加	JM
日本	JP
约旦	JO
哈萨克斯坦	KZ
肯尼亚	KE
基里巴斯	KI
朝鲜（民主主义人民共和国）	KP
韩国（共和国）	KR
科威特	KW
吉尔吉斯斯坦	KG
老挝人民民主共和国	LA
拉脱维亚	LV
黎巴嫩	LB
莱索托	LS
利比里亚	LR
阿拉伯利比亚民众国	LY

（续）

国　家	代　码
列支敦士登	LI
立陶宛	LT
卢森堡	LU
马其顿（前南斯拉夫共和国）	MK
马达加斯加	MG
马拉维	MW
马来西亚	MY
马尔代夫	MV
马里	ML
马耳他	MT
马绍尔群岛	MH
马提尼克	MQ
毛里塔尼亚	MR
毛里求斯	MU
马约特	YT
墨西哥	MX
密克罗尼西亚（联邦）	FM
摩尔多瓦（共和国）	MD
摩纳哥	MC
蒙古国	MN
蒙塞拉特岛	MS
摩洛哥	MA
莫桑比克	MZ
缅甸	MM
纳米比亚	NA
瑙鲁	NR
尼泊尔	NP
荷兰	NL
荷属安的列斯	AN
新喀里多尼亚	NC
新西兰	NZ

（续）

国　家	代　码
尼加拉瓜	NI
尼日尔	NE
尼日利亚	NG
纽埃	NU
诺福克岛	NF
北马里亚那群岛	MP
挪威	NO
阿曼	OM
巴基斯坦	PK
帕劳	PW
巴勒斯坦领土（被占）	PS
巴拿马	PA
巴布亚新几内亚	PG
巴拉圭	PY
秘鲁	PE
菲律宾	PH
皮特凯恩	PN
波兰	PL
葡萄牙	PT
波多黎各	PR
卡塔尔	QA
留尼旺	RE
罗马尼亚	RO
俄罗斯	RU
卢旺达	RW
圣赫勒拿	SH
圣基斯和尼维斯	KN
圣卢西亚	LC
圣皮埃尔和密克隆	PM
圣文森特和格林纳丁斯	VC
萨摩亚	WS

598

（续）

国　家	代　码
圣马力诺	SM
圣多美和普林西比	ST
沙特阿拉伯	SA
塞内加尔	SN
塞舌尔	SC
塞拉利昂	SL
新加坡	SG
斯洛伐克	SK
斯洛文尼亚	SI
所罗门群岛	SB
索马里	SO
南非	ZA
南乔治亚岛和南桑威奇群岛	GS
西班牙	ES
斯里兰卡	LK
苏丹	SD
苏里南	SR
斯瓦尔巴群岛和扬马延岛	SJ
斯威士兰	SZ
瑞典	SE
瑞士	CH
阿拉伯叙利亚共和国	SY
塔吉克斯坦	TJ
坦桑尼亚（联合共和国）	TZ
泰国	TH
多哥	TG
托克劳	TK
汤加	TO
特立尼达和多巴哥	TT
突尼斯	TN
土耳其	TR

（续）

国　家	代　码
土库曼斯坦	TM
特克斯和凯科斯群岛	TC
图瓦卢	TV
乌干达	UG
乌克兰	UA
阿拉伯联合酋长国	AE
英国	GB
美国	US
美国本土外小岛屿	UM
乌拉圭	UY
乌兹别克斯坦	UZ
瓦努阿图	VU
委内瑞拉	VE
越南	VN
维尔京群岛（英属）	VG
维尔京群岛（美属）	VI
瓦利斯和富图纳	WF
西撒哈拉	EH
也门	YE
南斯拉夫	YU
赞比亚	ZM

600

G.6　语言管理组织

- ISO 639 指定了一个向 ISO 639 的语言列表中进行添加或者对其进行修改的维护机构。此机构为：
  International Information Centre for Terminology（Infoterm，国际术语信息中心）
  P.O. Box 130
  A-1021 Wien
  Austria
  电话：+43 1 26 75 35 分机号：312
  传真：+43 1 216 32 72

- ISO 639-2 指定了一个向 ISO 639-2 的语言列表中进行添加或者对其进行修改的维护机构。此机构为：
  Library of Congress（美国国会图书馆）
  Network Development and MARC Standards Office（网络开发与 MARC 标准办公室）
  Washington, D.C. 20540
  USA
  电话：+1 202 707 6237
  传真：+1 202 707 0115
  URL：http://www.loc.gov/standards/iso639-2/
- ISO 3166（国家代码）的维护机构为：
  ISO 3166 Maintenance Agency Secretariat（ ISO 3166 维护机构秘书处）
  c/o DIN Deutsches Institut fuer Normung（ c/o DIN 德国标准化研究所）
  Burggrafenstrasse 6
  Postfach 1107
  D-10787 Berlin
  Germany
  电话：+49 30 26 01 320
  传真：+49 30 26 01 231
  URL：http://www.din.de/gremien/nas/nabd/iso3166ma/ 601

附录H

MIME字符集注册表

本附录描述了由 IANA 维护的 MIME 字符集注册表。表 H-1 给出了注册表中字符集的格式化表格。

H.1 MIME字符集注册表

MIME 字符集标记是由 IANA（http://www.iana.org/numbers.htm）注册的。字符集注册表是各项记录的平面文件文本数据库。每条记录中都包含一个字符集名称、参考引用、唯一的 MIB 号、一段资源描述和别名列表。名字或别名可能会被标识为“首选 MIME 名”。

下面是 US-ASCII 的记录：

```
Name: ANSI_X3.4-1968 [RFC1345, KXS2]^
MIBenum: 3
Source: ECMA registry
Alias: iso-ir-6
Alias: ANSI_X3.4-1986
Alias: ISO_646.irv:1991
Alias: ASCII
Alias: ISO646-US
Alias: US-ASCII (preferred MIME name)
Alias: us
Alias: IBM367
Alias: cp367
Alias: csASCII
```

602 RFC 2978（http://www.ietf.org/rfc/rfc2978.txt）记录了通过 IANA 注册字符集的过程。

H.2 首选MIME名

在编写本书时已注册的 235 个字符集中，只有 20 个包含了“首选 MIME 名”，即电子邮件和 Web 应用程序使用的常见字符集。其中包括：

Big5	EUC-JP	EUC-KR
GB2312	ISO-2022-JP	ISO-2022-JP-2
ISO-2022-KR	ISO-8859-1	ISO-8859-2
ISO-8859-3	ISO-8859-4	ISO-8859-5
ISO-8859-6	ISO-8859-7	ISO-8859-8
ISO-8859-9	ISO-8859-10	KOI8-R
Shift-JIS	US-ASCII	

H.3 已注册字符集

表 H-1 列出了 2001 年 3 月字符集注册表的内容。更多与此表内容有关的信息请直接参阅 http://www.iana.org。

表H-1 IANA MIME字符集标记

字符集标记	别 名	描 述	参考文献
US-ASCII	ANSI_X3.4-1968、iso-ir-6、ANSI_X3.4-1986、ISO_646.irv:1991、ASCII、ISO646-US、us、IBM367、cp367、csASCII	ECMA 注册表	RFC1345、KXS2
ISO-10646-UTF-1	csISO10646UTF1	通用传输格式（1）——这是以 ASCII-7 为子集的多字节编码；不存在字节顺序问题	
ISO_646.basic:1983	ref、csISO646basic1983	ECMA 注册表	RFC1345、KXS2
INVARIANT	csINVARIANT		RFC1345、KXS2
ISO_646.irv:1983	iso-ir-2、irv、csISO2IntlRefVersion	ECMA 注册表	RFC1345、KXS2
BS_4730	iso-ir-4、ISO646-GB、gb、uk、csISO4UnitedKingdom	ECMA 注册表	RFC1345、KXS2
NATS-SEFI	iso-ir-8-1、csNATSSEFI	ECMA 注册表	RFC1345、KXS2
NATS-SEFI-ADD	iso-ir-8-2、csNATSSEFIADD	ECMA 注册表	RFC1345、KXS2
NATS-DANO	iso-ir-9-1、csNATSDANO	ECMA 注册表	RFC1345、KXS2
NATS-DANO-ADD	iso-ir-9-2、csNATSDANOADD	ECMA 注册表	RFC1345、KXS2
SEN_850200_B	iso-ir-10、FI、ISO646-FI、ISO646-SE、se、csISO10Swedish	ECMA 注册表	RFC1345、KXS2
SEN_850200_C	iso-ir-11、ISO646-SE2、se2、csISO11SwedishForNames	ECMA 注册表	RFC1345、KXS2
KS_C_5601-1987	iso-ir-149、KS_C_5601-1989、KSC_5601、korean、csKSC56011987	ECMA 注册表	RFC1345、KXS2
ISO-2022-KR	csISO2022KR	RFC 1557（还可参见 KS_C_5601-1987）	RFC1557、Choi
EUC-KR	csEUCKR	RFC 1557（还可参见 KS_C_5861-1992）	RFC1557、Choi

603

（续）

字符集标记	别　　名	描　　述	参考文献
ISO-2022-JP	csISO2022JP	RFC 1468（还可参见 RFC 2237）	RFC1468、Murai
ISO-2022-JP-2	csISO2022JP2	RFC 1554	RFC1554、Ohta
ISO-2022-CN		RFC 1922	RFC1922
ISO-2022-CN-EXT		RFC 1922	RFC1922
JIS_C6220-1969-jp	JIS_C6220-1969、iso-ir-13、katakana、x0201-7、csISO13JISC6220jp	ECMA 注册表	RFC1345、KXS2
JIS_C6220-1969-ro	iso-ir-14、jp、ISO646-JP、csISO14JISC6220ro	ECMA 注册表	RFC1345、KXS2
IT	iso-ir-15、ISO646-IT、csISO15Italian	ECMA 注册表	RFC1345、KXS2
PT	iso-ir-16、ISO646-PT、csISO16Portuguese	ECMA 注册表	RFC1345、KXS2
ES	iso-ir-17、ISO646-ES、csISO17Spanish	ECMA 注册表	RFC1345、KXS2
greek7-old	iso-ir-18、csISO18Greek7Old	ECMA 注册表	RFC1345、KXS2
latin-greek	iso-ir-19、csISO19LatinGreek	ECMA 注册表	RFC1345、KXS2
DIN_66003	iso-ir-21、de、ISO646-DE、csISO21German	ECMA 注册表	RFC1345、KXS2
NF_Z_62-010_(1973)	iso-ir-25、ISO646-FR1、csISO25French	ECMA 注册表	RFC1345、KXS2
Latin-greek-1	iso-ir-27、csISO27LatinGreek1	ECMA 注册表	RFC1345、KXS2
ISO_5427	iso-ir-37、csISO5427Cyrillic	ECMA 注册表	RFC1345、KXS2
JIS_C6226-1978	iso-ir-42、csISO42JISC62261978	ECMA 注册表	RFC1345、KXS2
BS_viewdata	iso-ir-47、csISO47BSViewdata	ECMA 注册表	RFC1345、KXS2
INIS	iso-ir-49、csISO49INIS	ECMA 注册表	RFC1345、KXS2
INIS-8	iso-ir-50、csISO50INIS8	ECMA 注册表	RFC1345、KXS2
INIS-cyrillic	iso-ir-51、csISO51INISCyrillic	ECMA 注册表	RFC1345、KXS2
ISO_5427:1981	iso-ir-54、ISO5427Cyrillic1981	ECMA 注册表	RFC1345、KXS2
ISO_5428:1980	iso-ir-55、csISO5428Greek	ECMA 注册表	RFC1345、KXS2
GB_1988-80	iso-ir-57、cn、ISO646-CN、csISO57GB1988	ECMA 注册表	RFC1345、K5、KXS2

（续）

字符集标记	别　　名	描　　述	参考文献
GB_2312-80	iso-ir-58、chinese、csISO58GB231280	ECMA 注册表	RFC1345、KXS2
NS_4551-1	iso-ir-60、ISO646-NO、no、csISO60DanishNorwegian、csISO60Norwegian1	ECMA 注册表	RFC1345、KXS2
NS_4551-2	ISO646-NO2、iso-ir-61、no2、csISO61Norwegian2	ECMA 注册表	RFC1345、KXS2
NF_Z_62-010	iso-ir-69、ISO646-FR、fr、csISO69French	ECMA 注册表	RFC1345、KXS2
videotex-suppl	iso-ir-70、csISO70VideotexSupp1	ECMA 注册表	RFC1345、KXS2
PT2	iso-ir-84、ISO646-PT2、csISO84Portuguese2	ECMA 注册表	RFC1345、KXS2
ES2	iso-ir-85、ISO646-ES2、csISO85Spanish2	ECMA 注册表	RFC1345、KXS2
MSZ_7795.3	iso-ir-86、ISO646-HU、hu、csISO86Hungarian	ECMA 注册表	RFC1345、KXS2
JIS_C6226-1983	iso-ir-87、x0208、JIS_X0208-1983、csISO87JISX0208	ECMA 注册表	RFC1345、KXS2
greek7	iso-ir-88、csISO88Greek7	ECMA 注册表	RFC1345、KXS2
ASMO_449	ISO_9036、arabic7、iso-ir-89、csISO89ASMO449	ECMA 注册表	RFC1345、KXS2
iso-ir-90	csISO90	ECMA 注册表	RFC1345、KXS2
JIS_C6229-1984-a	iso-ir-91、jp-ocr-a、csISO91JISC62291984a	ECMA 注册表	RFC1345、KXS2
JIS_C6229-1984-b	iso-ir-92、ISO646-JP-OCR-B、jp-ocr-b、csISO92JISC62991984b	ECMA 注册表	RFC1345、KXS2
JIS_C6229-1984-b-add	iso-ir-93、jp-ocr-b-add、csISO93JIS62291984badd	ECMA 注册表	RFC1345、KXS2
JIS_C6229-1984-hand	iso-ir-94、jp-ocr-hand、csISO94JIS62291984hand	ECMA 注册表	RFC1345、KXS2
JIS_C6229-1984-hand-add	iso-ir-95、jp-ocr-hand-add、csISO95JIS62291984handadd	ECMA 注册表	RFC1345、KXS2
JIS_C6229-1984-kana	iso-ir-96、csISO96JISC62291984kana	ECMA 注册表	RFC1345、KXS2
ISO_2033-1983	iso-ir-98、e13b、csISO2033	ECMA 注册表	RFC1345、KXS2

（续）

字符集标记	别　　名	描　　述	参考文献
ANSI_X3.110-1983	iso-ir-99、CSA_T500-1983、NAPLPS、csISO99NAPLPS	ECMA 注册表	RFC1345、KXS2
ISO-8859-1	ISO_8859-1:1987、iso-ir-100、ISO_8859-1、latin1、l1、IBM819、CP819、csISOLatin1	ECMA 注册表	RFC1345、KXS2
ISO-8859-2	ISO_8859-2:1987、iso-ir-101、ISO_8859-2、latin2、l2、csISOLatin2	ECMA 注册表	RFC1345、KXS2
T.61-7bit	iso-ir-102、csISO102T617bit	ECMA 注册表	RFC1345、KXS2
T.61-8bit	T.61、iso-ir-103、csISO103T618bit	ECMA 注册表	RFC1345、KXS2
ISO-8859-3	ISO_8859-3:1988、iso-ir-109、ISO_8859-3、latin3、l3、csISOLatin3	ECMA 注册表	RFC1345、KXS2
ISO-8859-4	ISO_8859-4:1988、iso-ir-110、ISO_8859-4、latin4、l4、csISOLatin4	ECMA 注册表	RFC1345、KXS2
ECMA-cyrillic	iso-ir-111、csISO111ECMACyrillic	ECMA 注册表	RFC1345、KXS2
CSA_Z243.4-1985-1	iso-ir-121、ISO646-CA、csa7-1、ca、csISO121Canadian1	ECMA 注册表	RFC1345、KXS2
CSA_Z243.4-1985-2	iso-ir-122、ISO646-CA2、csa7-2、csISO122Canadian2	ECMA 注册表	RFC1345、KXS2
CSA_Z243.4-1985-gr	iso-ir-123、csISO123CSAZ24341985gr	ECMA 注册表	RFC1345、KXS2
ISO-8859-6	ISO_8859-6:1987、iso-ir-127、ISO_8859-6、ECMA-114、ASMO-708、arabic、csISOLatinArabic	ECMA 注册表	RFC1345、KXS2
ISO_8859-6-E	csISO88596E	RFC 1556	RFC1556、IANA
ISO_8859-6-I	csISO88596I	RFC 1556	RFC1556、IANA
ISO-8859-7	ISO_8859-7:1987、iso-ir-126、ISO_8859-7、ELOT_928、ECMA-118、greek、greek8、csISOLatinGreek	ECMA 注册表	RFC1947、RFC1345、KXS2
T.101-G2	iso-ir-128、csISO128T101G2	ECMA 注册表	RFC1345、KXS2
ISO-8859-8	ISO_8859-8:1988、iso-ir-138、ISO_8859-8、hebrew、csISOLatinHebrew	ECMA 注册表	RFC1345、KXS2

（续）

字符集标记	别　　名	描　　述	参考文献
ISO_8859-8-E	csISO88598E	RFC 1556	RFC1556、Nussbacher
ISO_8859-8-I	csISO88598I	RFC 1556	RFC1556、Nussbacher
CSN_369103	iso-ir-139、csISO139CSN369103	ECMA 注册表	RFC1345、KXS2
JUS_I.B1.002	iso-ir-141、ISO646-YU、js、yu、csISO141JUSIB1002	ECMA 注册表	RFC1345、KXS2
ISO_6937-2-add	iso-ir-142、csISOTextComm	ECMA 注册表及 ISO 6937-2:1983	RFC1345、KXS2
IEC_P27-1	iso-ir-143、csISO143IECP271	ECMA 注册表	RFC1345、KXS2
ISO-8859-5	ISO_8859-5:1988、iso-ir-144、ISO_8859-5、cyrillic、csISOLatinCyrillic	ECMA 注册表	RFC1345、KXS2
JUS_I.B1.003-serb	iso-ir-146、serbian、csISO146Serbian	ECMA 注册表	RFC1345、KXS2
JUS_I.B1.003-mac	macedonian、iso-ir-147、csISO147Macedonian	ECMA 注册表	RFC1345、KXS2
ISO-8859-9	ISO_8859-9:1989、iso-ir-148、ISO_8859-9、latin5、l5、csISOLatin5	ECMA 注册表	RFC1345、KXS2
greek-ccitt	iso-ir-150、csISO150、csISO150GreekCCITT	ECMA 注册表	RFC1345、KXS2
NC_NC00-10:81	cuba、iso-ir-151、ISO646-CU、csISO151Cuba	ECMA 注册表	RFC1345、KXS2
ISO_6937-2-25	iso-ir-152、csISO6937Add	ECMA 注册表	RFC1345、KXS2
GOST_19768-74	ST_SEV_358-88、iso-ir-153、csISO153GOST1976874	ECMA 注册表	RFC1345、KXS2
ISO_8859-supp	iso-ir-154、latin1-2-5、csISO8859Supp	ECMA 注册表	RFC1345、KXS2
ISO_10367-box	iso-ir-155、csISO10367Box	ECMA 注册表	RFC1345、KXS2
ISO-8859-10	iso-ir-157、l6、ISO_8859-10:1992、csISOLatin6、latin6	ECMA 注册表	RFC1345、KXS2
latin-lap	lap、iso-ir-158、csISO158Lap	ECMA 注册表	RFC1345、KXS2
JIS_X0212-1990	x0212、iso-ir-159、csISO159JISX02121990	ECMA 注册表	RFC1345、KXS2

（续）

字符集标记	别　名	描　述	参考文献
DS_2089	DS2089、ISO646-DK、dk、csISO646Danish	丹麦标准，DS 2089，1974 年 2 月	RFC1345、KXS2
us-dk	csUSDK		RFC1345、KXS2
dk-us	csDKUS		RFC1345、KXS2
JIS_X0201	X0201、csHalfWidthKatakana	JIS X 0201-1976—— 只有 1 个字节，等价于（与 ASCII 类似的）JIS/Roman 加上 8 位半角片假名	RFC1345、KXS2
KSC5636	ISO646-KR、csKSC5636		RFC1345、KXS2
ISO-10646-UCS-2	csUnicode	2 个 8 位字节的基本多语种平台，又名 Unicode—— 这需要指定网络字节序。标准中并未对其进行详细说明（它是个 16 位的整数空间）	
ISO-10646-UCS-4	csUCS4	完整的代码空间（与字节序的注释相同，是 31 位的数字）	
DEC-MCS	dec、csDECMCS	VAX/VMS 用户手册，订单号：AI-Y517A-TE，1986 年 4 月	RFC1345、KXS2
hp-roman8	roman8、r8、csHPRoman8	LaserJet IIP 打印机用户手册，惠普部件编号 33471-90901，惠普，1989 年 6 月	HP-PCL5、RFC1345、KXS2
macintosh	mac、csMacintosh	Unicode 标准 v1.0，ISBN 号 0201567881，1991 年 10 月	RFC1345、KXS2
IBM037	cp037、ebcdic-cp-us、ebcdic-cp-ca、ebcdic-cp-wt、ebcdic-cp-nl、csIBM037	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM038	EBCDIC-INT、cp038、csIBM038	IBM 3174 字符集参考文献，GA27-3831-02，1990 年 3 月	RFC1345、KXS2

607

（续）

字符集标记	别　　名	描　　述	参考文献
IBM273	CP273、csIBM273	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM274	EBCDIC-BE、CP274、csIBM274	IBM 3174 字符集参考文献，GA27-3831-02，1990 年 3 月	RFC1345、KXS2
IBM275	EBCDIC-BR、cp275、csIBM275	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM277	EBCDIC-CP-DK、EBCDIC-CP-NO、csIBM277	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM278	CP278、ebcdic-cp-fi、ebcdic-cp-se、csIBM278	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM280	CP280、ebcdic-cp-it、csIBM280	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM281	EBCDIC-JP-E、cp281、csIBM281	IBM 3174 字符集参考文献，GA27-3831-02，1990 年 3 月	RFC1345、KXS2
IBM284	CP284、ebcdic-cp-es、csIBM284	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM285	CP285、ebcdic-cp-gb、csIBM285	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM290	cp290、EBCDIC-JP-kana、csIBM290	IBM 3174 字符集参考文献、GA27-3831-02，1990 年 3 月	RFC1345、KXS2
IBM297	cp297、ebcdic-cp-fr、csIBM297	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM420	cp420、ebcdic-cp-ar1、csIBM420	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月，IBM NLS RM p 11-11	RFC1345、KXS2

（续）

字符集标记	别　　名	描　　述	参考文献
IBM423	cp423、ebcdic-cp-gr、csIBM423	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM424	cp424、ebcdic-cp-he、csIBM424	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM437	cp437、437、csPC8CodePage437	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM500	CP500、ebcdic-cp-be、ebcdic-cp-ch、csIBM500	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM775	cp775、csPC775Baltic	HP PCL 5 Comparison Guide（P/N 5021-0329）B-13 页，1996 年	HP-PCL5
IBM850	cp850、850、csPC850Multilingual	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM851	cp851、851、csIBM851	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM852	cp852、852、csPCp852	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM855	cp855、855、csIBM855	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM857	cp857、857、csIBM857	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM860	cp860、860、csIBM860	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM861	cp861、861、cp-is、csIBM861	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2

（续）

字符集标记	别　　名	描　　述	参考文献
IBM862	cp862、862、csPC862LatinHebrew	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM863	cp863、863、csIBM863	IBM 键盘布局和代码页，PN 07G4586，1991 年 6 月	RFC1345、KXS2
IBM864	cp864、csIBM864	IBM 键盘布局和代码页，PN 07G4586，1991 年 6 月	RFC1345、KXS2
IBM865	cp865、865、csIBM865	IBM DOS 3.3 参考文献（删节版），94X9575，1987 年 2 月	RFC1345、KXS2
IBM866	cp866、866、csIBM866	IBM NLDG 第二卷 SE09-8002-03，1994 年 8 月	Pond
IBM868	CP868、cp-ar、csIBM868	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM869	cp869、869、cp-gr、csIBM869	IBM 键盘布局和代码页，PN 07G4586，1991 年 6 月	RFC1345、KXS2
IBM870	CP870、ebcdic-cp-roece、ebcdic-cp-yu、csIBM870	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM871	CP871、ebcdic-cp-is、csIBM871	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM880	cp880、EBCDIC-Cyrillic、csIBM880	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM891	cp891、csIBM891	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM903	cp903、csIBM903	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2

（续）

字符集标记	别　　名	描　　述	参考文献
IBM904	cp904、904、csIBBM904	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM905	CP905、ebcdic-cp-tr、csIBM905	IBM 3174 字符集参考文献、GA27-3831-02，1990 年 3 月	RFC1345、KXS2
IBM918	CP918、ebcdic-cp-ar2、csIBM918	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
IBM1026	CP1026、csIBM1026	IBM NLS RM 第二卷 SE09-8002-01，1990 年 3 月	RFC1345、KXS2
EBCDIC-AT-DE	csIBMEBCDICATDE	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-AT-DE-A	csEBCDICATDEA	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-CA-FR	csEBCDICCAFR	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-DK-NO	csEBCDICDKNO	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-DK-NO-A	csEBCDICDKNOA	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-FI-SE	csEBCDICFISE	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-FI-SE-A	csEBCDICFISEA	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-FR	csEBCDICFR	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2

（续）

字符集标记	别　　名	描　　述	参考文献
EBCDIC-IT	csEBCDICIT	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-PT		IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-ES	csEBCDICES	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-ES-A	csEBCDICESA	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-ES-S	csEBCDICESS	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-UK	csEBCDICUK	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
EBCDIC-US	csEBCDICUS	IBM 3270 字符集参考文献第 10 章，GA27-2837-9，1987 年 4 月	RFC1345、KXS2
UNKNOWN-8BIT	csUnknown8BiT		RFC1428
MNEMONIC	csMnemonic	RFC 1345，也称作“mnemonic+ascii+38”	RFC1345、KXS2
MNEM	csMnem	RFC 1345，也称作“mnemonic+ascii+ 8200”	RFC1345、KXS2
VISCII	csVISCII	RFC 1456	RFC1456
VIQR	csVIQR	RFC 1456	RFC1456
KOI8-R	csKOI8R	RFC 1489，基于 GOST-19768-74、ISO-6937/8、INIS-Cyrillic、ISO-5427	RFC1489
KOI8-U		RFC 2319	RFC2319
IBM00858	CCSID00858、CP00858、PC-Multilingual-850+euro	IBM（参见 .../assignments/character-set-info/IBM00858）[Mahdi]	

（续）

字符集标记	别　　名	描　　述	参考文献
IBM00924	CCSID00924、CP00924、ebcdic-Latin9—euro	IBM（参见 .../assignments/character-set-info/IBM00924）[Mahdi]	
IBM01140	CCSID01140、CP01140、ebcdic-us-37+euro	IBM（参见 .../assignments/character-set-info/IBM01140）[Mahdi]	
IBM01141	CCSID01141、CP01141、ebcdic-de-273+euro	IBM（参见 .../assignments/character-set-info/IBM01141）[Mahdi]	
IBM01142	CCSID01142、CP01142、ebcdic-dk-277+euro、ebcdic-no-277+euro	IBM（参见 .../assignments/character-set-info/IBM01142）[Mahdi]	
IBM01143	CCSID01143、CP01143、ebcdic-fi-278+euro、ebcdic-se-278+euro	IBM（参见 .../assignments/character-set-info/IBM01143）[Mahdi]	
IBM01144	CCSID01144、CP01144、ebcdic-it-280+euro	IBM（参见 .../assignments/character-set-info/IBM01144）[Mahdi]	
IBM01145	CCSID01145、CP01145、ebcdic-es-284+euro	IBM（参见 .../assignments/character-set-info/IBM01145）[Mahdi]	
IBM01146	CCSID01146、CP01146、ebcdic-gb-285+euro	IBM（参见 .../assignments/character-set-info/IBM01146）[Mahdi]	
IBM01147	CCSID01147、CP01147、ebcdic-fr-297+euro	IBM（参见 .../assignments/character-set-info/IBM01147）[Mahdi]	
IBM01148	CCSID01148、CP01148、ebcdic-international-500+euro	IBM（参见 .../assignments/character-set-info/IBM01148）[Mahdi]	
IBM01149	CCSID01149、CP01149、ebcdic-is-871+euro	IBM（参见 .../assignments/character-set-info/IBM01149）[Mahdi]	
Big5-HKSCS	None	（参见 .../assignments/character-set-info/Big5-HKSCS）[Yick]	

611

（续）

字符集标记	别　　名	描　　述	参考文献
UNICODE-1-1	csUnicode11	RFC 1641	RFC1641
SCSU	None	SCSU（参见 .../assignments/character-set-info/SCSU）[Scherer]	
UTF-7	None	RFC 2152	RFC2152
UTF-16BE	None	RFC 2781	RFC2781
UTF-16LE	None	RFC 2781	RFC2781
UTF-16	None	RFC 2781	RFC2781
UNICODE-1-1-UTF-7	csUnicode11UTF7	RFC 1642	RFC1642
UTF-8		RFC 2279	RFC2279
iso-8859-13		ISO（参见 ...assignments/character-set-info/iso-8859-13）[Tumasonis]	
iso-8859-14	iso-ir-199、ISO_8859-14:1998、ISO_8859-14、latin8、iso-celtic、l8	ISO（参见 ...assignments/character-set-info/iso-8859-14）[Simonsen]	
ISO-8859-15	ISO_8859-15	ISO	
JIS_Encoding	csJISEncoding	JIS X 0202-1991；使用 ISO 2022 转义序列来切换代码集，如 JIS X 0202-1991 所述	
Shift_JIS	MS_Kanji、csShiftJIS	这个字符集是 csHalf WidthKatakana 的扩展——它在 JIS X 0208 中添加了图像字符。CCS 是 JIS X0201:1997 和 JIS X0208: 1997。JISX0208: 1997 的附录 1 给出了完整的定义。可以将此字符集用于顶级媒体类型“text”	

（续）

字符集标记	别　名	描　述	参考文献
EUC-JP	Extended_UNIX_Code_Packed_Format_for_Japanese、csEUCPkdFmtJapanese	由 OSF、UNIX 国际和 UNIX 系统太平洋实验室制定的标准。使用 ISO 2022 规则来挑选代码集。代码集 0：US-ASCII（单 7 位的字节集）；代码集 1：JIS X0208-1990（双 8 位的字节集），这两个字节都限制在 A0 ~ FF 之间；代码集 2：半角片假名（单 7 位的字节集）需要将 SS2 作为字符前缀；代码集 3：JIS X0212-1990（双 7 位的字节集），两个字节都要限制在 A0 ~ FF 之间，需要将 SS3 作为字符前缀	
Extended_UNIX_Code_Fixed_Width_for_Japanese	csEUCFixWidJapanese	在日本使用。每个字符都是 2 个字节。代码集 0：US-ASCII（单 7 位的字节集），第一个字节 00，第二个字节 20 ~ 7E；代码集 1：JIS X0208-1990（双 7 位的字节集），两个字节都限制在 A0 ~ FF 之间；代码集 2：半角片假名（单 7 位的字节集），第一个字节 00，第二个字节 A0 ~ FF；代码集 3：JIS X0212-1990（双 7 位字节集），第一个字节限制在 A0 ~ FF 之间，第二个字节限制在 21 ~ 7E 之间	
ISO-10646-UCS-Basic	csUnicodeASCII	Unicode 的 ASCII 子集。基本拉丁字母为集 1。参见 ISO 10646，附录 A	
ISO-10646-Unicode-Latin1	csUnicodeLatin1、ISO-10646	Unicode 的 ISO Latin-1 子集。基本拉丁字母和 Latin-1。增补为集 1 及集 2。参见 ISO 10646，附录 A 以及 RFC 1815	

612

（续）

字符集标记	别　名	描　述	参考文献
ISO-10646-J-1		ISO 10646 日语。参见 RFC 1815	
ISO-Unicode-IBM-1261	csUnicodeIBM1261	IBM Latin-2、-3、-5、扩展表示集，GCSGID: 1261	
ISO-Unicode-IBM-1268	csUnidoceIBM1268	IBM Latin-4 扩展表示集，GCSGID: 1268	
ISO-Unicode-IBM-1276	csUnicodeIBM1276	IBM 西里尔希腊语扩展表示集，GCSGID: 1276	
ISO-Unicode-IBM-1264	csUnicodeIBM1264	IBM 阿拉伯语扩展表示集，GCSGID: 1264	
ISO-Unicode-IBM-1265	csUnicodeIBM1265	IBM 希伯来语扩展表示集，GCSGID: 1265	
ISO-8859-1-Windows-3.0-Latin-1	csWindows30Latin1	用于 Windows 3.0 的扩展 ISO 8859-1 Latin-1。PCL 符号集 ID：9U	HP-PCL5
ISO-8859-1-Windows-3.1-Latin-1	csWindows31Latin1	用于 Windows 3.1 的扩展 ISO 8859-1 Latin-1。PCL 符号集 ID：19U	HP-PCL5
ISO-8859-2-Windows-Latin-2	csWindows31Latin2	用于 Windows 3.1 的扩展 ISO 8859-2 Latin-2。PCL 符号集 ID：9E	HP-PCL5
ISO-8859-9-Windows-Latin-5	csWindows31Latin5	用于 Windows 3.1 的扩展 ISO 8859-9 Latin-5。PCL 符号集 ID：5T	HP-PCL5
Adobe-Standard-Encoding	csAdobeStandardEncoding	PostScript 语言参考手册。PCL 符号集 ID：10J	Adobe
Ventura-US	csVenturaUS	Ventura US-ASCII plus 字符通常用于出版业，比如在 A0 ~ FF 范围之间的段落标记、版权、注册、商标、分节、剑号及双剑号。PCL 符号集 ID：14J	HP-PCL5

（续）

字符集标记	别　　名	描　　述	参考文献
Ventura-International	csVenturaInternational	Ventura International ASCII plus 编码字符与 Roman8 类似。PCL 符号集 ID：13J	HP-PCL5
PC8-Danish-Norwegian	csPC8DanishNorwegian	丹麦挪威使用的 PC 丹麦挪威 8 位 PC 集。PCL 符号集 ID：11U	HP-PCL5
PC8-Turkish	csPC8Turkish	PC 拉丁土耳其。PCL 符号集：9T	HP-PCL5
IBM-Symbols	csIBMSymbols	表示集，CPGID: 259	IBM-CIDT
IBM-Thai	csIBMThai	表示集，CPGID: 838	IBM-CIDT
HP-Legal	csHPLegal	PCL 5 Comparison Guide，惠普，惠普部件编号 5961-0510，1992 年 10 月。PCL 符号集 ID：1U	HP-PCL5
HP-Pi-font	csHPPiFont	PCL 5 Comparison Guide，惠普，惠普部件编号 5961-0510，1992 年 10 月。PCL 符号集 ID：15U	HP-PCL5
HP-Math8	csHPMath8	PCL 5 Comparison Guide，惠普，惠普部件编号 5961-0510，1992 年 10 月。PCL 符号集 ID：8M	HP-PCL5
Adobe-Symbol-Encoding	csHPPSMath	PostScript 语言参考手册。PCL 符号集 ID：5M	Adobe
HP-DeskTop	csHPDesktop	PCL 5 Comparison Guide，惠普，惠普部件编号 5961-0510，1992 年 10 月。PCL 符号集 ID：7J	HP-PCL5
Ventura-Math	csVenturaMath	PCL 5 Comparison Guide，惠普，惠普部件编号 5961-0510，1992 年 10 月。PCL 符号集 ID：6M	HP-PCL5
Microsoft-Publishing	csMicrosoftPublishing	PCL 5 Comparison Guide，惠普，惠普部件编号 5961-0510，1992 年 10 月。PCL 符号集 ID：6J	HP-PCL5

（续）

字符集标记	别　名	描　述	参考文献
Windows-31J	csWindows31J	日语版 Windows。是对 Shift_JIS 的进一步扩展，以包含 NEC 特殊字符（第 13 行）、NEC 选择的 IBM 扩展（第 89 ～ 92 行）、以及 IBM 扩展（第 115 ～ 119 行）。CCS 包含 JIS X0201:1997、JIS X0208:1997 和这些扩展。可将此字符集用于顶级媒体类型“text”，但用途有限，或仅用于特殊目的（参见 RFC 2278）。PCL 字符集 ID：19K	
GB2312	csGB2312	混合了 1 字节、2 字节集的中华人民共和国（PRC）汉语：20-7E 为 1 字节 ASCII；A1-FE 为 2 字节 PRC 汉字。参见 GB 2312-80。PCL 符号集 ID：18C	
Big5	csBig5	中国台湾省使用的汉语多字节集。PCL 符号集 id：18T	
windows-1250		微软（参见 .../character-set-info/windows-1250）[Lazhintseva]	
windows-1251		微软（参见 .../character-set-info/windows-1251）[Lazhintseva]	
windows-1252		微软（参见 .../character-set-info/windows-1252）[Wendt]	
windows-1253		微软（参见 .../character-set-info/windows-1253）[Lazhintseva]	
windows-1254		微软（参见 .../character-set-info/windows-1254）[Lazhintseva]	

614

（续）

字符集标记	别　名	描　述	参考文献
windows-1255		微软（参见 .../character-set-info/windows-1255）[Lazhintseva]	
windows-1256		微软（参见 .../character-set-info/windows-1256）[Lazhintseva]	
windows-1257		微软（参见 .../character-set-info/windows-1257）[Lazhintseva]	
windows-1258		微软（参见 .../character-set-info/windows-1258）[Lazhintseva]	
TIS-620		TISI，Thai Industrial Standards Institute（泰国工业标准协会）	[Tantsetthi]
HZ-GB-2312		RFC 1842、RFC 1843 [RFC1842、RFC1843]	

615

索引

索引中的页码为英文原书的页码，与本书页边标注的页码一致。

符号

: (colon)（冒号），use in headers（用在首部），47
= (equals sign)（等号），Base-64 encoding（Base-64 编码），572
/~ (slash-tilde)（斜线 - 波浪线），122

数字

8-bit identity encoding（8 位恒等编码），382
100 Continue status code（100 Continue 状态码），59，60
100-199 status codes（100 ～ 199 状态码），59 ～ 60，505
200-299 status codes（200 ～ 299 状态码），61，505
300-399 status codes（300 ～ 399 状态码），61 ～ 64，506
400-499 status codes（400 ～ 499 状态码），65 ～ 66，506
500-599 status codes（500 ～ 599 状态码），66，507
2MSL(maximum segment lifetime)（最大分段生存期），85

A

Absolute URLs（绝对 URL），30
Accept headers（`Accept` 首部），69，508
  robots（机器人），225
Accept-Charset headers（`Accept-Charset` 首部），371，375，509
  MIME charset encoding tags（MIME 字符集编码标记），374
Accept-Encoding headers（`Accept-Encoding` 首部），509
Accept-Instance-Manipulation headers（`Accept-Instance-Manipulation` 首部），367
Accept-Language headers（`Accept-Language` 首部），371，385，510
  content negotiation（内容协商），398
Accept-Ranges headers（`Accept-Ranges` 首部），510
access controls（访问控制），124
  proxy authentication（代理认证），156
access proxies（访问代理），137
advertising（广播），hit counts and caches（命中计数与缓存），194 ～ 196
age and freshness lifetime（使用期和新鲜生存期），188
Age headers（`Age` 首部），510
agents（Agent 代理），19
algorithms（算法）
  aging and freshness（老化及新鲜度），187 ～ 194
  document age calculation（文档使用期的计算），189 ～ 194
  instance-manipulation algorithms（实例操控算法），367
  LM-Factor，184
  message digest algorithms（报文摘要算法），291 ～ 294
    symmetric authentication（对称认证），298
  Nagle's algorithm（Nagle 算法），84
  redirection（重定向），enhanced DNS-based（基于 DNS 增强的），457
  resource-discovery algorithm (WPAD)（资源发现算法），143，465
  RSA，317
aliases (URLs)（别名），219
Allow headers（`Allow` 首部），159，511
<allprop> element（`<allprop>` 元素），437
anonymizers（匿名者），136
anycast addressing（任播寻址），457
Apache web servers（Apache Web 服务器），110
  content negotiation（内容协商），399
    MultiViews directive（MultiView 指令），400
    type-map files（类型映射文件），399
  DirectoryIndex configuration directive（DirectoryIndex 配置指令），123
  document root（文档根），setting（服务器设置），121
  HostnameLookups configuration directive（HostnameLookups 配置指令），115
  HTTP headers（HTTP 首部），control of（控制），186

IdentityCheck configuration directive（IdentityCheck 配置指令），116
magic typing（魔法分类），126
APIs (application programming interfaces)（应用编程接口），203
server extensions（服务器扩展），205
web services（Web 服务器），205
application/* MIME types（application/* MIME 类型），540 ～ 557
application programming interfaces (see APIs)（应用编程接口，参见 API）
application servers（应用服务器），123，203
ASCII character set（ASCII 字符集），379
asymmetric cryptography（对称加密），315
attacks（攻击），303 ～ 306
batched brute-force attacks（批量暴力型攻击），305
chosen plaintext attacks（选择明文攻击），305
dictionary attacks（词典攻击），304
enumeration（枚举），313
evidence of（证据），301
header tampering（首部篡改），303
hostile proxies（恶意代理），304
man-in-the-middle attacks（中间人攻击），304
replay attacks（重放攻击），284，303
preventing（防止），289
audio/* MIME types（audio/* MIME 类型），557 ～ 559
authentication（认证），277 ～ 280
basic (see basic authentication)［基本（参见基本认证）］
challenge/response framework（质询 / 响应框架），278
digest (see digest authentication) headers［摘要（参见摘要认证）首部］，278
HTCP，480
multiple authentication schemes（多重认证机制），risks of（认证风险），303
protocols（协议），278
proxy servers（代理服务器），156
server（服务器），using digital certificates（使用数字证书），321
(see also HTTPS)（还请参见 HTTPS）
Authentication-Info directives（Authentication-Info 指令），576
Authorization headers（Authorization 首部），281，511
directives（指令），575
preemptive generation（预先产生），295
automatic expansion of URLs（URL 的自动扩展），30

B

bandwidth（带宽）
bottlenecks（瓶颈），161
transfer times（传输时间），162
base URLs（基本 URL），32
base-64 encoding（Base-64 编码），570 ～ 572
alphabet（字母表），571
equals sign (=)（等号），572
HTTP，compatibility with（兼容于），570
padding（填充），572
Perl implementation（Perl 实现），572
purpose（目标），570
username/password（用户名 / 密码），282
bases（基础），31
basic authentication（基本认证），281 ～ 284
example（举例），281
headers（首部），281，300
insecurity of（不安全），283，286
protection space（保护空间），302
by proxy servers（通过代理服务器），283
username/password encoding（用户名 / 密码的编码），282
web server vs. proxy（Web 服务器与代理），283
(see also authentication)（还请参见认证）
batched brute-force attacks（批量暴力型攻击），305

Binary Wire Protocol（二进制连接协议），250，252
blind relays（盲中继），94
browsers（浏览器）
  Host headers and older versions of（Host 首部与较老版本），419
  HTTP，use of（使用），13
  parallel connections（并行连接），maximum（最大），90
  URLs（URL），automatic expansion of（自动扩展），34
byte hit rate（字节命中率），167

C

Cache Array Routing Protocol (see CARP)（缓存阵列路由协议，参见 CARP）
Cache-Control headers（Cache-Control 首部），175 ~ 177，182 ~ 186，189，511
  directives（指令），361
caches（缓存），18，133，161 ~ 196
  advertising and（广播），194 ~ 196
  aging and freshness algorithms（老化及新鲜度算法），187 ~ 194
  byte hit rate（字节命中率），167
  cache busting（缓存清除），492
  cache hits and misses（缓存命中及未命中），165，168
  cache meshes（网状缓存），170
  cache validators（缓存验证器），181
  Cache-Control headers（Cache-Control 首部），175 ~ 177，182 ~ 186，189，511
    directives（指令），361
  cookies（cookie），273
  digest authentication（摘要认证），302
  distance delays（距离时延），163
  DNS caching and load balancing（DNS 缓存及负载均衡），456
  document expiration（文档过期），175
  exclusion of documents from（从中排除文档），182
  expiration time（过期时间），setting（设置），182
  Expires headers（Expires 首部），175，183
  flash crowds and（瞬间拥塞），163
  freshness check（新鲜度检查），173
  heuristic expiration（试探性过期），184
  hit logs（命中日志），195
  hit rate（命中率），167
  HTTP-EQUIV tag（HTTP-EQUIV 标签），187
  logging（日志），174
  lookup（查找），173
  maintaining currency（维护流通性），175
  message truncation（报文截尾），344
  network bottlenecks（网络瓶颈），161
  parent caches（父缓存），169
  parsing（解析），172
  peering（对等），171
  processing steps（处理步骤），171 ~ 175
    flowchart（流程图），175
  receiving（接收），172
  response creation（创建响应），174
  revalidate hits（再验证命中），165
  revalidations（再验证），165 ~ 166
  sending（发送），174
  server revalidation（服务器再验证），175，177
  setting controls（设置控制），186
  sibling caches（兄弟缓存），171
  surrogate caches［替代物（反向代理）缓存］，421
  topologies（拓扑结构），168
  uses（使用），161
caching headers（缓存首部），68
caching proxies (see caches)（代理缓存）（参见缓存）

caching proxy servers（代理缓存服务器），169
canonicalizing of URLs（URL 的规范化），37，220
CARP (Cache Array Routing Protocol)（缓存阵列路由协议），475 ～ 478
  disadvantages（缺点），476
  ICP vs.，475
  redirection method（重定向方法），476
CAs (certificate authorities)（证书颁发机构），327
case sensitivity（区分大小写），language tags（语言标记），386
CDNs (content distribution networks)（内容分发网络），421
certificate authorities (CAs)（证书颁发机构），327
certificates (see digital certificates)（证书，参见数字证书）
CGI (Common Gateway Interface)（通用网关接口），203，204
challenge/response authentication model（质询 / 响应认证模块），278
  challenge headers（质询首部），72
  multiple challenges（多重质询），301
character encoding schemes（字符编码方案），377，381
character repertoire（字符库），377
character sets（字符集），35，371 ～ 376
  encoding（编码），370
    mechanisms（机制），36
  restricted characters（受限字符），36
characters（字符），378
  URLs（URL），legal in（合法），35
charset tags（字符集标记），371
  IANA MIME character set registry（IANA MIME 字符集注册表），371
chemical/* MIME types（chemical/* MIME 类型），559 ～ 561
child filters（儿童过滤器），131
chosen plaintext attacks（选择明文攻击），305
chunked encodings（分块编码），345
ciphers (see under cryptography)（密码，参见密码学条目下面的内容）
ciphertext（密文），310
cleartext（明文），310
client error status codes (400-499)［客户端错误状态码（400-499）］，65 ～ 66，505
client hostname identification（客户端主机名识别），115
client proxy configuration（客户端代理配置），141 ～ 144
  manual（手工），142
  PAC (Proxy Auto-configuration) protocol（代理自动配置）协议，142，463
  WPAD (Web Proxy Autodiscovery Protocol)（Web 代理自动发现协议），143
client-driven negotiation（客户端驱动的协商），396
  disadvantages（缺点），396
Client-ip headers（Client-ip 首部），258，260，512
clients（客户端）
  100 Continue status code（100 Continue 状态码），59
  freshness constraints（新鲜度限制），185
  Host header requirements（Host 首部需求），418
  identification（识别），257 ～ 276
    fat URLs（胖 URL），using for（用于），262
    IP addresses（IP 地址），using for（用于），259
    user logins（用户登录），using for（用于），260（see also cookies）（还请参见 cookie）
  supported character sets（支持的字符集），375
client-side gateways（客户端网关），199
  security accelerator gateways（安全加速器网关），202
client-side state（客户端状态），265
code width（代码宽度），377
coded character sets（编码后的字符集），377，379
coded characters（编码后的字符），376
coding space（代码空间），376
collections（集合），439
collisions（冲突），one-way digests（单向摘要），288
colon (:)（冒号），use in headers（用于首部），47
Combined Log Format（组合日志格式），485

Common Gateway Interface (CGI)（通用网关接口），203，204
Common Log Format（常见日志格式），484
composite MIME types（复合 MIME 类型），534
conditional requests（条件请求），362
  headers（首部），70，178 ~ 181
configuration URLs (CURLs)（配置 URL），465
CONNECT method（CONNECT 方法），206 ~ 208，336
connection handshake delays（连接握手时延），82
Connection headers（Connection 首部），86，512
connections (see HTTP connections)（连接，参见 HTTP 连接）
content distribution networks (CDN)（内容分发网络），421
content encodings（内容编码），345，351 ~ 354
content injection（内容注入），405
content negotiation（内容协商），395 ~ 403
  on Apache web servers（在 Apache Web 服务器上），399
  client-driven negotiation（客户端驱动的协商），396
  headers（首部），397
  other protocols（其他协议），405
  performance limitations（性能限制），405
  quality values（质量值），398
  server-driven negotiation（服务器驱动的协商），397 ~ 400
  techniques（技术），395
  transparent negotiation（透明协商），400 ~ 403
content routers（内容路由器），134，170
Content-Base headers（Content-Base 首部），513
Content-Encoding headers（Content-Encoding 首部），513
content-injection（内容注入），405
Content-Language headers（Content-Language 首部），371，384，513
Content-Length headers（Content-Length 首部），344 ~ 347，514
  content encoding（内容编码），345
  persistent connections（持久连接），345
Content-Location headers（Content-Location 首部），514
Content-MD5 headers（Content-MD5 首部），347，514
Content-Range headers（Content-Range 首部），515
Content-Type headers（Content-Type 首部），348 ~ 351，515
  character encodings（字符编码），349
  charset parameter（字符集参数），371
  META tags（META 标记），375
  MIME charset encoding tags（MIME 字符集编码标记），374
  multipart form submissions（多部分表格提交），349
continuation lines（延续行），in headers（位于首部），51
Continue status code (100)［Continue 状态码（100）］，59 ~ 60
Cookie headers（Cookie 首部），516
Cookie2 headers（Cookie2 首部），516
cookies，263 ~ 276
  browsers（浏览器），storage on（存储在），264
  caching（缓存），273
  domain attributes（域属性），267
  functioning（功能），264
  information contained in（包含在其中的信息），264
  Path attributes（Path 属性），268
  privacy（隐私），275
  security（安全性），275
  session tracking（会话跟踪），272
  Set-Cookie2 headers（Set-Cookie2 首部），271
  specifications（规范），268
  third-party vendors（第三方厂商），use by（所使用），267
  types（类型），264
  Version 0（版本 0），269
  Version 1（版本 1），270 ~ 272
    headers（首部），272
  version negotiation（版本协商），272

web site specificity of（Web 站点特有的），266
COPY method（COPY 方法），442
country codes（国家代码），388
country tokens（国家记号），388
crawlers（爬虫），215 ~ 224
aliasing（别名），219
canonicalizing of URLs（URL 的规范化），220
checkpoints（检查点），219
cycles（周期），avoiding（避免），217 ~ 218，222 ~ 224
dups（重复），218
filesystem link cycles（文件系统连接环路），220
hash tables（散列表），218
loops（循环），217
lossy presence bit maps（有损存在位图），218
partitioning（分区），219
root set（根集），216
search trees（搜索树），218
tracking of visited sites（受访站点的跟踪），218
traps（陷阱），220 ~ 224
CRLF，44
in entities（实体中的），343
cryptographic checksums（加密校验和），289
cryptography（密码学），309 ~ 317
asymmetrically keyed ciphers（对称密钥加密），315
cipher machines（密钥机制），311
ciphers（密钥），310 ~ 315
digital（数字），311
ciphertext（密文），310
cleartext（明文），310
enumeration attacks（枚举攻击），313
hybrid cryptosystems（混合加密系统），317
keyed ciphers（加密密钥），311
keys（密钥），311，312
key length（密钥长度），313
sharing（共享），logistical aspects of（逻辑方面的问题），315
public-key cryptography（公共密钥加密系统），315 ~ 317
computation speed（计算速度），317
digital signing with（数字签名），318
RSA algorithm（RSA 算法），317
symmetric-key ciphers（对称加密密钥），313
CURLs (configuration URLs)［CURL（配置 URL）］，465
cycles（周期），avoiding (web robots)［避免（Web 机器人）］，217 ~ 218，222 ~ 224
filesystem link cycles（文件系统连接环路），220

D

data formats（数据格式），conversion（转换），135
data formats（数据格式），392
Date headers（Date 首部），516
DAV headers（DAV 首部），431
compliance classes（遵从级别），445
decomposing of URLs（URL 的分解），33
dedicated web hosting（专用 Web 托管），412
delayed acknowledgements（延迟确认），83
DELETE method（DELETE 方法），58，441
delta encodings（差异编码），359，365 ~ 367
server disk space（服务器磁盘空间），368
delta generators and appliers（差异生成器和应用器），368
<depth> element（<depth> 元素），434
Depth headers（Depth 首部），431
Destination headers（Destination 首部），431
dictionary attacks（词典攻击），304

digcalc.c file（digcalc.c 文件），578
digcalc.h file（digcalc.h 文件），577
digest authentication（摘要认证），286 ~ 306，574-580
  algorithms（算法），291 ~ 295
    input data（输入数据），291
  authentication process（认证过程），287
  Authentication-Info directives（Authentication-Info 指令），576
  Authorization directives（Authorization 指令），575
  basic authentication（基本认证），compared to（与之比较），286
  caching（缓存），302
  digest calculations（摘要计算），291
  error handling（差错处理），301
  H(A1) and H(A2) reference code［H(A1) 和 H(A2) 参考代码］，577
  handshakes（握手），290
  headers（首部），300
  MD5 and MD5-sess（MD5 与 MD5-sess），291
  message-related data (A2)［报文相关的数据（A2）］，293，298
  nonces（随机数），289
  password files（密码文件），vulnerabilities of（脆弱性），305
  preemptive authorization（预授权），296
  protection space（保护空间），302
  request and response digest reference code（请求和响应摘要参考代码），577
  revalidating a session（对会话进行再验证），295
  rewriting URIs（重写 URI），302
  security（安全性），286
  security-related data (A1)［与安全有关的数据（A1）］，293
  session（会话），295
  symmetric authentication（对称认证），298
  WWW-Authenticate directives（WWW-Authenticate 指令），574 ~ 575
    (see also authentication)（还请参见认证）
digests（摘要），288
  algorithm input data（算法输入数据），291
  collisions（冲突），288
digital certificates（数字证书），319 ~ 322
  public-key cryptography（公共密钥加密），320
  server authentication（服务器认证），use for（用于），321
  universal standard（全球标准），lack of（缺乏），320
  virtual hosting（虚拟主机），328
  X.509v3 certificates（X.509v3 证书），320
digital cryptography (see cryptography)［数字密码（参见密码）］
digital signatures（数字签名），317 ~ 319
  example（举例），318
digtest.c file（digtest.c 文件），580
directory listings（目录列表），122
  disabling（禁止），123
discrete MIME types（离散 MIME 类型），534
distance delays（距离时延），163
distributed objects (HTTP:NG)［分布式对象（HTTP:NG）］，249
DNS
  caching（缓存），456
  DNS A record lookup（DNS A 记录查找），467
  redirection（重定向），453 ~ 457
    enhanced algorithms for（为其增强的算法），457
    multiple addresses and round-robin address rotation（多个地址及轮转地址的循环），455
  resolvers（解析程序），453
  round robin（轮转），454，455
  load balancing with（与其负载均衡），456
docroots (document roots)［docroots（文档根）］，120
  private（私有），122
  user home directory（用户主目录），122
  virtually hosted（虚拟主机），121

document access control（文档访问控制），132
document expiration（文档过期），175
  setting（设置），182
document hit rate（文档命中率），167
document roots (see docroot)（文档根，参见 docroot）
documents（文档）
  age and freshness lifetime（使用期及新鲜时长），188
  age-calculation algorithms（使用期计算算法），189 ~ 194
  caching（缓存），preventing（防止），182
  freshness and aging algorithms（新鲜度及老化算法），187 ~ 194
  heuristic expiration（试探性过期），184
Domain Name Service (see DNS)（域名服务，参见 DNS）
domain names（域名），internationalization of（国际化），392
downstream message flow（下游报文流），44
dups (Web robots)（Web 机器人），218
dynamic content resource mapping（动态内容资源映射），123

E

egress proxies（出口代理），137
embedded web servers（嵌入式 Web 服务器），111
encodings（编码），372
  chunked encodings（分块编码），345
  content encodings（内容编码），345，351 ~ 354
  delta encodings（差异编码），359，365 ~ 367
    impact on server disk space（对服务器磁盘空间的影响），368
  fixed-width（固定宽度），381
  transfer encodings（传输编码），354 ~ 359
end-of-line sequence（行终止序列），44
entities（实体），342
  body length（主体长度），determining（决定），346
  Content-Length headers（`Content-Length` 首部），344 ~ 347
  CRLF line（CRLF 行），343
entity bodies（实体的主体部分），44，47，52，343
  MIME types（MIME 类型），348
entity digests（实体摘要），347
entity headers（see under headers）（实体首部，参见首部条目下面的内容）
entity tags (see ETags)（实体标签，参见 `ETags`）
enumeration attacks（枚举攻击），313
equals sign(=)（等号），Base-64 encoding（Base-64 编码），572
escape sequences（转义序列），36
Etags (entity tags)（实体标签），180，298
  headers（首部），517
  using（使用），181
euc-jp encoding（euc-jp 编码），383
Expect headers（`Expect` 首部），517
experimental MIME types（实验性 MIME 类型），569
expiration of documents（文档过期），setting（设置），182
Expires headers（`Expires` 首部），175，183，517
explicit MIME typing（显式 MIME 分类），126
Extensible Markup Language (see XML)（扩展标记语言，参见 XML）
extension APIs（扩展 API），205
extension headers（扩展首部），51，68
extension methods（扩展方法），58

F

Fast CGI（快速 CGI），205
fat URLs（胖 URL），262
  limitations of（限制），263
file scheme（file 方案），39
filesystem link cycles（文件系统连接环路），220
FindProxyForURL() method（`FindProxyForURL()` 方法），143
fingerprint functions（指纹函数），289

first subtag（第一个子标记），387
fixed-width encodings（固定宽度编码），381
flash crowds（瞬间拥塞），163
format conversion（格式转换），404
FPAdminScriptUrl，426
FPAuthorScriptUrl，426
FPShtmlScriptUrl，426
Fpsrvadm，428
frag or fragment component（片段组件），URLs（URL），30
freshness and aging algorithms（新鲜度及老化算法），187 ~ 194
freshness lifetime（新鲜时长），188
From headers（From 首部），258，517
  robots（机器人），225
FrontPage，424 ~ 429
  client and server extension comunication（客户端和服务器扩展通信），426
  FrontPage Server Extensions (FPSE)（FrontPage 服务器扩展），424
  HTTP POST requests（HTTP POST 请求），425
    listExploreDocs element，POST request body（listExploreDocs 元素，POST 请求主体），427
  listHiddenDocs element，POST request body（listHiddenDocs 元素，POST 请求主体），427
  root web（根 Web），425
  RPC protocol（RPC 协议），426
  security（安全性），428
  server administrator utility (Fpsrvadm)［服务器管理员实用工具 Fpsrvadm)］，428
  service_name element，POST request body（service_name 元素，POST 请求主体），427
  subweb（子 Web），426
  virtual servers（虚拟服务器），425
ftp scheme（ftp 方案），39
full NAT（完全 NAT），461
full-text indexes（全文索引），243

G

gateways（网关），19，197 ~ 205
  client- and server-side（客户端及服务器端），199
  client-side security accelerator gateways（客户端安全加速器网关），202
  examples（举例），198
  protocol gateways（协议网关），200
  proxies（代理），contrasted with（与之对比），130
  resource gateways（资源网关），203
  server-side security gateways（服务器端安全网关），202
  server-side web gateways（服务器端 Web 网关），200
  Via headers（Via 首部），153
general headers（see under headers）（通用首部，参见首部条目下面的内容）
general-purpose software web servers（通用软件 Web 服务器），110
Generic Router Encapsulation (GRE)（通用路由器封装），472
GET command（GET 命令），virtual hosting issues（虚拟托管问题），414
GET messages（GET 报文），processing steps（处理步骤），171 ~ 175
GET method（GET 方法），53
getpeername function（getpeername 函数），259
glyphs（字形），378
GRE (Generic Router Encapsulation)（通用路由器封装），472

H

H(A1) and H(A2) reference code［H(A1) 和 H(A2) 参考代码］，577
half NAT（半 NAT），461
handshake delays（握手时延），82
handshakes（握手），digest authentication（摘要认证），290
hash tables（散列表），218
H(d)，one-way hash（单向散列），291
HEAD method（HEAD 方法），54
HEAD response（HEAD 响应），346

headers（首部），47，51，67 ～ 73，508 ～ 532
Accept，69，508
robots（机器人），225
Accept-Charset，371，375，509
MIME charset encoding tags（MIME 字符集编码标记），374
Accept-Encoding，509
Accept-Language，371，385，510
content negotiation（内容协商），398
Accept-Ranges，510
Age，510
Allow，159，511
authentication（认证），278
basic（基本），281，300
digest（摘要），300
Authentication-Info directives（Authentication-Info 指令），576
Authorization，281，511
directives（指令），575
preemptive generation（预先产生），295
Cache-Control，175 ～ 177，182 ～ 186，189，511
directives（指令），361
character set requirements（字符集需求），392
classification（分类），51
client identification using（客户端识别使用），258
Client-ip，258，260，512
Connection，86，512
content negotiation（内容协商），397
Content-Base，513
Content-Encoding，513
Content-Language，371，384，513
Content-Length，344 ～ 347，514
Content-Location，514
Content-MD5，347，514
Content-Range，515
Content-Type，348 ～ 351，515
charset parameter（字符集参数），371
continuation lines（延续行），51
Cookie，516
Cookie2，516
Date，516
DAV 431
compliance classes（遵从级别），445
Depth，431
Destination，431
entity headers（实体首部），51，67，72
content headers（内容首部），72
entity caching headers（实体缓存首部），73
HTTP/1.1，342
ETag，517
examples（举例），51
Expect，517
Expires，175，183，517
extension headers（扩展首部），68
From，517
robots（机器人），225
general headers（通用首部），51，67，68
caching headers（缓存首部），68
Heuristic Expiration Warning（试探性过期警告），184
Host，417，418，419，518
robots（机器人），225
HTCP cache headers（HTCP 缓存首部），480
If，43
If-Match，519

If-Modified-Since，166，178，518
If-None-Match，180，519
If-Range，519
If-Unmodified-Since，520
Last-Modified，520
Location，520
Lock-Token，431
max-age，183
Max-Forwards，156，521
for media types（用于媒体类型），348
Meter，196，493
MIME-Version，521
must-revalidate，183
no-cache，182
no-store，182
Overwrite，432，442
Pragma，68，182，521
Proxy-Authenticate，522
Proxy-Authorization，522
Proxy-Connection，96，523
Public，523
Range，524
Referer，259，524
  robots（机器人），225
request headers（请求首部），51，67，69 ~ 71
  Accept headers（Accept 首部），69
  client identification using（用其进行客户端识别），258
  conditional request headers（条件请求首部），70
  proxy request headers（代理请求首部），70
  request security headers（安全请求首部），70
response headers（响应首部），51，67，71 ~ 72
  negotiation headers（协商首部），71
  response security headers（安全响应首部），72
Retry-After，525
Server，525
Set-Cookie，264，525
  caching（缓存），273
  domain attributes（域名属性），267
Set-Cookie2，271，526
syntax（语法），51
tampering attack（篡改攻击），303
TE，526
Timeout，432，435
Title，527
Trailer，526
Transfer-Encoding，527
UA-，527
for uncachable documents（用于不可缓存文档），182
unsupported headers（不支持的首部），handling（处理），158
Upgrade，528
User-Agent，225，259，528
Vary，402，529
Via，151-154，529
Want-Digest，348
Warning，530
Warning，13，184
for WebDAV（用于 WebDAV），431
WWW-Authenticate，281，531，574 ~ 575
X-Cache，531
X-Forwarded-For，260，531
X-Pad，531
X-Serial-Number，532

heartbeat messages（心跳报文），473
heuristic expiration of documents（文档的试探性过期），184
Heuristic Expiration Warning headers（试探性过期警告首部），184
history expansion（历史扩展），34
hit logs（命中日志），195
hit metering（命中率测量），492 ~ 494
  Meter headers（Meter 首部），493
hit rate（命中率），167
host component（主机组件），URLs（URL），27
Host headers（Host 首部），417，418，518
  clients（客户端），requirements for（要求），418
  missing host headers（缺少 Host 首部），419
  proxies（代理），418，419
  robots（机器人），225
  web servers（Web 服务器），interpretation by（由其解释），419
hostile proxies（恶意代理），304
hosting services（托管服务），411
  dedicated web hosting（专用 Web 托管），412
hostname expansion（主机名扩展），34
hostnames（主机名），13
<href> element（<href> 元素），438
.htaccess，428
HTCP (Hyper Text Caching Protocol)（超文本缓存协议），478 ~ 481
  authentication（认证），480
  cache headers（缓存首部），480
  caching policies（缓存策略），setting（设置），480
  data components（数据组件），479
  message structure（报文结构），478
  opcodes（操作码），480
HTML (Hypertext Markup Language)（超文本标记语言）
  displaying resources using HTTP（使用 HTTP 显示资源），13
  documents（文档），relative URLs in（其中的相对 URL），31
  fragments（片段），referencing（引用），30
  robot-control META tags（机器人控制 META 标签），237
HTTP (Hypertext Transfer Protocol)（超文本传输语言），3 ~ 11，247
  authentication（认证），challenge/response framework（质询 / 响应框架），278
    (see also authentication)（还请参见认证条目）
  authentication schemes（认证方案），security risks（安全风险），303
  base-64 encoding（Base-64 编码），compatibility with（兼容于），570
  caching (see caches)（缓存，参见缓存）
  character sets (see character sets)（字符集，参见字符集）
  clients and servers（客户端及服务器），4
  commands（命令），8
  CONNECT method（CONNECT 方法），206 ~ 208
  connections (see HTTP connections)（连接，参见 HTTP 连接）
  entities (see entities)（实体，参见实体）
  headers (see headers)（首部，参见首部）
  hit metering extension（命中率测量扩展），492
  HTTP-NG (see HTTP-NG)（HTTP-NG，参见 HTTP-NG）
  informational resources（知识性资源），21
  instance manipulations（实例操控），359
  international content support（国际化内容支持），370
  limitations（局限性），248
  messages (see HTTP messages)（报文，参见 HTTP 报文）
  methods（方法），8
  performance considerations (see under TCP)（性能考虑，参见 TCP 条目下面的内容）
  proxy servers (see HTTP proxy servers)（代理服务器，参见 HTTP 代理服务器）
  redirection（重定向），452 ~ 453
  relays（中继），212
  reliability of（可靠性），3
  revalidations（再验证），165 ~ 166

robots（机器人），standards for（标准），225
secure HTTP (see HTTPS)（安全 HTTP，参见 HTTPS）
status codes（状态码），9，505 ~ 507
TCP，dependency on（取决于），80
textual basis of（文本基础），10
transactions（事务），8
delays（时延），causes of（原因），80
truncation detection（截尾检测），344
versions（版本），16
HTTP connections（HTTP 连接），74，75，86 ~ 104
closing（关闭），101 ~ 104
Connection headers（Connection 首部），86，512
establishing（建立），13，15 ~ 16
keep-alive connections (HTTP/1.0+)［keep-alive 连接（HTTP/1.0+)］，91 ~ 96
parallel (see parallel connections)（并行，参见并行连接）
persistent (see persistent connections)（持久，参见持久连接）
pipelined connections（管道化连接），99
Proxy-Connection headers（Proxy-Connection 首部），96，523
serial loading（串行加载），87
(see also TCP)（还请参见 TCP）
HTTP messages（HTTP 报文），8，10，43 ~ 73，474
entity bodies（实体的主体部分），47，52
example（举例），11
flow（流），43
GET messages，processing steps（GET 报文，处理步骤），171 ~ 175
headers (see headers)（首部，参见首部）
methods (see methods)（方法，参见方法）
reason phrases（原因短语），47，50
redirection of（对其重定向），450
request lines（请求行），48
robots（机器人），setting conditions for（为其设置条件），226
request URLs（请求 URL），46
response lines（响应行），48
robots（机器人），handling by（由其处理），227
start lines（起始行），47 ~ 51
status codes (see status codes)（状态码，参见状态码）
structure（结构），44
syntax（语法），45
tracing across proxies（追踪访问代理），150 ~ 157
Via headers（Via 首部），151 ~ 154
Version 0.9 messages（版本 0.9 报文），52
versions（版本），46，50
HTTP proxy servers（HTTP 代理服务器），129 ~ 160
authentication（认证），156
client proxy configuration（客户端代理配置），141 ~ 144
client traffic acquisition（客户端流量的获取），140
deploying（部署），137
interoperation（互操作），157 ~ 160
messages（报文），tracing（追踪），150 ~ 157
proxy and server requests（代理及服务器请求），handling（处理），146
proxy hierarchies（代理层次结构），138
public and private proxies（公共及私有代理），130
TRACE method and network diagnosis（TRACE 方法及网络诊断），155
unsupported headers and methods（不支持的首部和方法），handling（处理），158
URIs，in-flight modification of（转发过程中的修改），147
uses（使用），131 ~ 136
http scheme（http 方案），38
HTTP State Management Mechanism（HTTP 状态管理机制），265
HTTP: The Next Generation (see HTTP-NG)（HTTP：下一代，参见 HTTP-NG）
HTTP/0.9，16
HTTP/1.0，16

server requests to virtual hosts（服务器对虚拟主机的请求），problems with（存在的问题），413
HTTP/1.0+，17
HTTP/1.1，17
enhanced methods（增强的方法），444
entity header fields（实体首部字段），342
Host headers (see Host headers)（Host 首部，参见 Host 首部）
limitations（局限性），248
request pipelining（请求管道化），99
TRACE method（TRACE 方法），155
HTTP/2.0，17
HTTP-EQUIV tag（HTTP-EQUIV 标签），187
HTTP-NG (HTTP: The Next Generation)（HTTP：下一代），17，248 ~ 253
current status（当前状态），252
messsage transport layer（报文传输层），249-250
modularization（模块化），248
object types（对象类型），252
remote invocation layer（远程调用层），249 ~ 250
web application layer（Web 应用层），249，251
HTTPS，76，308 ~ 309，322 ~ 336
authentication（认证），326
clients（客户端），328 ~ 335
OpenSSL example（OpenSSL 举例），329 ~ 335
CONNECT method（CONNECT 方法），HTTP，206 ~ 208，336
connecting（连接），324
default port（默认端口），323
OpenSSL，328 ~ 335
schemes（方案），308，323
site certificate validation（站点证书的有效性），327
SSL handshake（SSL 握手），324 ~ 326
tunnels (see tunnels)（隧道，参见隧道）
https scheme（https 方案），38
Hyper Text Caching Protocol (see HTCP)（超文本缓存协议，参见 HTCP）
Hypertext Markup Language (see HTML)（超文本标记语言，参见 HTML）
Hypertext Transfer Protocol (see HTTP)（超文本传输协议，参见 HTTP）

I

IANA (Internet Assigned Numbers Authority)（因特网号码分配机构）
instance manipulations（实例操控），registered types（注册的类型），367
MIME charset registry（MIME 字符集注册表），602 ~ 615
MIME type registration（MIME 类型注册），537 ~ 539
registered language tags（已注册语言标签），386，582
ICP (Internet Cache Protocol)［ICP（因特网缓存协议）］，473
vs. CARP（与 CARP 对比），475
ident protocol（ident 协议），115
If headers（If 首部），431
If-Match headers（If-Match 首部），519
If-Modified-Since headers（If-Modified-Since 首部），166，178，518
If-None-Match headers（If-None-Match 首部），180，519
If-Range headers（If-Range 首部），519
If-Unmodified-Since headers（If-Unmodified-Since 首部），520
image/* MIME types（image/* MIME 类型），561 ~ 562
inbound messages（流入报文），43
inbound proxies（入口代理），138
indexes（索引），full-text（全文本），243
informational status codes (100-199)［信息性状态码（100-199）］，59 ~ 60，505
ingress proxies（入口代理），137
instance manipulations（实例操控），359，367 ~ 369
delta encodings（差异编码），365
IANA registered types（IANA 注册的类型），367
range requests（范围请求），364

integrity protection（完整性保护），299
intercepting proxies（拦截代理），140，146
  URI resolution with（对其进行 URI 解析），149
internationalization（国际化）
  date formats（日期格式），392
  domain names（域名），392
  headers（首部），character set for（字符集），392
  ISO 3166 country codes（ISO 3166 国家代码），594 ~ 600
  ISO 639 language codes（ISO 639 语言代码），583 ~ 594
  languages（语言），administrative organizations（管理组织），601
  URL variants（URL 变体），395
Internet Assigned Numbers Authority (see IANA)（因特网号码分配机构，参见 IANA）
Internet Cache Protocol (ICP)（因特网缓存协议），473
Internet search engines (see search engines)（因特网搜索引擎，参见搜索引擎）
IP address forwarding（IP 地址转发），460
IP (Internet protocol) addresses［IP（网际互连）地址］，13
  clients（客户端），identification using（用其进行识别），259
  virtual hosting（虚拟主机托管），416
IP MAC forwarding（IP MAC 转发），459
IP packets（IP 分组），76
iPlanet web servers（iPlanet Web 服务器），110
ISO 3166 country codes（ISO 3166 国家代码），594 ~ 600
ISO 639 language codes（ISO 639 语言代码），583 ~ 594
iso-2022-jp encoding（iso-2022-jp 编码），382
iso-8859 character set（iso-8859 字符集），380

J

Japanese encodings（日文编码），382，383
JIS X 0201, 0208, and 0212 character sets（JIS X 0201、0208 和 0212 字符集），380

K

KD(s,d) digest［KD(*s*,*d*) 摘要］，291
keep-alive connections (HTTP/1.0+)［keep-alive 连接（HTTP/1.0+）］，91 ~ 96
  (see also persistent connections)（还请参见持久连接）
keyed ciphers（使用了密钥的密码），311
keys（密钥），311，312
  key length（密钥长度），313

L

language preferences（语言偏好），configuring（配置），389
language tags（语言标记），370，384，581 ~ 600
  case sensitivity（大小写相关性），386
  first subtag rules（第一条子标记规则），581
  IANA-registered tags（IANA 注册的标记），386，582
  reference tables（参考表格），389
  second subtag rules（第二条子标记规则），582
  subtags（子标记），386
  syntax（语法），385
Last-Modified dates（`Last-Modified` 日期），using（使用），181
Last-Modified headers（`Last-Modified` 首部），520
layering of protocols（协议的分层），12
layout delay（布局时延），preventing（预防），88
ligatures（连笔），378
listExploreDocs element（`listExploreDocs` 元素），POST request body（POST 请求主体），427
listHiddenDocs element（listHiddenDocs 元素），POST request body（POST 请求主体），427
LM-Factor algorithm（LM-Factor 算法），184
load balancing（负载均衡），449
  DNS round robin（DNS 轮转），454 ~ 457

single clients（单个客户端），456
loading（加载），serial（串行），87
Location headers（Location 首部），520
LOCK method（LOCK 方法），433
status codes（状态码），436
lock refreshes（锁的刷新），435
<lockdiscovery> element（<lockdiscovery> 元素），435
<lockinfo> element（<lockinfo> 元素），434
locking（锁定），433
<locktoken> element（<locktoken> 元素），434
Lock-Token headers（Lock-Token 首部），431
logging（日志），483 ～ 492
commonly logged fields（通常会记录的字段），483
interpretation（解释），484
log formats（日志格式），484 ～ 492
Combined Log Format（组合日志格式），485
Common Log Format（常见日志格式），484
Netscape Extended 2 Log Format（网景扩展 2 日志格式），487 ～ 489
Netscape Extended Log Format（网景扩展日志格式），486
Squid Proxy Log Format（Squid 代理日志格式），489 ～ 492
privacy concerns（隐私考虑），495
loops (web robots)［循环（Web 机器人）］，217

M

MAC (Media Access Control) addresses（媒体访问控制地址），459
magic typing（魔法分类），126
mailto scheme（mailto 方案），38
man-in-the-middle attacks（中间人攻击），304
manual client proxy configuration（手工客户端代理配置），142
master origin server（主原始服务器），420
max-age response headers（max-age 响应首部），183
Max-Forwards headers（Max-Forwards 首部），155，521
MD5，288，291，293，347
MD5-sess，291，293
Media Access Control (MAC) addresses（媒体访问控制地址），459
media types（媒体类型），348
multipart（多部分），349
message body（报文主体），44
message digest algorithms（报文摘要算法），291 ～ 294
symmetric authentication（对称认证），298
message integrity protection（报文完整性保护），299
message/* MIME types（message/* MIME 类型），563
message transport layer (HTTP-NG)（报文传输层），249，250
message truncation（报文截尾），344
messages (see HTTP messages)（报文，参见 HTTP 报文）
<META HTTP-EQUIV>tag（<META HTTP-EQUIV> 标签），187
META tag directives（META 标签指令），239
meta-information（元信息），43
Meter headers（Meter 首部），196，493
methods（方法），46，48，53 ～ 59
CONNECT，206，208，336
DELETE，58，441
extension methods（扩展方法），58
GET，53
HEAD，54
OPTIONS，57，159，445
POST，55
PUT，54，444
redirection status codes（重定向状态码），64
TRACE，55
unsupported（不支持的），handling（处理），158
Microsoft FrontPage (see FrontPage)（微软的 FrontPage，参见 FrontPage）

Microsoft Internet Explorer（微软的 Internet Explorer）
  cookie storage（cookie 存储），266
  language preference configuration（语言偏好配置），389
Microsoft web servers（微软的 Web 服务器），110
MIME (Multipurpose Internet Mail Extensions)（多用途因特网邮件扩展）
  charset encoding tags（字符集编码标记），374
  charset registry（字符集注册表），602 ~ 615
    preferred MIME names（优选的 MIME 名），603
  “multipart” email messages（多部分电子邮件报文），349
  (see also MIME types)（还请参见 MIME 类型）
MIME types（MIME 类型），5，533 ~ 569
  application/* types（application/* 类型），540 ~ 557
  audio/* types（audio/* 类型），557 ~ 559
  chemical/* types（chemical/* 类型），559 ~ 561
  composite types（复合类型），534
  discrete types（离散类型），534
  documentation（文档），534
  experimental types（实验性类型），569
  IANA registration（IANA 注册），537 ~ 539
    media type registry（媒体类型注册），539
    process（处理），537
    registration trees（注册树），537
    rules（规则），538
    template（模版），538
  image/* types（image/* 类型），561 ~ 562
  message/* types（message/* 类型），563
  model/* types（model/* 类型），563
  multipart/* types（multipart/* 类型），535，564
  primary types（主类型），536
  structure（结构），534
  syntax（语法），536
  tables（表格），539 ~ 569
  text/* types（text/* 类型），565 ~ 568
  video/* types（video/* 类型），568
MIME typing（MIME 分类），125
MIME::Base64 Perl module（MIME::Base64 Perl 模块），572
MIME-Version headers（MIME-Version 首部），521
mirrored server farms（镜像服务器集群），420
MKCOL method（MKCOL 方法），440
mod_cern_meta module（mod_cern_meta 模块），Apache web server（Apache Web 服务器），186
model/* MIME types（model/* MIME 类型），563
mod_expires module（mod_expires 模块），Apache web server（Apache Web 服务器），186
mod_headers module（mod_headers 模块），Apache web server（Apache Web 服务器），186
MOVE method（MOVE 方法），442
multi-homed servers（多宿主服务器），425
multipart form-data encodings（多部分表格数据编码），349
multipart/* MIME types（multipart/* MIME 类型），535，564
multiplexed architectures（复用结构），119
  I/O web servers（I/O Web 服务器），119
  multithreaded web servers（多线程 Web 服务器），119
multiprocess（多处理器），multithreaded Web servers（多线程 Web 服务器），118
Multipurpose Internet Mail Extensions (see MIME; MIME types)（多用途因特网邮件扩展，参见 MIME，MIME 类型）
<multistatus> element（<multistatus> 元素），438
MultiViews directive（MultiViews 指令），400
must-revalidate response headers（must-revalidate 响应首部），183

N

Nagle's algorithm（Nagle 算法），84
namespace management（名字空间管理），439 ~ 444
  methods used for（用于其上的方法），440
    status codes（状态码），443
namespaces（名字空间），388
  language subtags（语言子标记），387 ~ 389
  XML，430
NAT (Network Address Translation)（网络地址转换），460
NECP (Network Element Control Protocol)（网元控制协议），461
negotiation headers（协商首部），71
Netscape Extended 2 Log Format（网景扩展 2 日志格式），487 ~ 489
Netscape Extended Log Format（网景扩展日志格式），486
Netscape Navigator
  cookies
    storage（存储），265
    Version 0（版本 0），269
  language preference configuration（语言偏好配置），389
Network Address Translation (NAT)（网络地址转换），460
network bottlenecks（网络瓶颈），161
Network Element Control Protocol (NECP)（网元控制协议），461
network exchange proxies（网络交换代理），137
news scheme（news 方案），39
no-cache response headers（`no-cache` 响应首部），182
nonces（随机数），289 ~ 298
  next nonce pregeneration（预先生成下一个随机数），297
  reuse（重用），297
  selection（选择），298
  time-synchronized generation（时间同步生成），297
no-store response headers（`no-store` 响应首部），182
.nsconfig，428

O

object types（对象类型），HTTP-NG，252
one-way digests（单向摘要），288
one-way hashes（单向散列），291
  functions（函数），289
opaquelocktoken scheme（opaquelocktoken 方案），433，434
OpenSSL，328 ~ 335
  example client（客户端实例），329 ~ 335
OPTIONS method（OPTIONS 方法），57，445
  requests（请求），159
    response headers to（对其的响应首部），445
origin servers（原始服务器），420
outbound messages（流出报文），43
outbound proxies（出口代理），138
Overwrite headers（`Overwrite` 首部），432，442

P

PAC files（PAC 文件），142
  autodiscovery（自动发现），465
PAC (Proxy Auto-Configuration) protocol［(代理自动配置）协议］，142，463
parallel connections（并行连接），88 ~ 90
  impression of speed（速度的影响），90
  loading speed（加载速度），88
  open connection limits（打开连接限制），90
  persistent connections（持久连接），91
parameters component（参数组件），URLs（URL），28
parent and child relationships（父子关系），138
parent caches（父缓存），169
password component（密码组件），URLs（URL），27
passwords（密码）

digest authentication password file（摘要认证密码文件），risks（风险），305
digest authentication（摘要认证），security（安全），287
path component（路径组件），URLs，28
Perl code for interaction with robots.txt files（与 robots.txt 文件交互的 Perl 代码），235
Perl web server（Perl Web 服务器），111
persistent connections（持久连接），90 ～ 99
Content-Length headers（Content-Length 首部），345
keep-alive connections (HTTP/1.0+)［keep-alive 连接（HTTP/1.0+)］，91 ～ 96
parallel connections（并行连接），91
restrictions and rules（限制及规则），98
persistent uniform resource locators (PURLs)（永久统一资源定位符），40
pipelined connections（管道化连接），99
plaintext（明文），security（安全性），310
port component（端口组件），URLS（URL），27
port exhaustion（端口耗尽），85
port numbers（端口号），13，77
default values（默认值），13
virtual hosting（虚拟主机托管），415
POST method（POST 方法），55
POST requests（POST 请求），FrontPage，425
Pragma headers（Pragma 首部），68，182，521
Pragma: no-cache headers（Pragma: no-cache 首部），182
precomputed dictionary attacks（预先计算的辞典攻击），305
preemptive authorization（预授权），295
presence bit arrays (web robots)［存在位数组（Web 机器人)］，218
presentation forms（表示形式），378
primary subtags（主子标记），386
privacy（隐私），495
cookies，275
robots（机器人），229
private caches（私有缓存），168
private docroots（私有 docroot），122
private proxies（私有代理），130
< prop> element（< prop> 元素），437
<propertyupdate> element（<propertyupdate> 元素），438
PROPFIND method（PROPFIND 方法），437
server response elements（服务器响应元素），438
XML elements，used with（在其中使用的 XML 元素），437
<propname> element（<propname> 元素），437
PROPPATCH method（PROPPATCH 方法），438 ～ 439
XML elements（XML 元素），used with（在其中使用），438
< propstat > element（<propstat> 元素），438
“protecting the header”（“对首部的保护”），87
protection spaces（保护空间），295，301
protocol gateways（协议网关），200
protocol stack（协议栈），76
protocols（协议），layering of（分层），12
proxies（代理）
100 Continue status code（100 Continue 状态码），60
authentication（认证），156
deploying（部署），137
egress proxies（出口代理），137
gateways（网关），contrasted with（与之对比），130
hostile（敌对的），304
HTTP proxies (see HTTP proxy servers)（HTTP 代理，参见 HTTP 代理服务器）
ingress proxies（入口代理），137
intercepting proxies（拦截代理），140，146
URI resolution with（URI 解析），149
interoperation（互操作），157 ～ 160
messages（报文），tracing（追踪），150 ～ 157
“missing scheme/host/port”problem（“缺少方案 / 主机 / 端口”的问题），146

proxy and server requests（代理和服务器请求），handling（处理），146
proxy hierarchies（代理层次结构），138
content routing（内容路由），139
dynamic parent selection（动态父路由选择），140
public and private（公共及私有），130
redirection（重定向），449
surrogates[替代物（反向代理）]，146
transparent negotiation（透明协商），400
transparent proxies（透明代理），140
tunnels（隧道），335
URIs，in-flight modification of（转发过程中的修改），147
uses（使用），131 ~ 136
(see also HTTP proxy servers)（还请参见 HTTP 代理服务器）
Proxy-Authenticate headers（`Proxy-Authenticate` 首部），522
proxy authentication（代理认证），283
Proxy-Authorization headers（`Proxy-Authorization` 首部），522
Proxy Auto-configuration (PAC) protocol（代理自动配置协议），142，463
(see also PAC files)（还请参见 PAC 文件）
proxy cache hierarchies（代理缓存层次结构），169
proxy caches（代理缓存），169
Proxy-Connection headers（`Proxy-Connection` 首部），96，523
proxy redirection（代理重定向）
CARP (Cache Array Routing Protocol)（缓存阵列路由协议），475 ~ 478
HTCP (Hyper Text Caching Protocol)（超文本缓存协议），478 ~ 481
ICP (Internet Cache Protocol)（因特网缓存协议），473
proxy redirection methods（代理重定向方法），462 ~ 469
explicit browser configuration（显式浏览器配置），463
disadvantages（缺点），463
PAC (Proxy Auto-configuration) protocol[（代理自动配置）协议]，463
WPAD (see WPAD)（参见 WPD）
proxy request headers（代理请求首部），70
proxy servers（代理服务器），18
networks（网络），use in securing（在安全方案中使用），335
(see also HTTP proxy servers)（还请参见 HTTP 代理服务器）
proxy URIs vs. server URIs（代理 URI 与服务器 URI 的对比），144
public caches（公共缓存），169
Public headers（Public 首部），523
public proxies（公共代理），130
public-key cryptography (see under cryptography)（公共密钥加密系统，参见加密条目下面的内容）
publishing systems（发布系统）
FrontPage (see FrontPage)（参见 FrontPage）
WebDAV (see WebDAV)（参见 WebDAV）
(see also web publishing)（也请参见 Web 发布）
PURLs (persistent uniform resource locators)（永久统一资源定位符），40
PUT method（PUT 方法），54，444

Q

qop (quality of protection)（保护质量），293
enhancements（增强），299
quality factors（质量因子），371
quality of protection (see qop)（保护质量，参见 qop）
quality values（质量值），398
query component（查询组件），URLs（URL），29

R

Range headers（`Range` 首部），524
range requests（范围请求），363
realm directive（realm 指令），280
realms (protection spaces)[域（保护空间）]，301
reason phrases（原因短语），9，47，50，505 ~ 507

redirection（重定向），126，448 ~ 481
  anycast addressing（任播寻址），457
  enhanced DNS-based algorithms（基于 DNS 的增强算法），457
  IP address forwarding（IP 地址转发），460
  IP MAC forwarding（IP MAC 转发），459
  load balancing（负载均衡），127，449
  methods（方法），450
    DNS redirection（DNS 重定向），453 ~ 457
    HTTP redirection（HTTP 重定向），452 ~ 453
    proxy methods（代理方法），462 ~ 469
    proxy techniques（代理技术），451
  NECP (Network Element Control protocol)（网元控制协议），461
  protocols（协议），450
  proxies（代理），role in（其中的角色），449
  purpose（用途），449
  techniques（技术），448
  temporary redirect（临时重定向），126
  transparent redirection（透明重定向），469
  URL augmentation（URL 增强），126
  WCCP (Web Cache Coordination Protocol)（Web 缓存协调协议），470 ~ 473
redirection status codes (300-399)［重定向状态码（300 ~ 399）］，61 ~ 64，505
Referer headers（Referer 首部），259，524
  robots（机器人），225
relative URLs（相对 URL），30 ~ 34
  bases（基础），31
  resolving（解析），33
relays（中继），212
  keep-alive connections（keep-alive 连接），212
relevancy ranking（相关性排名），245
reliable bit pipe（可靠的位传输管道），75
remote invocation layer (HTTP-NG)［远程调用层（HTTP-NG）］，249，250
remote procedure call (RPC) protocol（远程过程调用协议），FrontPage，426
<remove> element（<remove> 元素），439
replay attacks（重放攻击），284，289，303
  preventing（防止），289
replica origin servers（复制原始服务器），420
request digest reference code（请求摘要参考代码），577
request digests（请求摘要），294
request headers (see under headers)（请求首部，参见首部条目下面的内容）
request lines（请求行），48
request messages（请求报文），10，45，47
  methods（方法），46
  request URLs（请求 URL），46
request method (HTTP)［请求方法（HTTP）］，294
request pipelining（请求管道化），99
request security headers（安全请求首部），70
reserved characters (see restricted characters)（保留字符，参见受限字符）
resource gateways（资源网关），203
resource locator servers（资源定位符服务器），40
resource paths（资源路径），24
resources（资源），mapping and accessing of（映射及访问），120
response digest reference code（响应摘要参考代码），577
response entities（响应实体），125
response headers (see under headers)（响应首部，参见首部条目下面的内容）
response lines（响应行），48
response messages（响应报文），10，45，125
restricted characters（受限字符），36
Retry-After headers（Retry-After 首部），525
revalidate hits（再验证命中），165
revalidate misses（再验证未命中），166
revalidations（再验证），165-166
reverse proxies（反向代理），134，137

RobotRules object（RobotRules 对象），235
robots（机器人）
  conditional HTTP requests（条件 HTTP 请求），226
  entities（实体），227
  etiquette（规范），239 ～ 241
  excluding from web sites（从 Web 站点中排除），229 ～ 239
  HTML robot control META tags（HTML 机器人控制 META 标签），237
  HTTP（HTTP），225
    request headers（请求首部），identifying（识别），225
  META directives（META 指令），237
  Meta HTML tags（元 HTML 标签），227
  privacy（隐私），229
  problems caused by（引发的问题），228
  response handling（响应处理），227
  search engines（搜索引擎），242 ～ 246
  status codes（状态码），handling of（处理），227
  (see also robots.txt files)（还请参见 robots.txt 文件）
Robots Exclusion Standard（拒绝机器人访问标准），230
robots.txt files（robots.txt 文件），229
  caching and expiration（缓存及过期），234
    comments（注释），234
  disallow and allow lines（不允许行及允许行），233
  example（举例），236
  fetching（获取），231
  records（记录），232
  specification（规范），changes in（修改），234
  status codes for retrievals（解析状态码），231
  syntax（语法），232
  User-Agent line（`User-Agent` 行），232
  web sites（Web 站点），231
root set（根集），216
root Web（根 Web），425
round-robin load balancing（轮转负载均衡），452
  DNS round robin（DNS 轮转），454 ～ 457
routers and anycast addressing（路由器与任播寻址），457
RPC protocol，FrontPage（RPC 协议，FrontPage），426
RSA algorithm（RSA 算法），317
rtsp，rtspu schemes（rtsp，rtspu 方案），39

S

schemes（方案），7，24 27
  common formats（常见格式），38
  URIs for（... 的 URI），499 ～ 504
search engines（搜索引擎），242~246
  architecture（结构），242
  full-text indexes（全文索引），243
  queries（查询），244
  relevancy ranking（相关性排名），245
  results（结果），sorting and formatting（排序及格式化），244
  spoofing（欺诈），245
search trees（搜索树），218
second subtag（第二子标记），388
Secure Sockets Layer (see SSL)（安全套接字层，参见 SSL）
security（安全性）
  basic authentication（基本认证），283
  cookies（cookie），275
  digest authentication（摘要认证），286
  firewalls（防火墙），132
  FrontPage，security model（安全模块），428
  HTTP authentication schemes（HTTP 认证方案），associated risks（相关风险），303
  key length（密钥长度），314
  multiple authentication schemes（多重认证机制），risks（风险），303

security realms（安全域），280
WPAD security hole（WPAD 安全性漏洞），468
segments（分段），76
“sender silly window syndrome”（“发送端傻窗口综合症”），84
serial loading（串行装载），87
serial transaction delays（串行事务处理时延），87
server error status codes (500-599)［服务器错误状态码（500-599）］，66，505
server farms（服务器集群），413
mirrored servers（镜像服务器），420
Server headers（`Server` 首部），525
Server response header field（`Server` 响应首部字段），154
server URIs vs. proxy URIs（服务器 URI 与代理 URI 的对比），144
server-driven negotiation（服务器驱动协商），397 ~ 400
server-side extensions（服务器端扩展），400
servers（服务器）
100 Continue status codes（100 Continue 状态码），60
accelerators（加速器），134
certificates（证书），326
delta encodings（差异编码），impact on（对其影响），368
error status codes (500-599)［错误状态码（500-599）］，66
extension APIs（扩展 API），205
FrontPage Server Extensions (FPSE)（FrontPage 服务器扩展），424
Host headers（Host 首部），interpreting（解释），419
multi-homed servers（多宿主服务器），425
revalidation（再验证），177
server farms（服务器集群），413
master origin servers（主原始服务器），420
replica origin servers（复制原始服务器），420
supported functionality（所支持的功能），identifying（识别），159
validation（验证），175
server-side extensions（服务器端扩展），400
server-side gateways（服务器端网关），199
security gateways（安全网关），202
web gateways（Web 网关），200
server-side includes (SSIs)（服务器端包含），124
service groups（服务组），472
service_name element，POST request body（service_name 元素，POST 请求主体），427
sessions（会话），cookies，264
tracking with（用其进行跟踪），272
<set> element（`<set>` 元素），438
Set-Cookie headers（`Set-Cookie` 首部），264，525
caching（缓存），273
domain attributes（域属性），267
Set-Cookie2 headers（`Set-Cookie2` 首部），271，526
shared hosting（共享主机托管），413 ~ 419
shared keys（共享密钥），315
shared proxies（共享代理），130
sibling caches（兄弟缓存），171
Simple Object Access Protocol (SOAP)（简单对象访问协议），206
single-threaded web servers（单线程 Web 服务器），118
site certificate validation（站点证书有效性），327
slow hits（缓慢命中），165
s-maxage response headers（`s-maxage` 响应首部），183
SOAP (Simple Object Access Protocol)（简单对象访问协议），206
sockets API（套接字 API），78
calls（调用），78
software web servers（软件 Web 服务器），110
spiders（蜘蛛），215
spoofing（欺诈），245
Squid Proxy Log Format（Squid 代理日志格式），489 ~ 492
SSIs(server-side includes)（服务器端包含），124

SSL (Secure Sockets Layer)（安全套接字层），308
  authentication（认证），326
  handshakes（握手），324 ~ 326
  HTTPS，integration in（综合），323
  OpenSSL，328 ~ 335
  site certificate validation（站点证书有效性），327
  tunnels（隧道），209
  vs. HTTP/HTTPS gateways（与 HTTP/HTTPS 网关的对比），210
SSLeay，329
  (see also OpenSSL)（还请参见 OpenSSL）
start lines（起始行），47 ~ 51
status codes（状态码），9，49，59 ~ 67
  classes（分类），49
  client error codes (400-499)［客户端错误码（400-499）］，65 ~ 66，505
  HTTP codes（HTTP 码），505 ~ 507
  informational status codes (100-199)［信息性状态码（100-199）］，59 ~ 60，505
  LOCK method（LOCK 方法），436
  namespace management methods（名字空间管理方法），443
  redirection status codes (300-399)［重定向状态码（300-399）］，61 ~ 64，505
  robots（机器人），handling by（处理），227
  server error codes (500-599)［服务器错误码（500-599）］，66，505
  success status codes (200-299)［成功状态码（200-299）］，61，505
  UNLOCK method（UNLOCK 方法），436
<status> element（<status> 元素），438
strong validators（强验证器），181，363
subtags（子标记），386，389
  first subtag（第一子标记），387
  second subtag（第二子标记），388
subweb（子 Web），426
success status codes (200-299)［成功状态码（200-299）］，61，505
surrogate caches［替代物（反向代理）缓存］，421
surrogate proxies（替代物（反向）代理），137
surrogates［替代物（反向代理）］，134，146
symbolic links and cycles（符号连接与周期），220
symmetric authentication（对称认证），298
symmetric-key ciphers（对称密钥加密），313
syntax（语法），headers（首部），51

T

TCP slow start (see under TCP)（TCP 慢启动，参见 TCP 条目下面的内容）
TCP (Transmission Control Protocol)（传输控制协议），74 ~ 86
  connections（连接），75
    distinguishing values（识别值），77
    establishing（建立），13，79
    web server handling of（Web 服务器的处理），115
    (see also HTTP connections)（还请参见 HTTP 连接）
  network delays（网络时延），causes（原因），80
  performance considerations（性能考虑），80 ~ 86
    connection handshake delays（连接握手时延），82
    delayed acknowledgements（延迟确认），83
    delays（延迟），most common causes（最常见的原因），81 ~ 86
    Nagle's algorithm（Nagle 算法），84
    port exhaustion（端口耗尽），85
    TCP slow start（TCP 慢启动），83
    TCP_NODELAY，84
    TIME_WAIT accumulation（TIME_WAIT 累积），85
    port numbers（端口号），77
    reliability（可靠性），74
    serial loading（串行加载），87
    sockets API（套接字 API），78
    TCP slow start（TCP 慢启动），83

TCP/IP (Transmission Control Protocol/ Internet Protocol)（传输控制协议 / 网际协议），11
TCP_NODELAY parameter（TCP_NODELAY 参数），84
TE headers（TE 首部），526
Telnet example（Telnet 实例），15 ~ 16
telnet scheme（telnet 方案），40
text/* MIME types（text/* MIME 类型），565 ~ 568
<timeout> element（<timeout> 元素），434
Timeout headers（Timeout 首部），432，435
TIME_WAIT accumulation（TIME_WAIT 累积），85
Title headers（Title 首部），527
TLS (Transport Layer Security)（传输层安全），308
  (see also SSL)（还请参见 SSL）
TRACE method（TRACE 方法），55，155 ~ 157
  Max-Forwards，headers（首部），155
Trailer headers（Trailer 首部），526
transactional direction（事务处理的方向），messages（报文），43
transactions（事务），8
transcoders（转码器），135
transcodings（转码），403 ~ 406，395
  content injection（内容注入），405
  format conversion（格式转换），404
  information synthesis（信息综合），404
  types（类型），404
  vs. static pregenerated content（与静态预生成的内容对比），405
transfer encodings（传输编码），354 ~ 359
Transfer-Encoding headers（Transfer-Encoding 首部），527
Transmission Control Protocol/ Internet Protocol (TCP/IP)（传输控制协议 / 网际协议），11
transparent negotiation（透明协商），400 ~ 403
  caching（缓存），401
  Vary headers（Vary 首部），402
transparent proxies（透明代理），140
transparent redirection（透明重定向），469
Transport Layer Security (see TLS)（传输层安全，参见 TLS）
trees（树），218
truncation detection（截尾检测），344
tunnels（隧道），19，206 ~ 212
  authentication（认证），211
  HTTPS SSL，335 ~ 336
  security（安全），211
  SSL tunnels vs. HTTP/HTTPS gateways（SSL 隧道与 HTTP/HTTPS 网关的对比），210
type negotiation（类型协商），126
type-map files（type-map 文件），399
type-o-serve web server（type-o-serve Web 服务器），112

U

UA-headers（UA- 首部），527
UCS (Universal Character Set)（统一字符集），381
uncachable documents（不可缓存文档），182
uniform resource identifiers (see URIs)（统一资源标识符，参见 URI）
uniform resource locators (see URLs)（统一资源定位符，参见 URL）
uniform resource names (see URNs)（统一资源名，参见 URN）
Universal Character Set (UCS)（统一字符集），381
UNLOCK method（UNLOCK 方法），435
  status codes（状态码），436
Upgrade headers（Upgrade 首部），528
upstream message flow（上游报文流），44
uri-directive-value，294
URIs (uniform resource identifiers)（统一资源描述符），6，24

client autoexpansion and hostname resolution（客户端自动扩展和主机名解析），147
in-flight modification of（转发过程中的修改），by proxies（通过代理转发），147
internationalization（国际化），389 ~ 391
resolution（解析），144 ~ 150
proxy vs. server（代理与服务器的对比），144
with a proxy（通过代理转发），explicit（显式），149
without a proxy（没有代理），148
rewriting（重写），302
schemes（方案），499 ~ 504
URLs (uniform resource locators)（统一资源定位符），6，23 ~ 42
advantages of（优点），25
aliases（别名），219
augmentation（增强），126
automatic expansion（自动扩展），34
canonical form（规范形式），37
canonicalizing（规范化），220
character sets（字符集），35，35 ~ 38
CURLs，465
examples（实例），13
fat URLs（胖 URL），client identification using（用其进行客户端识别），262
informational resources（信息化资源），41
portability（可移植性），35
PURLs，40
relative URLs（相对 URL），30 ~ 34
restricted characters（受限字符），36
schemes（方案），7
schemes (see schemes)（方案，参见方案）
shortcuts（快捷方式），30
structure（结构），24
syntax（语法），26 ~ 30
URIs，as a subset of（作为其子集），24
variants［变体（备用候选）］，395
virtual hosting（虚拟主机托管），paths（路径），415
URNs (uniform resource names)（统一资源名），7，40
standardization（标准化），41
US-ASCII character set（US-ASCII 字符集），379
user agents（用户代理），19
user component（用户组件），URLs（URL），27
user home directory docroots（用户主目录 docroot），122
User-Agent headers（User-Agent 首部），225，259，528
user-tracking systems（用户追踪系统），content injection（内容注入），405
UTF-8 encoding（UTF-8 编码），382

V

validators（验证器），362
Last-Modified dates（Last-Modified 日期），using（使用），181
strong and weak（强弱），181，363
variable-length codes［可变长（变宽）编码］，372
variable-width modal encodings（可变宽有模态编码），381
variable-width nonmodal encodings（可变宽无模态编码），381
Vary headers（`Vary` 首部），402，529
Vermeer Technologies, Inc.（维美尔技术公司），424
version numbers（版本号），HTTP messages（HTTP 报文），50
Via headers（`Via` 首部），151 ~ 154，529
gateways and（网关），153
privacy and security（隐私和安全），154
request and response paths（请求和响应路径），153
Server response header fields（Server 响应首部字段），154
syntax（语法），152
video/* MIME types（video/* MIME 类型），568

virtual hosting（虚拟主机），225，413 ~ 419
  digital certificates and（数字证书），328
  docroots，121
  GET command（GET 命令），problems with（存在的问题），414
  Host headers（Host 首部），by，417 (see also Host headers)（还请参见 Host 首部）
  IP addresses（IP 地址），by，416
    problems for hosters（托管者的问题），417
  port numbers（端口号），by，415
  server requests，absence of host information in HTTP/1.0（服务器请求，在 HTTP/1.0 中缺少主机信息），413
    fixes（修正），415
  URL paths（URL 路径），by（通过），415
virtual servers（虚拟服务器），425

W

Want-Digest headers（Want-Digest 首部），348
Warning headers（Warning 首部），530
  Heuristic Expiration Warning（试探性过期警告），184
WCCP (Web Cache Coordination Protocol)（Web 缓存协调协议），470 ~ 473
  GRE packet encapsulation（GRE 分组封装），472
  heartbeat messages（心跳报文），473
  load balancing（负载均衡），473
  operation（操作），470
  service groups（服务组），472
  versions（版本），470
  WCCP2 messages（WCCP2 报文），470 ~ 472
    header and components（首部与组件），471
weak validators（弱验证器），181，363
web application layer (HTTP-NG)［Web 应用层（HTTP-NG)］，249，251
web architecture（Web 结构），17 ~ 20
Web Cache Coordination Protocol (see WCCP)（Web 缓存协调协议，参见 WCCP）
web caches（Web 缓存），18，161 (see also caches)（还请参考缓存）
web clients and servers（Web 客户端与服务器），4
Web Distributed Authoring and Versioning (see WebDAV)（Web 分布式写作和版本管理 , 请参见 WebDAV）
web hosting（Web 主机托管），411 ~ 422
  hosting services（托管服务），411
  shared or virtual hosting（共享或虚拟主机托管），413 ~ 419
web pages（Web 页面），9
Web Proxy Autodiscovery Protocol (see WPAD)（Web 代理自动发现协议，参见 WPAD）
web proxy servers（Web 代理服务器），129
web publishing（Web 发布）
  collaborative authoring（协作写作），429
  publishing systems（发布系统），424
web resources（Web 资源），4
web robots（Web 机器人），215 ~ 246
  crawlers (see crawlers)（爬虫，参见爬虫）
  examples（举例），215
  spiders（蜘蛛），215
web servers（Web 服务器），109
  access controls（访问控制），124
  appliances（应用），111
  client hostname identification（客户端主机名识别），115
  client identification（客户端识别），257 ~ 276
    cookies，using（使用），263 ~ 276
    fat URLs（胖 URL），using（使用），262
    headers（首部），using（使用），258
    IP address（IP 地址），using（使用），259
    user login（用户登录），using（使用），260
  connection input/output processing architectures（连接的输入 / 输出处理结构），117

connections（连接），handling new（处理新的），115
directory listings（目录列表），122
docroots（文档的根），120
dynamic content resource mapping（动态内容资源映射），123
embedded web servers（嵌入式 Web 服务器），111
explicit typing（显式分类），126
HTTP proxy servers (see HTTP proxy servers)（HTTP 代理服务器，参见 HTTP 代理服务器）
ident protocol（ident 协议），115
implementations（实现），109
logging（日志），127
MIME typing（MIME 分类），125
multiplexed I/O servers（复用的 I/O 服务器），119
multiplexed multithreaded servers（复用多线程服务器），119
multiprocess（多进程），multithreaded servers（多线程服务器），118
Perl example（Perl 举例），111
redirection responses（重定向响应），126
request handling（请求处理），449
request messages（请求报文）
  receiving（接收），116
  structure of（结构），117
resources（资源），mapping and accessing of（映射及访问），120
response entities（响应实体），125
response messages（响应报文），125
responses（响应），sending（发送），127
single-threaded servers（单线程服务器），118
software web servers（软件 Web 服务器），110
SSIs(server-side includes)（服务器端包含），124
tasks of（任务），113 ~ 114
type negotiation（类型协商），126
type-o-serve，112
  user authentication (see authentication)（用户认证，参见认证）
web services（Web 服务），206
web sites（Web 站点）
  personalizing of user experience（用户体验的个人化），257
  reliability（可靠性），improving（提高），419 ~ 422
    mirrored server farms（镜像服务器集群），420
  robots（机器人），exclusion（拒绝），229 ~ 239
  robots.txt files（robots.txt 文件），231
  speed（速度），improving（提升），422
web tunnels (see tunnels)（Web 隧道，参见隧道）
WebDAV (Web Distributed Authoring and Versioning)（Web 分布式写作和版本管理），429 ~ 446
  collaborative authoring（协作写作），429
  collections（集合），439 ~ 444
  DAV header（`DAV` 首部），431
  Depth header（`Depth` 首部），431
  Destination header（`Destination` 首部），431
  enhanced HTTP/1.1 methods（增强的 HTTP/1.1 方法），444
  headers（首部），431
  If header（`If` 首部），431
  LOCK method（LOCK 方法），433
  locking（锁定），432
  Lock-Token headers（`Lock-Token` 首部），431
  META data（META 数据），embedding（嵌入），436 ~ 439
  methods（方法），429
  namespace management（名字空间管理），439 ~ 444
  OPTIONS method（OPTIONS 方法），445
  Overwrite headers（`Overwrite` 首部），432
  overwrites（覆写），preventing（防止），432
  PROPATCH method（PROPATCH 方法），438 ~ 439
  properties（属性），436 ~ 439

PROPFIND method（PROPFIND 方法），437
PUT method（PUT 方法），444
Timeout headers（`Timeout` 首部），432
UNLOCK method（UNLOCK 方法），435
version management（版本管理），446
XML（XML），430
WebMUX protocol（WebMUX 协议），250，251
WPAD (Web Proxy Autodiscovery Protocol)（Web 代理自动发现协议），143，464 ～ 469
administration（管理），469
CURLs，465
DHCP discovery（DHCP 发现），467
DNS A record lookup（DNS A 记录查找），467
PAC file retrieval（PAC 文件获取），467
PAC file autodiscovery（PAC 文件自动发现），465
resource-discovery algorithm（资源发现算法），143，465
spoofing（欺骗），468
timeouts（超时），468
timing（计时），468
WWW-Authenticate headers（`WWW-Authenticate` 首部），281，531
directives（指令），574 ～ 575
WWW::RobotRules object（WWW::RobotRules 对象），235

X

X.509v3 certificates（X.509v3 证书），320
X-Cache headers（`X-Cache` 首部），531
X-Forwarded-For headers（`X-Forwarded-For` 首部），260，531
XML (Extensible Markup Language)（扩展标记语言），206，430
elements used in locking（锁定中使用的元素），434
namespace（名字空间），430
scheme definition（方案定义），XML documents（XML 文档），430
WebDAV（WebDAV），430
X-Pad headers（`X-Pad` 首部），531
X-Serial-Number headers（`X-Serial-Number` 首部），532

关于作者

David Gourley 是 Endeca 的首席技术官，负责 Endeca 产品的研究及开发。Endeca 开发的因特网及内部网络信息访问解决方案为企业级数据的导航及研究提供了一些新的方式。在到 Endeca 工作之前，David 是 Inktomi 基础工程组的一员，帮助开发了 Inktomi 的因特网搜索数据库，也是 Inktomi 的 Web 缓存产品的主要开发者。

David 在加州大学伯克利分校获得了计算机科学的学士学位，还拥有 Web 技术方面的几项专利。

Brian Totty 最近出任了 Inktomi 公司（这是 1996 年他参与建立的一家公司）研发部副总裁，在公司中他负责 Web 缓存、流媒体及因特网搜索技术的研发工作。他曾是 Silicon Graphics 公司的一名研究员，为高性能网络和超级计算机系统设计软件并对其进行优化。在那之前，他是苹果计算机公司高级技术组的一名工程师。

Brian 在伊利诺伊大学 Urbana-Champaign 分校获得了计算机科学的博士学位，在 MIT 获得了计算机科学及电子工程的学士学位，以及计算机系统研究的 Organick 奖。他还在加州大学分校开展并讲授了一些屡获殊荣的因特网技术方面的课程。

Marjorie Sayer 在 Inktomi 公司负责编写 Web 缓存方面的软件。在加州大学伯克利分校获得了数学硕士和博士学位之后，一直致力于数学课程的改革。从 1990 年开始致力于能量资源管理、并行系统软件、电话和网络方面的写作。

Sailu Reddy 目前在 Inktomi 公司负责嵌入式的性能增强型 HTTP 代理的开发。Sailu 从事复杂软件系统的开发已经有 12 年了，从 1995 年开始深入 Web 架构的研发工作。他是 Netscape 第一台 Web 服务器、Web 代理产品，以及后面几代产品的核心工程师。他具备 HTTP 应用程序、数据压缩技术、数据库引擎以及合作管理等

方面的技术经验。Sailu 在亚里桑那大学获得了信息系统的硕士学位并握有 Web 技术方面的多项专利。

Anshu Aggarwal 是 Inktomi 公司的工程总监。他管理 Inktomi 公司 Web 缓存产品的协议处理工程组，从 1997 年就开始参与 Inktomi 的 Web 技术设计工作。Anshu 在科罗拉多大学 Boulder 分校获得了计算机科学的硕士和博士学位，从事分布式多处理器的内存一致性技术研究。他还拥有电子工程的硕士和学士学位。Anshu 撰写了多篇技术论文，还拥有两项专利。

尾页

本书的封面设计得益于多方的参与，来自于读者意见、我们自己的揣摩以及各种发行渠道的反馈，共同的努力使它变得与众不同，既体现出我们诠释技术话题的独特手法，也使得有些原本枯燥的话题变得有趣。

封面上的动物叫多纹黄鼠（thirteen-lined ground squirrel，Spermophilus tridecemlineatus），在北美洲中部很常见。鼠如其名，沿着多纹黄鼠的背部有十三条条纹，上面有一行行的浅色斑点。它的颜色模式能让其融入周围的环境，以防捕食者发现它。多纹黄鼠是松鼠家族中的一员，松鼠家族中包括花栗鼠、地松鼠、树松鼠、土拨鼠和旱獭。它们的体型与东方花栗鼠类似，但比常见的灰松鼠小，平均长度大概有 11 英寸（加上一个 5 ~ 6 英寸的尾巴）。

多纹黄鼠会在十月份冬眠，并在三月末四月初苏醒过来。通常每只雌鼠会在每年五月产下一窝 7 ~ 10 只小鼠。小鼠会在四到五周大的时候离开地洞，并在六周的时候完全长成。它们喜欢在长有短草且排水良好的砂质开阔地或肥沃的土壤上打洞，它们不喜欢树木繁茂的地区——修剪过的草坪、高尔夫球场和公园是它们常见的栖息地。

地松鼠会引发一些问题，比如挖洞、挖出种下的种子，破坏菜园子。但它们同样也是几种捕食者的重要猎物，包括獾、草原狼、鹰、黄鼠狼和各种蛇，而且它们以很多有害的杂草、杂草种子及昆虫为食，会让人类直接受益。

本书的项目及文字编辑是 Rachel Wheeler。质量控制人员包括：Leanne Soylemez、Sarah Sherman 和 Mary Anne Weeks Mayo。其他辅助工作由 Derek Di Matteo 和 Brian Sawyer 完成。John Bickelhaupt 编写了索引部分。

本书封面是在 Edie Freedman 设计的一系列封面基础上完成的，并由 Ellie Volckhausen 完成具体的设计。封面的原始图片由 Lorrie LeJeune 绘制。封面布局是 Emma Colby 用 QuarkXPress 4.1 以 Adobe 的 ITC Garamond 字体绘制的。

David Futato 和 Melanie Wang 根据 David Futato 的一系列设计完成了内部版面的设计。Joe Wizda 用 FrameMaker 5.5.6 为制版准备了一些文件。书中的图片都是 Robert Romano 和 Jessamyn Read 用 Macromedia FreeHand 9 和 Adobe Photoshop 6 制作的。尾页是 Rachel Wheeler 编写的。

本书的在线版本是 Safari 产品组（John Chodacki、Becki Maisch 和 Madeleine Newell）用 Erik Ray、Benn Salter、John Chodacki 和 Jeff Liggett 编写并维护的一组 Frame-to-XML 转换及清理工具创建的。

Web Programming

O'REILLY®

HTTP权威指南



HTTP是Web客户端与服务器交互文档和信息时所使用的协议，是每个成功Web事务的幕后推手。众所周知，我们每天访问公司内部网络、搜索绝版书籍、研究统计信息时所使用的浏览器的核心就是HTTP。但HTTP的应用远不仅仅是浏览Web内容。由于HTTP既简单又普及，很多其他网络应用程序也选择了它，尤其是采用SOAP和XML-RPC这样的Web服务。

本书详细解释了HTTP协议，包括它是如何工作的，如何用它来开发基于Web的应用程序。但本书并不只介绍了HTTP，还探讨了HTTP有效工作所依赖的所有其他核心因特网技术。尽管HTTP是本书的中心内容，但本书的本质是理解Web的工作原理，以及如何将这些知识应用到Web编程和管理之中，主要涵盖HTTP的技术运作方式、产生动机、性能和目标以及一些相关技术问题。

本书是HTTP协议及相关Web技术方面的权威著作，主要内容包括：

- HTTP方法、首部以及状态码
- 优化代理和缓存的方法
- 设计Web机器人和爬虫的策略
- Cookies、认证以及安全HTTP
- 国际化及内容协商
- 重定向及负载平衡策略

本书由具有多年实践经验的专家编写，通过简洁、精确的语言和大量翔实的细节图解帮助读者形象地理解Web幕后所发生的事情，详细说明了Web上每条请求的实际运行情况。

要想高效地进行Web开发，所有Web程序员、管理员和应用程序开发者都应该熟悉HTTP。很多书籍只介绍了Web的使用方式，而本书则深入说明了Web的工作原理。

封面设计：Karen Montgomery　张健

图灵社区：www.ituring.com.cn
新浪微博：@图灵教育 @图灵社区
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

O'REILLY®
oreilly.com.cn

分类建议　计算机/网络技术/HTTP

人民邮电出版社网址：www.ptpress.com.cn

O'Reilly Media, Inc.授权人民邮电出版社出版

此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行
This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)

ISBN 978-7-115-28148-7

ISBN 978-7-115-28148-7
定价：109.00元

欢迎加入

图灵社区

电子书发售平台

电子出版的时代已经来临，在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版的梦想。你可以联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者，这极大地降低了出版的门槛。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果有意翻译哪本图书，欢迎来社区申请。只要通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

读者交流平台

在图灵社区，读者可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。欢迎大家积极参与社区开展的访谈、审读、评选等多种活动，赢取银子，可以换书哦！