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）。

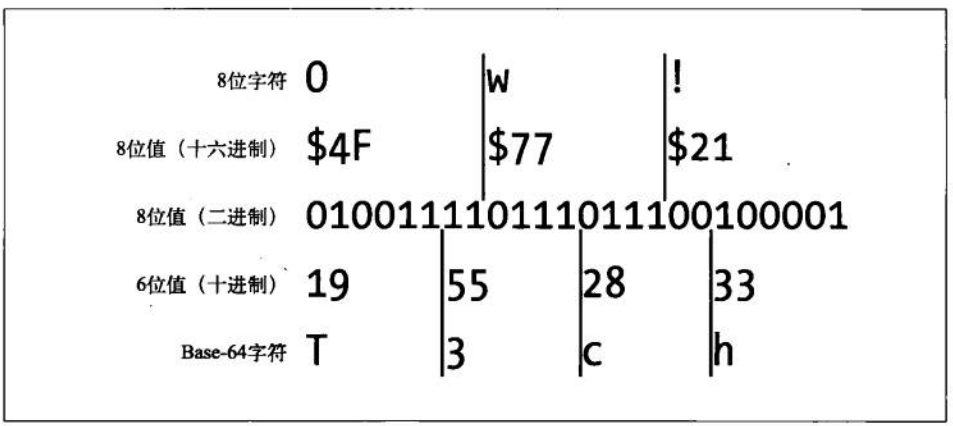

图 E-1 Base-64 编码实例

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。