SIP 协议描述（RFC2543）

协议版本：V2.0

原作者：M. Handley，ACIRI H.Schulzrinne

单位： IETF

目 录

1. Introduce（介绍）

1.1 功能简介（Overview of SIP Functionality）

SIP 是一个应用层的控制协议，可以建立，修改和结束多媒体的会话。这些多媒体会话包括多媒体会议，远程教育，网络电话和相关应用。SIP 既能够邀请用户也能够邀请机器，比如媒体存储服务。SIP 可以邀请多方进行多点传播和单点传播的会话；发起方不必是会话的成员，媒体和参与用户都可以被加入存在的会话。

SIP 能够启动会话，正如能够通过其他方法邀请其他成员加入会话一样。会话推荐使用多点传播

协议，比如 SAP，Email，新闻组，网页和目录服务等等。

SIP 支持名字映象和重定位服务，兼容 ISDN，IN 的服务。这些服务增强了用户的可移动性。IN 服务可以用一句话描述：“ 个人移动性是指用户以任何终端，在任何位置发出和接受通信服务，并且当用户移动时，可以定位用户位置。用户可移动性取决于唯一的个人身份鉴别机制，比如统一个人号码等 ”。终端的可移动性可以举个例子说明，比如当用户从系统的一个子网移动到另外的一个子网，这时候要保证正常的通信。

SIP 在五个层面上支持建立和终止多媒体通信：
User location（用户定位）：决定了用来通信的终端系统；
User capabilities（用户性能）：决定用户使用的媒体类型和媒体参数。
User availability（用户的可用性）：决定被叫方是否愿意参加通信。
Call setup（呼叫建立）：“ 振铃 ”，在主叫和被叫之间建立呼叫的参数
Call handling（呼叫处理）：包括呼叫转移和终止

SIP 用 MCU（多点控制单元）或 fully-meshed（全网连接）的交互连接发起多方呼叫，代替了原来的多点传播。连接 PSTN 的 ITG（Internet telephony gateways）也可以用 SIP 在它们之间建立连接。

SIP 是 IETF 在多媒体数据和控制体系结构方面相互结合的协议之一，类似的还有用于保留网络资源的 RSVP，用于传输实时数据和提供 QOS 保证的 RTP，用于控制媒体流传输的协议 RTSP，描述多媒体会话的 SDP 和 SAP。要注意的是，SIP 的功能和运作并不依靠这些协议。

SIP 也可以兼容其他呼叫的建立和其他协议的信令。在这个模式中，终端用户可以与已知的并且与协议无关的地址交换信息，以确定目标终端用户的地址和使用的协议。 比如，SIP 能够连接 H.323 的用户。先获得 H.245 的网关和用户地址，并用 H.225 建立呼叫连接。

另外一个例子，SIP 可以判断被叫是否可以通过 PSTN 到达，并用指定的电话号码呼叫，或者建议使用 Internet-to-PSTN 的网关。

SIP 并没有提供会议的控制服务，并没有规定投票系统（Voting）是如何被管理的。但 SIP 可以引出会议控制协议。SIP 不能够分配多点传播地址。

没有资源预约，SIP 也可以邀请用户加入会话。SIP 没有保留资源，但能够转达邀请系统的必要信息去申请资源。

1.2． Terminology（术语表－略）

1.3． Definitions（定义）

Client,Server 和 Proxy 的定义类似于 HTTP。下面的术语对 SIP 有重要的意义。

CALL（呼叫）： 呼叫包含会议中的所有参与者，这个会议往往由一个公共的源点发起的。SIP 呼叫有一个唯一的标识（Call-id Section 6.12 有介绍）。例如，如果一个用户加入到有几个用户的多点通话中时，每个参与者都是相同的呼叫中的一员。点到点的 Internet 电话映射为一个单个的 SIP 呼叫。 在一个多方会话单元中，每一个参与者使用独立的呼叫进入会话中？

CALL LEG（呼叫历程）： 由 CallID（呼叫标识）、From（源方）和 To（目的地）的组合来标识。

CLIENT（客户端）：是发送 SIP 请求的应用程序，User agents, proxy 甚至 servers 都可以是 Client。

CONFERENCE（会议）：用一个公共通话描述符标识的多媒体通话。可以包含零个或多个成员，包含多点会议、全网（full-mesh）会议、两方电话呼叫或者这些的组合的所有情况。呼叫的每一个成员均可建立会议。

DOWNSTREAM（下游）：由主叫发送到被叫的请求。（比如由客户端到服务端）

FINAL RESPONSE（最终相应）：即是终止一个 SIP 事务，一个临时的相应（1XX）不能终止会话事务。所有的 2xx、3xx、4xx、5xx 和 6xx response 都是最终相应。

INITIATOR，CALLING PARTY，CALLER：启动会议的一方。注意：呼叫端与会议的建立者并不需要一致。

INVITATION（邀请）：请求用户加入会话， 成功的 SIP 邀请包含两个事务：INVITE 和 ACK。

INVITEE、INVITED USER、CALLED PARTY、CALLEE：被呼叫端（Calling Party）邀请加入一个会议的人或业务。

ISOMORPHIC REQUEST OR RESPONSE： 如果两个请求或响应拥有相同的呼叫标识、源方、目的地和 Cseq 头域，则二者视为同构。另外，同构的请求拥有相同的 Request-URI。

LOCATION SERVER（定位服务器）： 定位服务由 SIP 的 Redirect Server 或者 Proxy Server 获得被叫者的可能位置信息。Location service 是 Location server 所提供的服务。Location servers 可以与 SIP 服务器相连，但服务器请求定位服务的方法超出这个文档的范围。

PARALLEL SEARCH（并行搜寻）： Proxy server 将接收到的请求同时发送到几个可能的用户，并不需要一个一个的发，一个一个的等待返回信息。没有等到上个请求的返回，下个请求也是可以开始的。

PROVISIONAL RESPONSE（临时响应）：接收到返回代码为 1XX 的响应，其他代码认为是结束响应。

PROXY，PROXY SERVER（代理服务器）：他们是一个中间体的程序，扮演着 CLIENT 和 SERVER 的角色。他们将信息由 Client 转化成一定的格式，转发到其他 Server 中。

REDIRECT SERVER（从定向服务器）：接受 SIP 的请求，映射请求的 IP 地址（零个或者多个）并且返回给客户端。与 Proxy Server 不同，Redirect Server 不能发起请求；与 User Agent Server 不同，Redirect Server 不能接收呼叫。

REGISTRAR（注册）：接受注册请求，注册服务器配合 Proxy 或者 Redirect Server 提供定位服务。

RINGBACK（回铃音）：由主叫客户端程序产生的信令音频，表明被叫正在振铃。（回铃音）

SERVER（服务器）：服务器接收请求并响应到请求者。服务器可以是 Proxy, Redirect，User agent 服务或者 Registrar 注册服务。

SESSION（会话）：如 SDP 协议（RFC2327）所说明：多媒体会话在发送者和接受者之间建立一系列的多媒体数据流。多媒体会议是多媒体会话的一个典型例子。因此，被叫可以在同一会话被不同的呼叫邀请几次。如果使用流 SDP，session 可以用用户姓名、通话标识、网络类型、地址类型和在数据源域中的地址元素组合来定义。

TRANSACTION（事务）：SIP 的事务发生在客户端到服务器之间的一系列请求响应，包括从客户端的第一个请求到服务器的最后一个响应所有消息。在同样事务将由 Cseq 号码来鉴别，ACK 请求与和其通信的 INVITE 请求有相同的 Cseq 号，但有自己的事务。（注意：Transaction 有相同的 Call Leg，即是相同的 CallID，From，To。A）

Upstream（上游）：从 User agent server 到 user agent client 的响应。

URL－encoded：根据 RFC 1738

User agent client（UAC），calling user agent：客户端发起一个 SIP 请求。

User agent server（UAS），called user agent：接收并响应客户端的请求。响应包括包括 accepts,rejects 和 redirects。

User agent（UA）：包括 User agent client 和 User agent server。应用程序既可以当服务端，也可以当客户端。

下表是各种 SIP Server 的属性表：

property	redirect server	proxy server	user agent server	registrar
also acts as a SIP client	no	yes	no	no
returns 1xx status	yes	yes	yes	yes
returns 2xx status	no	yes	yes	yes
returns 3xx status	yes	yes	yes	yes
returns 4xx status	yes	yes	yes	yes
returns 5xx status	yes	yes	yes	yes
returns 6xx status	no	yes	yes	yes
inserts Via header	no	yes	no	no
accepts ACK	yes	yes	yes	no

1.4．SIP 协议的总体描述

这一部分是协议的基本功能和基本操作。主叫和被叫通过 SIP 的地址鉴别，（1.4.1 中有描述），当产生一个呼叫时，呼叫首先定位合适的服务器（1.4.2），然后发送一个 SIP 请求（1.4.3）。SIP 最常用的操作就是 INVITE（1.4.4）。SIP 并不是直接访问被叫，而是通过 Redirect server 或者 Proxy 产生新的 SIP 请求（1.4.5）。用户能够通过他们的 SIP Server 注册他们的位置（4.2.6）。

1.4.1 SIP 的寻址

被 SIP 称为对象（objects）的是主机上的用户，通过 SIP 的 URL 鉴别。SIP URL 很类似信箱的

Mailto 或者 Telnet URL，比如 user@host。“ user ”部分是一个用户名或者电话号码；Host 部分是一个域名地址或者 IP 地址。第二部分将会详细的讨论 SIP 的 URL。

用户的 SIP 地址可以从 out-of-band 中获得，可以从存在的媒体代理获得，可以包含在邮件的消息头，可以记录在以前的交换 INVITE 中。在很多情况下，用户的 SIP URL 可以从 Email 地址中找出来。

SIP 的 URL 地址可以被单独的指定，包括团队中的第一个人或者整个团队。象这样的地址形式 SIP：sales@example.com 是不够的，通常主叫信息包含有主叫的意图。

如果用户或者某个服务从个人或者组织联系表中，获得一个可以到达的地址。传统的通过“ 黑名单 ”来保证的是很危险的。然而，SIP 不像传统的电话技术，SIP 提供认证和访问控制机制实现自身低层的安全，客户端将拒绝没有认证的，或者不需要的呼叫请求。

1.4.2 定位一个 SIP 服务器

当客户端想发送一个请求，要不就发送到本地确认的 SIP Proxy 服务器 —— 与 Request－URI 无关，要不就发送到 Request－URI 对应的端口和地址。

对于第二种情况，客户端必须确定请求服务器使用的协议，端口和地址。客户端应该通过如下的步骤去获得这些信息，但也有可选择的过程（详见附录 D）。在每一步，除非状态改变，客户端应该尝试用 Request－URI 中列出的端口号码通信，如果在 Request－URI 中没有写明，客户端则缺省的使用 5060。如果 Request-URI 确定一个协议，客户端必须使用指明的协议。如果没有指明协议，客户端尝试使用 UDP（支持此协议的话）。如果尝试失败，或者客户端部支持 UDP 只能支持 TCP，那么只有尝试 TCP。

如果客户端不能到达服务器，必须能够解析具体的原因，这样比单独返回 timeout 好很多。如果客户端发现服务器在某个特定的地址不能访问，就如客户端接收到 400 类的错误响应一样。

客户端通过 DNS，寻找一个或者多个 SIP 服务器的 IP 地址。这个过程如下：

1. 如果 Request-URI 的 HOST 部分是个 IP 地址，客户端用所给的地址联系服务器。否则，客户端要先执行到下一步。
2. 客户端以 Request-URI 的 Host 部分查询 DNS 服务器。如果 DNS 服务器返回没有地址记录，那么这个客户端停止动作，认为不能定位服务器。

没有强制地规定如何去选择 SIP 服务器的主机名字，不过鼓励用户以 sip.domainname 的习惯来命名他们的服务器（RFC2219 有说明）。不过，用户知道被叫用 Email 地址代替 SIP URL。在某些情况下，用 Email 中的地址中可以构造一个 SIP URL，这种方法到达 SIP Server 的可能性会增加。将来，会有更好的 DNS 技术，不必需要这种机制。比如在附录 D 中所定义的，将逐步有更广泛的应用。

客户端（Proxy）应该暂时保存一个成功的 DNS 请求。一个成功的查询是指在回答中包含记录，并且 server 与答案中的一个地址保持着联系。当客户端希望发送一个请求到同一主机，它必须首先在 Proxy 的 Cache 中查找。客户端必须依照 RFC1035 的流程，当 DNS 超时没有响应时，应该认为缓冲无效。

1.4.3 SIP Transaction（交互事务）

一旦解决来 SIP server 的 Host 部分，客户端开始跟 Server 进行请求－应答通信。一个请求和请求

的应答称为一个事务（Transaction)。所有请求的应答包含着相同的 Call-ID,Cseq,To 和 From 域。这样允许响应跟请求的匹配，注意：ACK 请求和随后的 INVITE 并不是一个事务，Cseq 并不一致。

如果使用 TCP，在一个 SIP 事务内的请求和响应将有相同的 TCP 连接（Section 10)。几个 SIP 请求从同一客户端到同一服务器有两种情况，一种是几个请求同用一个连接，另外一种情况是每个请求用一个连接。

如果客户端经过单点传输的 UDP 发送一个请求，响应发送到包含在 Via 头域（Section6.40）中的 IP 地址中，如果请求经过多点传输的 UDP，响应直接指向多点传播地址和目的端口。对于 UDP，可靠性通过使用重传机制来保证。

SIP 消息格式和消息的通讯域传输的协议无关。

1.4.4 SIP INVITE（SIP 邀请）

成功的 SIP 邀请包含两个请求，INVITE 和 ACK。INVITE 请求（Section4.2.1）要求被叫参加一个特定的会议或者建立一个双方通话。当被叫同意参加这个呼叫，主叫确认到被叫已经收到，并且返回一个 ACK（Section4.2.2）表示得到证实。如果主叫不再想参加这次呼叫，这时候返回的是 Bye 消息，而不是 ACK。

INVITE 请求一般都会包含一个会话描述，比如 SDP（RFC2327）格式，这样被叫将拥有足够的（媒体）信息加入这个会话。对于多点传输的会话，会话描述列举媒体的类型和格式，以便允许会话分发这些信息。对于单点传播，会话描述列举主叫希望用到的媒体的类型和格式，和主叫希望被叫发送的信息的地址。另外，如果被叫想接受呼叫，将返回一个希望用到的媒体列表。对于多点传播会话，被叫如果不能从主叫的描述中接收媒体指示，或者不能从单播中接收数据，将返回一个会话的描述。

Proxy Server 的 INVITE 交换过程可以从图 1 看出，而图 2 是个重定位的过程。

Figure 1:

1． 代理服务器接收 INVITE 请求

2． 取出部分或者全部的地址送到 Location Server 进行定位。

3． Location Server 返回一个确切的位置。

4． Proxy Server 发出一个 SIP INVITE 请求到指定的位置。

5． UAS 振铃用户。

6． 返回一个成功的指示到 proxy server。

7． Proxy server 返回一个成功的消息到主叫

8,9 发送证实消息。主叫发送的 ACK 可以经过 proxy，送到被叫。

Note： 收发的请求和相应有同用的 ID。

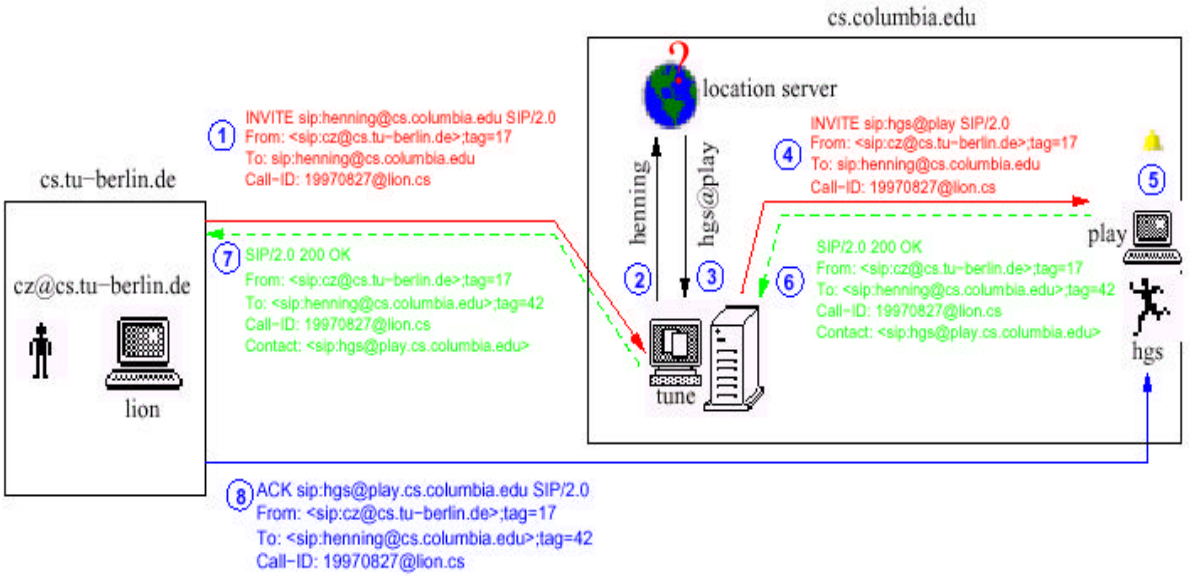

Figure 1: Example of SIP proxy server

重定位服务器说明（图 2）

1） Redirect Server **接受了** INVITE **请求**

2）,3）**向重定位服务器查询信息**

4）**返回地址到主叫**

5）**发送** ACK **给** Proxy Server

6) **主叫用以刚取到的地址发起一个新的请求，新的请求包含同样的** CALL ID **，但更高的** Cseq**。**

7) **在这个例子中，呼叫成功。**

8) **主叫被叫通过** ACK **完成了握手过程。**

下一部分将讨论如果 Location Service **返回更多的可能时如何做。**

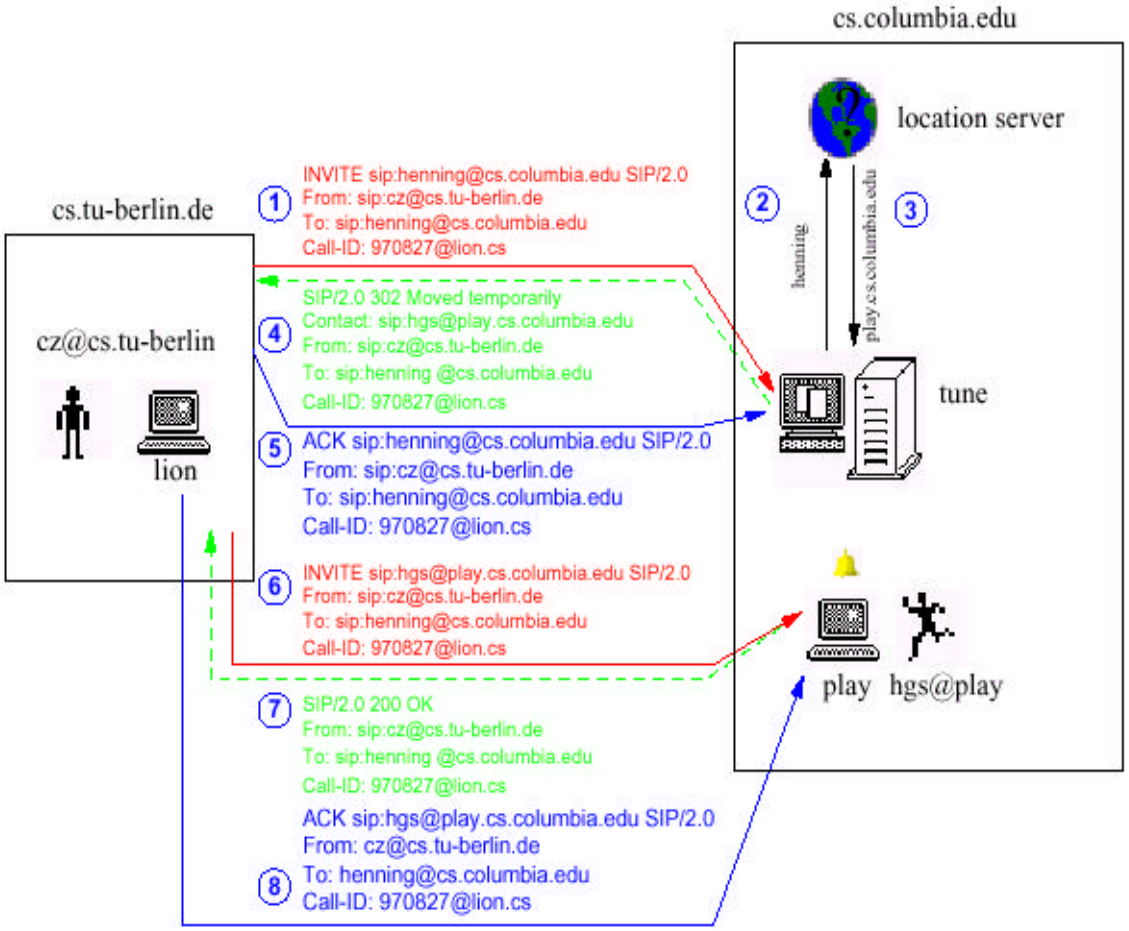

Figure 2: Example of SIP redirect server

1.4.5 Locating a user（定位用户）

被叫可以在不同的系统之间移动，这些位置能够动态的在 SIP Server 中注册（Serction 1.4.7,4.2.6）。Location Server 必须支持一个或者多个协议，比如 finger （RFC 1288[17]），rwhois （RFC 2167[18]）,LDAP(RFC 1777[19])等或者操作系统支持的机制来确定用户是否可到达。因为用户可以在几个地方注册，或者同时登陆到不同的注册服务器，或者定位服务器出现故障，所以 Location Server 可能会返回几个位置。SIP Server 会结合这些结果，写到联系列表中。

这个活动将会在 SIP Server 中接收到一系列的位置信息，SIP Redirect Server 在 Contact headers 返回这个位置列表到客户端（Section 6.13）。SIP Proxy Server 可以按顺序或者并行的尝试这些地址，直到呼叫返回成功(2XX response)，或者拒绝呼叫(6XX response)。随着这些顺序的尝试，Proxy Server 可以执行“anycast”服务。

如果 Proxy Server 转发 SIP 请求，它必须将自己的 IP 地址加入到 Via 部分的开头（Section 6.40）。这个 Via 的跟踪确保应答可以顺着原来的路径返回。在每个返回的路径，每个主机必须将自己的 IP 地址从 Via 中去掉，这样被叫和外部网络的路由信息可以蕴藏起来。

SIP 邀请能够经过更多的 SIP Proxy Server，比如采用“Forks”请求时，Proxy 接收到请求后将产生更多的请求。有可能客户端接收到不止一个此邀请的相互独立的重复，每一个副本都使用同样的 call-id，User Agent 必须返回与第一个响应同样的状态响应。复制请求是允许的。

1.4.6 Changing an Existing Session（改变已存通话）

在一些情况下，我们想改变一个正在进行的通话的参数。这一点可以通过重发 INVITE 来实现，使用相同的 CALL ID，但是必须使用一个新的或者不同的消息体或头域来传递新消息。INVITE 必须使用一个比前面的从客户端到服务器的请求数值更高的 Cseq。

例如：双方可能已经在通话，并且想加入第三方。其中的一个参与者使用新的多点地址邀请第三方，同时使用新的多点通话描述符向另一方发送 INVITE 请求，但必须是原先的呼叫标识。

1.4.7 Registration Services（注册服务）

Regester 允许客户让 proxy 或者 redirect 服务器知道客户的地址，客户可以用它来在服务器上安装呼叫处理功能部件。

1.5 Protocol Properties 协议属性

1.5.1 Minimal State（最小状态）

简单的会议呼叫包含 SIP 请求-响应事务。proxy server 并不为某个呼叫保存状态，然而，它们可能会为某个的 SIP 事务保持状态。为了高效性，服务器可以将定位服务请求的结果放入缓冲。

1.5.2 Lower-layer-Protocol Neutral（底层协议中立）

SIP 提供了传输层和网络层的最小的假设。底层提供了消息包或者比特流服务，可靠或不可靠服务。

在 Internet 上下文中，SIP 可以利用 UDP 和 TCP 作为传输协议。UDP 通信方式更高效，可以不用连接就可以并行的发送信息，方便多点广播。

同一个 SIP 呼叫的不同的 SIP 请求可能使用不同的 TCP 连接，或者同一个连接。本文档只介绍 SIP 作为一个 Internet 协议，实际上，SIP 也可以用在诸如 ATM AAL5、IPX、帧中继或者 X．25 协议中。

1.5.3 Text-Based（基于文本）

SIP 是基于文本的，使用 10646 编码。这使得容易使用 Java,Tcl 和 Perl 等语言调试，而且最重要的是，使 SIP 更加容易理解，更加容易被扩展。

2. SIP Uniform Resource Locators(SIP 的统一资源定位器)

SIP 的 URL 用在 SIP 消息中，指示着呼叫请求的起源（From），当前的目的地址（Resquest－URI）和最终的接收者（To），并且确定重定位地址（Contact）。SIP 的 URL 能被嵌入到网页，或者其他超级连接中，以指明一个特定的用户和服务能够通过 SIP 呼叫。当做超级链接使用时，INVITE 的方法通过 SIP 的 URL 来说明。SIP URL 可以设置 SIP 请求头域和 SIP 的消息体。

这里有个相应的用法 Mailto：URLs。这使得可以确定一个从网页或者 Email 消息的呼叫的主题，紧急性和媒体的类型。

SIP 的 URL 依从 RFC2396 的指导方针，语法如图 3。语法用 ABNF 描述（参见 Section C）。注意不能使用保留的字符，在所给 URI 单元定义的一套保留字符也不能使用。（如果想使用保留字符时，请用“%40”。40 是表示所要保留字符的 ASCII 码）

SIP URI 单元有如下含义：

```
SIP-URL          = "sip:" [ userinfo "@" ] hostport
                        url-parameters [ headers ]
userinfo         = user [ ":" password ]
user             = *( unreserved | escaped
                 | "&" | "=" | "+" | "$" | "," )
password         = *( unreserved | escaped
                 | "&" | "=" | "+" | "$" | "," )
hostport         = host [ ":" port ]
host             = hostname | IPv4address
hostname         = *( domainlabel "." ) toplabel [ "." ]
domainlabel      = alphanum | alphanum *( alphanum | "-" ) alphanum
toplabel         = alpha | alpha *( alphanum | "-" ) alphanum
IPv4address      = 1*digit "." 1*digit "." 1*digit "." 1*digit
port             = *digit
url-parameters   = *( ";" url-parameter )
url-parameter    = transport-param | user-param | method-param
                 | ttl-param | maddr-param | other-param
transport-param  = "transport=" ( "udp" | "tcp" )
ttl-param        = "ttl=" ttl
ttl              = 1*3DIGIT          ; 0 to 255
maddr-param      = "maddr=" host
user-param       = "user=" ( "phone" | "ip" )
method-param     = "method=" Method
tag-param        = "tag=" UUID
UUID             = 1*( hex | "-" )
other-param      = ( token | ( token "=" ( token | quoted-string )))
headers          = "?" header *( "&" header )
header           = hname "=" hvalue
hname            = 1*uric
hvalue           = *uric
uric             = reserved | unreserved | escaped
reserved         = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+" |
                   "$" | ","
digits           = 1*DIGIT
```

Figure 3: SIP URL syntax

User：如果主机是一个 Internet 电话网关，用户域所使用的电话描述符使用电话号码（图 4）。电话号码是用户名的一种特殊形式，不能被 BNF 区分。这样，URL 参数，用户联合在一起，区分电话号码和用户名。连接到电话网关时就使用电话标志符，一旦没有这些参数，而用户名命名空间的本地限制允许的话，SIP URL 的接收者会请求的 “@”符号前部分解析为电话号码。

Password： SIP 方案可以在“Userinfo”域中使用格式“User：Password”。但并不推荐 Password 在 Userinfo 中的使用，因为这里使用的是明文。

Host：The mailto：URL 和 RFC822 规定的 Email 地址要求数字的主机地址必须包含在方括号内，而没有方括号的地址适用于其他所有的 URLs。SIP URL 要求的是最终的形式，没有方括号。

Ipv6 的地址问题在 IETF 的另外文章里有讨论，SIP 会紧密跟踪。

Port：这个端口号就是发送请求的号码，如果当前没有，Section1.4.2 的流程说明如何决定一个端口号来发送一个请求。

URL parameters：SIP URL 可以定义一个具体的请求参数。URL 参数加入到主机单元后面，并以 ”;” 分隔。传输的参数决定传输的机制（UDP 或者 TCP）。但没有明确的指明传输的参数时，缺省地使用 UDP。maddr 地址参数提供用户所要连接的服务器的地址，这在主机域中支持。这个地址是典型的多点传送地址，但也可以是备份服务器的地址。当地址是一个多点传播地址，又是用 UDP 协议时，TTL 参数决定 UDP 多点传播包的生命期。用户的参数上面已经有所描述。例如，为了确定用多点传播呼叫 j.doe@big.com 到地址 239.255.255.1，使用 TTL 为 15，将使用下面的 URL：

SIP：j.doe@big.com;maddr=239.255.255.1;ttl=15

传输协议，地址，TTL 参数一定不能放在头域中的 From、To 和 Request-URI 中，否则将被忽略。

Headers：在 SIP URL 中，SIP 请求头可以被定义成“？”（原版本也是?）机制。”body”表明是 SIP INVITE 请求的消息体的值。Header 也不能用在 To, From 和 Request-URI 中，否则将被忽略。Hname 和 hvalue 分别包含 SIP Header 的名字和值。必须避免在 Header 的 name 域和 vlues 域中出现保留字符。

Method：SIP 请求的方法参数将确定所在地方法。同样不能够用于 From,To 和 Request-URI 中。

```
telephone-subscriber    = global-phone-number | local-phone-number
global-phone-number     = "+" 1*phonedigit [isdn-subaddress]
                                [post-dial]
local-phone-number      = 1*(phonedigit | dtmf-digit |
                                pause-character) [isdn-subaddress]
                                [post-dial]
isdn-subaddress         = ";isub=" 1*phonedigit
post-dial               = ";postd=" 1*(phonedigit | dtmf-digit
                                |   pause-character)
phonedigit              = DIGIT | visual-separator
visual-separator        = "-" | "."
pause-character         = one-second-pause | wait-for-dial-tone
one-second-pause        = "p"
wait-for-dial-tone      = "w"
dtmf-digit              = "*" | "#" | "A" | "B" | "C" | "D"
```

Figure 4: SIP URL syntax; telephone subscriber

sip:j.doe@big.com
sip:j.doe:secret@big.com;transport=tcp
sip:j.doe@big.com?subject=project
sip:+1-212-555-1212:1234@gateway.com;user=phone
sip:1212@gateway.com
sip:alice@10.1.2.3
sip:alice@example.com
sip:alice%40example.com@gateway.com
sip:alice@registrar.com;method=REGISTER

表 2 概括 SIP URL 哪些条目可以填写，如果没有写值时，缺省为多少。

	default	Req.-URI	To	From	Contact	external
user	--	x	x	x	x	x
password	--	x	x		x	x
host	mandatory	x	x	x	x	x
port	5060	x	x	x	x	x
user-param	ip	x	x	x	x	x
method	INVITE				x	x
maddr-param	--				x	x
ttl-param	1				x	x
transp.-param	--				x	x
headers	--				x	x

Table 2: Use and default values of URL components for SIP headers, Request-URI and references

在 SIP 消息内，URLs 用于表明请求的起始地址和终止地址，重定向地址和当前的目的地址。通常所有的这些域都包含 SIP URLs。

SIP URLs 大小写不敏感，所以比如 sip:j.doe@example.com 和 SIP:J.Doe@Example.com 是等价的。

SIP 头域可以包含非 SIP 的 URLs，比如，从 PSTN 通过 SIP 转化到 Internet 上，SIP 的 From 域将包含一个 Phone URL。

3. SIP Message Overview（SIP 消息的概述）

SIP 是基于文本的协议，采用 ISO10646 的字符集，发送者必须以 CRLF 结束一行，接收者接收到 CR 和 LF 后，都认为是行的结束。

除了字符集不同，大多数的消息语法跟 HTTP/1.1 相同。我们以 ABNF 描述 SIP。ABNF 将在 Section C 中有详细介绍。然而，SIP 不是 HTTP 的扩展。

不像 HTTP，SIP 可以使用 UDP。当在 TCP 或者 UDP 中传输时，多个 SIP 事务可以只用一个 TCP 连接或者 UDP 的数据报。包括头部的 UDP 数据报，不能大于 MTU，或者当 MTU 大小不知道时，长

度要小于 1500bytes。

最近研究表明 1500bytes 是一个可以接收的假设。MTU 下一层的 SLIP 协议的 MTU 值为 1006bytes 和 296bytes（对于低水平的端对端协议）。另外一个合适的值是 950Bytes（对于 SLIP）。

SIP 消息有两种类型，一种是从客户端发向服务端的请求，另外一种是服务端给客户端的响应。

SIP-Message = Request | Response

SIP 请求和响应消息都用 RFC822 来传送消息实体。消息包括：

起始行：一个或者更多的头域

空行（CRLF）：表示头域的结束

详见下表格式：

generic-message =	**start-line**	
	***message-header**	
CRLF		
	[message-body]	
start-line	=	Request-Line | ;Section 4.1
		Status-Line ;Section 5.1
message-header	=	(general-header
		| request-header
		| response-header
		| entity-header)

Table 3 Sip Message Structure

为了防止与 HTTP 类似的名字的头冲突，我们参考消息体的域来描述实体的头域。这些在下面的章节有具体的描述。

考虑到协议的健壮性，任何空行开始的消息必须被忽略。换句话来说，请求或者响应的消息以 CRLF，CR 或者 LF 开始，都必须被忽略。

4. Request(请求)

请求的消息格式如下所示：

```
Request = Request-Line          ;  Section 4.1
          *(  general-header
          |  request-header
          |  entity-header )
           CRLF
          [  message-body  ]    ;  Section  8
```

4.1 Request-line（请求行）

请求行以方法标记开始，接着就是 Request－URI 和协议版本，并以 CRLF 结束。这个原理以 SP（空格）字符结束。除了在消息的最后，其他行没有 CR 或者 LF 也是允许的。

Request-Line = Method SP Request-URI SP SIP-Version CRLF

general-header	=	Accept	; Section 6.7
		| Accept-Encoding	; Section 6.8
		| Accept-Language	; Section 6.9
		| Call-ID	; Section 6.12
		| Contact	; Section 6.13
		| CSeq	; Section 6.17
		| Date	; Section 6.18
		| Encryption	; Section 6.19
		| Expires	; Section 6.20
		| From	; Section 6.21
		| Record-Route	; Section 6.29
		| Timestamp	; Section 6.36
		| To	; Section 6.37
		| Via	; Section 6.40
entity-header	=	Content-Encoding	; Section 6.14
		| Content-Length	; Section 6.15
		| Content-Type	; Section 6.16
request-header	=	Authorization	; Section 6.11
		| Contact	; Section 6.13
		| Hide	; Section 6.22
		| Max-Forwards	; Section 6.23
		| Organization	; Section 6.24
		| Priority	; Section 6.25
		| Proxy-Authorization	; Section 6.27
		| Proxy-Require	; Section 6.28
		| Route	; Section 6.33
		| Require	; Section 6.30
		| Response-Key	; Section 6.31
		| Subject	; Section 6.35
		| User-Agent	; Section 6.39
response-header	=	Allow	; Section 6.10
		| Proxy-Authenticate	; Section 6.26
		| Retry-After	; Section 6.32
		| Server	; Section 6.34
		| Unsupported	; Section 6.38
		| Warning	; Section 6.41
		| WWW-Authenticate	; Section 6.42

Table 4: SIP headers

4.2 Methods（方法）

方法如下面所定义。没有被 Proxy **和** Redirect Server **支持的方法也被接收，就如他们是可选的方**

法,并根据情况转发,没有被 User agent server 或者 registrar 支持的方法将产生一个 501 的返回(Section 7)。如果在 HTTP，方法标记是大小写敏感的。

Method = "INVITE" | "ACK" | "OPTIONS" | "BYE"
| "CANCEL" | "REGISTER"

4.2.1 INVITE（邀请）

INVITE 邀请一个用户或者服务参加一个会话。这个消息体包含一个会话描述，指明哪一个被叫被邀请。对于两方呼叫，主叫指明它能够接收的媒体类型和发送目的地参数。成功的响应必须在消息体中，指明被叫希望接收和发送的媒体。

注意：不是所有的会话描述格式都能指示发送的媒体。

服务器应该自动的响应邀请，返回 200（OK）响应。对于已经加入会议的用户，应该在会话描述里用 CALL ID 或者全球统一标识区分开来,。

如果 User agent 接收到一个请求，它的 Call leg 比先前同个 Call-ID 的 INVITE 中有更高的 Cseq 数值，此时就必须检查会话描述以前的所有版本，如果没有版本标志符，则认为会话描述的内容改变了。同时也必须检查其他头域，如果改变了，User agent 必须及时更新信息。如果会话描述改变了，User agent 在向 USER 确认后，必须由此调整会话的参数。

INVITE 方法必须被 SIP Proxy，Redirect 和做为客户端的 User agent servers 所支持。

4.2.2 ACK（确认响应）

客户端以 ACK 确认 INVITE 请求的最终响应，（ACK 仅仅用于响应 INVITE 请求）。客户代理都知道 2XX 响应，其他的最终响应由第一个 Proxy 或者 User Agent 响应。Via 域总是初始化（写上）发送 ACK 消息的主机地址。 比如在 User Agent Client 在返回 2XX 响应，或者第一个接收到一个非 2XX 的最终响应之后。这个 ACK 在 Request—URI 响应 INVITE 请求。具体在 Section 10 中说明。

ACK 请求必须包含被叫最后会话的所有描述消息体。如果 ACK 消息体是空的，被叫必须用这个 INVITE 请求的会话描述。

Proxy 发送 3XX,4XX,5XX 或者 6XX 后接收一个 ACK 请求， 收到 ACK 之后必须决定 ACK 是返回给自己的，还是返回给进一步往下传的用户代理或代理服务器。此决定通过检查 to 域中的 tag 来实现。如果此 tag 与先前响应的 to 域中的 tag 相匹配，以及其他的 from,Cseq 和 Call－ID 等域也匹配，则此 ACK 是返回给此代理服务器的，否则此 ACK 应该继续向下传。

注意：User agent client 或者 proxy server 必须接收 3XX，4XX，5XX 或者 6XX 的一个请求路径的响应。这在各种不同的错误条件下都可以发生。典型的，当接收所有的 forking proxy 之前，forking proxy 将从有状态变为无状态。 为了消除歧义，除了每个响应都有不同的 tag（To 域中），不同的响应可以被确认。

此方法必须被 SIP Proxy，Redirect 和 User agent servers 支持。

4.2.3 OPTIONS（可选方式）

这个服务用于查询主机的性能，服务器确信可以联系到用户，例如用户所登记的并且最近处于活动状态的用户代理，可以对此请求做出响应，返回一个能力集，比如：600（busy）。象这样的服务应该返回一个 Allow 头域，指示支持的方法。Proxy 和 Redirect Server 没有指明他们的能力集，而简单的转达他们的请求。

这个方法应该被下列服务支持：SIP Proxy，Redirect，User agent Servers, Registrars 和 Clients 所

支持。

4.2.4 BYE（结束方式）

User agent client 用 BYE 向服务器表明释放这个呼叫。BYE 请求可以象 INVITE 一样被转发，并且可以由主叫或者被叫发起。呼叫的一方应该在释放呼叫之前发起一个 BYE 呼叫["hanging up"]。接收到 BYE 请求后必须停止媒体流的传输。

如果 INVITE 请求包含一个 Contact 头，被叫将发送一个 BYE 请求到 From 地址。

这个方法应该被 Proxy Server，Redirect 和 User agent 支持。

4.2.5 CANCEL（会话取消）

CANCEL 请求用（与 INVITE）相同的 Call ID，To，From 和 Cseq 头域，取消一个触发的请求，但不是一个完整的请求。（如果服务器返回一个最终的响应，请求才被认为是完整的）。

User agent client 或者 Proxy client 可以在任何时候发出 CANCEL 请求。特别是代理可以选择向某目的地发送 CANCEL 请求，此目的地在接收到一个 2XX 或者 6XX 响应后（对一个或多个并发搜寻请求所做出的）还未返回一个最终响应。接收到 CANCEL 请求后，代理应该将此请求转发到悬而未决的请求的目的地址处。

CANCEL 请求中的 Call-id、From、To 和 Cseq（中的序列号部分）头与初始请求中的一致。这一点保证了 CANCEL 请求与它所取消的请求相匹配。然而，为了保证能够区分 CANCEL 和原始的请求，CANCEL 方法中设置有 Cseq Method 成分。Via 头域由发送 CANCEL 的代理初始化。（如此，对此 CANCEL 作出的响应将只到达此代理。）

一旦 UAS 已经接收到 CANCEL 请求（而又接收到另外的 CANCEL），UAS 不必发送一个 2XX 回应它。

Redirect 或者 UAS 收到 CANCEL 请求，必须响应一个 200（OK）的状态码。否则，将不会有进一步的动作，存在的呼叫视为无效。

注意：BYE 请求不能用于取消并行分支的请求，因为几个分支可以通过交换媒体的代理，找到同一 UAS 并结束这个呼叫。结束未决的寻找，UAC 必须用 BYE 这个方法，而不是用 CANCEL。当 CANCEL 结束 INVITE 请求，非常清楚的看到，200 响应给 INVITE 还是 CANCEL，在 Cseq头域的方法项中加以区别。

这个方法应该被 Proxy Servers 和所有其他类型的 SIP Server 类型。

4.2.6 REGISTER （注册）

客户端使用 REGISTER 方式来将 To 头域中所列的地址登记在一个 SIP 服务器上。

REGISTER 请求在接收时按照发送的顺序接收。User agent 必须注册到本地的服务器，在启动的时候发送一个注册请求到已知的“ALL SIP SEVER” —— 多点地址“sip.mcast.net”(224.0.1.75)。这个请求应该有一个范围去保证不能超出管理系统的边界。依靠所在网络，这个可以通过或者 TTL 或者管理范围来完成。SIP User Agent 应该侦听该地址并变成其他本地用户的位置；然而，他们没有响应请求。User agent 可以与注册服务器配在同一地址上，User agent 也是在启动的时候，注册服务器初始化。

REGISTER 请求在接收的过程是按发送的顺序接收的。客户端如果没有收到上一个注册的响应，将不会发送新的注册。

注意：客户端可以在不同的 Location 注册，并且有必要用不同的 CALL－ID，这样 Seq 值就没有必要强制排序，因为注册是活动的，如果完全代替以前的值注册不会出现问题。

REGISTER 请求的头域的含义定义如下。定义“address-of-record”作为 SIP 地址。这个址知道如何注册。典型的有 User@domain 和“User@host。在第三方注册，实体发生一个请求是不同于实体被注册。

To：将被建立或者被修改的登记的 address-of-record；

From：登记的个人的 address-of-record，对于单方登记，与 to 头域的值一致。

Request-URI：命名登记请求的地址，也就是 registrar 的域名（domain）。User 名为空（user@domain）。通常此 domain 与 to 头域有相同的值；然而，此 domain 可能作为一个“visitor”登记而使用一个名字。一旦 REGISTER 请求到达一个在它的 Request-URI 中所列的授权域中的服务器时，它就不再继续发送。

Call-ID：来自同一个客户的所有的登记应该使用同样的 Call-ID 头值，至少是在同一个重新启动的循环中。

Cseq：同一个 Call-ID 的登记必须增加其 Cseq 值。同时，服务器不拒绝无次序的请求。

Contact：在 to 头域中给出的 non-REGISTER 请求应该被发送到 Contact 头所指示的地址处。

如果 REGISTER 中不包含 contact 域，则登记保持不变。

服务器会对于登记的生命期设置自己的时间参数。当在此时间段内，登记认为被更新，则应该被撤销。对于一个登记的响应应该包含一个期限域（Expires）或者期限 contact 参数，表明服务器撤销此登记的时间。一般默认为一个小时。用户可以申请一个登记的生命期，通过在此请求的 Expire 域中表明时间。

用户通过发送 REGISTER 请求来取消一个已存的登记，在此请求中，Expires 域的值为零秒。如果想取消全部的注册，则在 CONTACT 中，填上“*”号。

服务器应该在 200 响应的 Contact 头域中返回当前的登记列表。

REGISTER 请求被验证时是特别重要的，因为它允许重定位以后的请求（Section 13.2）

除了作为一个简便的定位服务以外，当在一个主机上有多个 SIP 服务器时，此 REGISTER 方式也是需要的。在这种情况下，这些服务器中只能有一个可以使用默认（缺省）的端口号。

4.3 Request-URI（请求 URI）

Request-URI 是 SIP URL 或者一个普通的 URI，它指示用户（服务）要到达的请求所定位的地址。跟 To 域不同的是，Request-URI 域可以由 proxy 来填写。（URL 是目标地址，URI 是动态的临时的地址）.

当使用 Request-URI，SIP URL 禁止包含 transport-param、maddr-param、ttl-param，或者头元素。如果一个服务器接收到含有这些参数的 Request-URL，在进一步处理前将删除这些参数。

注：典型的，假如维持一个很长的时期，UAC 设置 Request—URI 和 To 域到同一 SIP URL。然而，如果 UAC 已经获得一个更直接到被叫的路径，比如，对于前一响应 Contact 头域，To 域将包含一个长期的地址，而 Request-URI 将设置到临时的地址。

Proxy 和 redirect 服务器可以使用 Request-URI 和请求头域中的信息来处理请求，同时可能重新改写 Request-URI。

比如：一个请求地址 sip:sales@acme.com 是代理域特殊的人，如 SIP：bob@ny.acme.com。在 To 域中保持 SIP:sales@acme.com.在 ny.acme.com,Bob 指明 Alice 临时代替它。

Request-URI 的 host 部分应该与接收它的服务器的某个主机名一致。否则，服务器将此请求代理到所指示的地址，或者返回一个 404（Not Found）响应，如果它不这样作或者不能这样作（指前者）。例如：Request－URI 和 Server 主机否决处理 Outgoing Call 的防火墙代理。这个操作模式类

似于 HTTP 代理。

如果 SIP 服务器接收到一个请求，其 URI 指示了一个 scheme 而不是 SIP，而服务器又无法识别，此时服务器必须返回一个 400（Bad Request）响应。服务器必须这样作即使 To 头域中包含了服务器可识别的 scheme，因为 proxy 负责处理 Request-URI；To 域指示的是对端对端（end-to-end）的地址。

4.3.1 SIP Version（SIP 版本）

在请求和响应中均包含所使用的 SIP 版本，根据规定，发送 SIP 消息的应用程序必须使用 SIP 版本——“SIP/2.0”。

4.4 Option Tags（选项标签）

Option tags 是用来在 SIP 中指定新的选项的唯一标识，用在 Require(Section 6.30)和 Unsupport 头域(Section 6.38)中。

Option-tag = token

新的 SIP 选项的建立者或者用他们域名的倒转作为选项的前缀,或者将新的选项登记到 IANA 上。Option tags 区分大小写。

Section C 有 Token 的定义。新 SIP 功能的创建者用 IANA（Internet Assigned Numbers Authority），或者前缀相反域名或者注册新的功能。比如,“com.foo.mynewfeature”是一个倾向的特性,将可以达到“foo.com”。独立的机制确保可选名不冲突。可选的 IANA 注册有一个“org.iana.sip.”在 RFCs 可选的描述有前缀“org.ietf.rfc.N”，N 是 RFC 的数。可选的标志是大小写敏感的。

4.4.1 在 IANA 上登记新的 Option Tags

当登记一个新的 SIP option 时，应提供以下的信息。

--option 的名字与描述符。名字只能有字母组成，不超过 20 个字母。名字只能是表 3 所包含的字符集。

--指出改变此 option 上协议的组织或者公司。

--进一步描述的参考资料。如果可能，比如 RFC，印刷纸，专利表单，技术报表，文件资源代码或者计算机手册。

--联系方式（邮政的和 Email）。

登记应发送到 iana@iana.org

这个过程已经借用了 RTSP 到 RTP[4]/AVP[26].

5. Response（响应）

在接收到并且解释了一个请求信息后，接收者应该作出相应的响应。响应格式如下所示：

```
Response  =  Status-Line
             * (general-header | response-header | entity-header)
             CRLF
             [message-body]
```

5.1 Status-Line（状态行）

响应消息的第一行就是状态行，包括协议的版本，数字的状态码，相关的文本分析。他们之间的

每一项是以空格键分开的。最后一行可以 CRLF，没有 CR 或者没有 LF 也是允许的。

Status-Line = SIP-veision SP Status-Code SP Reason-Phrase CRLF

5.1.1 Status Code&Reason Phrase（状态码和原因短语）

Status Code 是一个 3 位的整形结果，表明对请求的输出。Reason Phrase 对 Status Code 进行简短的描述。Status Code 主要的作用是自动操作，而 Reason Phrase 是方便人工观察。在客户端并不需要将 Reason Phrases 显示出来。

```
Status-Code     =  Informational                    ;Fig. 5
                /  Success                         ;Fig. 5
                /  Redirection                     ;Fig. 6
                /  Client-Error                    ;Fig. 7
                /  Server-Error                    ;Fig. 8
                /  Global-Failure                  ;Fig. 9
                /  extension-code
extension-code  =  3DIGIT
Reason-Phrase   =  *<TEXT-UTF8, excluding CR, LF>
```

下面提供状态码的总体结构，在 Section 7 将提供完整的定义。总体码的第一个数字表示响应的类型。其他两位只是具体的代码，没有类别的作用。 SIP/2.0 允许第一个数字有 6 个值。

1xx：Informational—接收到请求，继续处理请求；
2xx：Success—动作被成功接收到、理解和接受；
3xx：Redirection—为完成请求需要采取进一步的行动；
4xx：Client-Error—请求包含错误的语法或者无法在此服务器上实现；
5xx：Server Error—请求明显有效，但服务器无法实现。
6xx：Global Failure—此请求在任何服务器上都不能实现。

Reason phrases 仅仅是推荐的，被本地设备代替后，也不影响这个协议。SIP 采用了许多 HTTP/1.1 的响应代码。SIP/2.0 增加从 *X80* 开始返回代码，以避免和新的 HTTP 代码冲突。

SIP 响应代码可以扩展。SIP 应用程序并不需知道所有的代码的意思，但必须知道每一个代码所属类型（由代码的第一个数字指示）。当接收到一个不认识的代码时，将其与同一类型的 *X00* 代码同样处理（除了特殊情况，此不认识的代码禁止存放于缓冲区）。比如收到响应代码是 431，知道这可能出了什么问题，这时候就当做 400 处理。在这种情况下，user agent 应该将响应返回的消息体传给用户，应为此消息体有可能包含着用户可以识别的信息，这个信息解释了此异常出现的状态的原因。

```
Informational  =  "100"  ;  Trying
               |  "180"  ;  Ringing
               |  "181"  ;  Call Is Being Forwarded
               |  "182"  ;  Queued
Success        =  "200"  ;  OK


Redirection  =  "300"  ;  Multiple Choices
```

```
                    |   "301"  ;  Moved Permanently
                    |   "302"  ;  Moved Temporarily
                    |   "303"  ;  See Other
                    |   "305"  ;  Use Proxy
                    |   "380"  ;  Alternative Service
```

Figure 6: Redirection status codes

```
     Client-Error  =  "400"  ;  Bad Request
                    |   "401"  ;  Unauthorized
                    |   "402"  ;  Payment Required
                    |   "403"  ;  Forbidden
                    |   "404"  ;  Not Found
                    |   "405"  ;  Method Not Allowed
                    |   "406"  ;  Not Acceptable
                    |   "407"  ;  Proxy Authentication Required
                    |   "408"  ;  Request Timeout
                    |   "409"  ;  Conflict
                    |   "410"  ;  Gone
                    |   "411"  ;  Length Required
                    |   "413"  ;  Request Entity Too Large
                    |   "414"  ;  Request-URI Too Large
                    |   "415"  ;  Unsupported Media Type
                    |   "420"  ;  Bad Extension
                    |   "480"  ;  Temporarily not available
                    |   "481"  ;  Call Leg/Transaction Does Not Exist
                    |   "482"  ;  Loop Detected
                    |   "483"  ;  Too Many Hops
                    |   "484"  ;  Address Incomplete
                    |   "485"  ;  Ambiguous
                    |   "486"  ;  Busy Here
```

Figure 7: Client error status codes

```
Server-Error  =  "500"  ;  Internal Server Error
               |   "501"  ;  Not Implemented
               |   "502"  ;  Bad Gateway
               |   "503"  ;  Service Unavailable
               |   "504"  ;  Gateway Time-out
               |   "505"  ;  SIP Version not supported
```

Figure 8: Server error status codes

```
Global-Failure |  "600"  ;  Busy Everywhere
               |  "603"  ;  Decline
```

```
| "604"  ;  Does not exist anywhere
| "606"  ;  Not Acceptable
```

Figure 9: Global failure status codes

6. Header Field Definition（头域定义）

SIP 头域在语法和语义上与 HTTP 头域是一致的。

Hop-by-hop(点对点)的头域必须出现在 end-to-end（端对端）的任何头域之前。 Proxy 不应该改变此顺序。Proxy 能增加 Via 域，可以增加其他的 hop-by-hop 域。

他们可以确定特定的头域，比如 Max-Forwards(Section 6.23) “ Fix up ”这个 Via 域用在 Section6.40.1 中描述的“ receive”。

Proxy 禁止改变任何已经得到验证的域（Section 13.2)。头域请求，可选并没有应用每个列在表 4 和表 5 的方法。表用“ o ”表示可选的，用“ m ”表示强制的，而“ - ”表示没有应用。“ * ”表示如果消息体不为空，则是必须的。

“ where ”列描述头域中请求或者响应中可以用的类型。“ R ”指的是可以被请求的头域（那就是，请求并产生这个头域)。“ r ”指明一个响应或者所以可适用的通用头域。数字值表明可以用的头域状态码。“ g ”和“ e ”指明通用（Section 6.1）和各自的实体头域（Section 6 .2)。如果头域标志为“ c ”，表明有请求 copy 到响应。

“ enc. ”列表明这个消息头域是否应该被 end-to-end 加密。“ n ”表明不能加密，而“ c ”表明如过需要，也可以被加密。

对于“ e-e ”列，“ e ”代表 end-to-end 而“ h ”代表 hop-to-hop。

	where	enc.	e-e	ACK	BYE	CAN	INV	OPT	REG
Accept	R		e	-	-	-	o	o	o
Accept	415		e	-	-	-	o	o	o
Accept-Encoding	R		e	-	-	-	o	o	o
Accept-Encoding	415		e	-	-	-	o	o	o
Accept-Language	R		e	-	o	o	o	o	o
Accept-Language	415		e	-	o	o	o	o	o
Allow	200		e	-	-	-	-	m	-
Allow	405		e	o	o	o	o	o	o
Authorization	R		e	o	o	o	o	o	o
Call-ID		gc	n	e	m	m	m	m	m	m
Contact	R		e	o	-	-	o	o	o
Contact	1xx		e	-	-	-	o	o	-
Contact	2xx		e	-	-	-	o	o	o
Contact	3xx		e	-	o	-	o	o	o
Contact	485		e	-	o	-	o	o	o
Content-Encoding	e		e	o	-	-	o	o	o
Content-Length	e		e	o	-	-	o	o	o
Content-Type	e		e	*	-	-	*	*	*
CSeq		gc	n	e	m	m	m	m	m	m

Date	g		e	o	o	o	o	o	o
Encryption	g	n	e	o	o	o	o	o	o
Expires	g		e	-	-	-	o	-	o
From	gc	n	e	m	m	m	m	m	m
Hide	R	n	h	o	o	o	o	o	o
Max-Forwards	R	n	e	o	o	o	o	o	o
Organization	g	c	h	-	-	-	o	o	o

Table 4: Summary of header fields, A—O

（包括：适合的请求和响应类型；是否需要在 end-to-end 中加密；属于 end-to-end 类型还是 hop-by-hop 类型；适合的方式）

他的头域可以根据需要增加；服务器必须忽略未在说明中定义的头域；proxy 禁止删除或者改变未在说明中定义的头域；同时，当请求必须封装在一个特定的包中，且包的大小已经确定（发送）时，可使用 UDP 方式的头域的压缩方式（Section 9）。

6.1 General Header Fields（通用头域）

通用头域适用于请求和响应域。通用头域的域名只有在协议版本改变时才可有效地扩展。不过，通信中的所有方均认为是“通用头域”的新的头域也是通用头域。不被认可的头域作为实体头域。

6.2 Entity Header Fields（实体头域）

实体头域定义了消息体信息之后的内容（如：Content-Length、Content-Type、Content-Encoding），或者如果没有消息体，则定义请求所指示的资源。

在响应体可以包含一个转换消息体的转换版本的地方，术语“实体头”是 HTTP 1.1 的术语。初始消息体参考“实体”。我们对于头域将保持同样的术语，但经常是参考消息体，因为两者是相同的。

6.3 Request Header Fields（请求头域）

头域客户将关于请求和关于客户自己的其他信息传送给服务器。这些域类似于请求的变量，语义上相当于可编程语言方式调用的参数。

结合协议版本的变化，请求头域可以被可靠的扩展。如果所有的通信方认为他们成为一个请求域，新的或者试验的头域可以由请求域的语义给出。不认识的头域将被认为是实体域。

6.4 Response Header Fields（响应头域）

响应头域允许服务器发送关于响应信息，这些信息无法放在 Status-Line 中。这些头域给出了由 Request-URL 指示的资源，关于服务器和进一步访问资源定位器。

响应头域的扩展与通用头域相同。

6.5 End-to-end and Hop-by-hop Headers（端到端&点到点的头域）

End-to-end 头在经过所有的 proxy 传输时禁止修改，而 Hop-to-hop 头可以被 proxy 修改或者

增加。

6.6 Header Field Format（头域格式）

头域（"general-header", "request-header", "response-header",and "entity-header"）允许同一类头的格式。正如如 Section 3.1 所给的格式。每个头域包含一个名称，“ :”和域值。头域名不区分大小写。这个域前可以由任意数量的 Leading White Space，当然，单个的空格是首先的。头域能够被扩展成多行，每一行之间必须由 SP 或者 Tab 字符。因为可能存在一些任务，它们不能执行普通表单之外的任务，所以应用必须依据普通表单产生结构。

```
message-header  =  field-name ":" [ field-value ] CRLF
field-name      =  token
field-value     =  *( field-content | LWS )
field-content   =  < the OCTETs  making up the field-value
                     and consisting of either *TEXT-UTF8
                     or combinations of token,
                     separators, and quoted-string>
```

不同的头域名之间的相对顺序并不重要。同一个头域名的多个头域可以存在于一条消息中，条件是此头域的整个 field-value 定义为用“ ,”分隔的格式。可以联合多个头域，成为一个数据对"field-name: field-value"，没有改变消息的语义，从第一个增加序列的域值，并以逗号格开。因为域名跟域值的对应关系相当重要，所以当消息传输时，proxy 禁止改变同一个头域名的域值的顺序。

域名大小写不敏感，但是域值大小写敏感。

6.7 Accept（接受）

Accept 域用于 INVITE、OPTIONS 和 REGISTER 请求方式中，指示在响应中能够接收的媒体的类型（指服务器应该接收的，客户所用的媒体类型）（缺省值为 application/sdp）。

详见[H14.1]。

Example:

Accept: application/sdp;level=1, application/x-private, text/html

6.8 Accept-Encoding（接受的编码）

Accept-Encoding 请求头域与 Accept 头域相似，但它限制在响应中可接受的 content-codings（详见[H3.4.3]、[H14.3]）。

6.9 Accept-Language(接受的语言)

Accept-Language 头域遵循[H14.4]定义的语法。定义语言的规则基于同样应用于 SIP 的 Q 参数。但我们用 SIP，Accept-Language 请求头域将服务器愿意接收的语言，包括消息体的下面几个部分：原因短语，会话描述，状态响应代码。代理将用这些域选择呼叫的目的地，比如，主叫所说的，又是人工操作者熟悉的语言。

Example:

Accept-Language: da, en-gb;q=0.8, en;q=0.7

6.10 Allow（允许）

Allow 实体头域列出了 Request-URI 指示的资源所支持的方式集。此域的目的是通知接收者与资源相联系的有效方式。在 *405（Method Not Allowed）* 响应中必须有 Allow 头域；在 OPTIONS 响应中应该有 Allow 头域。

Allow = “Allow ”“ :” 1#Method

6.11 Authorization（鉴权）

用户代理经常希望通过服务器来验证自己，不过这并不是必需的。在接收到 401（Unauthorized）响应后，可以在请求中包含一个 Authorization 请求头域。Authorization 域值由包含了用户代理对所做出的验证信息的信任组成。

Section13．2 介绍了 Authorization 头域的用法。

Section 15 描述了基于 PGP 的 authorization 的语法和语义。

6.12 Call-ID（呼叫标识）

Call-ID 通用头域唯一标识一个请求或者一个客户的注册。注意到单个的多媒体会议可以用几个不同 Call-ID 产生几个呼叫，例如，多次邀请一个用户加入同一个会议。

对于一个 INVITE 请求，如果用户先前已经对 INVITE 请求中的 Call-ID 作出了响应，主叫方用户代理服务器不应该警告用户。如果用户已经是会议的一个成员，同时包含在会话描述中的会议参数并未改变，那么被叫方用户代理服务器将不管 Call-ID，接受此呼叫。对于一个已存在的 Call-ID 邀请，或者可以改变会议参数的会话，客户应用可以决定向用户简单地指示会议参数已经改变，可以自动接受邀请或者可能需要用户的确认。

使用几个不同的 Call-ID 可以邀请一个用户加入同一个会议或者呼叫。如果需要的话，可以使用在会话描述中的标识来检测此副本。例如，SDP 的“ o ”域中包含了会话标识和版本号。

REGISTER 和 OPTIONS 方式使用 Call-ID 值来精确匹配请求和响应。一个单个的客户发布的所有的 REGISTER 请求应该使用同一个 Call-ID，至少在同一个有效循环中。

格式如下：

Call-ID = (“Call-ID” | “i”)”:”local-id”@”host

*Local-id = 1*uric*

i 是 Call-ID 的缩写形式。

“host”应该是一个真正的域名或者是一个全球性的 IP 地址。如此，”local-id”应该是一个由 URI 字符组成的标识，此标识在”host”中是唯一的。 建议使用经过加密的随机标识。因为有些域名不是 FQDN（正式域名），或者可以全球路由到达的 IP 地址（比如一个网卡，10 个地址），Local－ID(本地的 ID)这样能够在主叫内部定位。

Call-ID 区分大小写。

Call-ID 保证全球唯一。

Call-ID 的值禁止被重用于另一个不同的呼叫。

注意 :使用经过加密的随机标志防止了会话的拦截。Call-ID,To 和 From 确定一个 Call Leg。Call 和 Call Leg 关系到第三方的控制。

例子：

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@foo.bar.com
```

6.13 Contact（互通）

Contact 通用头域可出现在 INVITE、ACK 和 REGISTER 请求中，1XX、2XX、3XX 和 485（不明确的）响应中。通常，它提供了一个 URL，用户可以通过此 URL 来进行进一步的通信。

INVITE 和 ACK 请求：Contact 域表明请求从哪个位置发起；

这允许被叫方直接向主叫方发送请求，如 BYE，而不是通过一系列的代理。由于所想要的地址可能是代理的地址，所以只 Via 头域并不够。

INVITE 2XX 响应：一个用户代理服务器在发送一个限定的、肯定的响应（2XX）时，可以加入一个 Contact 响应头域，表明对于将来的有同一个 Call-ID 的请求它可以最直接到达的 SIP 地址，如 ACK 请求，就用同一 CallID。Contact 头域包含了服务器自己或者代理的地址，例如在一个防火墙之后的主机。

如果响应未包含 Record-Route 头,此 Contact 的值将复制到此呼叫的后来的请求的 Request-URI 中；如果响应包含了 Record-Route 头域， Contact 域的值将作为最后一项增加到 Record-Route 域中。

Contact 的值不应该在创建呼叫时被暂时存储起来，因为它往往不是最想要的位置。

INVITE 1XX 响应：UAS 发送一个临时的响应（1XX）时可以插入一个 Contact 响应域。

REGISTER request：REGISTER 请求中的 Contact 域表明用户的位置。 REGISTER 请求定义了一个通配的 Contact 域，“ * ”。 这种用于请求过时，并删除所有登记的用户。

可选的“ expires ”参数指示登记所想要的期限。如果 Contact 未使用此参数，则 Contact 域的值将使用默认值。如果这些机制都未采用，SIP URL 的期限为一个小时。其他的 URL 没有期限时间。

REGISTER 2XX 响应：REGISTER 注册响应返回当前用户可以到达的所有地址。

3XX 和 485 响应：Contact 头域指示一个或多个可选的地址。可以出现在 INVITE、BYE 和 OPTIONS 方法的响应中。Contact 头域包含的 URI 给出了新的位置和用户名，或者简单地说明其他的传输参数。*300 (Multiple Choise)*、*301(Moved Permanently)*、*302(Moved Temporarily)*或者 *485(Ambiguous)响应*应该包含一个含有可尝试的新地址的 URL 的 Contact 域。301 和 302 响应可以给出正在尝试的同样的位置和用户名，但指定了其他的传输参数，如一个不同的服务器或者多点地址，或者一个从 TCP 到 UDP，UDP 到 TCP 的 SIP 事务的改变。

客户将 Contact-URL 中的“ user ”、“ password ”、“ host ”、“ port ”、“ user-param ”复制到重定位请求的 Request-URL 中，同时使用 tranport 参数中的传输协议， 将此请求传到“ maddr ”和“ port ”参数所说明的地址处。如果“ maddr ”是一个多点地址，“ ttl ”值表明 time-to-live 值。Contact 头域可能指示一个不同于原始呼叫实体的实体。例如，与 GSTN 网关相连的 SIP 呼叫可能需要发送一个特殊的消息通知。Contact 头域可以包含任何合适的 URL 来指示被叫方的位置，而不只限于 SIP URL。比如，SIP 呼叫连接到 GSTN 网关，需要传送一个特别的信息宣布，比如“ 比较已经改动。

Contact 头域指示包含所有 URI 包含呼叫方到达的地方，没有限制 SIP 的 URLs。比如，可以包含 Phones,fax,irc 或者 mailto:URL。已经定义了下面的参数，附加的定义在其它说明书里。

q：“ qvalue ”指示所在服务器的相关性能（也可以说所在选项的推荐程度，q 越高，表明越推荐使用此项）。“ qvalue ”是一个从 0 到 1 的十进制数值，更高的值指示更好的性能。

Action：此参数用于 REGISTER 注册请求的时候。它表明 Server Proxy 或者 Redirect 对客户端是否有更多的请求。如果这个参数没有确定，那将取决于配置服务器。在这个响应中，注册应该指明使用模式。这个参数在其他请求中被忽略。

Expires:指明 URI 的有效时间。这个参数或者是一个表明秒的数值，或者是一个包含 SIP-date 的字符串。如果 Contact 中存在 expires 参数，则使用其表示的时间；若不存在，则使用 Expire 头域所表示的时间。如果值大于 2**32-1(13 年)，那么将截取为 13 年。

```
Contact=(“Contact”|“m”)”:”(“*”|(1#((name-addr|addr-spec)
          [*(“;”contact-params)][comment])))
name-addr=[display-name]”<”addr-spec”>”
addr-spec=SIP-URL / URI
display-name=*token | quoted-string
contact-param= “q”  “=”qvalue
              / “action”   ”=””proxy” / “redirect”
              / ”expires”  “=”delta-seconds / <”>SIP-date<”>
              / extension-attribute
extension-attribute = extension-name [“=”extension-value]
```

仅仅允许一个地址结束引语。因为 URIs 将逗号和分号作为保留字符，所以在参数分析时就会出现错误。对于 To 和 From 的头域，目前的语法允许使用显示名。

Example:

```
Contact: "Mr. Watson" <sip:watson@worcester.bell-telephone.com>;q=0.7; expires=3600,
    "Mr. Watson" <mailto:watson@bell-telephone.com> ;q=0.1
```

6.14 Content-Encoding（编码内容）

形式：

```
Content-Encoding=(“Content-Encoding” / “e”)”:”1#content-coding
```

Content-Encoding 实体头域作为“media-type”的一个修饰语。它的值指示适用于实体消息体的其他的内容编码，指示为了获得 Content-Type 头域所给出的 media-type，必须使用的编码方案。

Content-Encoding 主要用于压缩消息体，而不丢失它底层的媒体类型的标识。

如果多个编码适用于一个实体，那么内容便必须按照他们被应用的顺序列出。

所有的 Content-Encoding 值都区分大小写。IANA 是内容编码标志的注册处。关于 Content-Coding 的语法定义，请参考【3.5】。

客户可以将内容编码应用于请求消息体中。如果服务器不能对消息体解码，或者不能识别任何的 Content-Encoding 值，它必须发送一个 *415(Unsupported Media Type)响应*，在 Accept-Encoding 头域中列出可以接受的编码。

服务器可以将内容编码应用于请求消息体中，但它只能使用请求的 Accept-Encoding 头域中所列出的编码。

6.15 Content-Length（内容长度）

Content-Length 实体头域指示消息体的长度。形式上以八个比特为一个字节。

```
Content-Length = (“Content-Length” / “l”)”:” 1*DIGIT
```

应用程序应该使用此域来指示所传送的消息体的大小，而不管实体所用的媒体类型。Content-Length 的值应为非负数，0 表示没有消息体。

服务器如果收到一个不包含 Content-Length 域的 UDP 请求，那么它便认为此请求压缩了包的剩余部分。否则，如果它的值比消息体的实际长度大，客户则应产生一个 400 类的响应。

在 UDP 包中，如果接收完消息体的最后一个比特后，还有其他的数据，服务器必须将此数据视为另一个消息。服务器如果收到一个包含有 Content-Length 域的 UDP 请求。但它的值比消息体的实

际长度大，客户则应产生一个 400 类的响应。

也就是说，允许在一个 UDP 包中包含有多个消息。

如果一个响应中未包含 Content-Length，客户便认为此响应压缩了 UDP 包或者数据的剩余部分，直到关闭 TCP 连接。Section 8 描述了如何决定消息体的长度。

6.16 Content-Type（内容类型）

Content-Type 实体头域指示发送给接收者的消息体的媒体类型。媒体类型成分定义在[H3.7]。

Content-Type= ("Content-Type" | "c") " :" media-type

Example:

Content-Type: application/sdp

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

6.17 Cseq（命令序列）

对于每一个请求，客户必须使用 Cseq（Command sequence）通用头域。此头域包含了请求方法，和一个提出请求的客户所设定的十进制序列数，在同一个 Call-ID 中此 Cseq 值唯一。此序列数必须为一个 32 位的无符号整数，它的初始值是任意的，但必须小于等于 2**31。连续的请求在请求方式、头域和消息上是不同的，但有同样的 Call-ID，它们的 Cseq 也必须是严格单调增加的相邻的序列数；序列数不能形成环。重传请求用相同的 Cseq，但消息体或者头域不同（即是修改）的 INVITE 请求需要一个更高的 Cseq。 服务器必须在它的响应中回送请求中的 Cseq（注意，必须相同）。如果在所接收的 Cseq 头域中没有 method，服务器应该正确的填充。

ACK 和 CANCEL 请求必须包含与它们相关的 INVITE 请求同样的 Cseq。而当 BYE 请求释放一个请求时必须含有以更高数值的 Cseq。 如果 BYE 请求中的 Cseq 值不高，那么将产生一个 *400（Bad Request）响应*。

用户代理服务器必须记住同一个 Call-ID 的 INVITE 请求中 Cseq 的最高序列数。对于包含较低序列数的任何 INVITE 请求，服务器必须作出响应，然后放弃。

所有在并行搜寻中产生的请求拥有和触发此并行搜寻的请求一样的 Cseq 值。

格式：

*Cseq = "Cseq" " :" 1*DIGIT Method*

注意：对于任何可以被 BYE 或 CANCEL 请求取消的 SIP 请求，或者对于客户发送了连续的几个同一个 Call-ID 的请求的情况，都需要使用 Cseq 头域。如果没有序列值，对于 INVITE 请求的响应将会和对于取消（BYE、CANCEL）的响应相混淆。同样，如果网络复制了消息包，或者一个 ACK 请求耽搁了直到服务器发送了另一个响应，客户应该将此旧的响应作为对于一个之后很快重传的邀请的响应。使用 Cseq 头域，不用比较消息体就可以帮助服务器区分不同版本的邀请。

"Method"值使得客户将对于 INVITE 请求的响应和对于一个 CANCEL 请求的响应（一般是 200 响应）区分开来。代理可以产生 CANCEL 请求；如果代理增大序列数，那么有可能与同一呼叫中用户代理以后发送的请求发生冲突。

Cseq 的长度是 32 比特，在同一个呼叫中，服务器应该继承，如果一秒中发一个请求，那么 13 年之后才会重复。Cseq 的初值可以有用户指定，比如可以基于时间的序列等。

Fork 派生的请求必须有同样的 Cseq 值，否则在这些派生的请求，和客户用户代理以后所发送的 BYE 请求之间将含糊不清。

Example:

CSeq: 4711 INVITE

6.18 Date（日期和时间）

Date 是一个通用头域，语法形式如下：

SIP-date = rfc1123-date

Rfc1123-date 的定义请参考[H14.19]。不像 HTTP/1.1，SIP 仅仅支持最近的 RFC1123 的日期格式。

```
rfc1123-date = wkday "," SP date1 SP time SP "GMT"

date1      = 2DIGIT SP month SP 4DIGIT
                  ; day month year (e.g., 02 Jun 1982)
wkday      = "Mon" | "Tue" | "Wed" | "Thu" | "Fri" | "Sat" | "Sun"
month      = "Jan" | "Feb" | "Mar" | "Apr" | "May" | "Jun" |
```

(GMT)：Greenwich Mean Time

例如： Date： Tue, 15 Nov 1994 08:12:31 GMT

当请求或者响应被第一次发送时，Date 头域指示发送日期和时间，所以，重传将使用与相应的初始同样的 Date 头域。

6.19 Encryption（加密）

Encryption 通用头域说明消息内容已被加密。此头域用于对请求和响应的 End-to-End 加密。被加密的请求基于在 To 域中命名的公钥。被加密的响应基于 Response-Key 头域中的公钥。需要注意的是这些公钥不能用于加密。此依赖于所用的特定算法。

对于任何加密消息，至少消息体，或者还有其他的消息头域须被加密。当应用程序接收到包含 Encryption 头域的请求或响应时，应该将消息体解密，同时将已解密的文本与请求行和原始消息头相连接。（注意：如果只有消息体被加密，消息体须以 CRLF 为前缀，以使正确连接。）需要注意的是请求方式和 Request-URI 不能加密。

注意：Encryption 只是提供一种秘密保障：接收者不能保证请求或者响应来自 From 头域所列的地方，只能保证发送者使用接收者的公钥。然而，代理不能修改请求或者响应。

```
Encryption         =  "Encryption" ":" encryption-scheme 1*SP
                      #encryption-params
encryption-scheme  =  token
encryption-params  =  token "=" ( token | quoted-string )
```

标志指明使用的加密形式，这一点在 Section13 中描述。图 10 下面的例子显示应用“PGP –ea”加密的文本消息。

```
INVITE sip:watson@boston.bell-telephone.com SIP/2.0
   Via: SIP/2.0/UDP 169.130.12.5
   From: <sip:a.g.bell@bell-telephone.com>
   To: T. A. Watson <sip:watson@bell-telephone.com>
```

```
Call-ID: 187602141351@worcester.bell-telephone.com
Content-Length: 885
Encryption: PGP version=2.6.2,encoding=ascii

hQEMAxkp5GPd+j5xAQf/ZDIfGD/PDOM1wayvwdQAKgGgjmZWe+MTy9NEX8O25Red
hO/pyrd/+DV5C2BYs7yzSOSXaj1C/tTK/4do6rtjhP8QA3vbDdVdaFciwEVAcuXs
ODxlNAVqyDi1RqFC28BJIvQ5KfEkPuACKTK7WlRSBc7vNPEA3nyqZGBTwhxRSbIR
RuFEsHSVojdCam4htcqxGnFwD9sksqs6LIyCFaiTAhWtwcCaN437G7mUYzy2KLcA
zPVGq1VQg83b99zPzIxRdlZ+K7+bAnu8Rtu+ohOCMLV3TPXbyp+err1YiThCZHIu
X9dOVj3CMjCP66RSHa/eaOwYTRRNYA/G+kdP8DSUcqYAAAE/hZPX6nFIqk7AVnf6
IpWHUPTelNUJpzUp5Ou+q/5P7ZAsn+cSAuF2YWtVjCf+SQmBR13p2EYYWHoxlA2/
GgKADYe4M3JSwOtqwU8zUJF3FIfk7vsxmSqtUQrRQaiIhqNyG7KxJt4YjWnEjF5E
WUIPhvyGFMJaeQXIyGRYZAYvKKklyAJcm29zLACxU5alX4M25lHQd9FR9Zmq6Jed
wbWvia6cAIfsvlZ9JGocmQYF7pcuz5pnczqP+/yvRqFJtDGD/v3s++G2R+ViVYJO
z/lxGUZaM4IWBCf+4DUjNanZMOoxAE28NjaIZOrrldDQmO8V9FtPKdHxkqA5iJP+
6vGOFti1Ak4kmEzOvM/Nsv7kkubTFhRlO5OiJIGr9S1UhenlZv9l6RuXsOY/EwH2
z8X9N4MhMyXEVuC9rt8/AUhmVQ==
=bOW+
```

Figure 10: PGP Encryption Example

因为代理基于 SIP 头域中的转向选择，所以一个加密请求隐藏的头域不能保证可以随着其他确定的非加密请求，一起到达同一目标地址。

6.20 Expires（期限）

Expires 实体头域给出了消息内容活动的日期和时间。

此头域只用于 INVITE、REGISTER 方式。

对于 REGISTER 方式，它可以用于请求和响应头域。在 REGISTER 请求中，它指示登记的有效期限。在响应中，服务器指示所有登记最早的期限时间。服务器可以选择一个比客户请求的时间短的时间间隔，但不能比客户请求的时间长。

对于 INVITE 方式，他可以用于请求和响应头域。在 INVITE 请求中，主叫方可以限制邀请的有效性，例如，客户希望限制搜寻的期限或者会议邀请的期限。用户界面可以将此作为离开屏幕上的邀请窗口的一种暗示，即使用户目前不在工作站上。这同样限制了搜寻的期限。如果搜寻在此期限内未完成，代理将返回一个 *408（Request Timeout）响应*。在 *302 响应*中，服务器可以建议客户最大的重定向期限。

此域的值可以是一个 SIP-date，或者是一个以秒为单位的数字形式。

Expires = “Expires” “:” (SIP-date | delta-seconds)

Two examples of its use are

Expires: Thu, 01 Dec 1994 16:00:00 GM

Expires: 5

6.21 From（源地址）

请求和响应必须包含 From 通用头域，指示请求的初始者。From 域可以包含一个“tag”参数，服务器将 From 头域从请求复制到响应。可选的“display-name”意味着由用户接口提出（执行）。如果客户身份被隐藏，那么系统必须使用显示名“Anonymous”。

此 SIP-URL 禁止包含“transport-param”,“maddr-param”,“ttl-param”,“headers”。接收到含有以上元素的 SIP-URL 的服务器在执行下一步处理之前，应将这些元素删除。

即使“display-name”是空的，如果“addr-spec”包含了逗号，分号，问号，“name-addr”形式也必须使用。

From = (“From” | “f”) “:” (name-addr | addr-spec)
** (“;” addr-params)*

addr-params=tag-param

tag-param=“tag=” UUID

UUID=1* (hex | “-”)

Examples:

From: "A. G. Bell" <sip:agb@bell-telephone.com>

From: sip:+12125551212@server.phone2net.com

From: Anonymous <sip:c8oqz84zk7z@privacy.org>

“tag”可以出现在一个请求的 From 头域中，当共享同一个 SIP 地址的用户的两个实例使用同一个 Call-ID 发出邀请时，必须使用此“tag”。

“tag”必须是全球唯一的，并且是一个经过加密的至少 32 比特的随机数。单个的用户应该在一个 Call-ID 所标识的整个呼叫中保持同一个 tag。

Call-ID、To 和 From 用于标识一个 Call leg。呼叫和 Call-leg 的区别在于多个响应对 Forked 派生请求的呼叫。这个格式类似 RFC822[24]中的头，不同的是用 URI 代替 email 地址。

6.22 Hide（隐藏）

客户使用 Hide 请求头域来指示：它希望对随后的代理和用户代理隐藏 Via 头域指示的路径(Secton6.40)。此头域的使用有两种形式：Hide:route 和 Hide:hop。Hide 头域通常由客户用户代理来增加，但也可以由发送路径上的任何代理增加。

如果一个请求包含了“Hide:route”头域，所有紧跟的代理应该隐藏它们先前的 hop。如果请求包含了“Hide:hop”头域，只有下一个代理应该隐藏它先前的 hop，然后删除此 Hide 选项，除非它也想要保持匿名。

服务器通过使用它所选择的算法，对最顶端的 Via 头域的“host”和“port”部分加密，来隐藏先前的 hop。服务器应该将另外的”salt” 增加到已经经过加密的”host” 和”port”中，以防止下传路径中可能有恶意的代理基于相同的已加密的 Via 头域来猜测路径的前面部分。

被隐藏的 Via 域用“hidden”Via 选项来标记。

能够隐藏 Via 域的服务器必须也能够将所有标记了”hidden”的 Via 域解密，以便执行回路检测。不能隐藏 Via 域的服务器可以在它们的回路检测算法中忽略被隐藏的 Via 域。

需要注意的是：如果被隐藏的头域未被标识，代理需要将所有的头域都解密，来检测回路，以防止其中可能被加密的头域，例如 Hide:Hop，可能在发送的路径上被删除了。

主机禁止增加诸如"Hide:hop"的头域，除非它能确保将一个到此目的地的请求发送到相同的下一个 hop。原因是请求可能会从一个下传的代理通过相同的 hop 回传回来。如果下一个 hop 的选择已固定（调整），此回路应经过下一个 hop 的检测，但（回路）也可以循环任意多次。

对于请求“Hide:route”的客户来说，如果它将此请求发送到一个可信任的代理处，那么它只需要将请求路径保密（私有）。如果数据包中的请求结果直接在主叫/被叫二者的用户代理之间交换，那么隐藏请求路径也是有限的。

除非确实需要将路径保密，否则最好不要使用 Hide 头域；Hide 域的使用给代理增加了额外的处理开销和限制，同时可能产生 *482(Loop Detected)响应*，这种情况在未使用 Hide 头域时可以避免。

Hide = “Hide” “:” (“route” | “hop”)

6.23 Max-Forwards（最大前转数目）

Max-Forwards 请求头域适用于任何请求方式，用来限制前转请求的代理或者网关的数目。当客户跟踪一个出现了错误或者循环的请求链时，这个头域非常有用。

*Max-Forwards=“Max-Forwards”“ :” 1*DIGIT*

Max-Forwards 值是一个十进制的整数，表明了此请求消息允许被前转的剩余次数。

对于接收到包含有 Max-Forwards 头域的请求的代理或者网关来说，它必须检测并且修改此头域先前的值，以便前转此请求。如果域值是 0，那么接收者禁止前转此请求。相反，对于 OPTIONS 和 REGISTER 方式的请求来说，它（接收者）必须将自己作为最终的接收者来作出响应。对于其他的方式，服务器应返回 *483（Too many hops）响应*。

如果接收到的 Max-Forwards 域值大于 0，那么前转的请求中的 Max-Forwards 域的值必须应减 1

Example:
Max-Forwards: 6

6.24 Organization（组织）

Organization 通用头域表明了发送请求或者响应的实体所属的组织。位于某组织边界的代理也可以加入。

客户软件可以使用此头域来过滤呼叫。

Organization =“Organization” “ :” *TEXT-UTF8

6.25 Priority（优先级）

Priority 请求头域指示了客户所认为的请求的紧急程度。

Priority =“Priority” “ :” priority-value
priority-valur=“emergency” | “urgent” | “normal”
| “non-urgent”

建议“emergency”这个值等到生命，肢体或者财产受到威胁时才使用。

Examples:

Subject: A tornado is heading our way!

Priority: emergency

Subject: Weekend plans

Priority: non-urgent

（此头域通常与 Subject 头域联合使用）

6.26 Proxy-Authenticate（代理验证）

Proxy-Authenticate 响应头域必须包含在 407(Proxy Authenticate Required)响应中。此域的值由查询组成，此查询指示了对于 Request-URI 的适合于代理的验证方案和参数。

与在 HTTP 中的使用不同，Proxy-Authenticate 头（在 407 响应中）必须作为对 UAC 的响应上传（UAS->UAC）。在 SIP 中，只有 UAC 可以向代理验证自己。

```
Proxy-Authenticate  = "Proxy-Authenticate" ":" challenge
```

客户应该将用于特定代理服务器的信任保存，对于此服务器的下一个请求的 realm 也保存。信任通常一个特定代理服务器的 Request-URI 的具体值有效。如果一个客户与一个以前需要验证的代理服务器互通，而客户没有对此特定 Request-URI 的信任，那么它可以尝试使用最近所用的信任。如果客户猜测错误，服务器返回 401 响应(Unauthorized)。

6.27 Proxy-Authorization（代理鉴权）

Proxy-Authorization 请求头域允许客户向要求验证的代理来鉴别自己。Proxy-Authorization 头域的值由信任组成，此信任包含了用户代理向代理提供的验证信息和/或被申请的资源领域（realm of the resource requested）。

```
Proxy-Authorization  =  "Proxy-Authorization" ":" credentials
```

与 Authorization 不同，Proxy-Authorization 头域只应用于使用 Proxy-Authenticate 域要求验证的下一个输出的代理。当有多个代理时，Proxy-Authorization 头域被接受信任的第一个输出代理所使用。一个代理可以将信任从客户请求通过中继传到下一个代理，这种方式可以作为一种代理之间合作验证一个给定请求的机制。

6.28 Proxy-Require（需要请求）

Proxy-Require 头域用来指示代理必须支持的一种特征，即是否需要代理。如果不支持，对于客户，任何 Proxy-Require 头域特征必须被代理所否定。

代理服务器将此域与 Require 域等同看待。参考 Section 6.30，有具体的消息机制和用法例子。

6.29 Record-Route（路由复制）

Record-Route 请求和响应头域可以被任何服务器加到请求中，这些服务器坚持以后的同一个 Call leg 的请求使用同样的路径。它包含了一个唯一可达的 Request-URI 来指示代理服务器。每一个代理服务器将它的 Request-URI 加到序列的开始。

服务器将 Record-Route 头域不做改变地复制到响应中（Record-Route 只和 2XX 响应有关）。主叫方 UAC 将 Record-Route 头复制到随后的同一个 Call leg 的请求的 Route 头域中，只是颠倒了请求的顺序，以使第一个进入的离 UAC 最近。如果响应包含了一个 Contact 头域，主叫方的用户代理将它（Contact）的内容作为最后一个 Route 域的内容。除非这将引起循环，否则任何用户必须将任何随后的同一个 Call leg 请求发送到 Route 头域的第一个 Request-URI 中，同时删除此入口。

主叫方用户代理禁止在包含有 Route 头域的请求中使用 Record-Route 头域。

一些代理，例如那些控制防火墙或者在一个自动呼叫分配（ACD）系统中，需要保持呼叫状态，这样就需要接收任何一个此呼叫的 BYE 和 ACK 包。

```
Record-Route = “Record-Route” “:” 1#name-addr
```

代理服务器应该使用“maddr”URL 参数来包含它们的地址，以便保证随后的请求能够准确到达同一个服务器。

```
Record-Route: <sip:a.g.bell@bell-telephone.com>,
<sip:a.bell@ieee.org>
```

6.30 Require（需求）

客户使用 Require 请求头域，通知 UAS 客户希望服务器所支持的选项，以便合适地处理请求。如果服务器不能识别此选项，它必须返回 420（Bad Extension）响应，在 Unsupported 头中列出它不能识别的选项。

```
Require = “Require” “:” 1#option-tag

   Example:
   C->S:    INVITE sip:watson@bell-telephone.com SIP/2.0
            Require: com.example.billing
            Payment: sheep_skins, conch_shells

   S->C:    SIP/2.0 420 Bad Extension
            Unsupported: com.example.billing
```

代理和重定向服务器必须忽略不可识别的特征。如果一个特定的扩展名需要中介设备支持，那么此扩展名必须在 Proxy-Require 域中标记。

6.31 Response-Key（响应公钥）

客户使用 Response-Key 请求头域来申请被叫方用户代理用来加密响应的公钥。

```
Response-Key = “Response-Key” “:” key-scheme 1*SP #key-param
```

如果客户坚持服务器应返回一个经过加密的响应，那么它的请求中应包含一个 Require 头域：Require:org.ietf.sio.encrypt-response。

如果服务器由于某种原因不能加密，它必须跟随通常的 Require 头域之后，返回一个 420（Bad Extension）响应。如果无 Require 头域，但可以加密的话，服务器仍应该加密。

```
Response-Key  =  "Response-Key" ":" key-scheme 1*SP #key-param
key-scheme    =  token
key-param     =  token "=" ( token | quoted-string )
```

6.32 Retry-After（重试）

Retry-After 通用头域用在 *503（Service Unavailable）响应*中，向提出申请的客户指示，此服务预计多长时间无效。用在 *404（Not Found）,600(Busy)和 603（Decline）响应*中，指示被叫方多长时间内再次有效。此域的值可以是 SIP-date 和以秒为单位的整数值。

REGISTER 请求用“Contact: *; expires:0”删除登记时可以使用 Retry-After 头域。此时，Retry-After 域值指示用户多长时间内可以再次到达。登记器对未来的呼叫作出响应时可以包含此消息。

可以使用一个 commend 选项来指示关于重新呼叫的其他消息。“duration”选项参数指示从有效的初始时间开始，多长时间内被叫方有效到达。

```
Retry-After = ” Retry-After” ”:” (SIP-date | delta-seconds)
              [comment] [“:” “duration””=”delta-seconds]
```

6.33 Route（路由）

Route 请求头域决定了请求的路由。每一个主机将删除第一个入口，然后将此请求代理到那个入口所列的主机处，将它作为 Request-URI。

```
Route =” Route” “:” 1#name-addr
```

6.34 Server（服务器）

Server 响应头域包含了关于 UAS 用来处理请求的软件的信息。这个域的语法在[14.39]中定义。

```
Server       = "Server" ":" 1*( product | comment )
product-version = token
```

例如：

Server: CERN/3.0 libwww/2.17

product：所使用的服务器；

comment：服务器中的重要部分。

如果响应通过代理来前转，那么代理禁止修改此 Server 响应头域，它应该包含一个 Via 头域。

6.35 Subject（标题）

Subject 请求头域提供了一个摘要，或者指示了呼叫的实际情况，使得不必分析通话描述便可过滤呼叫。（当然，通话描述不必使用与邀请同样的标题）

```
Subject  =  ("subject" / "s") ":" *TEXT-UTF8
```

6.36 Timestamp（时间标记）

Timestamp 通用头域指示客户何时向服务器发送请求。此头域的值只对客户有用。服务器必须回送完全一样的数值，同时如果它有确切的消息，可以增加一个小数值指示从它接收到请求开始所过去的时间。客户使用 timestamp 头域来计算到达服务器的 round-trip 时间，以便调整重传的 timeout 时间。

```
Timestamp = "Timestamp" ":" *(DIGIT)[ "." *(DIGIT) ][delay]
Delay     = (DIGIT) [ "." *(DIGIT)]
```

6.37 To（目的地地址）

To 通用头域说明了请求的接收者：

```
To = ("To" / "t") ":" (name-addr / addr-spec)
       * (";" addr-params)
```

请求和响应必须包含 To 头域，指示请求的预定接收者。可选的“display-name”意味着由用户接口提出（执行）。UAS 或者重定向服务器将 To 头域的内容复制到它的响应中，同时如果请求包含了不止一个 Via 头域，则必须增加“tag”参数。

如果 Via 头域不止一个，那么表明请求至少经过一个代理服务器的处理。因为接收者不知道此请求是哪一个代理服务器派生的请求，所以从安全方面考虑，可认为它们都派生了请求。

此 SIP-URL 禁止包含“transport-param”,“maddr-param”,“ttl-param”,“headers”。接收到含有以上元素的 SIP-URL 的服务器在执行下一步处理之前，应将这些元素删除。

“tag”参数作为一种通用机制，用于区分由一个 SIP-URL 标识的用户的多个实例。因为代理可以派生请求，所以同一个请求可以到达用户的多个实例（例如：移动和住宅电话）；又由于每一个都可以响应，所以必须有一种方法来区分来自被叫方每一个实例的响应。这种情况也可由于多点传送（组播）请求而产生。“tag”参数用于区分 UAC 的响应。当请求有可能被一中间代理派生时，每一个实例都必须在它的响应中包含“tag”参数。“tag”参数必须可以被 UAS、登记器和重定向服务器增加，但禁止被加入到上传的响应中。“tag”参数可以增加到所有方式的所有已经定义的响应中，也可以加入到来自 UAS 或者重定向服务器的报告性（1xx）响应。两个实体间随后所有的事务都必须包含“tag”参数。

当响应与请求相匹配时，如果请求的 To 域中未包含“tag”参数，那么响应 To 域中的“tag”参数将忽略。“tag”允许代理派生同一个呼叫的未来的请求，而只对几个可能的响应 UAS 中的一个定位（寻址）。它也允许被叫方的多个实例发送可以区分的请求。

当 SIP 服务器接收到一个请求，此请求的 To 域中含有它不能识别的 URI 时，它应该返回一个 400（Bad Request）响应。

即使“display-name”是空的，如果“addr-spec”包含了“,”、“?”、“;”,“name-addr”形式也必须使用。

Call-ID、To 和 From 用于标识一个 Call leg。呼叫和 Call-leg 的区别在于多个响应对派生请求的呼叫。“tag”允许代理派生同一个呼叫的未来的请求，而只对几个可能的响应 UAS 中的一个定位（寻址）。它也允许被叫方的多个实例发送可以区分的请求。

The following are examples of valid To headers:

To: The Operator <sip:operator@cs.columbia.edu>;tag=287447
To: sip:+12125551212@server.phone2net.com

6.38 Unsupported（不支持）

Unsupported 响应头域列出了服务器不支持的特性，参考 Section 6 .30 的一个例子。(只用于 420 响应)

语法：

Unsupported = “Unsupported” “ :”1#option-tag

6.39 User-Agent（用户代理）

User-Agent 通用头域包含了关于发送初始请求的客户用户代理的消息。

此头域用于统计目的，跟踪违反协议的情况、用户代理的自动认可的情况，以便在编制响应时避免特定用户代理的限制。用户代理应在请求中包含此头域。

User-Agent = "User-Agent" ":" 1(product | comment)*

例如： User-Agent: CERN-LineMode/2.15 libwww/2.17b3

6.40 Via（路径）

Via 头域指示请求迄今为止所走的路径。它防止了请求的循环，同时确保了响应（回答）沿同样的路径返回，这一点可以通过防火墙遍历和其他的异常路径情况提供帮助。

6.40.1 Requests（请求）

初始(发送)请求的客户必须在请求中加入 Via 头域，此头域中包含了它自己的主机名或者网络地址，以及端口号（可使用默认的)。(需要注意的是：此端口号与请求的 UDP 源端口号不同。）建议使用有效的域名。随后的每一个代理服务器必须在已存在的 Via 头域之前加入它们自己的 Via 头域。接收到 3xx 重定向响应且循环搜寻的代理，必须使用与初始的被代理的请求同样的 Via 头域。

代理应该检测最顶端的 Via 头域，以确保它包含了发送者的正确的网络地址，就像从其代理本身看到的一样。如果发送者的地址不正确，代理需要增加一个“ received ”属性。(“ received ”的值是发送者的实际地址)。

注意：隐藏在网络地址翻译器（NAT）或者防火墙之后的主机可能不能在 Via 头域中增加一个网络地址，此 Via 头域可以在 NAT 的范围外通过下一个 hop 到达。“ received ”属性的使用允许 SIP 请求穿过能够修改源 IP 地址的 NAT。如果 NAT 可以修改端口号，那么被叫方网络地址端口翻译器 (NAPT) 将不能正确传输 SIP，当 SIP 在 UDP 上传输时；在这种情况下，需要使用应用层网关。当在 TCP 上传输时，SIP 更容易穿过 NAT。

将请求发送到多点地址的代理必须在它的 Via 域中增加“ maddr ”参数，并且应该增加“ ttl ”参数。当服务器接收到一个在最顶端的 Via 域中含有“ maddr ”参数的请求时，它应该按照“ maddr ”中所列的多点地址发送响应。

当代理服务器接收到一个在Via头域中包含有它自己的地址的请求时,它必须返回一个482(Loop Detected）响应。

代理服务器禁止将一个请求发送到已经在某一个 Via 域中存在的多点地址组。此规定防止了已出现故障的代理服务器导致循环回路。

6.40.2 Receiver-tagged Via Header Fields（标记了 received 的 Via 头域）

通常，每一个发送或者前转 SIP 消息的主机通过增加 Via 头域来指示经过的路径。然而很可能 NAT 改变了请求的源地址和端口，在这种情况下，Via 头域不能用来指示路由。为了防止这种情况的发生，代理应该检测最顶端的 Via 头域，以确保它包含了发送者的正确的网络地址，就像从其代理本身看到的一样。如果发送者的地址不正确，代理必须在前一个 hop 增加的 Via 头域中增加一个“received”属性。

6.40.3 Response（响应）

代理或者 UAC 根据以下的规定来处理响应中的 Via 头域：

1. 第一个 Via 头域指示处理此相应的代理或者客户。如果未指示，应放弃此消息，否则删除此 Via 头域。
2. 如果没有第二个 Via 头域，那么此响应的目的地便是此客户。否则，对响应的处理依赖于此 Via 头域是否包含了“maddr”参数，或者是否是一个标记了“received”的 Via 头域：
   ----如果第二个 Via 头域中包含了“maddr”参数，则使用“sent-by”中所标识的端口号（如果没有，则默认为 5060），将此响应发送到“maddr”参数所列的多点地址处。同时应使用“ttl”标识的 TTL（如果没有，则默认为 1）来发送此响应。考虑到健壮性，即使“maddr”所指示的不是一个多点地址，响应也应该被发送到此地址。
   ----如果第二个 Via 头域中未包含“maddr”参数，但此头域是一个标记了“received”的 Via 头域，则使用“sent-by”中所标识的端口号（如果没有，则默认为 5060），将此响应发送到“received”所指示的地址处。
   ----如果不是上面提到的两种情况，则将响应发送到第二个 Via 头域的“sent-by”所指示的地址处。

6.40.4 User Agent and Redirect Servers（用户代理和重定向服务器）

UAS 或者重定向服务器依据以下规则来发送响应：

----如果第二个 Via 头域中包含了“maddr”参数，则使用“sent-by”所标识的端口号（如果没有，则默认为 5060），将此响应发送到“maddr”参数所列的多点地址处。同时应使用“ttl”标识的 TTL（如果没有，则默认为 1）来发送此响应。考虑到健壮性，即使“maddr”所指示的不是一个多点地址，响应也应该被发送到此地址。

----如果第一个 Via 头域中“sent-by”的地址与包的源地址不同，则将此响应发送到包的实际源地址处，处理类似于标记了“received”的 Via 头域。

----如果不是上面提到的两种情况，则将响应发送到第二个 Via 头域的“sent-by”所指示的地址处。当响应使用 TCP 发送时，如果可能的话，使用已经存在的 TCP 连接。

6.40.5 Syntax（语法）

```
Via = ("Via" | "v") ":"1#(sent-prrotocol sent-by *(";"via-params)[comment])
via-param= via-hidden | via-ttl | via-maddr | via-received | via-branch
```

via-hidden= "hidden"
via-ttl= "ttl""="ttl
via-maddr= "maddr""="maddr
via-received= "received""=" host
via-branch="branch" "=" token
sent-protocol= protocol-name "/" protocol-version "/" transport
protocol-name="SIP" | token
transport="UDP" | "TCP" | token
sent-by=(host [":" port]) | (concealed-host)
concealed-host=token

"protocol"的默认值是 SIP，"transport"的默认值是 UDP。"maddr"和"ttl"参数只用于向多点地址发送的请求。注意到：代理的私人性依赖于与它相通信的下一个 hop。如果需要的话，应该知道 IP 地址和源主机的端口号。

"branch"参数用于每一个可以派生的代理。对于每一个代理派生产生的不同的请求来说，它的值必须是唯一的。CANCEL 请求必须拥有和相通信的派生的请求同样的 branch 值。当响应到达此代理时，它可以用 branch 值来指示与此响应相通信的分支。产生唯一请求的代理（非派生）也可以增加"branch"参数，但它的值只在一系列同构的请求中是唯一的。

6.41 Warning（警告）

Warning 响应头域中包含了关于响应状态的其他信息。

Warning = "Warning""" :" 1#warning-value
Warning-value = warn-code SP warn-agent SP warn-text
Warn-code = 3DIGIT
Warn-agent = (host[" :"port]) | pseudonym
Warn-text = quoted-string

一个响应中可以有多个 Warning 头域。"warn-text"使用自然语言。

任何一个服务器都可以在响应中加入 Warning 头。代理服务器必须将 Warning 头域加在任何一个 Authorization 头域之前，由于此限制，Warning 头域必须加在任何已存的未被 signature 覆盖的 Warning 域之后。代理服务器禁止删除它所接收到的响应中的任何 Warning 头域。

当响应中有多个 Warning 头域时，用户代理应该尽可能将它们按照在响应中出现的顺序都显示出来。如果不能全部显示，用户代理应显示在响应中出现的较早的警告。

"warn-code"包含了三个数字，第一个数字"3"表示 SIP 的专用警告。

下面列出已定义的警告，需要注意的是：所有的警告都由通话描述引起。

300--329：通话描述中的关键字出现的问题；
330—339：通话中所申请的基本网络业务相关的警告；
370—379：通话描述中所申请的定量的 QoS 相关的警告；
390—399：不属于以上所述类型的警告的混杂。
300：不兼容的网络协议；　　301：不兼容的网络地址格式；
302：不兼容的传输协议；　　303：不兼容的带宽单元；
304：无效的媒体类型；　　305：不兼容的媒体格式；
306：无法识别的属性；　　307：无法识别的通话描述参数；
330：无效的多点传送；(用户位置不支持多点传送)
331：无效的单点传送；(用户位置不支持单点传送，通常是由于防火墙的

存在）

370：无效的带宽；（带宽数超过允许范围）

399：混杂的警告；（接收到此警告的系统禁止自动采取措施）

10 ：Response is stale（旧的响应）当响应是旧的时，必须包含。

11 ：Revalidation failed（重新生效失败）由于重新发送响应失败，只能返回旧的响应时，必须包含。

12 ：Disconnected operation（非连接操作）

13 ：Heuristic expiration（探索超时）

14 ：Transformation applied（已用过的事务）

99 ：Miscellaneous warning（混杂的警告）

6.42 WWW-Authenticate（WWW 验证）

WWW-Authenticate 响应头域必须包含在 401（Unauthorized）响应中。此域的值包含了至少一个 challenge，用来表明验证方案和参数适用于 Request-URI。

WWW-Authenticate = "WWW-Authenticate" ":" 1#challenge

“realm”参数应该向用户显示。用户代理应该将对于给定的 To 域中的目的地地址的值的验证信任保存，以便在接收到下一个到此目的地的请求时重用这些值。

除了“basic”和“digest”验证方案，SIP 定义了一个新的方案：PGP。

7. Status Code Definitions（状态码定义）

7.1 Informational 1xx

1xx 响应表示，相关的代理或者服务器应执行进一步的处理，而不能作出一个最终的（决定性的）响应。客户应该等待服务器的进一步的响应，同时服务器应该不用提示就发送这样的一个响应。如果服务器预计获得一个最终响应需要 200ms 以上的时间，那么它应该发送一个 1xx 响应。服务器可以发送零个或者多个 1xx 响应，在顺序和唯一性上没有任何限制。需要注意的事，1xx 响应不能够可靠传输，也就是说，它们不能让客户发送 ACK。服务器可以自由的重传 1XX 响应，客户可以通过重发请求来查询呼叫处理的当前状态。

7.1.1 100 Trying（尝试）

代理代表此次呼叫正在采取一些的未指明的动作（例如，正在查询数据库），但用户尚未定位。

7.1.2 180 Ringing（振铃）

被叫方用户代理将用户定位在他最近所登记的位置上，并且正在试着提醒用户。

7.1.3 181 Call Is Being Forwarded（呼叫正被前转）

代理服务器适用此状态码表明呼叫正在被前转到一些不同的位置。

7.1.4 182 Queued（排队）

被叫方目前无效，但被叫方决定将此呼叫排队而不是拒绝此呼叫。当被叫方变为有效时，它将返回合适的最终状态响应。原因短语可以进一步给出关于此呼叫状态的的细节。服务器可以向主叫方发送几个 182 响应来修改此排队呼叫的状态信息。

7.2 Successful 2xx

请求成功，终止搜寻。

7.2.1 200 OK（成功）

此响应返回的消息依赖于请求中所使用的方式：

BYE：呼叫终止，消息体为空。

CANCEL：取消搜寻，消息体为空。

INVITE：被叫方同意加入；消息体指示被叫方的能力集。

OPTIONS：被叫方同意共享包含在它的消息体中的能力集。

REGISTER：登记成功。客户根据 Content-Type 来决定消息体。

7.3 Redirection 3xx

3xx 响应给出关于用户新位置的信息，或者是其他可能适合呼叫的服务的信息。它们应该终止目前的搜寻，并可能导致一个新的搜寻的初始。

任何一个 3XX 响应均禁止将请求中 Via 域中的地址路径加入 Contact 域。

为避免前转循环，UAC 或者 proxy 必须检测重定向服务器返回的地址是否是以前已经尝试过的地址。

7.3.1 300 Multiple Choices（多个选择）

在请求中会有多个地址选择，每一个都有自己特定的位置，用户（或用户代理）可以选择其中较好的一个通信端点，将其请求重定向到此位置。

此响应应该包含一个实体，此实体中有一系列的资源特性和位置，用户或用户代理可以选择这些位置中最适合的一个，同时这一个应该得到 Accept 请求头的允许。此实体格式通过 Content-Type 头域中所给出的媒体类型来说明。这些选择也应该在 Contact 域中列出。与 HTTP 不同的是，SIP 响应可以包含多个 Contact 域，或者是在一个 Contact 域中有多个地址。用户代理可以用 Contact 域来自动重定向或者让用户确定一个选择。

如果被叫方可以在几个不同的位置找到，同时服务器不能代理此请求时，此状态响应较为适当。

7.3.2 301 Moved Permanently（永久转移）

当使用 Request-URL 中的地址，而永远无法找到用户时，提出申请的客户应该尝试 Contact 头域中所给出的新地址。主叫方应该使用此新地址来修改所有的本地位置、地址本和用户位置缓冲，同时将未来的请求重定位到上述所列出的地址中。

7.3.3 302 Moved Temporarily（临时转移）

提出申请的客户应该在 Contact 头域所给出的新的地址处重新尝试请求。重定向的期限通过 Expires 头来指示。如果没有明确的期限，则此地址只对本次呼叫有效，同时，此地址不能缓冲用于未来的呼叫。

7.3.4 305 Use Proxy（使用代理）

申请的资源必须通过 Contact 头域所给出的代理来访问。Contact 头域给出了代理的 URI。接收者可以通过代理来重复此请求。305 响应只能由 UAS 产生。

7.3.5 380 Alternative Service（可选的服务）

呼叫不成功，但是可能完成可选服务。此可选服务在响应的消息体中描述。（目前未定义此种格式的消息体）

7.4 Request Failure 4xx(请求失败)

4XX 响应定义了来自特定服务器的错误响应。 客户不应该不做任何修改就重试原先同一个请求（例如，加入合适的鉴权信息）。然而，发送到另一个服务器的同一个请求可能是成功的。

7.4.1 400 Bad Request（错误请求）

请求中由于有错误的语法而无法识别。

7.4.2 401 Unauthorized（未鉴权）

请求需要得到用户鉴权。

7.4.3 402 Payment Required（需要支付（付款））

（预留为以后的使用。）

7.4.4 403 Forbidden（禁止）

服务器可以识别此请求，但是拒绝完成。鉴权不能提供帮助，并且不应该再重复此请求。

7.4.5 404 Not Found（未找到）

服务器作出确定：用户不在 Request-URI 中说明的地址处。当 Request-URI 中的域名和此请求的接受者所处理的任何一个域名都不匹配时，也可返回此响应。

7.4.6 405 Method Not Allowed（方式不允许）

请求行中所列的方式不能用于 Request-URI 所标识的地址。

此响应必须包含一个 Allow 头域，其中含有适合所给地址的有效方式。

7.4.7 406 Not Acceptable（不接受）

请求所标识的资源只能产生某种类型的响应实体。此实体的内容特征与请求中的 Accept 头域所列的内容不符。

7.4.8 407 Proxy Authentication Required（需要代理验证）

此响应类似于 401 响应，但表示客户必须首先通过代理验证自己。代理必须返回一个 Proxy-Authenticate 头域，此头域包含了适合代理对请求资源的查询。客户可以使用 Proxy-Authorization 头域来重复此请求。

此响应用于访问通信信道的应用程序（如：电话网关），而不是被叫方需要验证。

7.4.9 408 Request Timeout（请求超时）

在 Expires 请求头域所指示的时间内，服务器不能产生一个响应，例如，一个用户位置。在以后的任何时间内，客户可以重复此请求，而不用加以修改。

7.4.10 409 Conflict（冲突）

请求因为与被请求资源的当前状态冲突而无法完成。如果 REGISTER 请求中的 action 参数（可

参见 Contact 中的 action 参数的描述。）与已存的登记器有冲突，则返回此响应。

7.4.11 410 Gone（离开）

被请求的资源在服务器上无效，且不知道前转地址。这种情况一般是永久性的。如果服务器不知道，或者不能轻易决定，此情况是否是永久性的，则应该使用 *404（Not Found）响应*。

7.4.12 411 Length Required（需要长度）

服务器拒绝接受一个没有定义 Content-Length 的请求。客户可以在请求中增加了有效的 Content-Length 头域之后重新发送此请求，此头域中包含了请求消息体的长度。

7.4.13 413 Request Entity Too Large（请求实体太大）

由于请求实体的长度大于服务器所能处理的范围，服务器拒绝处理此请求。在这种情况下，服务器应该断开连接，以防止客户继续此请求。

如果这种情况是暂时性的，那么服务器可以在响应中包含一个 Retry-After 头域，来指示此情况是暂时的，并且客户可以在此时间之后重新请求。

7.4.14 414 Request-URI Too Long（Request-URI 太长）

由于 Request-URI 的长度大于服务器能够翻译的范围，服务器拒绝处理此服务。

7.4.15 415 Unsupported Media Type（不支持的媒体类型）

服务器拒绝为请求服务，因为此请求消息体的格式不能被请求资源所用的请求方式所支持。服务器应该在 Accept，Accept-Encoding 和 Accept-Language 头域中列出可以接受的格式。

7.4.16 420 Bad Extension（错误的扩展名）

服务器不能识别在 Require 头域中说明的协议扩展名。

7.4.17 480 Temporarily Unavailable（暂时无效）

被叫方终端系统已经成功连接，但被叫方目前无效。（例如，未 login 或者用预先与被叫方通信的方式 login）。

此响应可以在 Retry-After 头域中指示一个合适的时间。因为用户可能在其他的地方有效，所以，此响应不能终止任何搜寻。原因短语应该指示一个关于用户为何无效的更加详细的原因。此值应该被用户设置。486 响应可以用来更加详细地指示呼叫失败的原因。

此响应也可以被重定向服务器返回，此服务器通过 Request-URI 来识别用户，但目前没有一个对用户的有效的前转位置。

7.4.18 481 Call leg/Transaction Does Not Exist（事务不存在）

在以下两种情况下返回此响应：

--服务器接收到一个 BYE 请求，此请求与任何已存的 call leg 均不匹配。

--服务器接收到一个 CANCEL 请求，此请求与任何已存的事务均不匹配。

（服务器简单地丢弃指示一个未知事务的 ACK。）

7.4.19 482 Loop Detected（检测出回路）

服务器接收到一个在 Via 的路径中包含有自己的请求。

7.4.20 483 Too many hops（hop 数过大）

服务器接收到一个请求，在此请求中，Via 入口的个数（也就是 Via 头域的个数，也是 hop 数）大于 Max-Forwards 头域所允许的数目。

7.4.21 484 Address Incomplete（地址不完整）

服务器接收到的请求中 To 头域的地址或者 Request-URI 不完整。应该提供其他的信息。

此状态码允许重复拨号。对于重复拨号，客户并不清楚拨号串的长度，它发送不断增长的拨号串，促使用户继续输入，直到不再接收到一个 484 响应为止。

7.4.22 485 Ambiguous（不明确）

在请求中提供的被叫方的地址是不明确的。

此响应可以在 Contact 头中包含一系列可能的不明确的地址。

将可选的地址透露出来，有可能涉及个人或组织的隐私。如果此请求地址是不明确的，它必须能够配置一个服务器来作出 404(Not found)响应，或者禁止此一系列的选择。

```
Example response to a request with the URL lee@example.com :
485 Ambiguous SIP/2.0
     Contact: Carol Lee <sip:carol.lee@example.com>
     Contact: Ping Lee <sip:p.lee@example.com>
     Contact: Lee M. Foote <sip:lee.foote@example.com>
```

7.4.23 486 Busy Here（本地忙）

被叫方的终端系统被成功连接，但被叫方目前不想或者不能接受其他的呼叫。

此响应在 Retry-After 头域中指示了呼叫的合适时间。

如果用户在其他的地方有效，例如通过一个语音信箱系统，那么此相应不能中止任何搜寻。

如果客户知道没有其他的终端可以接受此呼叫时应返回 600 响应。

7.5 Server Failure 5xx(服务器失败 5XX)

当服务器本身出错时，返回 5XX 响应。它们不是确定的错误，并且如果其他的位置还未尝试过，则不能终止搜寻。

7.5.1 500 Server Internal Error（服务器内部错误）

服务器遇到异常情况，从而不能完成请求。客户可以显示出此特定错误情况，同时在几秒之后重新发送此请求。

7.5.2 501 Not Implemented（未实现）

服务器不支持用于实现请求的功能。当服务器不能识别请求方式，同时对于任何用户都不能支持此请求方式时，应返回此响应。

7.5.3 502 Bad Gateway（错误的网关）

作为网关或者代理的服务器从为完成请求所需访问的下传服务器接收到一个无效响应。

7.5.4 503 Service Unavailable（无效服务）

由于暂时的超负荷或者服务器的维护，服务器目前不能处理请求。含义是这只是一个暂时的情况，过一点时间会有所减轻。此段时间可以在 Retry-After 头域中标识，如果没有 Retry-After，客

户必须将此响应作为一个 500 响应。

注意：503 响应并不表示一旦超负荷，服务器就要用此响应。一些服务器可以简单地拒绝连接。

7.5.5 504 Gateway Time-out（网关超时）

作为网关的服务器不能接收一个来自为完成请求所需访问的服务器（例如：定位服务器）的及时的响应。

7.5.6 505 Version Not Supported（不支持的版本）

服务器不支持，或者拒绝支持请求消息中所用的 SIP 协议的版本。意思是：服务器不能或者不愿意使用与客户一样的主版本来完成请求，而不是使用错误的消息。响应中的消息体可以描述为什么不支持此版本以及服务器所支持的版本。（此消息体的格式未定义）

7.6 Global Failure 6xx

6XX 响应表明服务器有关于某特定用户的确切信息，而不是在 Request-URI 中所指示的特例。所有对于此用户的进一步搜寻注定是失败的，因此此搜寻应该中止。

7.6.1 600 Busy Everywhere（全忙）

被叫方的终端系统被成功连接，但被叫方正忙，此时并不想接受本次呼叫。

此响应在 Retry-After 头域中指示了呼叫的合适时间。

如果呼叫方不希望透露拒绝呼叫的原因，可以使用 603（Decline）响应。

此响应（600）只有在客户知道没有其他的终端（如一个语音信箱系统）可以回答此请求时才返回。否则应返回 *486 (Busy Here)*。

7.6.2 603 Decline（拒绝）

被叫方的机器被成功连接，但用户很明显不想或不能加入呼叫。

此响应在 Retry-After 头域中指示了呼叫的合适时间。

7.6.3 604 Does Not Exist Anywhere（不存在）

服务器验证出 To 域中所标识的用户并不存在，在其他地方继续搜寻结果亦如此。

7.6.4 606 Not Acceptable（不接受）

用户代理已经连接上，但不能接收通话描述中的某些因素，例如请求媒体、带宽、地址结构等。606 响应表示用户（被叫方）希望通信，但是不能完全支持所描述的通话。此响应可以包含多个 Warning 头域来指示不能支持所描述的通话的原因。当并不经常需要协商时，或者当一个新的用户被邀请加入一个已经存在的会议，协商是不可能时，可使用 606 响应。邀请的初始者决定是否使用 606 响应。

8. SIP Message Body（SIP 消息体）

8.1 Body Inclusion（是否包含消息体）

请求可以包含消息体（除非特定的指明）。BYE 请求禁止包含消息体。ACK、INVITE、OPTIONS 请求中的消息体通常是通话描述。REGISTER 的消息体有待研究。

对于响应消息，请求方式和响应状态码决定了消息体的类型和内容。所有的响应都可以包含一个消息体。

1XX：请求进展情况的咨询信息；

2XX INVITE：通话描述；
3XX：对于可代替的（选择的）目的地或者服务的描述；
4XX、5XX、6XX：其他的失败原因信息。
建议 1XX、3XX、4XX、5XX、6XX 中的消息体使用 text/plain 或者 text/html 类型。

8.2 Message Body Type（消息体类型）

必须使用 Content-Type 头域指示消息体中的媒体类型。

如果消息体经过编码（压缩），则必须在 Content-Encoding 头域中指示编码类型，否则应忽略 Content-Encoding 域。

消息体所使用的字符集在 Content-Type 头域的“charset”属性中指示。

8.3 Message Body Length（消息体长度）

应该使用 Content-Length 头域来指明消息体的长度（字节数）。

SIP 禁止使用 HTTP/1.1 中的“chunked”传输编码。

9. Compact Form（压缩形式）

当 SIP 在 UDP 上传输鉴权和通话描述时，可能会造成请求或者响应的长度大于 MTU 的能力，为解决这个问题，可使用 SIP 的压缩形式：定义通常使用的头域的缩写形式。

c：Content-Type；e：Content-Encoding；f：From；　i：Call-ID；
m：Contact；　l：Content-Length；　s：subject；t：To；
v：Via；

```
INVITE sip:schooler@vlsi.caltech.edu SIP/2.0
v:SIP/2.0/UDP 131.215.131.131;maddr=239.128.16.254;ttl=16
v:SIP/2.0/UDP 128.16.64.19
f:sip:mjh@isi.edu
t:sip:schooler@cs.caltech.edu
i:62729-27@128.16.64.19
c:application/sdp
CSeq: 4711 INVITE
l:187

v=0
o=user1 53655765 2353687637 IN IP4 128.3.4.5
s=Mbone Audio
i=Discussion of Mbone Engineering Issues
e=mbone@somewhere.com
c=IN IP4 224.2.0.1/127
t=0 0
m=audio 3456 RTP/AVP 0
```

客户可以混合使用常域名和压缩性的域名。服务器同样必须接受两种形式的域名。代理可以在两种形式之间来改变头域，但是禁止改变在 Authorization 头域之后的头域。

10. Behavior of SIP Clients and Servers(客户端和服务端的性能)

10.1 General Remarks(总括)

SIP 定义的可以使用的传输协议，既可以是 UDP（单点或者多点广播），或者是 TCP；SIP 本身提供可靠性的通信。

10.1.1 Requests(请求)

服务器丢弃相同类型的请求，但要先回应一个合适的响应。 从客户端接收 CANCEL 请求后，状态代理服务器如果还没有一个最终的响应，应该向所有的分支发送 CANCEL 信号。

当用户代理接收一个请求，将先检查正在进行的呼叫的 Call-ID。如果检查到了，就比较 To 域中用户的 tag 值。如果数值不匹配，将拒绝这个请求。在 From 头，如果已经存在的 Call Leg 匹配还有 tag 值匹配，服务器继续比较 Cseq 头域的值。如果小于或者等于目前的序列号码值，将重新发送请求。否则，则认为是一个新的请求。如果 From 头不匹配一个存在的 Call Leg，将创建一个新的 Call Leg。

如果没有发现 Call-ID,也将创建一个新的有 To，From 和 Call-ID 头的 Call Leg。在这种情况下，To 头域不应该包含 tag．服务器返回一个包含相同 To 值，和统一的 tag 值的消息。如果请求仅仅包含一个头域，Tag 可以被忽略。

10.1.2 Responses 响应

在发送最终响应之前，服务器可以发送一个或者多个临时的响应。对于有状态值的 Proxy，User agent server，redirect server 或者注册不能在 200ms 内发出一个最终的响应，应该尽可能的响应临时性的响应。没有状态的代理自己不能发出临时性的响应。

响应通过匹配下面域值的映射给请求：To，From,Call-ID,Cseq headers 和第一个 Via 头域的分支参数。如果 Via 头域指示 upstream server 使用 TCP，代理将打开 TCP 连接到那个地址。代理不得不准备接收被动的 TCP 连接的响应，即使大多数响应将到达活动的连接的输入部分。（一个活动的连接是一个被 Proxy 初始化的 TCP 连接，被动的连接被 Proxy 接受，但有其他实体初始化的实体）

100 响应不能被转发，其他 1XX 响应消息可以转发，能够排除刚刚发送出去的响应代码。2XX 响应根据这个 Via 域来转发。一旦有状态的代理接收一个 2XX 响应，它不必转发一个非－2XX 的最近响应。用 300 或者更高的响应状态，将被每个有状态的代理重传，直到上面的 proxy 发送回 ACK 或者 CANCEL。

状态代理必须在收到第一个非 200 响应后的 32 秒内维护这个状态，为了的是处理响应的重发。

注意：32 秒的窗口是由最大重传数决定的，时间是使用缺省时间的 200 类。当 ACK 在到被叫的 User agent 或者下个有状态的代理过程中丢失。

10.2 Source Addresses, Destination Addresses and Connections

（源地址，目标地址和连接）

10.2.1 Unicast UDP（单点传播）

返回到地址的响应列在 Via 头域中（Section 6.40）,不是请求的源地址。

无状态响应的下一跳并没有产生重新调用的响应，但由 Proxy Server 或者用户代理服务器产生。这样，无状态的代理仅仅可以使用 Via 头域转发这个响应。

10.2.2 多点传播的 UDP

请求可以被多点传播；多点请求正好描述一个独立的 Request-URI。这个请求应该有个范围，确保不是转发超出管理系统的边界。这或者由 TTL 或者管理范围[25]完成，依靠网络中的工具。

客户端接收多点的请求没有检查是否主机部分 Request-URI 匹配自己的主机或者域名。如果请求经过多点传播接收，响应同样也经过多点传播返回。多点传播请求的响应也用有同一 TTL 请求的多点传播，这里的 TTL 参数从 Via 头域的 ttl 参数派生（Section 6.40）

为了防止响应的冲突，服务器不应该回答用 2XX 或者 6XX 其他的状态代码。服务器延迟一段时间后才响应，这个延迟时间由内部定义的随机产生的数（在 1 秒钟之内）。服务器如果从其他集团成员听到一个更低的数或者 6XX 响应准备优先发送,那么此时应该禁止响应。服务器没有响应 CANCEL 的请求，这个请求经过多点传播以防止请求阻塞。一个代理或者 UAC 应该发送一个 CANCEL，关于接收第一个 2XX 或者 6XX 的响应的多点传播请求。

注意：服务器响应抑制是一个可以因为请求一个服务器去分享一些消息处理规则。比如说 A 发送一个单点传播请求，B，C 接收到了，D，B 发送一个 200 响应。响应的 Via 域将包含地址 A。C 也将接收到这些响应，也可以禁止它的响应。然而，C 通常没有检查这个响应，而这个 Via 并不是他们自己的。正常情况下，一个响应接收一个不正确的最高 Via 必须删除，但不在这一情况下。为了从丢失路由或者区分环路的包将是相当复杂，正是这个原因，这个过程是应该。CANCEL，相反的，提供一个简单或者更标准的方式来执行响应的一只。也是因为这个原因，这里 CANCEL 只能是可以。

10.3 TCP

一个简单的 TCP 连接可以服务一个或者两个 SIP 事务。一个事务包含几个（包括零个）对应于最终响应的临时的响应。（典型的，事务包含一个最终响应，但没有特别的环境，比如可以产生多个 200 响应）

这个客户端应该至少保持这个连接，直到第一个响应到达。如果客户端在收到最终响应之前关闭或者复位 TCP 连接，服务器把这个视为等价于 CANCEL 请求。

注意：这个行为不看为是故障，客户导致一个 Proxy server 不确定的保持连接状态。

服务器直到发送最终响应后，才关闭 TCP 连接，或者希望关闭 TCP 连接。通常，关闭连接是客户端的责任。

如果服务器让连接打开着，还有如果客户端还用相同的协议族发送请求，还有再次使用一个连接（比如 HTTP 或者流工作命令）。如果服务器小于返回一个响应到客户端并且保持连接到客户端，它可以打开一个连接到列在 Via 头的地址。 这样，Proxy 或者 User agent 必须准备以“被动”的连接接收和发送。

10.4 Reliability for BYE, CANCEL, OPTIONS, REGISTER Requests

（BYE，CANCEL，OPTIONS 和 REGISTER 请求的可靠性）

10.4.1 UDP

SIP 的客户端使用 UDP 在一定的时间后重传 BYE，CANCEL，OPTIONS 或者 REGISTER。在 T1 时间开始，使每个包的发送间隔时间增大两倍。这说明在第一个包发送后的 T1 秒后再次发送，下个包将是 2×T1 秒，再下一个是 4×T1 秒，等等。直到时间间隔为 T2 秒。如果客户端接收一个临时性的响应，将继续重传这个请求，但要在 T2 秒之后。当客户端发送 11 个包，或者接收一个定义的响应后，取消重传机制。缺省的 T1 和 T2 值分别是 500ms 和 4 秒。客户可以使用更大的值，但不能比这个小。服务器根据重传请求重传重传响应。服务器发送一个最终请求后，不能够确保客户端已经接受这个响应，这样最少保持这个结果 10×T2 时间内。比如，根据接收请求传送来联系 User 或者定位服务器。

注意：使用 exponential backoff 是为了进行拥塞控制。然而，back－off 必须关闭，因为请求的重传机制用于除了服务器的重传响应。没有这个能力，单一响应的丢失将明细增加潜在危险。

启动重传定时器的值小于 TCP 的网络交互通信的时间，小于 500ms。为了拥塞控制的目的，重传数不得不跳跃。所给的最多事务希望包含一个请求和几个响应，来回的时间估计不是很有用。如果 RTT 希望更快速的发现一个丢失的最终响应，每个重传请求需要加上自己本身的 TimeStamp（Section 6.36），在响应钟返回。服务器将保持这个结果，等到客户端确信不再需要重传同样的请求。

每个在代理链中的服务器产生自己的最终响应去取消请求。服务器立即响应接收到的 CANCEL 请求，而不是等待到从 CANCEL 请求中接收到最终的响应。

BYE 和 OPTIONS 最终响应有 redirect 和 user agent server 产生 注册的最终响应有 registrar 产生。注意对比在 Section 10.5 中描述的可靠性机制。这些请求的响应没有被周期性的重传，并且没有通过 ACK 确认。

10.4.2 TCP

客户端使用 TCP 不需要重传请求。

10.5 Reliability for INVITE Requests（INVITE 请求的可靠性）

特别考虑到 INVITE 方法的应用。

1. 在接收一个邀请之后，服务器分析的时间对于决定这个输出，可以消除时间。比如，如果这个呼叫方“ rung”或者执行扩展的搜索，在请求和定义的响应到了几十秒。要么主叫要么被叫是没有人直接控制的服务器，呼叫的尝试的时间可以到很大。
2. 如果电话用户的接口是模型或者我们需要到这个 PSTN 的接口，主机用户接口将提供“ ringback（回铃音）”，这是一个被叫产生的信号。（状态响应 180（Ringing）用于初始化回铃音）一旦被叫摘机，主叫需要知道可以使能语音路径和停止回铃。被叫响应的邀请将丢失。除非响应可靠传输，主叫将继续提到回铃音，直到被叫认为呼叫存在。
3. 客户端不得不结束一个向外的请求，比如，因为它不再需要去等待这个连接或者成功的搜索。服务器将等待几个重传间隔解析缺少的重传请求，作为呼叫的结束。如果呼叫在主叫放弃的时候返回成功，被叫将“ 摘机 ”并不被“ 连接 ”。

10.5.1 UDP

对于 UDP，SIP 客户端将用开始于 T1 时间的间隔，然后以双倍的时间后再发送包。客户端接收到临时的或者有定义的响应，或者发送 7 次请求后没有响应，将取消重传。

传输临时响应的服务器应该重传一个复制请求的接受者。重传最终响应的服务器应该以一定的间隔重传。比如再 T1 时间开始，以后每个包的成倍的增加。如果发现下面情况，重传机制将取消：

1. 接收到同一事务的 ACK 请求。
2. 接收到同一 Call Leg 的 BYE 请求。
3. 接收到同一 Call Leg 的 CANCEL 请求和最后的响应代码等于或者大于 300
4. 响应重传 7 次。

只有 UAC 产生一个 2XX 最终响应的 ACK，如果这个响应包含联系头域，这个 ACK 应该发送到列在联系头域中的地址。如果响应没有包含联系头，客户端用同一 To 头域和 Request－URI 作为 INVITE 请求，并发送 ACK 到同一目的地址，作为初始的 INVITE 请求。非 2XX 的 ACK 最终请求将送到初始请求的同一服务器，并且用相同的 Request－URI。注意，ACK 的 To 头域从响应中拷贝，这样，特别的包含 Tag 参数。也注意到不像 2XX 最终响应，代理产生对于 non-2XX

产生的最终 ACK 响应。

ACK 请求不能保证回路响应可以被认知。图 12，图 13 表示邀请的客户和服务器的状态图。

注意：对于 INVITE，因为在 INVITE 和 Final Response 之间有较长的延迟，所有 Sec.10.4 将不会很好的工作。如果 200 响应丢失了，被叫将相信存在的呼叫，但语音路径将“死掉”。因为主叫不知道被叫已经摘机了。这样，INVITE 的重传间隔将成为一个秒或者两个来限制状态的冲突。用 exponential back-off 帮助重传这个响应，帮助确认收到响应，没有给网络增加负担。

10.5.2 TCP

UA 使用 TCP 不需要重传请求，但与 UDP 一样，使用相同的算法（Section 10.5.1）去传送响应，直到接收到 ACK。

必须重发 2XX 响应如它们的可靠性，确保点到点的可靠性。如果中间代理链有个 UDP 连接，这将丢失其响应，并且不可能恢复。为了简单性，尽管这样这样做并不需要，我们还是重传非 2XX 响应的消息。

10.6 ACK 请求的可靠性

ACK 的请求不能产生响应，仅仅产生在响应一个 INVITE 请求（参考 10.5）。这个行为与传输协议有关。注意到 ACK 请求应该采用与起始 INVITE 请求不同的路径，并产生一个新的 TCP 连接。

10.7 ICMP Handling（处理 ICMP）

ICMP 消息是一种直接传输的 UDP 消息。对于请求，主机，网络，端口，或者协议不可到达的错误将被看成收到 400 类的响应。对于响应，这些错误将导致服务器取消重传响应。

源的 ICMP 消息应该被忽略。TTL 超出的错误应该被忽略。参数问题的错误应该被看成接收到 400 类型的响应。

11. SIP User Agent 的行为

这个部分描述 UAC 和 UAS 产生请求和响应的规则。

11.1 Caller Issues Initial INVITE Request（主叫发出初始邀请请求）

当 UAC 希望初始化一个呼叫，就发出一个 INVITE 请求。请求中的 To 域包含被叫的地址。Request－URI 包含同一个地址。From 域包含主叫的地址。如果 From 地址可以出现在请求中，并在同一呼叫中，由其他 User agnet clients 产生，主叫必须插入 tag 出参数到 From 域中。UAC 应该可选的增加一个 Contact Header 包含一个可以联系的地址，返回被叫到主叫的事务。

11.2 Callee Issues Response（被叫发出的响应）

被叫接收到初始的 INVITE 请求，被叫可以接受，重定向，或者拒绝呼叫。在所有的这些情况中，都要返回一个响应。响应的信息必须从请求域中拷贝下面的域：To,From,Call-ID,Cseq 和 Via 域。另外，如果在请求中包含一个或者多个 Via 头域，必须在 To 域的响应中增加 tag 参数。

因为一个从 UAC 的请求应该 Fork（分叉）并到达多个主机，UAC 将根据 tag 区分来自不同 UAS’s 的响应。UAS 可以在响应中增加一个 Contact header 域。这里包含一个被叫希望连接的被

叫地址，包括目前 INVITE 请求的 ACK。UAS 保存 TO 和 FROM 的值，包括任何 Tags。这些分别变成本地或者远端 Call Leg 地址。

11.3 Caller Receives Response to Initial Request（主叫接收并响应初始化的请求）

因为 Forking proxy，单个 INVITE 请求，多个响应可以到达 UAC。每个响应通过 To 头域中的“ tag ”参数区别。主叫应该通过每次响应的 UAS，选择识别或者结束呼叫。为了识别，它发送一个 ACK 请求，为了结束，发送一个 BYE 请求。在 ACK 或者 BYE 的 To 头域中，包括任何类型的 tag，From 头域，必须与 200 响应的 To 域一致。缺省时，ACK 或者 BYE 的 Request－URI 被设置成在 200 响应中的地址。

相反的，UAC 可以从 To 头域拷贝地址到 Request－URI。在每个响应中，UAC 也会注意到 To 和 From 的值。对于每个呼叫历程（Call Leg）,To 头域变成远端地址，而 From 头域变成本地地址。

11.4 Caller or Callee Generate Subsequent Requests（主叫或者被叫产生二级并发请求）

一旦呼叫已经建立，主叫或者被叫可以产生 Invite 或者 Bye 请求，来改变或者结束呼叫。不管是主叫还是被叫产生新的请求，在请求中的头域如下设置。对于期望的 Call Leg，To 头域设置为远端地址，From 头域设置为本地地址（两者包含任何 tags）。联系头域域与先前响应请或者请求的 Contact 域不同。Request-URI 应该设置成联系头域的值，这个联系头域被先前的从远端发送过来的请求或者响应接收，或者设置成远端地址。

11.5 Receiving Subsequent Request（接收的并发请求）

当请求后来被接收，将进行下面的检查：

1. 如果 Call－ID 是新的，不管是否是 To 和 From 头域的值，认为请求是一个新的呼叫。
2. 如果 Call－ID 已经存在，那么请求是对于一个存在的呼叫。如果 To，From，Call－ID 和 Cseq 值与先前接收的一致，则这个请求是一个重传请求。
3. 如果上面两步没有匹配到，To 和 From 域比较存在的 Call Leg 和本地及远端的地址。如果这里匹配，在请求中的 Cseq 比上一个 Cseq 值，这样对于存在的 Call Leg，这个请求是一个新的事务。

12 SIP Proxy 和 Redirect Servers 的功能

这一节详细的描述了 SIP redirect 和 proxy servers 的功能。Proxy servers 能“ fork ”连接，比如由一个进入的请求“ fork ”出几个呼出的请求。

12.1 Redirect Server

redirect server 自己不提交 SIP 请求。当收到除了“ CANCEL ”之外的请求时，它找到可供选择的地址列表并返回 3xx 类型的最终回应消息或者拒绝该请求。对于一个正常的“ CANCEL ”请求，它应该返回 2xx 类型的回应。这个回应终结这个 SIP 事务。redirect server 维护整个 SIP 事务的事务状态。redirect servers 之间的环路问题由客户端来解决。

12.2 User Agent Server

User agent servers 的功能和 redirect servers 类似，除了 User agent servers 能够接受请求并返回一个 2xx 类型的响应。

12.3 Proxy Server

这一节概述了 proxy servers 的功能特征。一个 proxy servers 可以是有状态的也可以是无状态的。如果是有状态的，那么它就会记住进入请求和外出请求的状态。如果是无状态的，那么它在产生外出请求后会丢失所有的状态。一个 forking proxy 必须是有状态的。那些接受 TCP 连接的 Proxies 在处理 TCP 连接时也必须是有状态的。一个 proxy servers 发送多点广播请求时也必须是有状态的。否则，当 proxy servers 丢失它的请求时，TCP 客户端将不会重发请求。

当一个有状态的 proxy servers 向上发送了最终的响应，并且收到了对方的最终响应过 32 秒后才能变成无状态。

一个有状态的 proxy servers 实际上和 UAS/UAC 相似，但是它并不是 UAS 和 UAC 直接结合在一起。（事实上 proxy servers 除了 ACK 和 CANCEL 外并不发起其他的请求。）一个 proxy servers 实现了 UAS 的状态机用来接受进入的请求，实现 UAC 的状态机用来产生外出的请求，只是当它收到一个回应 INVITE 请求的 2xx 响应时处理会有差别。proxy servers 收到一个 2xx 响应时它并不是产生一个 ACK 请求而是将该响应向上转发给主叫。此外，同一个无状态的 proxy servers 一样，对应于 200 响应的 ACK 请求总是向下转发给 UAS。

一个无状态的 proxy servers 转发它收到的所有向下请求，转发它从上面收到的所有响应。

12.3.1 代理请求

proxy server 在代理一个请求前必须检查前向环路。在下面的情况下表明该请求处理出现了环路：

当 proxy server 发现它自己的地址出现在 Via 头域中；

当针对 10.46.6 节列举的域进行的哈希求值结果和包含了 proxy server 地址的 Via 头域的“branch”参数的哈希部分相同时。

当 via 头域中包含该广播组的地址时，proxy server 不能将请求转发给广播组。

proxy server 必须将所有的请求的头域 copy 到外出请求的对应部分。它也可以添加其它的头域。

proxy server 总是将自己的地址添加到那些由进入的请求所产生外出请求的 via 头域中。每一个 proxy server 必须添加一个“branch”参数（10.46 节）。

除了 outbound proxies 外的所有代理服务器应该改变 Request-URI 以便指示它打算将请求发送到哪一个服务器。

12.3.2 代理响应

代理服务器只处理那些最上层 via 域的地址与它自己的地址相同的那些响应消息（见 10.46 节）。一个响应消息的 via 域如果不匹配，那么它必须被丢弃。

12.3.3 无状态的代理服务器：代理响应

一个无状态的代理服务器必须遵从 10.46 节的规定以决定将响应送往的地址。

一个无状态的服务器不能产生它自己的临时响应。它必须向上转发包括 100 的所有的临时响应。

12.3.4 有状态的代理服务器：接收请求

当一个用有状态的代理服务器接收到一个请求，它将对照以前的请求记录检查 to(包括 tags)，from(包括 tags)，Call-ID,Cseq,Request-URI 以及最上层的 via。如果存在相同的记录，那么该请求是一个重传的请求。当一个临时或者最终的响应是重传的响应，那么将按照服务器的状态机来处理。如果不存在相同的记录，那么该请求对应一个新的事务，该请求将被代理转发。

一个有状态的代理服务器可以产生它自己临时的响应。

12.3.5 有状态的代理服务器：接收 ACKs

当接收到一个 ACK 请求后，它被代理转发除非该请求的 to(包括 tags)，from(包括 tags)，Call-ID,Cseq 头域和该代理服务器发送出去的那些非 2xx 类型响应消息的头域相同。在头域相同的情况下，该请求将在本地处理并停止重传响应消息。

12.3.6 有状态代理服务器：接收响应

当一个代理服务器接收到一个响应并且该响应已通过 via 头域检查，那么该服务器将对照前面请求的对应值来检查响应消息的 to(不包括 tags)，from(包括 tags)，Call-ID,Cseq 域值。当检查到没有相同的记录时，该响应将被向上转发。如果存在相同的记录，那么将检查 via 域中的“ branch ”标签。当该标签和一已知的标签相同时，那么该响应就是对应标签的分支的响应，并由对应该分支的虚拟客户端来处理。否则，该响应被丢弃。

一个有状态的代理服务器应该遵从 17.4 节的规则来决定响应消息是否应该向上转发处理。如果响应消息将被转发处理，对于无状态的代理服务器应该遵从上面的规则，对应 TCP，下面是一些附加的内容。如果通过 TCP 接收到一个请求（在最上层的 via 头域中指示），代理服务器将会检查对应的地址当前是否有开放的连接。如果有，对应的响应消息将从该连接上发送。否则，将依照 via 头域中的地址和端口来建立一个新的 TCP 连接，响应消息将从新建立的连接上发送。注意到这意味着一个 UAC 或者代理服务器必须在 TCP 连接上准备接收响应消息。即使通过 TCP 传输，代理服务器必须重传最终非 200 类型的响应消息。

12.3.7 无状态，非派生代理服务器

这一类型的代理服务器对于一个进入的请求只产生单个地外出请求，也就是说他们不派生请求。然而，就像 17.4 节中描述的那样，服务器可以选择控制自己的模式以让自己可以派生出多个外出请求。

该服务器能转发请求和响应。对应 SIP 事务，它不保存任何状态。由服务器链中的重定向服务器或者有状态的代理服务器来保证状态。

为了加快重发的速度，一个代理服务器应该缓存地址转换的结果和响应消息。当缓存的条目过期之后，代理服务器将不知道一个进入的请求是否是上次请求的重发请求。服务器将把该请求当作一个新的请求并开始新的搜索处理。

12.4 派生代理服务器

如果服务器估计得到一个最终响应需要多余 200ms 的时间，那么它必须对一个请求（除了 ACK ）立刻回一个 100 类型的响应。

对应于 INVITE 请求的成功响应可以包含一个 Contact 头域，这样接下来的 ACK 或者 BYE 消息就不用进行代理搜索。如果代理服务器希望将来的请求通过它路由，它就应该将 Record-Route 头加到请求中（10.34 节）。

下面的 C 代码描述了一个代理服务器由一个进入的 INVITE 请求产生多个外出请求。

request(r, a, b)函数发送一个 branch id 为 b，请求类型为 r，地址为 a 的 SIP 请求。
await_response()函数等待直到收到一个响应并且返回该响应。
close(a)函数关闭连接到地址为 a 的客户端的 TCP 连接。
response(r)函数给客户端发送一个响应。
ismulticast()函数当地址是广播地址时返回 1，否则返回 0。
timeleft 变量指示离最大响应时间到达之前还有多长时间。
recurse 变量指示当收到一个 3xx 类型的响应时是否继续尝试其它的服务器。
一个服务器可以决定只继续尝试特定的地址，比如，那些和该代理服务器在同一个域里的服务器。这样一个初始的多播请求能够导致额外的单播请求。

```
/* 请求类型 */
typedef enum {INVITE, ACK, BYE, OPTIONS, CANCEL, REGISTER} Method;

process_request(Method R, int N, address_t address[], int expires)
{
  struct {
    char *branch;         /* branch token */
    int branch_seq;       /* branch sequence number part */
    int done;             /* has responded */
  } outgoing[];
  char *location[];       /* list of locations */
  int heard = 0;          /* number of sites heard from */
  int class;              /* class of status code */
  int timeleft = expires; /* expiration value */
  int loc = 0;            /* number of locations */
  struct {                /* response */
    int status;           /* response: CANCEL=-1 */
    int locations;        /* number of redirect locations */
    char *location[];     /* redirect locations */
    address_t a;          /* address of respondent */
    char *branch;         /* branch token */
    int branch_seq;       /* branch sequence number */
  } r, best;              /* response, best response */
  int i;

  best.status = 1000;
```

```
for (i = 0; i < N; i++) {
  request(R, address[i], i);
  outgoing[i].done = 0;
  outgoing[i].branch = "";
  outgoing[i].branch_seq = i;
}

while (timeleft > 0 && heard < N) {
  r = await_response();
  class = r.status / 100;

  /* If final response, mark branch as done. */
  if (class >= 2) {
    heard++;
    for (i = 0; i < N; i++) {
      if (r.branch_seq == outgoing[i].branch_seq) {
        outgoing[i].done = 1;
        break;
      }
    }
  }
  /* CANCEL: respond, fork and wait for responses */
  /* terminate INVITE with 40
  else if (class < 0) {
    best.status = 200;
    response(best);
    for (i = 0; i < N; i++) {
      if (!outgoing[i].done)
        request(CANCEL, address[i], outgoing[i].branch);
    }
    best.status = -1;
  }

  /* Send an ACK */
  if (class != 2) {
    if (R == INVITE) request(ACK, r.a, r.branch);
  }

  if (class == 2) {
    if (r.status < best.status) best = r;
    break;
  }
  else if (class == 3) {
    /* 服务器可以选择重复请求。服务器必须检查以前是否尝试过这个
```

```
            * 地址，该地址是否包含在进入请求的 via 域中。
            * 为了简单一些，这里的检查就忽略掉了。
            * 如果服务器没有重复请求，那么多播地址不能返回给客户端。
            */
           if (recurse) {
             multicast = 0;
             N += r.locations;
             for (i = 0; i < r.locations; i++) {
               request(R, r.location[i]);
             }
           } else if (!ismulticast(r.location)) {
             best = r;
           }
         }
         else if (class == 4) {
           if (best.status >= 400) best = r;
         }
         else if (class == 5) {
           if (best.status >= 500) best = r;
         }
         else if (class == 6) {
           best = r;
           break;
         }
       }

       /* We haven't heard anything useful from anybody. */
       if (best.status == 1000) {
         best.status = 408; /* request expired */
       }
       if (best.status/100 != 3) loc = 0;
       response(best);
     }
```

响应按下面的方式处理。当所有的请求收到最终的响应（对应单播）或者 60 秒过期（对于多播），处理过程就结束了。代理服务器应该发送 CANCEL 给所有未完成的分支交易。如果过了 60 秒或者请求头域中指示过期时间后响应没有全部收到，服务器应该发送最合适的最终响应给客户端。如果没有收到响应，服务器返回 408（超时）给客户端。

转发消息时，代理服务器必须转发整个响应，包括指定响应的所有头域和 body 域。

1xx：代理服务器必须向上转发所有的大于 100 的临时响应并且不应转发 100 响应。

2xx：对于 INVITE 请求，代理服务器必须向上转发响应给客户端，它不必向下发送 ACK。对于其它请求，如果它没有向上转发其它响应，它只需向上转发响应。收到 2xx 消息后，服务器应该通过发送 CANCEL 请求来中断所有未完成的请求。（中断未完成的请求可以发现额外消耗的资源以节约资源。同样，当被叫当前多次注册时，INVITE 请求能同时呼叫多个终端。）

如果请求类型不是 INVITE，当代理服务器已经向上转发了最终的响应，它应该丢掉 2xx 类型响应。

3xx：对于 INVITE 请求，服务器必须发送一个 ACK。它可以给 Contact 中列出的地址循环发送 ACK。否则，如果没有 2xx 或者 6xx 类型的响应，那么应返回数字小的响应。

地址列表不能合并因为这样将不能转发鉴权响应。同样，响应可能包含消息体，所以合并并不可行。

4xx，5xx：对于 INVITE 请求，代理服务器必须发送 ACK。如果该响应的状态码比以前的 4xx 和 5xx 响应小，它将记下该响应。最后，如果没有 2xx,3xx 或者 6xx 类型的响应，那么将返回最低响应类型的响应。在一系列的最低响应类型的响应中，代理服务器可以选择任一返回。

如果 401 或者 407 是最小数字响应，代理服务器应该从所有的 401 和 407 响应中收集所有的 WWW-Authenticate 和 Proxy-Authenticate 头并且返回他们。

6xx:对于 INVITE 请求，代理服务器发送 ACK。它转发 6xx 类型的响应除非收到一个 2xx 类型的响应。就像 2xx 响应里描述的一样，其它的未完成请求应该通过发送 CANCEL 来中断。和 2xx 响应不同的是，只有一个 6xx 类型响应被转发，因为 ACKs 是在本地产生。

根据响应的 via 头域，代理服务器依据 Call-IDs 转发所有的已完成事务的响应。对于 call legs 未知的 BYE 请求，用户代理服务器发送状态码为 481（事务不存在）的响应。对于 call legs 未知的 ACK 请求，他们只是将请求丢弃。

对于 ACK 和 BYE 的请求，选择转发地址有特别的考虑。在大多数的情况下，这些请求将不通过代理服务器而直接发送给对方，不由代理服务器来决定转发。

代理服务器可以自己选择维护呼叫状态的时间长度。如果代理服务器依然保存着它转发 INVITE 请求的目 的地址列表，它应该直接将 ACK 请求只发送给它对应的下游服务器。

13 安全考虑

13.1 机密和安全：加密

13.1.1 End-to-End 加密

SIP 请求和响应可能包含关于通信模式以及个人通信内容的敏感信息。SIP 消息也可能包含了自己的会话加密密钥。SIP 支持两个补充的加密方式以保护秘密：

对于 SIP 消息体以及敏感头域的 End-to-End 加密；

hop-by-hop 加密以防止偷听跟踪主叫和被叫；

SIP 请求或者响应不能作为一个整体进行 End-to-End 的加密,因为代理服务器需要检查 To 和 via 等头域以便 SIP 请求能够被正确的路由。hop-by-hop 加密对整个 SIP 请求或者响应加密，这样包检漏器或者偷听者就不能看到主叫和被叫。注意代理服务器仍然能看见主叫和被叫，而且通过网络流量分析这些信息也可以得到。因此，这种方法提供了一种有限的但仍值得去做的保护。

End-to-End 加密依赖请求中双方共享出来的加密键值。典型例子是发送的消息使用接收者的公开密钥加密，这样只有接收者能读到该消息。

一个 SIP 请求（或者响应）消息被分成两部分来加密，一部分是需要加密的部分，另一部分是一个简短的头域没有加密。SIP 消息的一些部分，比如请求行、响应行以及一些表 4 和表 5 的“proxy”列中表上了 “r”的头域需要被代理服务器读取和返回，所以它们不能被 End-to-End 加密。包括呼叫的目的地址（to）和 转发路径(via)在内可能敏感的信息需要作为明文保留。如果鉴权头域包含数字

签名，那么它必须作为明文保留。但是当它包含“basic”或者“digest”鉴权时它可以被加密。

其它的头域可以依据发送的意愿被加密或者可以作为明文传输。Subject、Allow 和 Content-Type 头域通常被加密。Accept、 Accept-Language、Date、 Expires、 Priority、 Require、 Call-ID 、 Cseq 以 及 Timestamp 头域将会作为明文保留。

所有未加密的域必须放在那些被加密域的前面。消息的加密部分从将被加密的头域的第一个字符开始一直到消息体的结尾。像下面展示的那样，如果没有头域被加密，加密将从最后一个头域的第二个 CRLF 开始。回车和换行符被显示成“$”。消息的被加密部分略述如下。

```
INVITE sip:watson@boston.bell-telephone.com SIP/2.0$
Via: SIP/2.0/UDP 169.130.12.5$
To: T. A. Watson <sip:watson@bell-telephone.com>$
From: A. Bell <sip:a.g.bell@bell-telephone.com>;tag=7abm$
Encryption: PGP version=5.0$
Content-Length: 224$
Call-ID: 187602141351@worcester.bell-telephone.com$
Content-Type: message/sip
CSeq: 488$
$
*******************************************************
* Subject: Mr. Watson, come here.$                    *
* Content-Type: application/sdp$                      *
* $                                                   *
* v=0$                                                *
* o=bell 53655765 2353687637 IN IP4 128.3.4.5$        *
* s=Mr. Watson, come here.$                           *
* t=0 0$                                              *
* c=IN IP4 135.180.144.94$                            *
* m=audio 3456 RTP/AVP 0 3 4 5$                       *
*******************************************************
```

必须加上加密头域以指示加密所用的机制。添加上 Conten-Length 域以指示被加密的消息体的长度。同普通的 SIP 消息体一样，被加密的消息体的前面有一空行。

拥有正确解密密钥的被叫用户代理收到消息后，消息被解密并且将明文部分加到明文头域后。由于被解密后实际消息体的长度是不明确的，因此不需要在被加密的消息体里保留额外的 Content-Length 头域。

指示“message/sip”的 Content-Type 可以被加上，但是接收到后将不处理。

当没有 SIP 头域需要被加密时，消息将会如下所示。注意到被加密的消息体必须包含一个空行（以 CRLF 开始）以隔开可能出现的 SIP 头域和 SIP 消息体。

```
INVITE sip:watson@boston.bell-telephone.com SIP/2.0$
Via: SIP/2.0/UDP 169.130.12.5$
To: T. A. Watson <sip:watson@bell-telephone.com>$
```

```
From: A. Bell <a.g.bell@bell-telephone.com>;tag=7abm$
Encryption: PGP version=5.0$
Content-Type: application/sdp$
Content-Length: 107$
Call-ID: 187602141351@worcester.bell-telephone.com$
CSeq: 488$
$

**********************************************
* $                                          *
* v=0$                                       *
* o=bell 53655765 2353687637 IN IP4 128.3.4.5$  *
* c=IN IP4 135.180.144.94$                   *
* m=audio 3456 RTP/AVP 0 3 4 5$              *
**********************************************
```

18.1.2 SIP 响应的安全

SIP 请求可以安全的通过 End-to-End 的加密和鉴权发送给被叫用户代理，该用户代理可以以非安全方式发送响应。虽然 SIP 安全模式里容许这样，但是这并不是一个好主意。然而，除非清楚的指出正确的行为，否则被叫用户代理并不是总能推断出什么是合适的行为。所以，当一个原始的请求采用 End-to-End 的加密时，响应消息需要用到的密钥应该在请求中给出（10.36 节）。如果不这样做，被叫用户代理在响应的时候可能无法得到合适的密钥。所以为了防止依靠猜测键值来处理，我们规定当一个被叫用户代理收到一个没有规定响应中所需密钥的请求时，响应消息应该不被加密。

任何一个在请求中被加密的头域在响应中也应该被加密。如果 Contact 头域包含敏感信息，那么它可以被加密，或者也可以不加密以便代理服务器有更多的机会进行位置搜寻。

13.1.3 被代理服务器加密

通常，代理服务器不容许改变 End-to-End 的头域和消息体。然而，代理服务器可以采用接收者的密钥加密未签名的请求或者响应。

如果终端系统不能加密，代理服务器需要加密 SIP 请求以加强安全。

13.1.4 hop-by-hop 加密

也可以采用传输层或者网络层的安全机制来保证 SIP 请求和响应的安全。这里没有定义或者推荐特殊的机制。两个可能的加密机制是 IPSEC 或者 TLS。特殊的加密机制通常用在带外的情况下，比如说用手工配置。

13.2 消息完整和访问控制：鉴权

必须采用保护方法来防止活跃的攻击者发送修改过的 SIP 请求和响应。我们采用用来保证 SIP 消息真实性的加密方法来对初始发送者进行鉴权。然而，“basic”和“digest”鉴权机制只提供鉴权功能

，不保证消息的完整性。

传输层或者网络层鉴权可以用来支持 hop-by-hop 鉴权。SIP 也扩展了 WWW-Authenticate(10.48 节) 和 Authorization (10.11 节)头域以及它们代理服务器的配对物以包括更强大的数字签名。SIP 也支持 HTTP 的"basic" 和 "digest"机制(见 19 节)以及其它的将要定义的 HTTP 鉴权机制。

SIP 请求可以采用包含了多个头域的数字签名的 Authorization 头域来鉴权。多个头域包括 request method 和 version number 以及 payload，它们每一个在客户端和被叫用户代理间传送时都不能被修改。请求里的 Authorization 头域用来对 End-to-End 发送消息到代理服务器和被叫用户代理的初始发送者进行鉴权，通过它，我们可以对返回它们自己错误码的被叫用户代理或者代理服务器进行鉴权。如果需要，也可以提供 hop-by-hop 鉴权，比如说，通过 IPSEC Authentication 头。

通常，我们对一些特定的请求 URI、realm 以及一个保护域进行 SIP 鉴权。因此，对于 basic 和 digest 鉴权来说，每一个这样的保护域都有它们自己的用户群和 secrets。当一个用户代理不关心不同的请求 URIs,并且一个特殊的请求 URI 没有它自己的 realm 或者用户群，那么建立一个全局的用户名、secrets 和基本询问的 realm 是有意义的。同样的，SIP 实体同 PSTN 网关一样包含许多用户，当被要求验证时它可以不依赖请求的 URI 而尝试预先配置好的全局用户名和 secrets。

对于鉴权，SIP 没有规定使用哪一种数字签名机制。像上面指出的那样，SIP 实现者也可以采用 "basic" 和 "digest"鉴权机制以及其它为 HTTP 定义的鉴权机制。注意“basic”鉴权作为一种安全机制有它的局限性。以下所述不适合该机制。

对 SIP 请求进行加密签名时，SIP 头域的顺序非常重要。当出现一个鉴权头域时，它表示跟在该头域后面所有的头域都包括进了数字签名。因此，那些必须或者应该被代理服务器修改的 hop-by-hop 头域必须放在鉴权头域的前面，它们通常会被代理服务器修改。那些可能被代理服务器修改的 hop-by-hop 头域可以放在鉴权头域的前面或者后面。当放在前面时，他们可以被代理服务器修改。当放在后面时，他们不能被代理服务器修改。

客户端通过下面的域来构造一个消息来完成一个签名的请求：request method(大写)及随后的域(不包括 LWS)；SIP version number 及随后的域(不包括 LWS)；将要被签名的请求头以及消息体。这样被构建的消息就被签名了。

比如说，如下所示的 SIP 请求：

```
INVITE sip:watson@boston.bell-telephone.com SIP/2.0
Via: SIP/2.0/UDP 169.130.12.5
Authorization: PGP version=5.0, signature=...
From: A. Bell <sip:a.g.bell@bell-telephone.com>;tag=7abm
To: T. A. Watson <sip:watson@bell-telephone.com>
Call-ID: 187602141351@worcester.bell-telephone.com
Subject: Mr. Watson, come here.
Content-Type: application/sdp
Content-Length: ...

v=0
o=bell 53655765 2353687637 IN IP4 128.3.4.5
s=Mr. Watson, come here.
```

```
t=0 0
c=IN IP4 135.180.144.94
m=audio 3456 RTP/AVP 0 3 4 5
```

这样被签名的数据块如下所示：

```
INVITE SIP/2.0From: A. Bell <sip:a.g.bell@bell-telephone.com>;tag=7abm
To: T. A. Watson <sip:watson@bell-telephone.com>
Call-ID: 187602141351@worcester.bell-telephone.com
Subject: Mr. Watson, come here.
Content-Type: application/sdp
Content-Length: ...

v=0
o=bell 53655765 2353687637 IN IP4 128.3.4.5
s=Mr. Watson, come here.
t=0 0
c=IN IP4 135.180.144.94
m=audio 3456 RTP/AVP 0 3 4 5
```

那些希望鉴权的客户端必须采用规范的表格在鉴权头域的下面构建一部分的消息。这样代理服务器就可以分析消息、分拆消息并且重新构建它而不会因为额外的空白区域而导致鉴权失败。规范表格的构建遵从下面的规则：

?没有缩写的头域；
?如这篇文章所示头域名大写；
?在头域名和冒号之间没有空白区域；
?冒号后有一个空格；
?采用 CRLF 换行；
?没有折叠的行；
?头域的值列表之间没有逗号；每一个都必须表现为一个单独的头；
?在 token 之间、在 token 和引用的字符串之间以及在引用的字符串之间只有一个单独的 SP;在 token 或者引用的字符串后没有 SP;
?除了如上所述在头域的冒号后，在 token 和 separator 之间不能有 LWS;
?即使 display-name 是空的，To 和 From 头域总是包含“<”和“>”分割符；

注意到，当一个消息同时采用加密和数字签名授权时，应该先加密，然后产生数字签名。接收到消息后，先检查数字签名然后解密。

一个客户端可以要求包含请求头域：signed-response 以要求服务器对响应签名。客户端通过 WWW-Authenticate 头域来指示需要的鉴权方法。当请求服务器返回一个鉴权响应而实际上返回未鉴权的响应时，我们按照 18.2.1 节所述处理。

13.2.1 信任的响应

一个偷听者有可能监听请求然后产生一个未授权的响应去中断、重定向或者干扰这次呼叫。(即使一个加密的请求也包含了足够的信息去伪造一个响应。)

客户端必须特别谨慎的处理 3xx 类型的重定向响应。因此，比如说当一个客户端知道被叫用户代理的公共密钥并且要求响应被鉴权时收到一个 301 响应时，那么应该将该响应当作可疑的响应处理。在这种情况下正确的做法是被叫用户产生一个包含了 Contact 域的注明了日期的响应，然后对它进行签名并将签名后的响应发给即将提供重定向服务的代理服务器。这样响应就能正确的被鉴权。客户端不能在未警告用户鉴权失败的情况下自动重定向该请求。

可能出现其它的问题，比如 6xx 错误响应只是简单的中断搜寻，或者 4xx 和 5xx 响应返回失败。

如果采用 TCP,代理服务器应该将 4XX 和 5XX 看作有效，因为它们不是中断一个搜寻。然而，一个欺诈代理服务器伪造一个 6XX 响应将会不正确的中断搜寻。如果客户端要求，6XX 响应应该被鉴权。如果不这样做将会导致客户端丢弃 6XX 响应并且继续搜寻。

如果采用 UDP,对于 6XX 响应存在同样的问题，只是一个活跃的偷听者能够产生一个响应使代理服务器或者客户端相信产生了一个实际上并没有产生的错误。通常 4XX 和 5XX 响应没有被被叫用户代理进行签名，因此没有简单的方法去发现伪造的响应。在 SIP 请求中采用 hop-by-hop 加密可以有效的防止相对于 TCP 使用 UDP 而产生的一些额外问题。

这些攻击通过要求进行响应鉴权以及丢掉未鉴权的响应而得到有效的防治。当服务器端不能进行响应鉴权时返回一个普通的 420 响应。

13.3 被叫安全

用户地址和 SIP 发起的呼叫可能会侵犯被叫的安全。实现者应该能够针对每一个用户来限制该类被叫用户可以接受的地址以及可能的信息。

13.4 拒绝服务

攻击者能够欺骗 via 头域以直接对第三方进行响应，使 SIP UAS 或者代理服务器产生流量。我们可以采用要求一个已经存在的安全连接比如说 TLS 或者 Ipsec 来防止这种攻击。这是一个合适的方法，比如说，在那些需要交换大量信息流量的代理服务器间或者在用户代理和它的 outbound 代理服务器间。

如果这样的安全连接并不可行，客户端和代理服务器应该对未鉴权的请求只是简单的返回 401 或者 407 而不是采用普通的响应重传算法。重传 401 或者 407 状态响应会使攻击者伪造 VIA 头域将流量导向第三方 的问题被放大。

13.5 已知的安全问题

无论对于 TCP 或者 UDP,一个已经存在的拒绝服务的例子是欺骗服务器发送 6XX 响应。虽然如果客户端要求鉴权应该选择丢弃这样的响应，但是代理服务器不能这样做。它必须将 6XX 响应转发给客户端。客户端可以丢弃该响应，但是如果它重复请求它将可能重新到达同一个欺诈代理服务器，并且处

理过程将会重复。

14 使用 HTTP Basic 和 Digest 机制来进行 SIP 鉴权

SIP 实现者可以使用 HTTP 的 basic 和 digest 鉴权机制（RFC2617）来提供初步的安全机制。这一节概述了在 SIP 中使用这些机制。SIP 中的 basic 操作和 HTTP 中的几乎一样。这一节大概的描述了该操作并指出了当它应用在 SIP 中时的一些差别，在 RFC2617 中将会详细的描述该操作。由于 RFC2543 的鉴权机制基于 RFC2069 的 HTTP basic 和 digest,支持 RFC2617 的 SIP 服务器必须保证它们能向后兼容 RFC2069。处理该向后兼容的方法在 RFC2617 中描述。

14.1 框架

SIP 鉴权的框架和 HTTP 中（RFC2617）相似。特别是，auth-scheme、auth-param、challenge、realm、 realm-value、credentials 的 BNF 是同样的。在 SIP 中用户代理服务器使用 401 响应对客户端的鉴权表示异议。注册服务器和重定向服务器可以对鉴权请求使用 401 响应，但是代理服务器不能这样，它使用 407 响应。同 RFC2617 一样，我们需要在各种信息中包含 Proxy-Authenticate、Proxy-Authorization、 WWW-Authenticate 和 Authorization 域。

由于 SIP 没有规范的 root URL 的概念，对于 SIP 来说保护区域的概念有不同的理解。对于那些 SIP Request-URI 中的 userinfo、host 以及 port 的值相同的 SIP URI 来说，realm 就是一个保护域。举例如下：

```
INVITE sip:alice.wonderland@example.com SIP/2.0
WWW-Authenticate:  Basic realm="business"
and
INVITE sip:aw@example.com SIP/2.0
WWW-Authenticate: Basic realm="business"
```

依据这一规则定义不同的保护域。

当 UAC 接收到 401 或者 407 响应并重传带上了 credential 的请求时，它必须增加 Cseq 头域的值就像它发送了一个更新的请求后通常所做的一样。

14.2 basic 鉴权

basic 鉴权的规则在[41]中定义，只是其中的"origin server"被替换成"user agent server,redirect server,或者 registrar"。

由于 SIP URIs 部分层，在[41]中所描述的"所有与 Request-URI 中的 path 域的最后一个元素处于同一深度或者更深的通路也在由当前 challenge 的 Basic realm 值所描述的保护空间中。"对于 SIP 并不适用。

SIP 客户端在未从服务器收到其它的 challenge 时可以抢先在相同的保护域（如上所定义）中发送相应的鉴权头以请求 SIP URIs。

由于它的低安全性，我们不推荐使用 basic 鉴权方法。然而，服务器应该支持它以便处理那些仍然使用它们的老的 RFC2543 终端。

14.3 Digest 鉴权

digest 鉴权的规则遵从[41]的定义，除了将“HTTP 1.1”替换成“SIP/2.0”还有如下的差别：

1. challenge 中的 URI 有如下的 BNF:

   URI = SIP-URL

2. RFC2617 种的 BNF 有一个错误：URI 没有加上引号。（3.5 节的例子是正确的）对于 SIP,URI 必须加上引号。

3. digest-uri-value 的 BNF 是：

   digest-uri-value = Request-URI ; 如同 4.3 节中定义

4. 依据 Etag 来选择一个 nonce 的例子对 SIP 无效。

5. RFC2617 [41]中的 cache 操作对 SIP 无效。

6. RFC2617 [41]要求服务器检查请求行中的 URI 以及鉴权头中的 URI 是否指向同一个地方。在 SIP 中，由于一些代理服务器的转发，这两个 URI 实际上可能指向不同的用户。因此，在 SIP 中，一个服务器可能检查鉴权头中的 request-uri 是否对应服务器希望进行呼叫用户。

RFC2543 不容许使用 Authentication-Info 头（它在 RFC2069 中可以使用）。然而，我们现在使用该头域，因为它提供了对消息体的完整性检查以及相互鉴权。RFC2617 [41]定义了向后兼容在请求中使用 qop 属性的机制。这些机制必须有服务器使用来确定客户端是否支持了 RFC2069 中没有定义而 RFC2617 支持的新机制。

14.4 代理鉴权

Proxy-Authentication 和 Proxy-Authorization 的使用除了一点不同之外，其它和[41]中描述的类似。 代理服务器不能添加 Proxy-Authorization 头。同其它响应的处理过程一样，407 响应必须向上转发给客户端。当代理服务器要求鉴权时，客户端应该加上包含了 credential 的 Proxy-Authorization 头域。

如果代理服务器重新发送包含了 Proxy-Authorization 头域的请求，那么在新的请求中它必须增大 Cseq 的值。然而，这意味着由于 Cseq 值的不同提交初始请求的 UAC 将会丢弃从 UAS 收到的响应。

10.31 和 10.32 节描述了在 SIP 中使用这些域的额外信息。

如果请求被派生，401 响应也可能包含几个 challenges。

15． 例子

在下面的例子中，为了简短我们常常忽略消息体以及对应的 Content-Length 和 Content-Type 头域。

15.1 注册

在启动时，一个在 saturn.bell-tel.com 主机上的用户通过多播在本地注册，本地 SIP 服务器的名字叫 bell-tel.com。在这个例子中，在 saturn 上的用户代理希望在 UDP 端口 3890 上接收 SIP 请求。

```
C->S: REGISTER sip:bell-tel.com SIP/2.0
      Via: SIP/2.0/UDP saturn.bell-tel.com
      From: <sip:watson@bell-tel.com>;tag=19al
      To: sip:watson@bell-tel.com
      Call-ID: 70710@saturn.bell-tel.com
      CSeq: 1 REGISTER
      Contact: <sip:watson@saturn.bell-tel.com:3890;transport=udp>
      Expires: 7200
```

注册信息在两小时后过期。将来任何从 sip.bell-tel.com 来的对 watson@bell-tel.com 的邀请将会被从 定向到 watson@saturn.bell-tel.com 的 UDP 端口 3890。

如果 Watson 想要注册到其它的地方，比方说，当移动时使用在线业务，它应该取消自己的注册地址后更新自己的信息。

```
 C->S: REGISTER sip:bell-tel.com SIP/2.0
      Via: SIP/2.0/UDP saturn.bell-tel.com
      From: <sip:watson@bell-tel.com>;tag=19al
      To: sip:watson@bell-tel.com
      Call-ID: 70710@saturn.bell-tel.com
      CSeq: 2 REGISTER
      Contact: *
      Expires: 0

C->S: REGISTER sip:bell-tel.com SIP/2.0
      Via: SIP/2.0/UDP saturn.bell-tel.com
      From: <sip:watson@bell-tel.com>;tag=19al
      To: sip:watson@bell-tel.com
      Call-ID: 70710@saturn.bell-tel.com
      CSeq: 3 REGISTER
      Contact: sip:tawatson@example.com
```

现在，服务器会将任何关于 Watson 的请求的 Request-URI 置为 tawatson@example.com 然后转发到 example.com 的服务器上。对于位于 example.com 的服务器，要想到达 Watson，它必须发送 REGISTER，或者通过其它的方法通知 Watson 当前位置所在的服务器。

可以使用第三方注册。这里，秘书 jon.diligent 为她的老板 T.Watson 注册：

```
C->S: REGISTER sip:bell-tel.com SIP/2.0
      Via: SIP/2.0/UDP pluto.bell-tel.com
      From: <sip:jon.diligent@bell-tel.com>;tag=7eff
      To: sip:watson@bell-tel.com
      Call-ID: 17320@pluto.bell-tel.com
      CSeq: 1 REGISTER
      Contact: sip:tawatson@example.com
```

该请求可能发送到位于 bell-tel.com 注册服务器或者位于 example.com 的服务器。在后一种情况下，位于 example.com 的服务器将会将请求转发到 Request-URI 中所指示的地址。然而，Max-Forwards 头可以用来限制注册到该服务器。

15.2 邀请参加多方会议

下面是邀请 bob@example.com 参加多方会议的第一个例子。所有的例子使用 SDP(RFC2327)来描述会话描述格式。

15.2.1 请求

```
C->S: INVITE sip:bob@one.example.com SIP/2.0
      Via: SIP/2.0/UDP sip.example.com;branch=7c337f30d7ce.1
        ;maddr=239.128.16.254;ttl=16
      Via: SIP/2.0/UDP mouse.wonderland.com
      From: Alice <sip:alice@wonderland.com>;tag=1ija
      To: Bob <sip:bob@example.com>
      Call-ID: 602214199@mouse.wonderland.com
      CSeq: 1 INVITE
      Contact: Alice <sip:alice@mouse.wonderland.com>
      Subject: SIP will be discussed, too
      Content-Type: application/sdp
      Content-Length: 187

      v=0
      o=user1 53655765 2353687637 IN IP4 128.3.4.5
      s=Mbone Audio
      t=3149328700 0
      i=Discussion of Mbone Engineering Issues
      e=mbone@somewhere.com
```

```
c=IN IP4 224.2.0.1/127
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

上面的请求头域 From 表示该请求由 alice@wonderland.com 发出并且由 To 头域指示发往 bob@example.com. Via 头域列出了从发起者（列表中最后一个元素）到被叫的通路上的服务器。在上面的例子中，消息的 ttl 被社置为 16 并从 sip.example.com 服务器发出最后广播到管理范围群为 239.128.16.254 的广播地址。第二个 Via 头域表示它从 outbound proxy mouse.wonderland.com 发出。

Request-URI 表示该请求当前被发送到 bob@one.example.com,该地址是位于 example.com 域的 SIP 服务器 查找到的被叫本地地址。

在这个例子中，像 Content-Type 头域中所述，使用 SDP 进行会话描述。头部以空行结束，后面接着包含了会话描述的消息体。

15.2.2 响应

被叫用户代理，直接或间接通过代理服务器，指示正在给被叫方振铃。

```
S->C: SIP/2.0 180 Ringing
      Via: SIP/2.0/UDP sip.example.com;branch=7c337f30d7ce.1
        ;maddr=239.128.16.254;ttl=16
      Via: SIP/2.0/UDP mouse.wonderland.com
      From: Alice <sip:alice@wonderland.com>;tag=1ija
      To: Bob <sip:bob@example.com> ;tag=3141593
      Call-ID: 602214199@mouse.wonderland.com
      CSeq: 1 INVITE
```

下面是一个对邀请进行响应的例子。响应的第一行指出了 SIP 版本号，这是一个 200 响应，它表示请求成功。Via 头域从请求中 copy 过来，当响应消息从请求消息的路径折回时，via 头域元素随着转发被去掉。

如果要求，被叫用户代理可以加上一个新的鉴权头域。在请求消息中，从 Call-ID 到随后的头域直接从初始请求中取出。From 域的初始意义保持不变（也就是说，它表示会话的发起者）。

另外，Contact 头域指出了用户所在主机的详细信息，或者是主叫主机将要到达的相关代理服务器联系点。

```
S->C: SIP/2.0 200 OK
      Via: SIP/2.0/UDP sip.example.com;branch=7c337f30d7ce.1
        ;maddr=239.128.16.254;ttl=16
      Via: SIP/2.0/UDP mouse.wonderland.com
      From: Alice <sip:alice@wonderland.com>;tag=1ija
      To: Bob <sip:bob@example.com> ;tag=3141593
```

```
        Call-ID: 602214199@mouse.wonderland.com
        CSeq: 1 INVITE
        Contact: <sip:bob@one.example.com>
```

主叫通过发送 ACK 请求消息给 Contact 头域中的地址来确认邀请。

```
C->S: ACK sip:bob@one.example.com SIP/2.0
        Via: SIP/2.0/UDP mouse.wonderland.com
        From: Alice <sip:alice@wonderland.com>;tag=1ija
        To: Bob <sip:bob@example.com> ;tag=3141593
        Call-ID: 602214199@mouse.wonderland.com
        CSeq: 1 ACK
```

20.3 两方呼叫

对于两方 internet 电话呼叫，响应消息必须包含发送地址的描述。在下面的例子中，Bell 呼叫 Watson。Bell 指出了它能够接收 RTP 语音编码 0（PCMU）,3(GSM),4(G.723)以及 5（DVI4)。

```
C->S: INVITE sip:watson@boston.bell-tel.com SIP/2.0
        Via: SIP/2.0/UDP kton.bell-tel.com
        From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To: T. Watson <sip:watson@bell-tel.com>
        Call-ID: 662606876@kton.bell-tel.com
        CSeq: 1 INVITE
        Contact: <sip:a.g.bell@kton.bell-tel.com>
        Subject: Mr. Watson, come here.
        Content-Type: application/sdp
        Content-Length: ...

        v=0
        o=bell 53655765 2353687637 IN IP4 128.3.4.5
        s=Mr. Watson, come here.
        e=a.g.bell@bell-tel.com
        t=3149328600 0
        c=IN IP4 kton.bell-tel.com
        m=audio 3456 RTP/AVP 0 3 4 5
        a=rtpmap:0 PCMU/8000
        a=rtpmap:3 GSM/8000
        a=rtpmap:4 G723/8000
        a=rtpmap:5 DVI4/8000

S->C: SIP/2.0 100 Trying
```

```
        Via: SIP/2.0/UDP kton.bell-tel.com
        From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To: T. Watson <sip:watson@bell-tel.com> ;tag=37462311
        Call-ID: 662606876@kton.bell-tel.com
        CSeq: 1 INVITE
        Content-Length: 0

S->C: SIP/2.0 180 Ringing
        Via: SIP/2.0/UDP kton.bell-tel.com
        From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To: T. Watson <sip:watson@bell-tel.com> ;tag=37462311
        Call-ID: 662606876@kton.bell-tel.com
        CSeq: 1 INVITE
        Content-Length: 0

S->C: SIP/2.0 182 Queued, 2 callers ahead
        Via: SIP/2.0/UDP kton.bell-tel.com
        From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To: T. Watson <sip:watson@bell-tel.com> ;tag=37462311
        Call-ID: 662606876@kton.bell-tel.com
        CSeq: 1 INVITE
        Content-Length: 0

S->C: SIP/2.0 182 Queued, 1 caller ahead
        Via: SIP/2.0/UDP kton.bell-tel.com
        From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To: T. Watson <sip:watson@bell-tel.com> ;tag=37462311
        Call-ID: 662606876@kton.bell-tel.com
        CSeq: 1 INVITE
        Content-Length: 0

S->C: SIP/2.0 200 OK
        Via: SIP/2.0/UDP kton.bell-tel.com
        From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To: <sip:watson@bell-tel.com> ;tag=37462311
        Call-ID: 662606876@kton.bell-tel.com
        CSeq: 1 INVITE
        Contact: sip:watson@boston.bell-tel.com
        Content-Type: application/sdp
        Content-Length: ...

        v=0
        o=watson 4858949 4858949 IN IP4 192.1.2.3
        s=I'm on my way
```

```
e=watson@bell-tel.com
t=3149329600 0
c=IN IP4 boston.bell-tel.com
m=audio 5004 RTP/AVP 0 3
a=rtpmap:0 PCMU/8000
a=rtpmap:3 GSM/8000
```

该例子解释了状态响应码的使用。在这里，接收到的呼叫立刻通过 100 响应消息确认，然后，可能经过一 些数据库处理延时后，返回 call rings 消息，接着被放到呼叫队列中，呼叫的状态被周期性的更新。

Watson 只能使用 PCMU 和 GSM 编码。Watson 将发送语音数据到 c.bell-tel.com 的 3456 端口，Bell 将发送数据到 boston.bell-tel.com 的 5004 端口。

缺省情况下，媒体会话是一个 RTP 会话。Watson 将在端口 5005 接收到 RTCP 包，Bell 将在端口 3457 接收到该数据包。

由于双方对于支持的媒体类型取得了一致的意见，所以 Bell 确认呼叫时没有装入另外一个会话描述：

```
C->S: ACK sip:watson@boston.bell-tel.com SIP/2.0
      Via: SIP/2.0/UDP kton.bell-tel.com
      From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To: T. Watson <sip:watson@bell-tel.com> ;tag=37462311
      Call-ID: 662606876@kton.bell-tel.com
      CSeq: 1 ACK
```

15.4 中断呼叫

主叫可以发送一个如下所示格式的 BYE 请求来中断呼叫。

```
C->S: BYE sip:watson@boston.bell-tel.com SIP/2.0
      Via: SIP/2.0/UDP kton.bell-tel.com
      From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To: T. A. Watson <sip:watson@bell-tel.com> ;tag=37462311
      Call-ID: 3298420296@kton.bell-tel.com
      CSeq: 2 BYE
```

如果被叫希望中断呼叫，它同样发送一个 BYE 请求。只是，该 BYE 请求的 To 域和上面 BYE 请求的 From 域内容相同，而该请求的 From 域和上面请求的 To 域的内容相同。

15.5 Forking Proxy

在这个例子中，当前位于 c.bell-tel.com 主机的 Bell(a.g.bell@bell-tel.com)（C）想要呼叫 Watson(t.watson@ieee.org)。在呼叫的时候，Watson 在两台工作站 t.watson@x.bell-tel.com（X）和 watson@y.bell-tel.com(Y)上登录，并且通过叫做 sip.ieee.org 的 IEEE 代理服务器（P）注册。IEEE 服务器还有 Watson 在本地机器 watson@h.bell-tel.com（H）上的注册信息，以及在 watson@acm.org（A）上的永久注册信息。为了简短，例子忽略了包含会话描述的消息体。

Bell 的用户代理从 ieee.org 域发送邀请到 SIP 服务器：

```
C->P: INVITE sip:t.watson@ieee.org SIP/2.0
      Via:     SIP/2.0/UDP c.bell-tel.com
      From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To:      T. Watson <sip:t.watson@ieee.org>
      Call-ID: 31415@c.bell-tel.com
      CSeq:    1 INVITE
      Contact: a.g.bell@c.bell-tel.com
```

位于 ieee.org 的 SIP 服务器并行的尝试四个地址。它发送下面的信息到本地机器：

```
P->H: INVITE sip:watson@h.bell-tel.com SIP/2.0
      Via:     SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.1
      Via:     SIP/2.0/UDP c.bell-tel.com
      From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To:      T. Watson <sip:t.watson@ieee.org>
      Call-ID: 31415@c.bell-tel.com
      CSeq:    1 INVITE
      Contact: a.g.bell@c.bell-tel.com
```

由于 Watson 当前不在本地登录，所以针对该请求立刻产生一个 404 响应：

```
H->P: SIP/2.0 404 Not Found
      Via:     SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.1
      Via:     SIP/2.0/UDP c.bell-tel.com
      From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To:      T. Watson <sip:t.watson@ieee.org>;tag=87454273
      Call-ID: 31415@c.bell-tel.com
      CSeq:    1 INVITE
```

代理服务器发送 ACK 响应这样主机 H 就可以停止重传它：

```
P->H: ACK sip:watson@h.bell-tel.com SIP/2.0
      Via:     SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.1
```

```
From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
To:      T. Watson <sip:t.watson@ieee.org>;tag=87454273
Call-ID: 31415@c.bell-tel.com
CSeq:    1 ACK
```

同意，P 试图通过 ACM 服务器找到 Watson:

```
P->A: INVITE sip:watson@acm.org SIP/2.0
      Via:     SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.2
      Via:     SIP/2.0/UDP c.bell-tel.com
      From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To:      T. Watson <sip:t.watson@ieee.org>
      Call-ID: 31415@c.bell-tel.com
      CSeq:    1 INVITE
      Contact: a.g.bell@c.bell-tel.com
```

并行的，下一次尝试开始了，INVITE 请求发送往 X 和 Y:

```
P->X: INVITE sip:t.watson@x.bell-tel.com SIP/2.0
      Via:     SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.3
      Via:     SIP/2.0/UDP c.bell-tel.com
      From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To:      T. Watson <sip:t.watson@ieee.org>
      Call-ID: 31415@c.bell-tel.com
      CSeq:    1 INVITE
      Contact: a.g.bell@c.bell-tel.com

P->Y: INVITE sip:watson@y.bell-tel.com SIP/2.0
      Via:     SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.4
      Via:     SIP/2.0/UDP c.bell-tel.com
      From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To:      T. Watson <sip:t.watson@ieee.org>
      Call-ID: 31415@c.bell-tel.com
      CSeq:    1 INVITE
      Contact: a.g.bell@c.bell-tel.com
```

接下来，在 X 旁的 Watson 和在 Y 旁的同事听到电话振铃并摘机。X 和 Y 将通过代理服务器返回 200 响应给 Bell。

```
X->P: SIP/2.0 200 OK
      Via:      SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.3
      Via:      SIP/2.0/UDP c.bell-tel.com
```

```
        From:     A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:       T. Watson <sip:t.watson@ieee.org> ;tag=192137601
        Call-ID:  31415@c.bell-tel.com
        CSeq:     1 INVITE
        Contact:  sip:t.watson@x.bell-tel.com

   Y->P: SIP/2.0 200 OK
        Via:      SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.4
        Via:      SIP/2.0/UDP c.bell-tel.com
        Contact:  sip:t.watson@y.bell-tel.com
        From:     A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:       T. Watson <sip:t.watson@ieee.org> ;tag=35253448
        Call-ID:  31415@c.bell-tel.com
        CSeq:     1 INVITE
```

两个响应通过 Via 信息转发给 Bell。在这时，ACM 服务器仍然在搜寻它的数据库。P 现在可以取消它的尝试了。

```
   P->A: CANCEL sip:watson@acm.org SIP/2.0
        Via:      SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.2
        From:     A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:       T. Watson <sip:t.watson@ieee.org>
        Call-ID: 31415@c.bell-tel.com
        CSeq:     1 CANCEL
```

ACM 服务器停止它的神经网络数据库的查询并返回 200 响应。由于 P 是 via 头中的第一个元素，所以 200 响应不会更进一步的传下去。

```
   A->P: SIP/2.0 200 OK
        Via:      SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.2
        From:     A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:       T. Watson <sip:t.watson@ieee.org>
        Call-ID: 31415@c.bell-tel.com
        CSeq:     1 CANCEL
```

另外，487 响应将发送给初始的 INVITE 请求者：

```
   A->P: SIP/2.0 487 Request Terminated
        Via:      SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.2
        Via:      SIP/2.0/UDP c.bell-tel.com
        From:     A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:       T. Watson <sip:t.watson@ieee.org>;tag=0303
```

```
        Call-ID: 31415@c.bell-tel.com
        CSeq:    1 INVITE
```

该响应在 P 结束。P 为 487 响应产生一个 ACK 确认：

```
P->A: ACK sip:watson@acm.org SIP/2.0
        Via:     SIP/2.0/UDP sip.ieee.org ;branch=3d8a50dbf5a28d.2
        From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:      T. Watson <sip:t.watson@ieee.org>;tag=0303
        Call-ID: 31415@c.bell-tel.com
        CSeq:    1 ACK
```

Bell 很快的从 X 和 Y 收到两个 200 响应并且给两个 ACK 确认。Bell 现在可以保存所有的 call leg 或者使用 BYE 请求中断一个。这里，它临时保存两个以确定 Watson 在什么地址。

```
C->X: ACK sip:t.watson@x.bell-tel.com SIP/2.0
        Via:     SIP/2.0/UDP c.bell-tel.com
        From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:      T. Watson <sip:t.watson@ieee.org>;tag=192137601
        Call-ID: 31415@c.bell-tel.com
        CSeq:    1 ACK

C->Y: ACK sip:watson@y.bell-tel.com SIP/2.0
        Via:     SIP/2.0/UDP c.bell-tel.com
        From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:      T. Watson <sip:t.watson@ieee.org>;tag=35253448
        Call-ID: 31415@c.bell-tel.com
        CSeq:    1 ACK
```

Bell 和 X、Y 经过简短的交谈后，非常清楚的确定了 Watson 是在 X。（注意这里不是三方通话；只有 Bell 可以和 X、Y 通话，X 和 Y 之间不能相互通话。）这样 Bell 发送 BYE 给 Y,Y 给出响应：

```
C->Y: BYE sip:watson@y.bell-tel.com SIP/2.0
        Via:     SIP/2.0/UDP c.bell-tel.com
        From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:      T. Watson <sip:t.watson@ieee.org>;tag=35253448
        Call-ID: 31415@c.bell-tel.com
        CSeq:    2 BYE

Y->C: SIP/2.0 200 OK
        Via:     SIP/2.0/UDP c.bell-tel.com
        From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
        To:      T. Watson <sip:t.watson@ieee.org>;tag=35253448
        Call-ID: 31415@c.bell-tel.com
```

```
        CSeq:     2 BYE
```

20.6 重定向

301 响应或者 302 响应在 Contact 头域中指出了另外一个地址。接着上面的例子，在 ieee.org 的服务器 P 决定重定向而不是转发该请求：

```
P->C: SIP/2.0 302 Moved temporarily
      Via:     SIP/2.0/UDP c.bell-tel.com
      From:    A. Bell <sip:a.g.bell@bell-tel.com>;tag=3pcc
      To:      T. Watson <sip:t.watson@ieee.org>;tag=72538263
      Call-ID: 31415@c.bell-tel.com
      CSeq:    1 INVITE
      Contact: sip:watson@h.bell-tel.com,
               sip:watson@acm.org, sip:t.watson@x.bell-tel.com,
               sip:watson@y.bell-tel.com
```

举例来说，假定 Alice(A)希望在她从 1998/7/29 日开始的度假期间由 Bob(B)代理她的电话。任何一个呼叫她的电话将会到达 Bob,请求的 To 头域指出了 Bob 代理的角色。Charlie(C)呼叫 Alice(A),Alice 的服务器返回响应：

```
A->C: SIP/2.0 302 Moved temporarily
      From: Charlie <sip:charlie@caller.com>;tag=5h7j
      To: Alice <sip:alice@wonderland.com> ;tag=2332462
      Call-ID: 27182@caller.com
      Contact: sip:bob@example.com
      Expires: Wed, 29 Jul 1998 9:00:00 GMT
      CSeq: 1 INVITE
```

Charlie 然后发送下面的请求到 example.com 域的 SIP 服务器。注意到位于 example.com 的服务器根据 Request-URI 转发请求给 Bob。

```
C->B: INVITE sip:bob@example.com SIP/2.0
      Via: SIP/2.0/UDP h.caller.com
      From: <sip:charlie@caller.com>;tag=5h7j
      To: sip:alice@wonderland.com
      Call-ID: 27182@caller.com
      CSeq: 2 INVITE
      Contact: sip:charlie@h.caller.com
```

在第三个关于重定向的例子中，我们假定所有的外出请求都直接通过位于 caller.com 的本地防火墙 F(“outbound proxy”),仍然是 Charlie 呼叫 Alice:

```
C->F: INVITE sip:alice@wonderland.com SIP/2.0
```

```
        Via: SIP/2.0/UDP h.caller.com
        From: <sip:charlie@caller.com>;tag=5h7j
        To: Alice <sip:alice@wonderland.com>
        Call-ID: 27182@caller.com
        CSeq: 1 INVITE
        Contact: sip:charlie@h.caller.com
```

位于 caller.com 的本地防火墙正好超载，所以将从 Charlie 来的呼叫重定向到第二个服务器 S:

```
 F->C: SIP/2.0 302 Moved temporarily
        Via: SIP/2.0/UDP h.caller.com
        From: <sip:charlie@caller.com>;tag=5h7j
        To: Alice <sip:alice@wonderland.com>
        Call-ID: 27182@caller.com
        CSeq: 1 INVITE
        Contact: <sip:alice@wonderland.com:5080;maddr=spare.caller.com>
```

基于上面的响应，Charlie 通过端口 5080 将同样的请求发送到第二个服务器 spare.caller.com,Request-URI 不做改动：

```
 C->S: INVITE sip:alice@wonderland.com SIP/2.0
        Via: SIP/2.0/UDP h.caller.com
        From: <sip:charlie@caller.com>;tag=5h7j
        To: Alice <sip:alice@wonderland.com>
        Call-ID: 27182@caller.com
        CSeq: 2 INVITE
        Contact: sip:charlie@h.caller.com
```

15.7 协商

一个 606 响应的例子如下所示：

```
 S->C: SIP/2.0 606 Not Acceptable
        Via: SIP/2.0/UDP c.bell-tel.com
        From: A. Bell <sip:a.g.bell@bell-tel.com>;tag=3ab6
        To: T. Watson <sip:t.watson@ieee.org> ;tag=7434264
        Call-ID: 14142@c.bell-tel.com
        CSeq: 1 INVITE
        Warning: 370 "Insufficient bandwidth (only have ISDN)",
          305 "Incompatible media format",
          330 "Multicast not available"
        Content-Type: application/sdp
```

```
Content-Length: ...

v=0
o=c 3149329138 3149329165 IN IP4 38.245.76.2
s=Let's talk
e=t.watson@ieee.org
t=3149328630 0
b=CT:128
c=IN IP4 x.bell-tel.com
m=audio 3456 RTP/AVP 5 0 7
a=rtpmap:5 DVI4/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:7 LPC/8000
m=video 2232 RTP/AVP 31
a=rtpmap:31 H261/90000
```

在这个例子中，初始的请求指定的带宽比接入链路可以支持的带宽高，要求广播，指定媒体编码集。该响 应指出只支持 128kb/s 的带宽，只支持 DVI,PCM 或者 LPC 语音编码并且在参数中指定了编码的优先选择顺序。

该响应也指出了不支持多播。在这种情况下，比较合适的做法是通过一个转换网关并且重新呼叫。

15.8 OPTIONS 请求

主叫 Alice 可以使用 OPTIONS 请求去查找潜在被叫 Bob 的能力，而不用"ringing"目的地址。Bob 返回一个描述指示他能支持接收语音编码 PCM mu-law(RTP 有效载荷类型 0)，1016（有效载荷类型 1），GSM(有效载荷类型 3)，SX7300/8000(动态有效载荷类型 99)，以及视频编码 H.261(有效载荷类型 31)，H.263(有效载荷类型 34)。

```
C->S: OPTIONS sip:bob@example.com SIP/2.0
      Via: SIP/2.0/UDP cat.wonderland.com
      From: Alice <sip:alice@wonderland.com>;tag=1gloo
      To: Bob <sip:bob@example.com>
      Call-ID: 6378@cat.wonderland.com
      CSeq: 1 OPTIONS
      Accept: application/sdp

S->C: SIP/2.0 200 OK
      From: Alice <sip:alice@wonderland.com>;tag=1gloo
      To: Bob <sip:bob@example.com> ;tag=376364382
      Call-ID: 6378@cat.wonderland.com
      Content-Length: 81
      Content-Type: application/sdp
```

```
v=0
o=alice 3149329138 3149329165 IN IP4 24.124.37.3
s=Security problems
e=bob@example.com
t=3149328650 0
c=IN IP4 24.124.37.3
m=audio 0 RTP/AVP 0 1 3 99
a=rtpmap:0 PCMU/8000
a=rtpmap:1 1016/8000
a=rtpmap:3 GSM/8000
a=rtpmap:99 SX7300/8000
m=video 0 RTP/AVP 31 34
a=rtpmap:31 H261/90000
a=rtpmap:34 H263/90000
```

附录：

A 最小实现

A.1 传输层协议支持

所有的实体必须支持 UDP。用户代理应该实现 TCP 传输。有状态的代理服务器，注册服务器，以及重定向服务器必须实现 TCP 传输。其它的传输层协议可以在任一实体上实现。

A.2 客户端

所有的客户端必须能够产生 INVITE 和 ACK 请求。客户端必须产生和解析 Call-ID,Content-Length,Content-Type,Cseq,From,Record-Route,Route 以及 To 头域。客户端必须也能析 Require 头域。一个最小实现必须理解 SDP(RFC2327,[6])。它必须能识别从 1 到 6 类型的状态码并能做对应的处理。Uas 必须能使用 outbound proxies。

下面的最小能力集基于前面描述的最小实现来构建。通常，下面列出的每一个能力集基于它上面的能力集来构建：

Basic：basic 实现加上了对 BYE 方法的支持以允许对未完成的呼叫进行中断。它在请求里包括了一个 User-Agent 头并且通过 Accept-Language 头指定了首选的语言。

Redirection：为了支持呼叫转发，一个客户端必须能够理解 Contact 头，但只是 SIP-URL 部分，不包括参数。

Negotiation：客户端必须能使用 OPTIONS 请求，能理解 380 状态以及 Contact 参数以参加终端和媒体的协商。它应该能够解析 Warning 响应头以给主叫提供有用的反馈。

Authentication：如果客户端想要呼叫需要主叫鉴权的被叫，那么它必须能识别 401 状态码，必须能产生 Authorization 请求头，必须能理解 WWW-Authenticate 响应头。

如果客户端想要使用需要主叫鉴权的代理服务器，它必须能识别 407 状态码，必须能产生 Proxy-Authorization 请求头，必须能理解 Proxy-Authenticate 响应头。

A.3 服务器

一个最小实现服务器必须理解 INVITE,ACK,OPTIONS 以及 BYE 请求。一个代理服务器还必须理解 CANCEL。它必须能解析产生 Call-ID，Content-Length，Content-Type，CSeq，Expires，From，Max-Forwards， Require，To 以及 Via 头域。它必须在响应中重复 Cseq 和 Timestamp 头。它应该在响应中包括 Server 头。

A.4 头处理

对于头域的数目和头域的长度，实现者不应该有固定的限制，但长度也不能太大。标示事务的头域应该出现在消息的前面，这样当实现者不能处理全部的消息时至少能返回一个响应状态，比如说 513（消息太大 ）。

表 6 中列出了不同实现者支持的头域。UAC 指的是用户代理客户端（主叫用户代理），UAS 指的是用户代理服务器（被叫用户代理）。

表中域的意义如下所示。Type 和表 4、表 5 一样。“-”表示该域在这个系统里没有意义（虽然它可能由它产生）。“m”表示该域必须被解释。“b”表示该域应该被 basic 实现所解释。“r”表示如果系统声称支持重定向该域应该被解释。“a”表示如果系统声称支持鉴权该域应该被解释。“e”表示如果系统声称支持加密该域应该被解释。“o”表示对该域的支持是可选的。对所有的实现者是可选支持的域没有显示出来。

B 使用会话描述协议（SDP）

这一节描述了会话描述协议的使用（SDP）(RFC2327 [6])。SDP 表示为 Content-Type “application/sdp”。每一个 SIP 消息必须包含 0 或 1 个 SDP 消息。虽然 SDP 规范允许多个会话描述组合进一个大的 SDP 消息，但是用在 SIP 里的 SDP 必须只包含一个会话描述。

代理服务器通常不改变会话描述，但是如果需要再没有对会话描述采取加密完整性保护机制的情况下也可以这样做，比如说，进行网络地址转换。

B.1 通用方法

在 SIP 里的 SDP 遵循“发起-响应”模式。发起者提出关于会话的观点，另一方响应以对应的意见。“发起-响应”模式可以在两个方向上发生。会话发起可以放在 INVITE 请求中，在这种情况下会话响应必须放在 200 类型的响应中，并且 ACK 不能包含 SDP。或者，INVITE 可以不包含 SDP,在这种情况下，会话发起放在 200 类型响应中，并且 ACK 必须包含响应。

type UAC proxy UAS registrar

Accept	R	-	o	m	m
Accept-Encoding	R	-	-	m	m
Accept-Language	R	-	b	b	b
Allow	405	o	-	-	-
Authorization	R	a	o	a	a
Call-ID	g	m	m	m	m
Contact	R	-	-	m	m
Contact	r	m	r	-	-
Content-Encoding	g	m	-	m	m
Content-Length	g	m	m	m	m
Content-Type	g	m	-	m	m
CSeq	g	m	m	m	m
Encryption	g	e	-	e	e
Expires	g	-	o	o	m
From	g	m	o	m	m
Max-Forwards	R	-	b	-	-
Proxy-Authenticate	407	a	-	-	-
Proxy-Authorization	R	-	a	-	-
Proxy-Require	R	-	m	-	-
Record-Route	R	m	-	m	m
Require	R	m	-	m	m
Response-Key	R	-	-	e	e
Route	R	m	m	m	-
Timestamp	g	o	o	m	m
To	g	m	m	m	m
Unsupported	r	b	b	-	-
User-Agent	g	b	-	b	-
Via	g	m	m	m	m
WWW-Authenticate	401	a	-	-	-

Table 6: Header Field Processing Requirements

不能被接收的发起会话会被拒绝。如果由 INVITE 发起会话，拒绝消息是 488 或者 606 响应。如果由 200 类型响应发起会话，UAC 必须发送 ACK 请求，并给出一个有效的会话响应。它可以发送 BYE 请求中断呼叫，或者可以修改成能够接收的会话参数然后重新通过 INVITE 发起会话。

SDP 的处理不依赖于会话通过 INVITE 或者 200 类型响应发起。

B.2 产生初始会话

发起的会话（以及会话响应）必须是有效的 SDP，必须遵循 RFC2327 [6]。这表示它必须包含一个 v 行，o 行，s 行以及 t 行。必须包含 e 行或者 p 行任一个。然而，我们推荐所有的实现者接受不

包含 e,p，或者 s 行的 SDP。在 o 行中出现的 session id 和 version 必须是 64 位无符号的整数。

在一个多方会话中，SDP 中的“s”行和 SIP Subject 头域有不同的意义。该会话描述行描述了多方会话的主题，而 SIP Subject 头域描述了邀请的原因。20.2 节的例子阐述了这一观点。对于两方会话的发起者，SDP 中的“s=”行可以包含单个的空格字符（0x20）。

不幸的是，SDP 不允许“s=”行空着。

B.2.1 单播

发起的会话必须包含 0 或者多个媒体单元。0 个媒体会话表示会话发起者希望通信，只是会话媒体流在稍后通过 re-INVITEs 加上。

如果发起者的会话描述只包含一个标示 send(receive) only 的媒体流，它表示发起者只愿意发送（接收）该媒体流，而不愿意接收（发送）。媒体流分别通过 SDP 中的”a=sendonly”和”a=recvonly”属性来标示 媒体流只发送或者只接收。如果两个属性都没有指定，缺省是可以发送接收。（可以显式的用”a=sendrecv” 指出）。

对于只接收和只发送流，会话描述里的端口号和地址指示媒体流应该发送到哪里去。对于只发送 RTP 流，端口号和地址指示了 RTCP 报告应发送到哪里去。需要特别指出的是，RTCP 报告发送到比指定的端口数字多一的端口上。会话描述中出现的端口号和地址和会话发起者将要发送 RTP 和 RTCP 包的源端口号和源地址无关。发起会话中的端口号为 0 表示已发送媒体流但是不应该被使用。在初始的 INVITE 中这是毫无用处的语法规则，但却因为完整性的原因而存在，因为响应可以包含一个 0 端口以表示拒绝该媒体流（B.3 节）。此外，已经存在的媒体流可以通过将端口号设置为 0 而中断（B.4 节）。通常，端口号为 0 表示不需要该媒体流。

每一个媒体流的有效载荷类型列表包括了两部分信息，一部分是发送者支持的发送和接收（依赖于方向属性）的 codecs，另一部分是用来表示那些 codecs 的 RTP 有效载荷类型数字。如果列出了多个 codecs，它表示发起者在呼叫的过程中可以使用那些 codecs 中的任何一个。换句话说，响应者可以在呼叫的过程中使用那些列表改变 codecs 而不需要发送 SIP 消息。对于只发送的流，发起会话应该指出发起者发送该流时将会使用的 codecs 系列。对于只接收的流，发起会话应该指出发起者接收该流时将会使用的 codecs 系列。对于一个发送接收流，会话发起应该指出发起者发送接收流时将会使用的 codecs。

对于只接收流，有效载荷类型数字表示对于该 codec 会话发起者希望接收的 RTP 包中的有效载荷域的值。

对于只发送流，有效载荷类型数字表示对于该 codec 会话发起者希望发送的 RTP 包中的有效载荷域的值。

对于只接收发送流，有效载荷类型数字表示对于该 codec 会话发起者希望接收发送的 RTP 包中的有效载荷域的值。这表示对于一个 codec 有效载荷类型在两个方向上都是相同的。 所有的媒体描述应该包含“a=rtpmap”以将 RTP 有效载荷类型映射到编码。如果没有包含”a=rtpmap”， 将使用 RFC1890 [27]中的静态有效载荷表。

这样就更容易从静态有效载荷映射了。

m 行中的 codecs 总是按照优先顺序排列，列表中的第一个 codec 是首选。在这里，首选表示接收者应该使用优先级最高的 codec。

如果出现了不同类型的多个媒体流，它表示会话发起者希望同时使用这些流。一种典型的情况是视频会议中使用音频和视频流。

如果出现了相同类型的多个媒体流，它表示会话发起者希望同时发送（或接收）多个媒体流。当发送多个相同类型的媒体流时，流的发送者（像 microphone 或者网关）发送多次，每次发送一个流。每一个流可以使用不同的编码。当接收到多个相同类型的流时，必须在播放它们之前将它们混合。一个典型的例子是预付费卡应用，用户可以在呼叫的任一时刻按键以挂断使用同一张卡进行新的呼叫。这需要流从用户到远方网关，到需要按键信息的系统。

有一些 codecs 只能对一些特殊的媒体内容编码，比如说对 DTMF 音频信号、数字以及舒适噪声的 RTP 有效载荷格式编码。当这些 codecs 的一个出现在只发送或发送接收流里面时，它表示当该媒体流内容能够使用该 codec 编码时发送者将只采用该 codec 发送。当媒体流内容不能使用该 codec 编码时，可以使用 m 行中指出的另一个 codec。如果 m 行中没有其它的 codec,将什么也不发送。当 UA 遵从 RFC2833 发送 DTMF 到远方服务器时，这一点很有用。这表示单个媒体行只包含 DTMF codec。

B.2.2 多播

构建一个多播发起会话的会话描述的过程除了几点差异基本上和上面的过程相同。

c 行中列出的地址必须是多播地址。它指出了会话发起者希望从什么地址接收包。

对于只发送和只接收，多播媒体会话和单播会话并不相同。对于多播，只发送表示会话描述的接收者（主叫或被叫）只能发送媒体流到指定的地址和端口。只接收表示会话描述的接收者（主叫或被叫）只能从指定的地址和端口接收媒体流。

B.3 产生会话响应

我们基于发送的 SDP 之上对它做出响应。如果会话响应在某些方面不同（比如说不同的 IP 地址和端口），由于会话响应是由不同的实体产生，会话响应中的 origin 行必须不同。在这种情况下，会话响应中的 o 行中的版本号与会话发起中的 o 行的版本号无关。

对于会话发起中的 m 行，会话响应中必须有一个对应的 m 行。会话响应中包含的 m 行数必须和会话发起中相同。这样就允许流基于它们的顺序对应。这表示如果会话响应包含 0 个 m 行，那么会话响应必须包含 0 个 m 行。

发起的流可能因为任一原因在会话响应中被拒绝。拒绝的意思是会话发起者和响应者都不能通过该流发送媒体（或者 RTCP 包）。会话响应通过将响应中对应流的端口号设置为 0。媒体格式列表将被忽视。按照 SDP 中规定，必须至少出现一个媒体格式类型。

B.3.1 单播

如果流在发送时标示为只发送，那么会应中对应的流必须被标示为只接收。而且，会话响应对应的流必须包含至少一个在会话发送中列出的并且自己将会使用来接收数据的 codec。该流中可以包含另外没有在发起会话中列出的 codecs 但自己可以使用来接收处理的 codecs。连接地址和端口号指出了会话响应者从何处接收 RTP（接收 RTCP 的端口号比 RTP 高一个）。

如果会话发起中的媒体流标示为只接收，会话响应中的媒体流必须被标示为只发送。然而，会话响应中的对应流必须包含至少一个在发起会话中列出的且自己将会用来发送的 codec。IP 地址和端口号指出了会话响应者希望从什么地址接收 RTCP（接收 RTCP 的端口号比 SDP 中列出的数字大一个）。如果发起会话中的媒体流标示为接收发送（或者没有包含方向属性，在者种情况下缺省表示接受发送），会话响应中的对应流可以被表示为只发送，只接收，或者发送接收。缺省时发送接收。如果会话响应中的流标示为只发送，它必须包含在发起会话中列出的且自己将用来发送的 codec。

IP 地址和端口号指出了会话响应者从何处接收 RTCP。如果会话响应中的流标示为只接收，它必须包含在发起会话中列出的且自己将用来接收的 codec。该流中可以包含另外没有在发起会话中列出的 codecs 但自己可以使用来接收处理的 codecs。连接地址和端口号指出了会话响应者从何处接收 RTP、RTCP（接收 RTCP 的端口号比 SDP 中列出的数字大一个）。如果会话响应中的流标示为接收发送，它必须包含在发起会话中列出的且自己将用来发送接收的 codec。该流中可以包含另外没有在发起会话中列出的 codecs 但自己可以使用来接收处理的 codecs。连接地址和端口号指出了会话响应者从何处接收 RTP（接收 RTCP 的端口号比 SDP 中列出的数字大一个）。

在流中，对于一个特定的 codec，有效载荷类型数目在会话发起和会话响应中必须相同。换句话说，对于同一个 codec，不能在两个方向使用不同的动态有效载荷类型数目。

在 m 行中列出的 codecs 总是按照优先顺序排列，列在第一位的是首选。在者种情况下，首选表示会话响应的接收者应该使用它能接收的最高优先级。

虽然会话响应者可以按照优先级列出 codecs,我们仍然推荐除非是因为特殊的原因，会话响应中列出 codecs 的的顺序应和会话发起中的相同。换句话说，如果会话发起流中按照顺序 8，22，48 列出 codecs，并且会话响应者只支持 codecs 8 和 48，我们推荐如果会话响应者没有其它的原因去改变它，那么会话响应中列出 codecs 的顺序应为 8，48，而不是 48，8。这保证了在会话的两个方向使用相同的 codec。

如果对于特定会话发送的流，会话响应者没有双方共同支持的媒体格式，会话响应者必须拒绝该媒体流。

如果对于所有的媒体流，双方没有共同支持的媒体格式，整个发起的会话必须被拒绝。

B.3.2 多播

对于多播，接收和发送的多播地址相同并且所有的呼叫方使用相同的端口号来接收媒体数据。如果会话响应者可以接收会话发起者的会话描述，会话响应者可以选择不在响应中包含会话描述或者将会话描述 copy 到响应中。

会话响应者可以返回一个将一些有效载荷类型删除掉或者将端口号设为 0（而不是其它的值）的会话描述。这表示会话响应者不支持被删除掉的那些媒体类西或者指定的数据流。会话响应者不能改变

给定的流是否只发送，只接收，或者是发送接收。

如果会话响应者不支持多播，它应该拒绝该会话描述。

B.4 修改会话

在呼叫中的任何一点，参与的双方都可以发送 re-INVITE 以改变会话的特征。re-INVITE 的会话发起-响应过程和上面定义的完全相同。这表示 re-INVITE 可以不包含 SDP，而让对 re-INVITE 进行响应的 200 OK 来包含会话发起。在这个例子中，如果会话发起者没有什么原因去改变什么，那么它发起的 SDP 应和它之前发起的 SDP 相同。

re-INVITE 中的会话发起可以和刚才发送给另一方的 SDP（可能在会话发起或者会话发送中发出）相同，也可以不同。在这里 last SDP 和 previous SDP 相同。如果会话发起相同，会话响应可以和刚才从会话响应者接收的 SDP 相同，也可以不同。如果发起的 SDP 和刚才的 SDP 不同，在 SDP 的构建上会有一些限制，下面会讨论这一点。

几乎会话的所有方面都可以被修改。可以添加新的数据流，可以删除已存在的数据流，并且已经存在的流的参数可以被修改。当发起一个请求来修改会话时，新 SDP 的 o 行必须和上次 SDP 的相同，只是 origin 域中的版本号必须在上一个 SDP 的基础上增加。如果 origin 行的版本号不增加，那么 SDP 必须和具有相同版本号的 SDP 相同。会话响应者必须准备接收到一个包含了版本号没有改变的 SDP 的会话发起；对该请求不做处理。然而，会话响应者必须遵循 B.3 节中所定规程来产生一个有效的会话响应。（可以和会话响应者上一个的 SDP 相同，也可以不同）

如果一个发起的 SDP 和上一个 SDP 不同，新的 SDP 必须有相匹配的媒体节对应上一个 SDP 的每一个媒体节。换句话说，如果上一个 SDP 有 N 个媒体行，新的 SDP 必须有至少 N 个媒体行。从顶上数下来，上一个 SDP 的第 I 个媒体流应该和新的 SDP 的对应媒体流匹配。这样的匹配时有必要的，它可以让会话响应者确定新 SDP 中的哪一个流和上一个 SDP 中的流相对应。由于这一要求，在流中的 m 行的数目永远不会减少，只会增加。上一个 SDP 中被删除掉的媒体流在新的 SDP 中不能被删除掉。

B.4.1 添加一个媒体流

通过添加在现有的媒体描述下添加另外一个来创建新的媒体流。新的媒体节必须在已有的媒体节的下面出现。这些媒体节的格式规范和那些在 B.2 节中描述的相同。

当会话响应者接收到一个比上一个 SDP 有更多媒体描述的 SDP 时，会话响应者知道新的媒体流被添加上了 。可以拒绝它或者在会话响应中加入相应的媒体描述来响应。在会话响应中构建一个新的媒体描述的规则。

在 B.3 节中描述。

B.4.2 删除一个媒体流

可以通过创造一个将对应该流的端口设置为 0 的 SDP 来将一个已存在的媒体流删除掉。另外，媒体描述应该被格式以和上一个 SDP 的相同。

一个端口设置为 0 的发起的流必须在会话响应中将端口设置为 0。另外，媒体描述应该被格式以和上一个 SDP 的相同。

B.4.3 改变一个媒体流

媒体流的几乎所有特征都可以被修改。

流的端口号可以被修改。会话发起者构建一个新的媒体描述，并将 m 行中的端口号设置成和上一个 SDP 对应的不相同。如果只是改变端口号，媒体流描述的其它部分应该保留不被修改。在会话发起发送出去后，会话发起者必须准备同时在新旧端口接收媒体。会话发起者一直到在新的端口上接收到媒体才能停止在旧的端口上监听媒体。在那时，就可以停止在旧的端口上监听媒体。

会话响应中相应的媒体流可以和会话响应者上一个 SDP 的流相同，也可以不同。如果更新的流被会话响应者接收，对于该数据流会话响应者应该立即在新的端口上发送数据。在会话响应发送出去后，会话响应者必须准备同时在新旧端口接收媒体。会话响应者一直到在新的端口上接收到媒体才能停止在旧的端口上监听媒体。在那时，就可以停止在旧的端口上监听媒体。

我们遵循相同的规程来改变媒体发往的 IP 地址和端口号。它们仅有的差别是一个时更新连接行，一个是更新端口号。

会话中使用的 codecs 列表可以被修改。会话发起者构建一个新的媒体描述，并且 m 行中的媒体格式列表和上一个 SDP 中对应的不同。该列表可以添加新的 codecs,也可以将已有的删除。当一个新的 codec 使用一个动态有效载荷类型数目，它不能在会话中使用以前使用的动态有效载荷类型数目。

会话响应中相应的媒体流按照 B.3 节所述描述。如果一个流的新 codecs 列表改变选择使用的 codec,新的 codec 应该立刻被使用。这表示会话发起者在它发送会话发起后必须立即准备使用新的 codec 来接收媒体，并且会话接收者在它发送会话响应后必须立即准备使用新的 codec 来接收媒体。

可以改变流的媒体类型（声频，视频等等）。这在在一个单独的流中改变语音和传真媒体类型时特别有用处。会话发起者构建一个新的媒体描述，将需被改变的上一个 SDP 中的对应描述使用新的媒体类型替代。

流的 IP 地址和端口可以被改变，也可以不改变。然而，对于该流，新的 codecs 的有效载荷类型列表数目必须和以前使用的不同。

会话响应中相应的媒体流按照 B.3 节所述描述。假定该流能被接受，会话响应应该在它接收到会话发起后立即使用新的媒体类型和 codecs 发送。

可以改变流的传输方式。改变传输方式和改变端口的方法一样，只是一个是改变传输方法，一个是改变端口。

在媒体描述中的其它一些属性可以在会话发起被更新。

B.4.4 将一个媒体流保持

如果呼叫中的一方想要将另外一个保持，也就是说请求临时停止发送一个或多个媒体流，一方给

另一方发送更新 SDP。该 SDP 将需要被保持的流的连接地址设置为 0（0.0.0.0）。这个处理方法和其它的改变地址的方法没有区别，只是地址 0.0.0.0 被解释成"dev/null"，而且停止发送媒体。需要特别指出的是，在通信的两个方向数据流分开来保持。每一个流都独立的保持。对于一个被保持的流来说，会话发起的接收者不应该自动返回一个会话响应并将相应的数据流保持。一个将所有的数据流保持的 SDP 叫做保持 SDP。

当一个 UA 使用保持 SDP 来响应是，一些三方呼叫控制会不能工作。

典型的，当一个用户按下保持键，UA 对所有的流发送 re-INVITE 请求并且将请求中 SDP 的响应地址设置为 0.0.0.0，本地将静音，这样就没有媒体发送到远端。

B.5 例子

举例来说，假定主叫在 INVITE 请求中包含下面的描述。它包括一个双向的音频流和两个双向的使用 H.261(有效载荷类型 31)和 MPEG（有效载荷类型 32）的视频流。发起的 SDP 是：

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=New board design
e=alice@foo.org
t=0 0
c=IN IP4 host.anywhere.com
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31
a=rtpmap:31 H261/90000
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

被叫 Bob 不愿意接收或者发送第一个视频流，所以它返回如下的媒体描述作为会话响应：

```
v=0
o=bob 2890844730 2890844730 IN IP4 host.example.com
s=New board design
e=bob@bar.com
t=0 0
c=IN IP4 host.example.com
m=audio 47920 RTP/AVP 0 1
a=rtpmap:0 PCMU/8000
m=video 0 RTP/AVP 31
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

过一段时间后，Bob 决定改变它接收声频流（从 47920 到 6400）的端口，并且同时添加另外一个音

频流， 该流具有只接收属性，使用事件 RTP 有效载荷类型。在 INVITE 中 Bob 发送如下 SDP：

```
v=0
o=bob 2890844730 2890844731 IN IP4 host.example.com
s=New board design
e=bob@bar.com
t=0 0
c=IN IP4 host.example.com
m=audio 6400 RTP/AVP 0 1
a=rtpmap:0 PCMU/8000
m=video 0 RTP/AVP 31
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
m=audio 8864 RTP/AVP 110
a=rtpmap:110 telephone-events
a=recvonly
```

Alice 接收另外的媒体流，所以产生如下的会话响应：

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.anywhere.com
s=New board design
e=alice@foo.org
t=0 0
c=IN IP4 host.anywhere.com
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31
a=rtpmap:31 H261/90000
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
m=audio 4520 RTP/AVP 110
a=rtpmap:110 telephone-events
a=sendonly
```

C 扩充的 BNF 的摘要

在两篇文章中描述了本文中指出的所有机制并且扩充的 BNF 和 RFC822 [26]、RFC2234 [44]中使用的相似。为了理解本文，实现者应该熟悉这些规则。扩充的 BNF 包括如下结构：

```
name = definition
```

规则的名字仅仅只是它自己的名字（不包含"<"和">"）并且使用"="和它的定义分开。在续行中使用空格缩进以指示规则定义扩展到多行。一些基本的规则使用大写表示，比如

SP,LWS,HT,CRLF,DIGIT,ALPHA 等等。三角括号在定义中使用，使用它们可以清楚的辨识名字和定义。

"literal"

引号用来引用数字符号。除非另外指定，文字与大小写无关的。

rule1 | rule2

被"|"分开的元素可以被选择，比如说，"yes | no"可以选择 yes 或者 no。

(rule1 rule2)

被园括号包含的多个元素被当作一个单独的元素。所以"(elem (foo | bar) elem)"表示成"elem foo elem"和 "elem bar elem"。

*rule

在元素前出现的"*"表示重复。完全的格式"<n>*<m>element"表示至少<n>并且至多<m>次重复元素。缺省的值是 0 到无限，这样"*(elelment)"允许任何数字，包括 0；"1*element"要求大于 1；"1*2element" 允许 1 或者 2。

[rule]

方括号包含可选的元素；"[foo bar]"和"*1(foo bar)"相同。

N 规则

特殊的重复："<n>(element)"和"<n>*<n>(element)"相同；这表示重复(element)元素<n>次。所以，2DIGIT 表示两个数字的数，3ALPHA 表示 3 个字符的字符串。

#rule

和"*"相似，"#"用来定义元素系列。完全的格式是"<n>#<m> element"表示至少<n>并且至多<m>元素。

每一个被一个或多个",",分开，LWS 是可选择的。这让列表通常格式非常简单；一个如下的规则可以表示为 1# element。

(*LWS element * (*LWS "," *LWS element))

使用这种规则时，可以允许空元素，但是不计算在元素数目中。这表示，"(element),,(element)"可以被允许，但是只算作两个元素。因此，至少需要一个元素，至少有一个非空的元素。缺省值从 0 到无限， 这样"#element"允许任何数字，包括 0;"1#element"包括至少一个；"1#2element"允许 1 或者 2。

； comment

分号放置在规则文字的左边，它表示从分号开始到行末的文字时注释。这给在规范书中同时进行注释提供了一个简单的方法。

implied *LWS

本规范书中描述的语法基于词。除非特别说明，LWS 可以放在两个邻近的词（标记或者引用字符串），可以放在两个邻近的字符和分隔符之间而不会改变该域的意义。在任何两个标记之间必须至少有一个分隔符（LWS 或者分隔符），如果不这样，它们将被当作一个单独的标记。注意，URLs 不包含 LWS。

C.1 基础规则

在本规范书中依从下面的规则来描述基本解析结构。US-ASCII 编码字符集在 ANSI X3.4-1986 中定义。

C.1 Basic Rules

下面的规则用在本 RFC 中来描述基本的解析架构。US-ASCII 编码字符集由 ANSI X3.4-1986 定义。

```
OCTET      =  %x00-ff ; any 8-bit sequence of data
CHAR       =  %x00-7f ; any US-ASCII character (octets 0 - 127)
upalpha    =  "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
              "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
              "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
lowalpha   =  "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
              "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
              "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
alpha      =  lowalpha | upalpha
DIGIT      =  "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
              "8" | "9"
alphanum   =  alpha | DIGIT
CTL        =  %x00-1f | %x7f ; (octets 0 -- 31) and DEL (127)
CR         =  %d13 ; US-ASCII CR, carriage return character
LF         =  %d10 ; US-ASCII LF, line feed character
SP         =  %d32 ; US-ASCII SP, space character
HT         =  %d09 ; US-ASCII HT, horizontal tab character
CRLF       =  CR LF ; typically the end of a line
```

对于 SIP URI 下面的内容在 RFC2396 [10]中定义：

```
unreserved  =  alphanum | mark
mark        =  "-" | "_" | "." | "!" | "~" | "*" | "'"
            |  "(" | ")"
escaped     =  "%" hex hex
```

SIP 头域值可以被折叠成多行，续行以空格或者 tab 键开始。所有的 LWS 包括折叠和 SP 的意义相同。在解析域值或者向下转发消息前接收者可以将任何 LWS 体换成单个的 SP。

```
LWS  =  *( SP | HT ) [CRLF] 1*( SP | HT ) ; linear whitespace
```

TEXT-UTF8 规则只能用来描述那些不被消息解析器解析的域内容和值。*TEXT-UTF8 词包含 UTF-8 中的字符集（RFC2279 [23]）。TEXT-UTF8-TRIM 规则用来描述不包含在引号中的字符串，该串中起始和结尾的 LWS 都没有意义。在这一点上，SIP 和 HTTP 不同，HTTP 使用 ISO 8859-1 字符集。

```
TEXT-UTF8       =  *(TEXT-UTF8char | LWS)
TEXT-UTF8-TRIM  =  *TEXT-UTF8char *(*LWS TEXT-UTF8char)
TEXT-UTF8char   =  %x21-7e
                |  UTF8-NONASCII
UTF8-NONASCII   =  %xc0-df 1UTF8-CONT
                |  %xe0-ef 2UTF8-CONT
                |  %xf0-f7 3UTF8-CONT
                |  %xf8-fb 4UTF8-CONT
                |  %xfc-fd 5UTF8-CONT
UTF8-CONT       =  %x80-bf
```

在 TEXT-UTF8 中定义了 CRLF,并且它只是作为头域续行的一部分。在对 TEXT-UTF8 解析之前，下面的 LWS 将会被单个的 SP 所代替。

十六进制数字字符在几个协议元素中使用。

```
HEX  =  "A" | "B" | "C" | "D" | "E" | "F"
        | "a" | "b" | "c" | "d" | "e" | "f" | DIGIT
```

许多 SIP 头域值由由 LWS 或者特殊字符分开的词组成。除非另外指定，标记与大小写无关。下面这些特殊的字符使用在参数值中时必须包含在引号中。

```
token                          =  1*(alphanum | "-" | "." | "!" | "%" | "*"
| "_" | "+" | "`" | "'" | "~" )
separators                     =  "(" | ")" | "<" | ">" | "@" |
                                  "," | ";" | ":" | "\" | <"> |
                                  "/" | "[" | "]" | "?" | "=" |
                                  "{" | "}" | SP | HT
```

在 SIP 头域中可以通过将文字包含在圆括号中进行注释。只有那些域值定义中包含"comment"的域才允许注释。在所有其它的域中，圆括号是域值的一部分。

```
comment  =  "(" *(ctext | quoted-pair | comment) ")"
ctext    =  < any TEXT-UTF8  excluding "("  and ")">
```

包含在双引号中的字符将被解析为单个的词。在被引用的字符串中，引号和"\"需要被转义。

```
quoted-string  =  ( <"> *(qdtext | quoted-pair ) <"> )
qdtext         =  LWS | %x21 | %x23-5b | %x5d-7e
                  |  UTF8-NONASCII
```

只有在引用的字符串和注释中，"\"可以被用来作为但字符引用机制。和 HTTP/1.1 不同，字符 CR 和 LF 不能通过这种机制转义以避免和回车和换行混淆。

```
quoted-pair  =  "\" (%x00 - %x09 | %x0b | %x0c | %x0e - %x7f)
```

D IANA 考虑

4.4 节描述了命名空间和 SIP 选项注册机制。10.47 节描述了用来注册 SIP 警告码的命名空间。SIP 头域名使用 IANA 注册。它们不需要工作组或者工作组主席评论，但是应该在 RFC 或者 internet 草案中记载。对于 internet 草案，IANA 使草案作为注册数据库的一部分。到 RFC 发表的时候，冲突的名字可能已经出现了。

下面的信息需要提交给 IANA 以注册一个新的头名字：

?进行注册的个人的姓名和 email 地址。

?被注册的头域的名字。

?如果被定义了，提供该头域紧凑格式的版本。

?定义了头的草案或者 RFC 名字。

?定义了头的草案或者 RFC 的拷贝。

SIP 响应状态码使用 IANA 注册。由于状态码空间被限制，它们需要工作组或者工作组主席评审，并且必须记录在 RFC 或者 internet 草案中。对于 internet 草案，IANA 使草案作为注册数据库的一部分。

为了注册一个新的响应码下面的信息必须提供给 IANA。

?进行注册的个人的姓名和 email 地址。

?被注册的响应码的数字。

?用来描述状态码的缺省原因。

?定义了状态码的草案或者 RFC 名字。

?定义了状态码的草案或者 RFC 的拷贝。

新的头域或者新的状态码都是基于 I-D 创建，并且该 I-D 会成为一个 RFC。进行注册的人必须通知 IANA 去改变注册点到 RFC 而不是 internet 草案。

头域不应该使用 X 前缀符号，并且不能使用 SMTP 或者 HTTP 中使用的头域的名字，除非该语法是兼容的超集并且规则相似。一些公共的广泛使用的头域可以分配一个字符的紧凑格式（13 节）。分配紧凑格式必须经过 SIP 工作组的评审。一个指定的专家也可以代替工作组来评审请求。

E 从 RFC2543 中改变的一些地方

本文在 RFC2543 描述的 SIP 基础上作了如下的修改和增加：

? IETF 所作的扩充不在使用 org.ietf 前缀。

? 规定了 Tag 的语法规则。

? 扩展 Via 头中的 branch 参数以允许"spirals"，两个只是 request URI 不同的请求不在被当作拷贝

? 添加了新的 optional 头域 Alert-Info,Call-Info,In-Reply-To。

F 版本 00 中的改变

? 在 14.4.1 节，指出了重传失败后 UAC 应该发送 CANCEL 和 BYE。

? 分号和问号被当作 SIP URLs 用户部分的非保留字符以便正确的处理 tel: URLs。

? 统一处理跳转超过最大转发数：返回 483。注意这和 HTTP/1.1 不同，HTTP/1.1 只在 OPTIONS 和 TRACE 允许该头，但是在跳转值到 0 时会作为最终的接收者响应。

? 一个 forking proxy 只对 INVITE 请求发送 ACKs。

? 澄清了 DNS 缓存的定义。增加了关于"negative caching"的描述，也就是描述了请求主机失败后怎么处理。如果 DNS ttl 值仍有几分钟，那么简单的将该主机从 DNS 列表中删除并不是好注意，因为也有可能是由于虽然主机恢复在线但负载均衡可能有一段时间没有工作了。特别当服务器群从少量的服务器收到大量的请求时这一点更可能发生，就像主要的 ISPs 之间的呼叫情况。然而，除了不定的、复杂的重发机制外也难找到通用的算法。也许简单的限制黑名单的时间间隔到几分钟会有益处。

? 增加了选项头域 Call-Info 和 Alert-Info 以用来描述主叫和在警告里将使用的信息。

? SDP "s=" 行不能为空。

? 注意 maddr 可能也包含单播地址，但是如果请求通过多播发送过来，那么它应该包含多播地址。(10.46,14.1 节)

? 响应被发送到根据 Via 中 sent-by 值设置的端口。

? 澄清增加"other-*"值到用户 URL 参数以及 Hide、Content-Disposition 头。

? 澄清在 forking proxies 产生 timeout(408)并提及 Expires 头。(17.4)

? 澄清 CANCEL 和 INVITE 是不同的事务。所以当收到 CANCEL 或者 BYE 后 INVITE 请求产生 487（请求被中断）响应。

? 澄清 Record-Route 应该被放到每一个请求中，但是路由一旦建立就一直存在。当被叫 UAS 损坏时这提供了更高的可靠性。

? 强调 proxy,redirect,registrar 和 location 是逻辑实体而不是物理实体，并且 UAC 和 UAS 基于 request-by-request 定义。(1.4 节)

? 在 10.46 节，注意到当进行 Via 隐藏时 maddr 和接收参数也需要被加密。

? 简化图 11 以只显示 INVITE 事务。

? 增加了在 OPTIONS 中使用 Contact 的定义。(10.14 节)

? 增加了 HTTP/RFC822 中的头 Content-Lanuage 和 MIME-Version。

? 在 A 节中指出了 UA 应该支持 UDP。

? 在 15.5 节增加了说明来解释当接收一个初始的带 tag 的 INVITE 时，UA 应该怎么处理。

? 澄清 UA 和 proxy 对 302 响应的处理（11.3.3 节)。

? 增加了当 UAS 收到一个带 tag 的对 call leg 未知的呼叫进行 INVITE 请求应该怎样处理的详细描述。当 UAS 崩溃并且 UAC 发送一个 re-INVITE 或者因为 BYE 丢失而 UAC 仍然相信它在呼叫中时会发生这样的情况。

? 增加了将 4xx,5xx 和 6xx 响应中的 Contact 重定义为更多的错误描述。

? 在 17.4 节增加了 forking proxy 在响应中收集*-Authenticate 的描述。它允许几个分支同时被鉴权。

? 改变了 URI 语法规则以不使用引号对 URL 进行转义。

? 对电话用户部分参照 RFC2806 改变了 SIP URL 的定义。

? 澄清 To URI 基本上应该被接收的 UAS 丢弃除非它的 call legs 和请求相匹配。特别指出的是，包含对被叫是未知的草图或者名字的 To 头应该被接收。

? 在 10.46.1 节澄清 maddr 可以被任何客户端添加，包括 proxy 和 UAC。

？ 增加响应码 488 来表示在制定的目的地没有该媒体。（606 指出该错误在全局发生）

？ 在 10.24 节，提到注册更新可以缩短有效周期。

？ 在 19.3 节，增加了将 URI 包含在引号中。RFC2617 中的 BNF 是错误的。

？ 澄清 registrars 使用 Authorization 和 WWW-Authenticate 而不是 proxy authentication。

？ 增加了对"headers"从 Contact 拷贝到新的请求中的注解。

？ 改变了 URL 语法规则这样端口描述至少需要一个数字，这和像"http"等其它的 URL 格式相符。以前，允许空的端口号。

？ 在 B 节，增加了一节来描述怎样使用 re-INVITE 来添加和删除流。

？ IETF 的扩展名现在有了简短的名字，去掉了 org.ietf 前缀。

？ Cseq 不仅在一个 call 中是唯一的，在一个 call leg 中也是唯一的。（10.20 节）

？ 遵循 RFC2732 [45]在 SIP URL 定义中添加了 Ipv6 地址。修改 Ipv4 地址中的段至少有三位数字。

？ 在 7 节，修改了注册过程这样它就明确的参考了 URL 对比。现在可以将过期事件修改为更短的时间。

？ 对于只发送媒体，SDP 仍然必须指出地址和端口，因为在 RTCP 消息中需要使用它们作为地址。（Section B）

？ 将 DNS SRV 记录的参考对象从 RFC2052 调整到被推荐的标准 RFC2782。将 SRV 合并到 1 节中的搜寻过程并且删掉了 SRV 附录。唯一可以看见的改变是将所有的协议和服务名加上了一个"-"前缀。增加了指示 maddr 优先级的词。

？ 允许 Record-Route 和 Route 头中的参数。

？ 在表 2 中，将 udp 作为 SIP URI 中 transport 参数的缺省值。

？ 将 From 头改为不能被加密。因为该头需要用来作 call leg 标识。

？ 在 15 节中增加了如下描述：如果对一个 INVITE 请求返回 200 OK 响应，UAC 只将 To 标记拷贝到后续的事务中。这避免了当收到比如说 407（或者可能是 500，503，504，305，400，411，413，408）响应时重发请求。在以前的规则中，这些请求会有一个 tag,并且由于被叫 UAS 没有该 tag 的记录而将请求拒绝。

？ 修改成通过检查 request-URI 来进行环路检查（17.3 和 10.46.6 节）。这可以让有不同 request-URI 的请求两次访问服务器。

？ 对 URL 比较和 From/To 域比较进行了详细描述。

？ 增加了 np-queried 用户参数。

？ 将 tag 的语法从 UUID 改为符号，应为没有什么原因来将它限制为 hex。

？ 基于以前的描述消息体的功能标签的讨论增加了 Content-Disposition 头。

？ 澄清：proxies 必须加上 To 标记以便本地产生响应。

？ 澄清：多播可以用来进行并发注册。

？ 特征：增加 Supported 头。客户端可以使用它来标示服务器能在响应中返回的事件。

？ Bug：From，To 和 Via 头都缺少扩展参数。Encryption 和 Response-Key 头域现在正式允许由一个标记组成的参数，而不只是"token = value"。

？ Bug：关于 405 响应的允许选项事件列表列在表 4 中。它是强制性的。

？ 在 6 节增加了：无论是主叫或者被叫方都可以使用 BYE 请求来中断未完成的 INVITE 请求，但是 INVITE 请求事务必须使用最终响应来终结。

？ 在 5.1 节澄清：如果对一个已存在会话的 INVITE 请求失败，上一次成功 INVITE 请求事务中的会]话描述保持有效。

？ 在 5.1 节澄清：两个 INVITE 请求在线上同方向或者相对相遇的处理：

对于相同的 call leg,在前一个事务完成之前 UAC 不能发送另外一个 INVITE 请求。在对一个 Cseq 数字更小的 INVITE 进行返回最终响应之前，UAS 必须返回一个 400（错误请求）响应并且必须包含一个值为 0 到 10 秒之间随机的数字的 Retry-After 头域。当一个 INVITE 请求事务未完成的 UA 收到一个 INVITE 请求时返回 500（内部服务器错误）响应并且也包含一个 Retry-After 头域。

？ 过期头澄清：仅在 INVITE 事务期间存在限制，而不是当前的会话。SDP 的过期限制存在于当前会话期间。

？ 在 10.26 节中增加了 In-Retry-To 头。

？ 对于 WWW-Authenticate 由两个不兼容的 BNF。一个为 PGP 而定义，一个从 HTTP 中借鉴过来。对于 basic 或者 digest：

WWW-Authenticate：basic realm="Wallyworld"

对于 PGP:

WWW-Authenticate: pgp; realm="Wallyworld"

后面一个不正确并且分号被删掉。

? 增加了从主叫或者被叫 UA 构建路由的规则。

? 在 BYE 和 CANCEL 请求中现在允许 Accept 和 Accept-Encoding 头。没有特别的原因不允许这样，因为两个请求在理论上都可以返回响应，特别适当它们和其它的信令系统交互时。

? PGP 的"pgp-pubalgorithm"允许服务器请求需要的 public-key 算法。

? 现在 ABNF 规则明确的描述标记。

? 注册服务器应该仔细的检查 Date 头是否过期。

? 必须包含 Content-Length 头；表 4 错误的将它标识为可选的。

? User-Agent 当作一个请求头定义而不是当作一个通用头定义。

? 澄清被标记条目的顺序并将 realm 包含在列表中。

? 允许在 401 和 484 响应中使用 Record-Route 头。

? 只有当需要鉴权时，hop-by-hop 才需要放在 end-to-end 头前。(10 节)

? 1xx 响应消息体现在可以包含会话描述。

? 参考对象改变为 HTTP/1.1,鉴权借鉴于最新的 RFCs。

? 增加了 487（请求中断）状态响应。当初始的请求被 CANCEL 或者 BYE 中断时发送该响应。

? 规范说明书对 call leg 的定义不清楚。1.3 节说它是 To,From,Call-ID 的组合。然而，被叫到主叫的请求将 To 和 From 顺序翻转，所以这个定义不是非常正确。另外，call leg 的定义应该包含"tag"域。规范说明书现在规定 call leg 定义为 local-address,remote-address,call-id 的组合，并且这些地址中包含 tags。

6.21 节新增规则强调：From 和 To 头指定了请求的发起者而不是 call leg 的发起者。

? 除了 method 外所有的 URI 参数可以出现在 Request-URI 头中。因此更新了在 10.14 节中描述的那些参数从 3xx 响应中拷贝出来。

? 使用 CRLF,CR 或者 LF 来换行引起了混乱。更进一步，使用两个换行来结束头域。3 节中只允许发送 CRLF,但要求发送者可以接收 CR 和 LF 并且除了两个 CRLF 外只允许 CR CR,LF LF 作为消息头

和消息体的分隔符。

? Contact 头中的圆括号自 HTTP 继承而来并且非常难以实现。它们用处不大所以被删除。

? 规范说明书提到 proxy 是 UAS 和 UAC 的组合。这差不多是对的，但不完全对。比如说，一个 UAS 应该将一个 tag 加到临时响应里，但是 proxy 不需要。这一点需要阐明。

? RFC 中的 6.13 节在 BNF 后开始中间段落。下面的文字在转为 ASCII 后丢失：

如果"addr-spec"包含一个逗号，分号或者问号，即使"display-name"是空的，也必须使用"name-addr"。

G 版本 01 中的修改

? 对分号参数的统一语法说明：

```
Foo        =  "Foo" ":" something *( ";" foo-param )
foo-param  =  "bar" "=" token
           |   generic-param
```

? 删除掉 np-queried 用户参数因为这已经是电话 URL 扩展参数的一部分。

? 在 B 节中提到：由于 SDP 语法规则限制，即使能力交集是空的，也必须返回一个虚拟格式列表。

? 重新组织了表 4 和表 5 来显示 proxy 通过头域来交互而不仅仅是"end-toend"或者"hop-by-hop"。

H 版本 02 中的改变

? 在 10.46.1 节中描述接收头是加上了"or UAS"。这使即使 Via 列表中最后的 IP 地址不正确响应规则仍然能正常处理。

? 暂时去掉了在 CANCEL 请求中不能有 Route 头的限制。

? 暂时在 BYE 请求中加上了 Also 头，因为它被广泛的实现并且是实现无监控呼叫转发的简单方法。

如果有人抗议可以将它删除。

? 如果一个 proxy 使用 UDP(TCP)发送请求,规则允许将 Via 头域中的传输层参数设置为 TCP(UDP)。也包含了传输层实际使用的协议。

? 在 Contact 头中的 q 参数没有定义缺省值。这不是特别需要，但是使用它来同一在 UAC 和第归的 proxies 的行为很有作用。现在缺省值为 0.5。

？ 阐明：呼叫自己时为了简化请求匹配，To 和 From 标记的值应该不同。

？ 去掉了在单个的 UDP 包中传输多个请求。（10.18 节）

？ 增加了可以在请求中包含 Allow 头以在相同的 call ID 中指出请求者的能力。

？ 在 10.21 节中增加了指示注册服务器必须包含一个 Date 头来使 Uas 认识到不能有绝对时间的概念。

？ 在 7 节中增加了允许非 SIP URIs。

？ 重写了服务器查询部分以便更准确更像伪码并将"gotos"体换成嵌套。

？ 删除掉的注解

两个 URLs example.com 和 example.com:5060 被看成一样，但可能会到达不同的服务器，因为前者会导致一个 DNS SRV 查询，而后者只是使用一个记录。现在情况不是这样的。

？ 强调 proxies 必须使用未知的方法转发请求。

？ 将 call leg 定义和 URI 对比规则相联系。

？ 要求第二个分支参数使全局唯一，这样在下面所示的螺旋上升环境中一个 proxy 能够区别不同的分支：

```
                B  ---> P1 -------> P2 ------------> P1 ----------------> A
   BYE B    B/1       P1/2,B/1     P2/3,P1/2,B/1     P1/4,P2/3,P1/2,B/1
```

这里，A/1 使用主机 A 和分支参数 1 来标示 Via 记录。这样，这要求更新 isomorphic requests 的定义，因为对于所有被记录路由的 BYE 请求 Request-URI 都是相同的。

？ 从规范说明书中删除 Via 隐藏，原因如下：

复杂性，特别是隐藏在多个地方显现出来的"gotchas"；

和换路监测和排错冲突；

不像 HTTP，它将所有的数据包含在请求或者响应中所以 via 隐藏有意义，在 SIP 中 Via 隐藏不能将主叫和被叫隐藏，因为地址信息出现在多个地方：

```
Contact;
Route/Record-Route;
SDP,including the o= and c= lines;
possibly accidental leakage in User-Agent header and Call-ID headers.
```

除非在每个地方实现，否则该特征不是很有用处。明显的是几乎所有现存的 proxies 简单的忽视 Hide 头域。

? 增加了 Error-Info 头域。

I Version 03 中的改变

? 对 Route 和 Record-Route 的描述分开到不同的节，节 16 是新加上的。所有的 Uas 现在必须支持该机制（A 节）。

? 删除状态码 411，因为它不可能发生。

? 为了反映新的机制重写了 Record-Route 部分。特别是，从被叫到主叫的请求现在在相对的方向使用相同的路径，而不用替换 From 头域的值。Maddr 参数现在是可选的。

? 不允许 SIP URLs 中只包含一个密码而没有用户名。RFC1738 中的原型也不允许这样。

? 允许 registrar 设置过期时间。

? Cseq（10.20 节）在一个 call leg 里计数而不是在一个呼叫里。

? 删除掉语句：连接关闭和 CANCEL 或者 500 响应等价。当该连接用在多播事务里时并不是这样并且还有其它的问题。

? 整理了 Cseq 节。删掉了插入 Cseq 方法的文字。阐明 Cseq 对于所有的请求增加，而不只是 invite。阐明所有错序的请求而不只是错序的 INVITE 被 400 类型的响应拒绝。阐明"initial"序列数字的意义。阐明在请求派生后，每一个 200 OK 都是一个分开的 call leg，并且是分开的 Cseq 空间。阐明 Cseq 数字和 call leg 的每一个方向无关。

? 重新组织和整理了 SDP 部分。介绍了 offer-answer 模型的概念。阐明 m 行中的 codecs 集可以同时使用。增加了 o 行中显示值的限制。明确的描述了派生媒体。通过在 SDP 的下面添加新的媒体行来增加流。

? 删除 Also。

? 增加了对 Require 和 Proxy-Require 的描述，使得它可以在去掉不支持的扩展后重新请求。

? 增加了对 415 响应的描述，提到 UAC 在去掉不支持的部分后应该重试请求。

? 增加了对 CANCEL 和 ACK 的描述。

? 修改了 Content-Type，指出了当消息体为空时它也可以出现。

? From 标记必须有。

？旧的文章指出如果你在发送 ACK 之前挂断，那么你不需要发送 ACK。这是错误的。ACK 总是发送。

？旧的文章指出如果你发送 INVITE 后总收不到响应，UAC 应该同时发送 INVITE 和 CANCEL。这并没有意义。更正确的是，它什么也不做并且认为呼叫被中断。

？增加了规则：一个未完成的请求收到 BYE 后返回一个 487 响应。

？更新了 2.2 节，明确了 BYE 不使用 Contact。

？阐明了 Via 处理规则。增加了文字描述当 proxies 使用除 request URI 之外的头来路由时怎样处理回路。增加了文字描述当 sent-by 包含域名时怎样处理。增加了文字描述使用开放的 TCP 连接向上发送响应。

？阐明 1xx 响应和 100 响应不同。

？去掉了在 REGISTER 中使用 Retry-After。

？阐明永久连接的使用。

？阐明支持 RFC2617 重的 HTTP basic 或者 digest 的服务器应向后兼容 RFC2069。

？阐明 ACK 包含的 branch ID 和它响应的请求相同。

？增加了对 spiral,B2BUA 的定义。

？重定义了 UAC,UAS,Call,call-leg,caller,callee,以使定义更具体。

？URL 比较忽视那些没有在双方 URLs 中出现的参数只对未知的参数有效。

？阐明 Contact 中的"*"只是在 Expires 头为 0 的 REGISTER 中使用。在描述 Contact 语法的章节中谈到了"*"的使用。

？删除掉语句：UA 可以在 2xx 中插入 Contact 头以指示 proxy 的地址。通常这使不对的。

？删除掉 SDP 描述：对于相同的媒体类型可以将媒体流合并以便处理系统崩溃和重起的情况。

？在呼叫的中间收到 481 响应会中断该 call leg。在 IETF 49 上达成了该意见。

？介绍了常规事务的定义---指的是除了 ACK 和 CANCEL 的非 INVITE 请求。

？阐明了重叠事务的规则。

？ Forking proxies 必须是有状态的（以前说是应该）。发送多播请求的 proxies 必须是有状态的（以前没有提到）。

？ 增加了描述：registrars 允许 From 域中的记录可以注册 To 域中的地址记录。

？ 必须将非 100 临时响应向上转发（以前说是应该）。

？ 删除掉 PGP。

J 版本 04 中的修改

？ 将 Unsupported 作为一个请求头从表 5 中删除掉。

？ 阐明修改 IP 地址和端口的 SDP 过程。特别是，清楚的指出了 UA 在旧的地址和端口上接收媒体持续的时间。

？ 增加了描述：会话响应中 codecs 的顺序和会话发起中的相同，以保证在通常的情况下 codec 均衡操作。

？ 修改了 SDP 节中例子的 bug，在那里提到新的媒体行在头部列出。实际应该在底部列出。

？ 授权书应该基于 To 头中的 URL 缓存，而不是像 10.48 中所说的整个 To 头。

？ 在 10.31 节提到，对于 proxy-authenticate 指出了如果客户端猜测错误服务器应返回 401。这是错误的。它应该返回 407。

？ 在 10.14 节，删除掉相关的文字：Contact 头允许 INVITE 请求在终端系统间直接路由，因为它引起混乱。一些人解释说它的意思是当出现 Contact 头时 Record-Route 被忽略。

？ 增加了相关的 SCTP RFC。

？ 更新了 2.2 节以允许在 OPTIONS 中使用非 SIP URLs 并且可以对 OPTIONS 返回 2xx 响应。

？ 修改了 20.5 节中的例子。增加了对 487 响应的 ACK 请求，并且在 487 响应中增加了 To 标记。

？ 阐明了更进一步的 URL 对比。

？ 在 1.5.2 节，所有的实体强制支持 UDP。UA 应该支持 TCP，proxy、registrar、redirect servers 必须支持 TCP。

？ 是文字和 postscript 版本的 Contact，Via 和 SIP URL 语法规则相同。

？ 更新了节 6 中的文字：头域的顺序和 HTTP 中的相同，除了 Via 头是 order matters。然而，HTTP

规范也是 order matters,这样该语句是冗余混乱的。该语句被删除。

? 在 SDP 例子中增加 e 行。

? 重写了关于 Allow 的讨论，更正式的定义了它的语法和使用例子。

? 更新了关于 604 状态码的描述，以指示它基于 Request-URI 而不是 To。

? 增加了响应注册提交给 IANA 考虑。对注册过程提供了更详细的描述。

? 阐明了只有当 To 标记和本地的值不匹配时 UAS 拒绝该请求。

? 阐明无状态 proxies 只需要基于静态规则进行路由。

? 在派生的例子中，proxy 和 UAC 应该对 2xx,6xx 响应发送 CANCEL；以前是可以发送。

? 增加了描述：对于建立的 call leg，如果 UAS 发送 2xx 后不能收到 ACK 请求那么它应该发送一个 BYE 请求。

? 增加了描述 :对于 re-INVITE 的响应 2xx,如果 UAS 不能收到 ACK 请求那么它应该发送 re-INVITE。

? 增加了对 503 响应处理的描述：客户端在收到 503 响应后应该尝试其它的不同服务器，除非一个 proxy 不能发送更多的请求，否则它不应该向上转发 503 响应。

? 删除掉 10.43 节关于 Via 头的描述，因为它和它前面的文字不一致。

? IPSec 参照 RFC2401，以前是参照 RFC1825。

? 在 17.3.4 节中更新了重传的定义以便和规范说明书中其它的部分一致。

? 在 record-route 头中增加 transport 参数。特别是，在支持特殊传输层协议的私网中可以插入该参数。

? 更新了 B2BUA 的定义。

? 增加了对 420 响应处理的描述：客户端必须去掉响应中不支持的扩展头重试请求。

? 允许在 HTTP digest 中使用 Authentication-Info 头。

K 感谢信