1 介绍

有许多 Internet 应用程序需要创建和管理会话，其中会话被视为参与者关联之间的数据交换。这些应用程序的实施因参与者的实践而变得复杂：用户可能在端点之间移动，他们可能可以通过多个名称寻址，并且他们可能在几种不同的媒体中进行通信——有时是同时进行的。已经编写了许多协议来承载各种形式的实时多媒体会话数据，例如语音、视频或文本消息。会话发起协议 (SIP) 与这些协议协同工作，使 Internet 端点（称为用户代理）能够发现彼此并就他们想要共享的会话的特征达成一致。为了定位潜在的会话参与者和其他功能，SIP 支持创建网络主机（称为代理服务器）的基础设施，用户代理可以向其发送注册、会话邀请和其他请求。 SIP 是一种灵活的通用工具，用于创建、修改和终止会话，它独立于底层传输协议工作，并且不依赖于正在建立的会话类型。

2 SIP 功能概述

SIP 是一种应用层控制协议，可以建立、修改和终止多媒体会话（会议），例如 Internet 电话呼叫。 SIP 还可以邀请参与者加入已经存在的会话，例如多播会议。 可以将媒体添加到现有会话中（或从中删除）。 SIP 透明地支持名称映射和重定向服务，从而支持个人移动性 [27] - 用户可以维护一个外部可见的标识符，而不管他们的网络位置如何。

SIP 支持建立和终止多媒体通信的五个方面：

用户位置：确定用于通信的终端系统；

用户可用性：确定被叫方参与通信的意愿；

用户能力：确定要使用的媒体和媒体参数；

会话建立：“振铃”，在被叫和主叫双方建立会话参数；

会话管理：包括会话的转移和终止、会话参数的修改、服务的调用。

SIP 不是一个垂直集成的通信系统。 SIP 是一个可以与其他 IETF 协议一起使用以构建完整的多媒体架构的组件。 通常，搜索架构将包括诸如用于传输实时数据和提供 QoS 反馈的实时传输协议 (RTP) (RFC 1889 [28])、用于控制的实时流协议 (RTSP) (RFC2326 [29]) 流媒体传输，用于控制公共交换电话网络 (PSTN) 网关的媒体网关控制协议 (MEGACO) (RFC 3015 [30])，以及用于描述多媒体会话的会话描述协议 (SDP) (RFC 2327 [1])。 因此，SIP 应该与其他协议结合使用，以便为用户提供完整的服务。 但是，SIP 的基本功能和操作不依赖于这些协议中的任何一个。

SIP 不提供服务。 相反，SIP 提供了可用于实现不同服务的原语。 例如，SIP 可以定位用

户并将不透明的对象传送到他的当前位置。 例如，如果该原语用于传递以 SDP 编写的会话描述，则端点可以就会话的参数达成一致。 如果使用相同的原语来传递呼叫者的照片以及会话描述，则可以轻松实现“呼叫者 ID”服务。 如本例所示，单个原语通常用于提供多种不同的服务。

SIP 不提供会场控制或投票等会议控制服务，也没有规定如何管理会议。 SIP 可用于启动使用其他会议控制协议的会话。 由于 SIP 消息和它们建立的会话可以通过完全不同的网络，SIP 不能也不会提供任何类型的网络资源预留能力。

所提供服务的性质使得安全性尤为重要。 为此，SIP 提供了一套安全服务，包括拒绝服务预防、身份验证（用户对用户和代理对用户）、完整性保护以及加密和隐私服务。

SIP 适用于 IPv4 和 IPv6。

3 术语

在本文档中，关键词“必须”、“不得”、“要求”、“应”、“不得”、“应该”、“不应”、“推荐”、“不推荐”、“可能”，和“可选”将按照 BCP 14、RFC 2119 [2] 中的描述进行解释，并指示合规 SIP 实现的要求级别。

4 操作概述

本节通过简单的示例介绍 SIP 的基本操作。 本节本质上是教程，不包含任何规范性陈述。

第一个示例显示了 SIP 的基本功能：端点的位置、通信愿望的信号、会话参数的协商以建立会话，以及会话一旦建立就拆除。

图 1 显示了两个用户 Alice 和 Bob 之间的 SIP 消息交换的典型示例。（每条消息都标有字母“F”和文本引用的数字。）在此示例中，Alice 使用她 PC 上的 SIP 应用程序（称为软电话）通过 Internet 呼叫 Bob 上的 SIP 电话 . 还显示了两个 SIP 代理服务器，它们代表 Alice 和 Bob 来促进会话建立。 这种典型排列通常被称为“SIP 梯形”，如图 1 中虚线的几何形状所示。

Alice 使用他的 SIP 身份“呼叫” Bob，这是一种称为 SIP URI 的统一资源标识符 (URI)。 SIP URI 在第 19.1 节中定义。 它具有与电子邮件地址类似的形式，通常包含用户名和主机名。在这种情况下，它是 sip:bob@biloxi.com，其中 biloxi.com 是 Bob 的 SIP 服务提供商的域。Alice 的 SIP URI 为 sip:alice@atlanta.com。 Alice 可能已经输入了 Bob 的 URI，或者可能单击了超链接或地址簿中的条目。 SIP 还提供了一个安全的 URI，称为 SIPS URI。 一个例子是 sips:bob@biloxi.com。 对 SIPS URI 的调用可确保使用安全、加密的传输（即 TLS）将所有 SIP 消息从调用者传送到被调用者的域。 从那里，请求被安全地发送到被调用者，但安全机制取决于被调用者域的策略。

SIP 基于类似 HTTP 的请求/响应事务模型。 每个事务由调用服务器上特定方法或函数的请求和至少一个响应组成。在此示例中，事务从 Alice 的软电话发送一个指向 Bob 的 SIP URI 的 INVITE 请求开始。 INVITE 是 SIP 方法的一个示例，它指定请求者 (Alice) 希望服务器 (Bob) 采取的操作。 INVITE 请求包含许多标头字段。 标头字段是提供有关消息的附加信息的命名属性。 INVITE 中包含的信息包括呼叫的唯一标识符、目标地址、Alice 的地址以及 Alice 希望与 Bob 建立的会话类型的信息。 INVITE（图 1 中的消息 F1）可能如下所示：

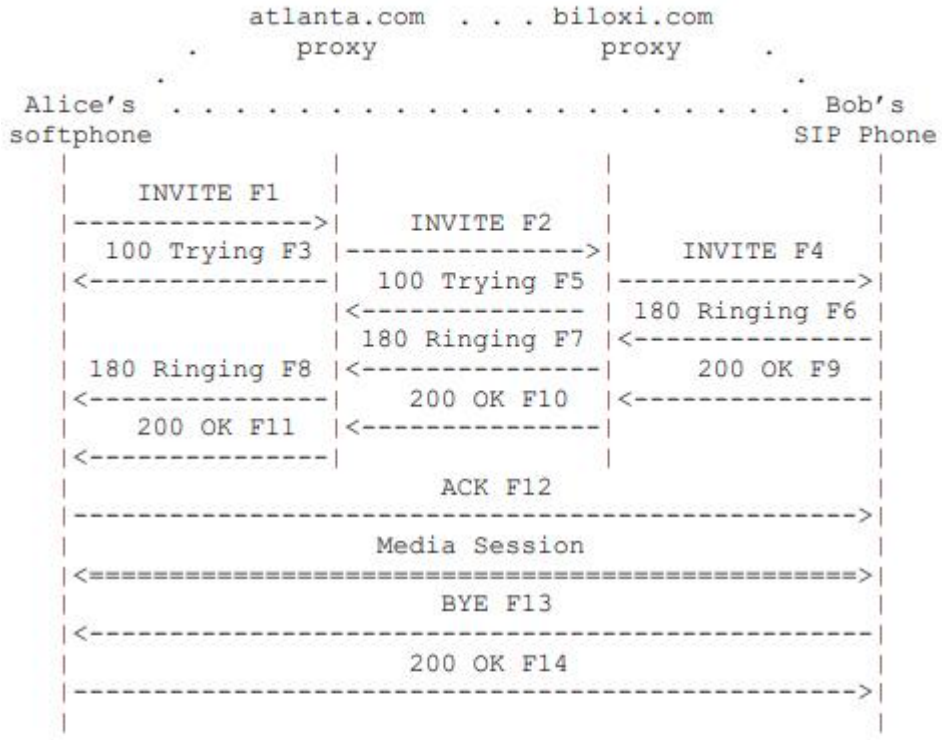

Figure 1: SIP session setup example with SIP trapezoid

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

文本编码消息的第一行包含方法名称 (INVITE)。 接下来的行是标题字段的列表。 此示例包含最低要求集。 头字段简要描述如下：

Via 包含 Alice 期望接收对此请求的响应的地址 (pc33.atlanta.com)。 它还包含一个标识此事务的分支参数。

To 包含请求最初指向的显示名称 (Bob) 和 SIP 或 SIPS URI (sip:bob@biloxi.com)。 显示名称在 RFC 2822 [3] 中描述。

From 还包含一个显示名称 (Alice) 和一个 SIP 或 SIPS URI (sip:alice@atlanta.com)，指示请求的发起者。 此标头字段还有一个标签参数，其中包含由软电话添加到 URI 的随机字符串 (1928301774)。 它用于识别目的。

Call-ID 包含此呼叫的全局唯一标识符，由随机字符串和软件电话的主机名或 IP 地址组合生成。To 标签、From 标签和 Call-ID 的组合完全定义了 Alice 和 Bob 之间的对等 SIP 关系，称为对话。

Cseq 或命令序列包含一个整数和一个方法名称。 Cseq 编号对于对话中的每个新请求都会增加，并且是传统的序列号。

Contact 包含一个 SIP 或 SIPS URI，它表示与 Alice 联系的直接路由，通常由完全限定域名 (FQDN) 中的用户名组成。 虽然首选 FQDN，但许多终端系统没有注册域名，因此允许使用 IP 地址。 Via 头域告诉其他元素将响应发送到哪里，而 Contact 头域告诉其他元素将未来的请求发送到哪里。

Max-Forwards 用于限制请求在到达目的地的途中可以进行的跳数。 它由一个整数组成，在每一跳减一。

Content-Type 包含消息正文的描述（未显示）。

Content-Length 包含消息正文的八位字节（字节）计数。

完整的 SIP 头字段集在第 20 节中定义。

会话的细节，例如媒体类型、编解码器或采样率，没有使用 SIP 描述。 相反，SIP 消息的主体包含会话的描述，以某种其他协议格式编码。 一种这样的格式是会话描述协议 (SDP) (RFC 2327 [1])。 该 SDP 消息（示例中未示出）由 SIP 消息以类似于通过电子邮件消息携带文档附件或在 HTTP 消息中携带网页的方式由 SIP 消息携带。

由于软件电话不知道 Bob 的位置或 biloxi.com 域中的 SIP 服务器的位置，因此软件电话将 INVITE 发送到服务于 Alice 的域 atlanta.com 的 SIP 服务器。 例如，atlanta.com SIP 服务器的地址可能已在 Alice 的软电话中配置，或者可能已被 DHCP 发现。

atlanta.com SIP 服务器是一种称为代理服务器的 SIP 服务器。代理服务器接收 SIP 请求并代表请求者转发它们。在此示例中，代理服务器接收到 INVITE 请求并将 100（尝试中）响应发送回 Alice 的软电话。 100（正在尝试）响应表明已收到 INVITE，并且代理正在代表她将 INVITE 路由到目的地。 SIP 中的响应使用三位数字代码，后跟一个描述性短语。此响应在 Via 中包含与 INVITE 相同的 To、From、Call-ID、CSeq 和分支参数，这允许 Alice 的软电话将此响应与发送的 INVITE 相关联。 atlanta.com 代理服务器在 biloxi.com 上定位代理服务器，可能通过执行特定类型的 DNS（域名服务）查找来查找为 biloxi.com 域提供服务的 SIP 服务器。这在 [4] 中有描述。结果，它获得了 biloxi.com 代理服务器的 IP 地址，并在那里转发或代理 INVITE 请求。在转发请求之前，atlanta.com 代理服务器添加了一个额

外的 Via 头字段值，其中包含它自己的地址（INVITE 已经包含 Alice 在第一个 Via 中的地址）。 biloxi.com 代理服务器接收到 INVITE 并向 atlanta.com 代理服务器返回 100（尝试）响应，表明它已收到 INVITE 并正在处理请求。代理服务器查询数据库，通常称为位置服务，其中包含 Bob 的当前 IP 地址。 （我们将在下一节看到如何填充这个数据库。）biloxi.com 代理服务器将另一个带有自己地址的 Via 头字段值添加到 INVITE 并将其代理到 Bob 的 SIP 电话。

Bob 的 SIP 电话收到 INVITE 并提醒 Bob 有来自 Alice 的来电，以便 Bob 决定是否接听电话，即 Bob 的电话响铃。 Bob 的 SIP 电话在 180（振铃）响应中表明了这一点，该响应通过两个代理反向路由回。 每个代理使用 Via 头字段来确定将响应发送到哪里，并从顶部删除自己的地址。 因此，尽管需要 DNS 和位置服务查找来路由初始 INVITE，但可以将 180（响铃）响应返回给调用者，而无需查找或在代理中维护状态。 这还具有理想的属性，即每个看到 INVITE 的代理也将看到对 INVITE 的所有响应。

当 Alice 的软电话收到 180（振铃）响应时，它会将这个信息传递给 Alice，可能使用音频回铃音或通过在 Alice 的屏幕上显示消息。

在此示例中，Bob 决定接听电话。 当他拿起听筒时，他的 SIP 电话会发送一个 200（OK）响应，表示呼叫已被应答。 200（OK）包含一个消息体，其中包含 Bob 愿意与 Alice 建立的会话类型的 SDP 媒体描述。 因此，SDP 消息的交换分为两阶段：Alice 向 Bob 发送了一个，Bob 将一个返回给 Alice。 这种两阶段交换提供了基本的协商能力，并基于简单的 SDP 交换提供/应答模型。 如果 Bob 不想接听电话或正忙于另一个电话，则会发送错误响应而不是 200（OK），这将导致没有建立媒体会话。 SIP 响应代码的完整列表在第 21 节中。200（OK）（图 1 中的消息 F9）在 Bob 发出时可能如下所示：

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

响应的第一行包含响应代码 (200) 和原因短语 (OK)。其余行包含标题字段。Via、To、From、Call-ID 和 CSeq 标头字段是从 INVITE 请求中复制的。 （有三个 Via 头字段值 - 一个由 Alice 的 SIP 电话添加，一个由 atlanta.com 代理添加，一个由 biloxi.com 代理添加。） Bob 的 SIP 电话在 To 头字段中添加了一个标记参数。 此标签将由两个端点合并到对话中，并将包含在此调用中的所有未来请求和响应中。 Contact 标头字段包含一个 URI，Bob 可以通过该 URI 直接与他的 SIP 电话联系。 Content-Type 和 Content-Length 指的是包含 Bob 的 SDP 媒体信息的消息体（未显示）。

除了本示例中显示的 DNS 和位置服务查找之外，代理服务器还可以做出灵活的“路由决策”来决定将请求发送到何处。例如，如果 Bob 的 SIP 电话返回 486（这里忙）响应，biloxi.com 代理服务器可以将 INVITE 代理到 Bob 的语音邮件服务器。代理服务器还可以同时向多个位置发送邀请。这种类型的并行搜索称为分叉。

在这种情况下，200（OK）通过两个代理路由返回并被 Alice 的软电话接收，然后停止回铃音并指示呼叫已被应答。最后，Alice 的软电话向 Bob 的 SIP 电话发送确认消息 ACK，以确认收到最终响应（200（OK））。在这个例子中，ACK 直接从 Alice 的软电话发送到 Bob 的 SIP 电话，绕过了两个代理。发生这种情况是因为端点通过 INVITE/200 (OK) 交换从 Contact 标头字段中了解了彼此的地址，而在发送初始 INVITE 时不知道这一点。不再需要两个代理执行的查找，因此代理退出调用流程。这样就完成了用于建立 SIP 会话的 INVITE/200/ACK 三向握手。会话设置的完整细节在第 13 节。

Alice 和 Bob 的媒体会话现在已经开始，他们使用他们在 SDP 交换中同意的格式发送媒体包。通常，端到端媒体包采用与 SIP 信令消息不同的路径。

在会话期间，Alice 或 Bob 都可能决定改变媒体会话的特征。这是通过发送包含新媒体描述的 re-INVITE 来完成的。这个 re-INVITE 引用了现有的对话，以便另一方知道它是要修改现有的会话而不是建立新的会话。对方发送 200（OK）以接受更改。请求者使用 ACK 响应 200（OK）。如果对方不接受变更，则发送 488（Not Acceptable Here）等错误响应，同样收到 ACK。然而，re-INVITE 的失败不会导致现有呼叫失败——会话继续使用先前协商的特性。会话修改的完整细节在第 14 节。

通话结束时，Bob 首先断开（挂断）并生成 BYE 消息。此 BYE 直接路由到 Alice 的软电话，再次绕过代理。Alice 用 200（OK）响应确认收到 BYE，终止会话和 BYE 交易。不发送 ACK - 仅发送 ACK 以响应对 INVITE 请求的响应。稍后将讨论对 INVITE 进行这种特殊处理的原因，但与 SIP 中的可靠性机制、响铃电话被应答所需的时间长度以及分叉有关。因此，SIP 中的请求处理通常分为 INVITE 或非 INVITE，指的是除 INVITE 之外的所有其他方法。会话终止的完整细节在第 15 节。

第 24.2 节完整描述了图 1 中显示的消息。

在某些情况下，SIP 信令路径中的代理可以在会话期间查看端点之间的所有消息传递。例如，如果 biloxi.com 代理服务器希望保留在初始 INVITE 之外的 SIP 消息传递路径中，它将向 INVITE 添加一个称为 Record-Route 的必需路由标头字段，其中包含解析为主机名或 IP 地址的 URI 的代理。Bob 的 SIP 电话和（由于 Record-Route 标头字段在 200 (OK) 中传回）Alice 的软电话都将收到此信息，并在对话期间存储。然后，biloxi.com 代理服务器将接收 ACK、BYE 和 200（OK）并将其代理到 BYE。每个代理可以独立决定接收后续消息，这些消息将通过所有选择接收它的代理。此功能经常用于提供呼叫中功能的代理。

注册是 SIP 中的另一个常见操作。注册是 biloxi.com 服务器了解 Bob 当前位置的一种方式。在初始化时，Bob 的 SIP 电话会定期向 biloxi.com 域中的服务器发送 REGISTER 消息，该服务器称为 SIP 注册器。REGISTER 消息将 Bob 的 SIP 或 SIPS URI (sip:bob@biloxi.com) 与他当前登录的机器相关联（在 Contact 头字段中作为 SIP 或 SIPS URI 传送）。注册商

将此关联（也称为绑定）写入称为位置服务的数据库，在 biloxi.com 域中的代理可以使用该关联。 通常，域的注册服务器与该域的代理位于同一位置。 SIP 服务器类型之间的区别是逻辑上的，而不是物理上的，这是一个重要的概念。

Bob 不限于从单个设备注册。 例如，他在家中的 SIP 电话和办公室中的 SIP 电话都可以发送注册信息。 此信息一起存储在定位服务中，并允许代理执行各种类型的搜索以定位 Bob。 同样，可以同时在一台设备上注册多个用户。

位置服务只是一个抽象概念。 它通常包含允许代理输入 URI 并接收一组零个或多个 URI 的信息，这些 URI 告诉代理将请求发送到哪里。 注册是创建此信息的一种方式，但不是唯一方式。 管理员可以自行配置任意映射功能。

最后，需要注意的是，在 SIP 中，注册用于路由传入的 SIP 请求，而在授权传出请求时没有任何作用。 授权和认证在 SIP 中通过挑战/响应机制在逐个请求的基础上进行处理，或者使用第 26 节中讨论的较低层方案来处理。

此注册示例的完整 SIP 消息详细信息集在第 24.1 节中。

SIP 中的其他操作，例如使用 OPTIONS 查询 SIP 服务器或客户端的功能，或使用 CANCEL 取消挂起的请求，将在后面的部分中介绍。

5 协议结构

SIP 被构造为分层协议，这意味着它的行为是根据一组相当独立的处理阶段来描述的，每个阶段之间只有松散的耦合。 为了表示的目的，协议行为被描述为层，允许在单个部分中描述跨元素的通用功能。 它不以任何方式规定实施。 当我们说一个元素“包含”一个层时，我们的意思是它符合该层定义的规则集。

并非协议指定的每个元素都包含每一层。 此外，SIP 指定的元素是逻辑元素，而不是物理元素。 物理实现可以选择充当不同的逻辑元素，甚至可能在逐个事务的基础上。

SIP 的最低层是它的语法和编码。 其编码是使用增强的 Backus-Naur 形式语法 (BNF) 指定的。 完整的 BNF 在第 25 节中指定； 可以在第 7 节中找到 SIP 消息结构的概述。

第二层是传输层。 它定义了客户端如何发送请求和接收响应以及服务器如何通过网络接收请求和发送响应。 所有 SIP 元素都包含一个传输层。 传输层在第 18 节中描述。

第三层是交易层。 事务是 SIP 的基本组成部分。 事务是客户端事务（使用传输层）向服务器事务发送的请求，以及从服务器事务发送回客户端的对该请求的所有响应。 事务层处理应用层重传、对请求的响应匹配以及应用层超时。 用户代理客户端 (UAC) 完成的任何任务都使用一系列事务进行。 事务的讨论可以在第 17 节中找到。用户代理包含一个事务层，有状态代理也是如此。 无状态代理不包含事务层。 事务层有一个客户端组件（称为客户端事务）和一个服务器组件（称为服务器事务），每一个都由一个有限状态机表示，该状态机

被构造用于处理特定请求。

事务层之上的层称为事务用户（TU）。 除了无状态代理之外，每个 SIP 实体都是事务用户。当一个 TU 希望发送一个请求时，它会创建一个客户端事务实例并将请求连同目标 IP 地址、端口和传输请求一起传递给它。 创建客户端事务的 TU 也可以取消它。 当客户端取消事务时，它会请求服务器停止进一步处理，恢复到事务启动之前存在的状态，并对该事务生成特定的错误响应。 这是通过 CANCEL 请求完成的，该请求构成了它自己的事务，但引用了要取消的事务（第 9 节）。

SIP 元素，即用户代理客户端和服务器、无状态和有状态代理和注册器，包含将它们彼此区分开来的核心。 除了无状态代理之外，核心都是事务用户。 虽然 UAC 和 UAS 核心的行为取决于方法，但所有方法都有一些通用规则（第 8 节）。 对于 UAC，这些规则管理请求的构建； 对于 UAS，它们控制请求的处理和生成响应。 由于注册在 SIP 中扮演着重要角色，处理 REGISTER 的 UAS 被赋予特殊名称注册器。 第 10 节描述了 REGISTER 方法的 UAC 和 UAS 核心行为。 第 11 节描述了 OPTIONS 方法的 UAC 和 UAS 核心行为，用于确定 UA 的能力。

某些其他请求在对话中发送。对话是两个用户代理之间持续一段时间的对等 SIP 关系。该对话框有助于消息的排序和用户代理之间请求的正确路由。 INVITE 方法是本规范中定义的唯一建立对话的方法。 当 UAC 在对话上下文中发送请求时，它遵循第 8 节中讨论的通用 UAC 规则，但也遵循对话中间请求的规则。 第 12 节讨论了对话，并介绍了对话的构建和维护过程，以及在对话中构建请求。

SIP 中最重要的方法是 INVITE 方法，用于在参与者之间建立会话。 会话是参与者的集合，以及他们之间的媒体流，用于通信目的。 第 13 节讨论如何启动会话，从而产生一个或多个 SIP 对话。 第 14 节讨论了如何通过在对话中使用 INVITE 请求来修改该会话的特征。最后，第 15 节讨论了如何终止会话。

第 8、10、11、12、13、14 和 15 节的程序完全处理 UA 核心（第 9 节描述取消，适用于 UA 核心和代理核心）。 第 16 节讨论了代理元素，它有助于用户代理之间的消息路由。

6 定义

以下术语对 SIP 具有特殊意义。

记录地址：记录地址 (AOR) 是一个 SIP 或 SIPS URI，它指向具有位置服务的域，该位置服务可以将 URI 映射到用户可能可用的另一个 URI。 通常，位置服务是通过注册来填充的。AOR 通常被认为是用户的“公共地址”。

背靠背用户代理：背靠背用户代理 (B2BUA) 是一个逻辑实体，它接收请求并将其作为用户代理服务器 (UAS) 进行处理。 为了确定应如何回答请求，它充当用户代理客户端 (UAC) 并生成请求。 与代理服务器不同，它维护对话状态，并且必须参与在它已建立的对话上发送的所有请求。 由于它是 UAC 和 UAS 的串联，因此不需要对其行为进行明确定义。

呼叫：呼叫是一个非正式术语，指的是对等方之间的某种通信，通常是为了多媒体对话而建立的。

Call Leg：对话的另一个名称[31]： 本规范中不再使用。

呼叫有状态：如果代理保持从发起 INVITE 到终止 BYE 请求的对话状态，则它是呼叫有状态的。 调用状态代理始终是事务状态的，但反过来不一定是正确的。

客户端：客户端是发送 SIP 请求和接收 SIP 响应的任何网络元素。 客户可能会也可能不会直接与人类用户交互。 用户代理客户端和代理是客户端。

会议：包含多个参与者的多媒体会话（见下文）。

核心：核心指定特定类型的 SIP 实体特定的功能，即特定于有状态或无状态代理、用户代理或注册器。 除无状态代理之外的所有核心都是事务用户。

对话：对话是两个 UA 之间持续一段时间的对等 SIP 关系。 对话由 SIP 消息建立，例如对 INVITE 请求的 2xx 响应。 对话由呼叫标识符、本地标签和远程标签标识。 在 RFC 2543 中，对话以前称为呼叫支路。

下游：事务中消息转发的方向，指的是请求从用户代理客户端流向用户代理服务器的方向。

最终响应：终止 SIP 事务的响应，与不终止的临时响应相反。 所有 2xx、3xx、4xx、5xx 和 6xx 响应均为最终响应。

标头：标头是 SIP 消息的一个组件，用于传达有关消息的信息。 它被构造为一系列标题字段。

标头字段：标头字段是 SIP 消息标头的组成部分。 标题字段可以显示为一个或多个标题字段行。 标题字段行由标题字段名称和零个或多个标题字段值组成。 给定标题字段行上的多个标题字段值用逗号分隔。 某些标题字段只能有一个标题字段值，因此始终显示为单个标题字段行。

Header Field Value：一个 header 字段值是一个单一的值； 标头字段由零个或多个标头字段值组成。

归属域：为 SIP 用户提供服务的域。 通常，这是注册地址记录中 URI 中存在的域。

信息响应：与临时响应相同。

发起方、主叫方、主叫方：使用 INVITE 请求发起会话（和对话）的一方。 呼叫者从发送建立对话的初始 INVITE 到该对话终止时一直保留此角色。

邀请：一个邀请请求。

受邀者、受邀用户、被叫方、被叫方：接收 INVITE 请求以建立新会话的一方。 被叫方从收到 INVITE 到由该 INVITE 建立的对话终止时一直保留此角色。

位置服务：SIP 重定向或代理服务器使用位置服务来获取有关被叫方可能位置的信息。 它包含记录地址键与零个或多个联系人地址的绑定列表。可以通过多种方式创建和删除绑定；本规范定义了一个更新绑定的 REGISTER 方法。

循环：到达代理的请求，被转发，然后返回同一代理。 第二次到达时，它的 Request-URI 和第一次是一样的，其他影响代理操作的头域不变，这样代理就会对它第一次发出的请求做出同样的处理决策。 循环请求是错误，协议描述了检测和处理它们的过程。

松散路由：如果代理遵循本规范中定义的用于处理 Route 头字段的过程，则称它为松散路由。 这些过程将请求的目的地（存在于 Request-URI 中）与需要沿途访问的代理集（存在于 Route 头字段中）分开。 符合这些机制的代理也称为松散路由器。

消息：作为协议的一部分在 SIP 元素之间发送的数据。 SIP 消息是请求或响应。

方法：方法是请求在服务器上调用的主要功能。 该方法是在请求消息本身中携带的。 示例方法是 INVITE 和 BYE。

出站代理：从客户端接收请求的代理，即使它可能不是由 Request-URI 解析的服务器。 通常，UA 手动配置出站代理，或者可以通过自动配置协议了解一个。

并行搜索：在并行搜索中，代理在接收到传入请求时向可能的用户位置发出多个请求。 与在顺序搜索中发出一个请求然后在发出下一个请求之前等待最终响应不同，并行搜索在不等待先前请求的结果的情况下发出请求。

临时响应：服务器用来指示进度的响应，但不会终止 SIP 事务。 1xx 回复为临时回复，其他回复视为最终回复。

代理，代理服务器：作为服务器和客户端的中间实体，用于代表其他客户端发出请求。 代理服务器主要扮演路由的角色，这意味着它的工作是确保将请求发送到另一个“更接近”目标用户的实体。 代理对于执行策略也很有用（例如，确保允许用户拨打电话）。 代理在转发请求消息之前解释并在必要时重写请求消息的特定部分。

递归：当客户端对响应中的 Contact 标头字段中的一个或多个 URI 生成新请求时，客户端在 3xx 响应上递归。

重定向服务器：重定向服务器是一个用户代理服务器，它对收到的请求生成 3xx 响应，指导客户端联系一组备用 URI。

注册商：注册商是一个服务器，它接受注册请求并将在这些请求中接收到的信息放入它处理

的域的位置服务中。

常规事务：常规事务是具有除 INVITE、ACK 或 CANCEL 之外的方法的任何事务。

请求：从客户端发送到服务器的 SIP 消息，用于调用特定操作。

Response：从服务器发送到客户端的 SIP 消息，用于指示从客户端发送到服务器的请求的状态。

回铃音：回铃音是由主叫方应用程序产生的信号音，指示被叫方正在被提醒（振铃）。

路由集：路由集是有序 SIP 或 SIPS URI 的集合，它们表示发送特定请求时必须遍历的代理列表。 可以通过诸如 Record-Route 之类的标头来学习路由集，也可以对其进行配置。

服务器：服务器是一个网络元素，它接收请求以便为它们提供服务并将响应发送回这些请求。服务器的示例是代理、用户代理服务器、重定向服务器和注册商。

顺序搜索：在顺序搜索中，代理服务器按顺序尝试每个联系人地址，只有在前一个已生成最终响应后才继续下一个。 2xx 或 6xx 类最终响应总是终止顺序搜索。

会话：来自 SDP 规范："多媒体会话是一组多媒体发送者和接收者以及从发送者流向接收者的数据流。多媒体会议是多媒体会话的一个示例。"（RFC 2327 [1]）（为 SDP 定义的会话可以包含一个或多个 RTP 会话。）根据定义，可以通过不同的呼叫多次邀请被叫方加入同一会话。 如果使用 SDP，则会话由 SDP 用户名、会话 ID、网络类型、地址类型和源字段中的地址元素的串联定义。

SIP 事务：SIP 事务发生在客户端和服务器之间，包括从客户端发送到服务器的第一个请求到从服务器发送到客户端的最终（非 1xx）响应的所有消息。 如果请求是 INVITE 并且最终响应是非 2xx，则事务还包括对响应的 ACK。 对 INVITE 请求的 2xx 响应的 ACK 是一个单独的事务。

螺旋：螺旋是一个 SIP 请求，它被路由到代理，向前转发，然后再次到达该代理，但这次不同的方式会导致与原始请求不同的处理决策。 通常，这意味着请求的 Request-URI 与之前到达的不同。 与循环不同，螺旋不是错误条件。 一个典型的原因是呼叫转移。 用户调用 joe@example.com。 example.com 代理将其转发到 Joe 的 PC，而后者又将其转发到 bob@example.com。 此请求被代理回 example.com 代理。 然而，这不是一个循环。 由于请求针对的是不同的用户，因此它被认为是螺旋式的，并且是有效条件。

有状态代理：在处理请求期间维护本规范定义的客户端和服务器事务状态机的逻辑实体，也称为事务有状态代理。 第 16 节进一步定义了有状态代理的行为。（事务）有状态代理与调用有状态代理不同。

无状态代理：处理请求时不维护本规范中定义的客户端或服务器事务状态机的逻辑实体。无状态代理转发它在下游接收到的每个请求和它在上游接收到的每个响应。

严格路由：如果代理遵循 RFC 2543 的路由处理规则和该 RFC 的许多先前工作版本，则称其为严格路由。 当存在 Route 标头字段时，该规则导致代理破坏 Request-URI 的内容。 本规范中没有使用严格的路由行为，而是支持松散的路由行为。 执行严格路由的代理也称为严格路由器。

目标刷新请求：在对话中发送的目标刷新请求被定义为可以修改对话的远程目标的请求。

事务用户 (TU)：位于事务层之上的协议处理层。 事务用户包括 UAC 核心、UAS 核心和代理核心。

上游：事务中消息转发的方向，指的是响应从用户代理服务器流回用户代理客户端的方向。

URL 编码：根据 RFC 2396 第 2.4 节 [5] 编码的字符串。

用户代理客户端 (UAC)：用户代理客户端是一个逻辑实体，它创建一个新请求，然后使用客户端事务状态机制发送它。 UAC 的角色仅在该交易期间持续。 换句话说，如果一个软件发起一个请求，它会在该事务的持续时间内充当一个 UAC。 如果它稍后收到请求，它将承担用户代理服务器的角色来处理该事务。

UAC 核心：UAC 所需的一组处理功能，位于事务和传输层之上。

用户代理服务器 (UAS)：用户代理服务器是生成对 SIP 请求的响应的逻辑实体。 响应接受、拒绝或重定向请求。 此角色仅在该交易期间持续。 换句话说，如果一个软件响应了一个请求，它就会在该事务的持续时间内充当一个 UAS。 如果它稍后生成请求，它将承担用户代理客户端的角色来处理该事务。

UAS 核心：UAS 所需的一组处理功能，位于事务和传输层之上。

用户代理 (UA)：既可以作为用户代理客户端又可以作为用户代理服务器的逻辑实体。

UAC 和 UAS 以及代理和重定向服务器的角色是在逐个事务的基础上定义的。 例如，发起呼叫的用户代理在发送初始 INVITE 请求时充当 UAC，在接收到被叫方的 BYE 请求时充当 UAS。 类似地，同一软件可以充当一个请求的代理服务器和下一个请求的重定向服务器。

上面定义的代理、位置和注册服务器是逻辑实体； 实现可以将它们组合成一个应用程序。

7 SIP 消息

SIP 是基于文本的协议，使用 UTF-8 字符集 (RFC 2279 [7])。

SIP 消息要么是从客户端到服务器的请求，要么是从服务器到客户端的响应。

请求（第 7.1 节）和响应（第 7.2 节）消息都使用 RFC 2822 [3] 的基本格式，尽管语法在字符集和语法细节上有所不同。（例如，SIP 允许标头字段不是有效的 RFC 2822 标头字段。）这两种类型的消息都包含一个起始行、一个或多个标头字段、一个指示标头字段结束的空行和一个可选的 邮件正文。

```
generic-message  =  start-line
                    *message-header
                    CRLF
                    [ message-body ]
start-line       =  Request-Line / Status-Line
```

起始行、每个消息标题行和空行必须由回车换行序列 (CRLF) 终止。 请注意，即使没有消息正文，也必须存在空行。

除了上述字符集的不同之外，SIP 的大部分消息和头域语法与 HTTP/1.1 相同。 我们没有在这里重复语法和语义，而是使用 [HX.Y] 来引用当前 HTTP/1.1 规范（RFC 2616 [8]）的第 X.Y 节。

但是，SIP 不是 HTTP 的扩展。

7.1 请求

SIP 请求的区别在于起始行有一个 Request-Line。 Request-Line 包含方法名称、Request-URI 和由单个空格 (SP) 字符分隔的协议版本。

Request-Line 以 CRLF 结尾。 除行尾 CRLF 序列外，不允许使用 CR 或 LF。 任何元素中都不允许出现线性空格 (LWS)。

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

方法：本规范定义了六种方法：REGISTER 用于注册联系信息，INVITE、ACK 和 CANCEL 用于设置会话，BYE 用于终止会话，以及 OPTIONS 用于向服务器查询其能力。 标准跟踪 RFC 中记录的 SIP 扩展可以定义其他方法。

Request-URI：Request-URI 是第 19.1 节中描述的 SIP 或 SIPS URI 或通用 URI (RFC 2396 [5])。它指示此请求所针对的用户或服务。 Request-URI 不得包含未转义的空格或控制字符，并且不得包含在“<>”中。

SIP 元素可以支持具有“sip”和“sips”以外的方案的请求 URI，例如 RFC 2806 [9] 的“tel” URI 方案。 SIP 元素可以使用任何机制转换非 SIP URI，从而产生 SIP URI、SIPS URI 或其他一些方案。

SIP-Version：请求和响应消息都包含正在使用的 SIP 版本，并遵循[H3.1]（其中 HTTP 被 SIP 替换，HTTP/1.1 被 SIP/2.0 替换）关于版本排序、合规性要求和升级 的版本号。 为了符合本规范，发送 SIP 消息的应用程序必须包含“SIP/2.0”的 SIP 版本。 SIP-Version 字符串不区分大小写，但实现必须发送大写。

与 HTTP/1.1 不同，SIP 将版本号视为文字字符串。 在实践中，这应该没有什么区别。

7.2 回应

SIP 响应与请求的区别在于将状态行作为起始行。 状态行包含协议版本，后跟数字状态代码及其相关的文本短语，每个元素由单个 SP 字符分隔。

除了最后的 CRLF 序列外，不允许使用 CR 或 LF。

```
Status-Line  =  SIP-Version SP Status-Code SP Reason-Phrase CRLF
```

状态代码是一个 3 位整数结果代码，表示尝试理解和满足请求的结果。 原因短语旨在给出状态代码的简短文本描述。 Status-Code 是供自动机使用的，而 Reason-Phrase 是供人类用户使用的。 客户不需要检查或显示原因短语。

虽然本规范建议了原因短语的特定措辞，但实现可以选择其他文本，例如，在请求的 Accept-Language 标头字段中指示的语言中。

状态码的第一个数字定义了响应的类别。 最后两位数字没有任何分类作用。 因此，任何状态码在 100 到 199 之间的响应称为“1xx 响应”，任何状态码在 200 到 299 之间的响应称为“2xx 响应”，依此类推。 SIP/2.0 允许第一位数字有六个值：

1xx：临时——收到请求，继续处理请求；

2xx：成功——动作被成功接收、理解、接受；

3xx：重定向——为了完成请求需要采取进一步的行动；

4xx：客户端错误——请求包含错误的语法或无法在此服务器上完成；

5xx：服务器错误——服务器未能完成一个明显有效的请求；

6xx：全局失败——请求无法在任何服务器上完成。

第 21 节定义了这些类并描述了各个代码。

7.3 标题字段

SIP 标头字段在语法和语义上都类似于 HTTP 标头字段。 特别是，SIP 标头字段遵循 [H4.2] 消息标头语法定义以及将标头字段扩展到多行的规则。 但是，后者是在 HTTP 中指定的，带有隐式空格和折叠。 该规范符合 RFC 2234 [10] 并且仅使用显式空格和折叠作为语法的组成部分。

[H4.2]还规定，多个相同字段名且值为逗号分隔列表的头域可以合并为一个头域。 这也适用

于 SIP，但由于语法不同，具体规则也不同。 具体来说，任何 SIP 标头，其语法格式为：

```
header  =  "header-name" HCOLON header-value *(COMMA header-value)
```

允许将同名的标题字段组合成一个逗号分隔的列表。 Contact 头字段允许使用逗号分隔的列表，除非头字段值为“*”。

7.3.1 头域格式

标头字段遵循与 RFC 2822 [3] 的第 2.2 节中给出的相同的通用标头格式。每个标头字段由一个字段名称后跟一个冒号（“：”）和字段值组成。

```
field-name: field-value
```

第 25 节中指定的消息头的形式语法允许在冒号的任一侧有任意数量的空格； 但是，实现应避免在字段名称和冒号之间使用空格，并在冒号和字段值之间使用单个空格 (SP)。

```
Subject:            lunch
Subject      :      lunch
Subject            :lunch
Subject: lunch
```

因此，以上都是有效和等价的，但最后一种是首选形式。

标题字段可以通过在每个额外行之前至少一个 SP 或水平制表符 (HT) 来扩展多行。 下一行开头的换行符和空格被视为单个 SP 字符。 因此，以下是等价的：

```
Subject: I know you're there, pick up the phone and talk to me!
Subject: I know you're there,
         pick up the phone
         and talk to me!
```

具有不同字段名称的头字段的相对顺序并不重要。但是，建议代理处理所需的标头字段（例如，Via、Route、Record-Route、Proxy-Require、Max-Forwards 和 Proxy-Authorization）出现在消息的顶部，以便于快速解析.具有相同字段名称的标题字段行的相对顺序很重要。当且仅当该头字段的整个字段值被定义为逗号分隔列表时（即，如果遵循第 7.3 节中定义的语法），具有相同字段名称的多个头字段行可能出现在消息中.必须可以将多个标题字段行组合成一个“字段名称：字段值”对，而不改变消息的语义，方法是将每个后续字段值附加到第一个字段值，每个字段值用逗号分隔。此规则的例外是 WWW-Authenticate、Authorization、Proxy-Authenticate 和 Proxy-Authorization 头字段。具有这些名称的多个标题字段行可能出现在消息中，但由于它们的语法不遵循第 7.3 节中列出的一般形式，它们不得组合成单个标题字段行。

实现必须能够以每行单值或逗号分隔值形式的任意组合处理具有相同名称的多个标题字段行。

以下标题字段行组有效且等效：

```
Route: <sip:alice@atlanta.com>
Subject: Lunch
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>

Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>
Subject: Lunch

Subject: Lunch
Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>,
       <sip:carol@chicago.com>
```

以下每个块都是有效的，但不等同于其他块：

```
Route: <sip:alice@atlanta.com>
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>

Route: <sip:bob@biloxi.com>
Route: <sip:alice@atlanta.com>
Route: <sip:carol@chicago.com>

Route: <sip:alice@atlanta.com>,<sip:carol@chicago.com>,
       <sip:bob@biloxi.com>
```

标头字段值的格式是按标头名称定义的。 它总是要么是不透明的 TEXT-UTF8 八位字节序列，要么是空格、标记、分隔符和引号字符串的组合。 许多现有的标头字段将遵循值的一般形式，后跟以分号分隔的参数名称、参数值对序列：

```
field-name: field-value *(;parameter-name=parameter-value)
```

即使可以将任意数量的参数对附加到标头字段值，任何给定的参数名称也不得出现超过一次。

比较标头字段时，字段名称始终不区分大小写。 除非在特定头字段的定义中另有说明，否则字段值、参数名称和参数值不区分大小写。 标记始终不区分大小写。 除非另有说明，否则表示为带引号的字符串的值是区分大小写的。 例如，

```
Contact: <sip:alice@atlanta.com>;expires=3600
```

相当于

```
CONTACT: <sip:alice@atlanta.com>;ExPiReS=3600
```

和

```
Content-Disposition: session;handling=optional
```

相当于

```
content-disposition: Session;HANDLING=OPTIONAL
```

以下两个标头字段不等效：

```
Warning: 370 devnull "Choose a bigger pipe"
Warning: 370 devnull "CHOOSE A BIGGER PIPE"
```

7.3.2 头域分类

某些标头字段仅在请求或响应中才有意义。 这些分别称为请求头字段和响应头字段。 如果标题字段出现在与其类别不匹配的消息中（例如响应中的请求标题字段），则必须忽略它。 第 20 节定义了每个标题字段的分类。

7.3.3 紧凑形式

SIP 提供了一种以缩写形式表示公共头域名称的机制。 当消息变得太大而无法在可用的传输上传输（例如，在使用 UDP 时超过最大传输单元 (MTU)）时，这可能很有用。 这些紧凑形式在第 20 节中定义。在不改变消息语义的情况下，可以随时用紧凑形式替换头字段名称的较长形式。 标题字段名称可以在同一消息中以长格式和短格式出现。 实现必须接受每个标头名称的长格式和短格式。

7.4 消息体

请求，包括在本规范扩展中定义的新请求，可以包含消息体，除非另有说明。 正文的解释取决于请求方法。

对于响应消息，请求方法和响应状态码决定了任何消息体的类型和解释。 所有响应都可能包含正文。

7.4.1 消息体类型

消息体的 Internet 媒体类型必须由 Content-Type 头域给出。 如果正文经历了任何编码，例如压缩，那么这必须由 Content-Encoding 头字段指示；否则，必须省略 Content-Encoding。 如果适用，消息正文的字符集被指示为 Content-Type 标头字段值的一部分。

RFC 2046 [11] 中定义的“multipart” MIME 类型可以在消息正文中使用。 如果远程实现通过不包含多部分的 Accept 头字段请求，则发送包含多部分消息正文的请求的实现必须将会话描述作为非多部分消息正文发送。

SIP 消息可以包含二进制正文或正文部分。 当发送者没有提供明确的字符集参数时，“text”类型的媒体子类型被定义为具有“UTF-8”的默认字符集值。

7.4.2 消息体长度

以字节为单位的正文长度由 Content-Length 标头字段提供。 20.14 节详细描述了这个头域的必要内容。

HTTP/1.1 的“分块”传输编码绝不能用于 SIP。 （注意：分块编码会修改消息的主体，以便

将其作为一系列块传输，每个块都有自己的大小指示符。）

7.5 组帧 SIP 消息

与 HTTP 不同，SIP 实现可以使用 UDP 或其他不可靠的数据报协议。 每个这样的数据报都携带一个请求或响应。 请参阅第 18 节关于使用不可靠传输的限制。

通过面向流的传输处理 SIP 消息的实现必须忽略开始行 [H4.1] 之前出现的任何 CRLF。

Content-Length 标头字段值用于定位流中每个 SIP 消息的结尾。 当 SIP 消息通过面向流的传输方式发送时，它将始终存在。

8 一般用户代理行为

一个用户代理代表一个终端系统。 它包含一个生成请求的用户代理客户端 (UAC) 和一个响应请求的用户代理服务器 (UAS)。 UAC 能够根据一些外部刺激（用户单击按钮或 PSTN 线路上的信号）生成请求并处理响应。 UAS 能够接收请求并根据用户输入、外部刺激、程序执行的结果或某些其他机制生成响应。

当 UAC 发送请求时，请求会通过一些代理服务器，这些代理服务器将请求转发给 UAS。 当 UAS 生成响应时，该响应被转发给 UAC。

UAC 和 UAS 程序很大程度上取决于两个因素。 第一，基于请求或响应是在对话内部还是外部，第二，基于请求的方法。 对话在第 12 节中详细讨论； 它们代表用户代理之间的对等关系，并通过特定的 SIP 方法（例如 INVITE）建立。

在本节中，我们将讨论在处理对话之外的请求时 UAC 和 UAS 行为的独立于方法的规则。 当然，这包括本身建立对话的请求。

对话之外的请求和响应的安全程序在第 26 节中描述。具体来说，存在 UAS 和 UAC 相互验证的机制。 通过使用 S/MIME 对正文进行加密，还支持一组有限的隐私功能。

8.1 UAC 行为

本节介绍对话之外的 UAC 行为。

8.1.1 生成请求

由 UAC 制定的有效 SIP 请求必须至少包含以下头字段：To、From、CSeq、Call-ID、Max-Forwards 和 Via； 所有这些标头字段在所有 SIP 请求中都是必需的。 这六个标头字段是 SIP 消息的基本构建块，因为它们共同提供了大多数关键消息路由服务，包括消息寻

址、响应路由、限制消息传播、消息排序以及唯一标识 交易。 这些标头字段是对强制请求行的补充，其中包含方法、请求 URI 和 SIP 版本。

在对话之外发送的请求示例包括建立会话的邀请（第 13 节）和查询能力的选项（第 11 节）。

8.1.1.1 Request-URI

消息的初始 Request-URI 应该设置为 To 字段中 URI 的值。 一个值得注意的例外是 REGISTER 方法。 设置 REGISTER 的 Request-URI 的行为在第 10 节中给出。出于隐私原因或方便将这些字段设置为相同的值也可能是不可取的（尤其是如果始发 UA 期望 Request-URI 在传输过程中会被更改）。

在某些特殊情况下，预先存在的路由集的存在会影响消息的 Request-URI。 预先存在的路由集是一组有序的 URI，它们标识了一系列服务器，UAC 将向其发送对话之外的传出请求。 通常，它们由用户或服务提供商在 UA 上手动配置，或者通过其他一些非 SIP 机制配置。 当提供商希望为 UA 配置出站代理时，建议通过向其提供具有单个 URI 的预先存在的路由集（即出站代理的 URI）来完成此操作。

当存在预先存在的路由集时，必须遵循第 12.2.1.1 节中详述的填充 Request-URI 和 Route 头字段的过程（即使没有对话），使用所需的 Request-URI 作为远程目标 URI。

8.1.1.2 To

To 标头字段首先指定请求的所需“逻辑”接收者，或作为此请求目标的用户或资源的记录地址。 这可能是也可能不是请求的最终接收者。 To 头域可以包含一个 SIP 或 SIPS URI，但它也可以在适当的时候使用其他的 URI 方案（例如，tel URL (RFC 2806 [9])）。 所有 SIP 实现必须支持 SIP URI 方案。 任何支持 TLS 的实现都必须支持 SIPS URI 方案。 To 标题字段允许显示名称。

UAC 可以通过多种方式学习如何为特定请求填充 To 标头字段。通常，用户会通过人机界面建议 To 标头字段，可能是手动输入 URI 或从某种地址簿中选择它。通常，用户不会输入完整的 URI，而是输入一串数字或字母（例如，“bob”）。 UA 可以自行决定如何解释该输入。使用字符串构成 SIP URI 的用户部分意味着 UA 希望在域中将名称解析到 SIP URI 中 at-sign 的右侧 (RHS)（例如，sip:bob @example.com）。使用字符串形成 SIPS URI 的用户部分意味着 UA 希望安全地通信，并且名称将在域中解析为 at-sign 的 RHS。 RHS 通常是请求者的主域，它允许主域处理传出请求。这对于需要解释主域中的用户部分的“快速拨号”等功能很有用。当 UA 不希望指定应该解释用户输入的电话号码的域时，可以使用 tel URL。相反，请求通过的每个域都将获得该机会。例如，机场中的用户可能会通过机场中的出站代理登录并发送请求。如果他们输入“411”（这是美国本地查询服务的电话号码），则需要由机场的出站代理进行解释和处理，而不是用户的主域。在这种情况下，电话：411 将是正确的选择。

对话外的请求不得包含 To 标记； 请求的 To 字段中的标签标识对话的对等方。 由于没有

建立对话，因此不存在标签。

有关 To 标头字段的更多信息，请参阅第 20.39 节。 以下是有效 To 标头字段的示例：

```
To: Carol <sip:carol@chicago.com>
```

8.1.1.3 From

From 头域表示请求发起者的逻辑身份，可能是用户的记录地址。 与 To 标头字段一样，它包含一个 URI 和一个可选的显示名称。 SIP 元素使用它来确定将哪些处理规则应用于请求（例如，自动呼叫拒绝）。 因此，From URI 不包含运行 UA 的主机的 IP 地址或 FQDN 非常重要，因为这些不是逻辑名称。

From 标题字段允许显示名称。 如果要保持隐藏客户端的身份，UAC 应该使用显示名称“匿名”，以及语法正确但无意义的 URI（如 sip:thisis@anonymous.invalid）。

通常，在特定 UA 生成的请求中填充 From 标头字段的值是由用户或用户本地域的管理员预先配置的。 如果一个特定的 UA 被多个用户使用，它可能具有可切换的配置文件，其中包括一个与配置文件用户的身份相对应的 URI。 请求的接收者可以对请求的发起者进行身份验证，以确定他们是 From 头字段声称的身份（有关身份验证的更多信息，请参见第 22 节）。

From 字段必须包含一个新的“tag”参数，由 UAC 选择。 有关选择标签的详细信息，请参阅第 19.3 节。

有关 From 标头字段的更多信息，请参阅第 20.20 节。 例子：

```
From: "Bob" <sips:bob@biloxi.com> ;tag=a48s
From: sip:+12125551212@phone2net.com;tag=887s
From: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

8.1.1.4 Call-ID

Call-ID 标头字段充当唯一标识符，将一系列消息组合在一起。 对于任一 UA 在对话中发送的所有请求和响应，它必须相同。 在来自 UA 的每个注册中，它应该是相同的。

在由 UAC 在任何对话之外创建的新请求中，除非被特定方法的行为覆盖，否则 UAC 必须选择 Call-ID 头字段作为空间和时间上的全局唯一标识符。 所有 SIP UA 必须有一种方法来保证它们产生的 Call-ID 报头字段不会被任何其他 UA 无意中生成。 请注意，当请求在请求修改请求的某些失败响应（例如，验证挑战）之后重试时，这些重试请求不被视为新请求，因此不需要新的 Call-ID 头字段； 见第 8.1.3.5 节。

建议在生成呼叫 ID 时使用加密随机标识符 (RFC 1750 [12])。 实现可以使用“localid@host”的形式。 呼叫 ID 区分大小写，并且简单地逐字节进行比较。

使用加密随机标识符提供了一些防止会话劫持的保护，并减少了无意的 Call-ID 冲突的可能性。

选择请求的 Call-ID 标头字段值不需要配置或人机界面。

有关 Call-ID 标头字段的更多信息，请参阅第 20.8 节。

例子：

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

8.1.1.5 CSeq

CSeq 标头字段用作识别和订购交易的一种方式。 它由一个序列号和一个方法组成。 该方法必须与请求的方法相匹配。 对于对话之外的非注册请求，序列号值是任意的。 序列号值必须可以表示为 32 位无符号整数，并且必须小于 2**31。 只要遵循上述准则，客户端就可以使用任何它想要选择 Cseq 头字段值的机制。

第 12.2.1.1 节讨论了对话中请求的 Cseq 构造。

例子：

```
CSeq: 4711 INVITE
```

8.1.1.6 Max-Forwards

Max-Forwards 标头字段用于限制请求在到达其目的地的途中可以传输的跃点数。 它由一个整数组成，在每一跳减一。 如果在请求到达目的地之前 Max-Forwards 值达到 0，它将被拒绝并返回 483（Too Many Hops）错误响应。

UAC 必须在它发起的每个请求中插入一个 Max-Forwards 标头字段，其值应该是 70。这个数字被选择得足够大，以保证在没有循环时不会在任何 SIP 网络中丢弃请求， 但不会大到在发生循环时消耗代理资源。 应谨慎使用较低的值，并且仅在 UA 已知拓扑的网络中使用。

8.1.1.7 Via

Via 标头字段指示用于事务的传输，并标识要发送响应的位置。 只有在选择了将用于到达下一跳的传输之后才添加 Via 头字段值（这可能涉及使用 [4] 中的过程）。

当 UAC 创建一个请求时，它必须在该请求中插入一个 Via。 头字段中的协议名称和协议版本必须分别为 SIP 和 2.0。 Via 头域值必须包含一个分支参数。 此参数用于标识该请求创

建的事务。 客户端和服务器都使用此参数。

对于 UA 发送的所有请求，分支参数值必须在空间和时间上是唯一的。 此规则的例外是非 2xx 响应的 CANCEL 和 ACK。 如下所述，CANCEL 请求将具有与其取消的请求相同的分支参数值。如第 17.1.1.3 节所述，非 2xx 响应的 ACK 也将具有与它确认其响应的 INVITE 相同的分支 ID。

为了便于将其用作事务 ID，分支 ID 参数的唯一性属性不是 RFC 2543 的一部分。

由符合本规范的元素插入的分支 ID 必须始终以字符“z9hG4bK”开头。 这 7 个字符被用作魔术 cookie（7 个被认为足以确保旧的 RFC 2543 实现不会选择这样的值），以便接收请求的服务器可以确定分支 ID 是按照本文描述的方式构建的 规范（即全局唯一）。 除了这个要求之外，分支标记的精确格式是实现定义的。

Via 头 maddr、ttl 和 sent-by 组件将在传输层处理请求时设置（第 18 节）。

代理的通过处理在第 16.6 节第 8 节和第 16.7 节第 3 节中进行了描述。

8.1.1.8 Contact

Contact 标头字段提供了一个 SIP 或 SIPS URI，可用于联系该 UA 的特定实例以获取后续请求。 在任何可能导致对话建立的请求中，Contact 头域必须存在并且包含一个 SIP 或 SIPS URI。 对于本规范中定义的方法，仅包括 INVITE 请求。 对于这些请求，Contact 的范围是全球性的。也就是说，Contact 头域的值包含了 UA 想要接收请求的 URI，并且这个 URI 必须是有效的，即使在任何对话之外的后续请求中使用。

如果 Request-URI 或顶部 Route 头域的值包含一个 SIPS URI，Contact 头域也必须包含一个 SIPS URI。

有关 Contact 标头字段的更多信息，请参阅第 20.10 节。

8.1.1.9 支持和要求

如果 UAC 支持服务器可以应用到响应的 SIP 扩展，UAC 应该在请求中包含一个 Supported 头字段，列出这些扩展的选项标签（第 19.2 节）。

列出的选项标签必须仅引用标准跟踪 RFC 中定义的扩展。这是为了防止服务器坚持让客户端实现非标准的、供应商定义的功能来接收服务。 由实验性和信息性 RFC 定义的扩展被明确排除在请求中的 Supported 标头字段的使用中，因为它们也经常用于记录供应商定义的扩展。

如果 UAC 希望坚持 UAS 理解 UAC 将应用于请求以处理请求的扩展，它必须在请求中插入 Require 头字段，列出该扩展的选项标签。 如果 UAC 希望对请求应用扩展并坚持任何

被遍历的代理都理解该扩展，它必须在请求中插入一个 Proxy-Require 头字段，列出该扩展的选项标签。

与 Supported 头字段一样，Require 和 Proxy-Require 头字段中的选项标签必须仅引用标准跟踪 RFC 中定义的扩展。

8.1.1.10 附加消息组件

在创建新请求并正确构建上述标头字段后，将添加任何其他可选标头字段，以及特定于该方法的任何标头字段。

SIP 请求可以包含一个 MIME 编码的消息体。 无论请求包含何种类型的正文，都必须制定某些标头字段来表征正文的内容。 有关这些标头字段的更多信息，请参阅第 20.11 至 20.15 节。

8.1.2 发送请求

然后计算请求的目的地。 除非本地政策另有规定，否则目的地必须通过应用 [4] 中描述的 DNS 程序来确定，如下所示。 如果路由集中的第一个元素表示一个严格的路由器（导致形成第 12.2.1.1 节中描述的请求），则过程必须应用于请求的 Request-URI。 否则，过程将应用于请求中的第一个 Route 头字段值（如果存在），或者如果不存在 Route 头字段，则应用于请求的 Request-URI。 这些过程产生一组有序的地址、端口和传输尝试。 与 [4] 的过程中使用哪个 URI 作为输入无关，如果 Request-URI 指定一个 SIPS 资源，UAC 必须遵循 [4] 的过程，就好像输入 URI 是一个 SIPS URI。

本地策略可以指定要尝试的一组备用目的地。如果 Request-URI 包含 SIPS URI，则必须与 TLS 联系任何备用目的地。除此之外，如果请求不包含 Route 标头字段，则对备用目的地没有任何限制。这为预先存在的路由集提供了一种简单的替代方案，作为指定出站代理的一种方式。但是，不推荐使用这种配置出站代理的方法；应该使用带有单个 URI 的预先存在的路由集。如果请求包含 Route 头字段，则请求应该发送到从其最高值派生的位置，但可以发送到 UA 确定将遵守本文档中指定的 Route 和 Request-URI 策略的任何服务器（如与 RFC 2543 中的内容相反）。特别是，配置了出站代理的 UAC 应该尝试将请求发送到第一个 Route 头字段值中指示的位置，而不是采用将所有消息发送到出站代理的策略。

这确保了未添加 Record-Route 标头字段值的出站代理将退出后续请求的路径。它允许无法解析第一个 Route URI 的端点将该任务委托给出站代理。

UAC 应该遵循 [4] 中为有状态元素定义的过程，尝试每个地址，直到联系到服务器。 每个尝试都构成一个新事务，因此每个尝试都带有一个不同的最顶层 Via 头字段值和一个新的分支参数。 此外，Via 标头字段中的传输值设置为为目标服务器确定的任何传输。

8.1.3 处理响应

响应首先由传输层处理，然后传递到事务层。 事务层执行其处理，然后将响应传递给 TU。 TU 中的大多数响应处理都是特定于方法的。 但是，有一些与方法无关的一般行为。

8.1.3.1 事务层错误

在某些情况下，事务层返回的响应不会是 SIP 消息，而是事务层错误。 当从事务层收到超时错误时，必须将其视为已收到 408（请求超时）状态代码。 如果传输层报告了致命的传输错误（通常是由于 UDP 中的致命 ICMP 错误或 TCP 中的连接失败），则必须将条件视为 503（服务不可用）状态代码。

8.1.3.2 无法识别的响应

UAC 必须将它不能识别的任何最终响应视为等同于该类的 x00 响应代码，并且必须能够处理所有类的 x00 响应代码。 例如，如果 UAC 收到无法识别的响应代码 431，它可以安全地假定其请求有问题，并将响应视为收到 400（错误请求）响应代码。 UAC 必须将任何不同于 100 且它不能识别为 183（会话进度）的临时响应处理。 UAC 必须能够处理 100 和 183 响应。

8.1.3.3 Vias

如果响应中存在多个 Via 头字段值，则 UAC 应该丢弃该消息。

在请求的发起者之前存在额外的 Via 头字段值表明消息路由错误或可能已损坏。

8.1.3.4 处理 3xx 响应

在收到重定向响应（例如，301 响应状态码）后，客户端应该使用 Contact 头字段中的 URI 来根据重定向请求制定一个或多个新请求。这个过程类似于在 16.5 和 16.6 节中详述的 3xx 类响应上的代理递归过程。客户端从一个初始目标集开始，该目标集恰好包含一个 URI，即原始请求的 Request-URI。如果客户端希望基于对该请求的 3xx 类响应来制定新请求，它会将要尝试的 URI 放入目标集中。根据本规范中的限制，客户端可以选择将哪些联系人 URI 放入目标集中。与代理递归一样，处理 3xx 类响应的客户端不得多次将任何给定的 URI 添加到目标集。如果原始请求在 Request-URI 中有一个 SIPS URI，客户端可以选择递归到一个非 SIPS URI，但应该通知用户重定向到一个不安全的 URI。

任何新请求都可能收到包含原始 URI 作为联系人的 3xx 响应。 可以将两个位置配置为相互重定向。 将任何给定的 URI 放置在目标集中仅一次可防止无限重定向循环。

随着目标集的增长，客户端可以以任何顺序生成对 URI 的新请求。 一种常见的机制是根据 Contact 头字段值中的“q”参数值对集合进行排序。对 URI 的请求可以串行或并行生成。一种方法是串行处理递减的 q 值组，并并行处理每个 q 值组中的 URI。另一种方法是仅以 q 值递减顺序执行串行处理，在相等 q 值的触点之间任意选择。

如果联系列表中的地址导致失败，如下一段中所定义，则元素将移动到列表中的下一个地址，直到列表用尽。 如果列表用尽，则请求失败。

应通过故障响应代码（大于 399 的代码）检测故障； 对于网络错误，客户端事务将向事务用户报告任何传输层故障。 注意一些响应码（在 8.1.3.5 中有详细说明）表示请求可以重试；重新尝试的请求不应被视为失败。

当接收到特定联系地址的失败时，客户端应该尝试下一个联系地址。 这将涉及创建一个新的客户端事务来传递一个新的请求。

为了基于 3xx 响应中的联系地址创建请求，UAC 必须将整个 URI 从目标集复制到 Request-URI 中，“method-param”和“header”URI 参数除外（参见第 这些参数的定义见 19.1.1）。 它使用“标头”参数为新请求创建标头字段值，根据第 19.1.5 节中的指南覆盖与重定向请求关联的标头字段值。

请注意，在某些情况下，已在联系地址中传达的标头字段可能会附加到原始重定向请求中的现有请求标头字段。 作为一般规则，如果标头字段可以接受逗号分隔的值列表，则新的标头字段值可以附加到原始重定向请求中的任何现有值。 如果头域不接受多个值，原始重定向请求中的值可能会被联系地址中通信的头域值覆盖。 例如，如果返回具有以下值的联系人地址：

```
sip:user@host?Subject=foo&Call-Info=<http://www.foo.com>
```

然后原始重定向请求中的任何 Subject 标头字段都将被覆盖，但 HTTP URL 仅附加到任何现有的 Call-Info 标头字段值。

建议 UAC 重用在原始重定向请求中使用的相同 To、From 和 Call-ID，但 UAC 也可以选择更新新请求的 Call-ID 头字段值，例如。

最后，一旦构建了新请求，它就会使用新的客户端事务发送，因此必须在顶部 Via 字段中具有新的分支 ID，如第 8.1.1.7 节所述。

在所有其他方面，收到重定向响应后发送的请求应该重用原始请求的头字段和正文。

在某些情况下，Contact 标头字段值可能会根据收到的状态代码和到期间隔的存在临时或永久缓存在 UAC 中； 见第 21.3.2 和 21.3.3 节。

8.1.3.5 处理 4xx 响应

某些 4xx 响应代码需要特定的 UA 处理，与方法无关。

如果收到 401（未授权）或 407（需要代理身份验证）响应，UAC 应该遵循第 22.2 节和第 22.3 节的授权过程以使用凭据重试请求。

如果收到 413（请求实体太大）响应（第 21.4.11 节），则请求包含的正文比 UAS 愿意接受的要长。 如果可能，UAC 应该重试请求，要么省略主体，要么使用较小长度之一。

如果收到 415（不支持的媒体类型）响应（第 21.4.13 节），则请求包含 UAS 不支持的媒体类型。 UAC 应该重试发送请求，这一次只使用响应中 Accept 头字段中列出的类型、响应中 Accept-Encoding 头字段中列出的编码以及 Accept-Language 中列出的语言的内容。回复。

如果收到 416（不支持的 URI 方案）响应（第 21.4.14 节），则请求 URI 使用了服务器不支持的 URI 方案。 客户端应该重试请求，这一次，使用 SIP URI。

如果收到 420（错误扩展）响应（第 21.4.15 节），则请求包含一个 Require 或 Proxy-Require 标头字段，其中列出了代理或 UAS 不支持的功能的选项标签。 UAC 应该重试请求，这一次忽略响应中 Unsupported 标头字段中列出的任何扩展。

在上述所有情况下，通过创建具有适当修改的新请求来重试请求。 这个新请求构成一个新事务，并且应该具有与前一个请求相同的 Call-ID、To 和 From 值，但 Cseq 应该包含一个比前一个高一个的新序列号。

对于其他 4xx 响应，包括那些尚未定义的响应，重试可能会也可能不会，具体取决于方法和用例。

8.2 UAS 行为

当 UAS 处理对话之外的请求时，会遵循一组处理规则，与方法无关。 第 12 节给出了关于 UAS 如何判断请求是在对话内部还是对话之外的指导。

请注意，请求处理是原子的。 如果请求被接受，则必须执行与其相关的所有状态更改。 如果被拒绝，则不得执行所有状态更改。

UAS 应该按照本节后面的步骤顺序处理请求（即，从身份验证开始，然后检查方法、标头字段等，贯穿本节的其余部分）。

8.2.1 方法检查

一旦请求被认证（或认证被跳过），UAS 必须检查请求的方法。 如果 UAS 识别但不支持请求的方法，它必须生成 405（不允许方法）响应。 生成响应的程序在第 8.2.6 节中描述。 UAS 还必须在 405（方法不允许）响应中添加一个 Allow 头域。 Allow 头域必须列出生成消息的 UAS 支持的方法集。 Allow 标头字段在第 20.5 节中介绍。

如果该方法是服务器支持的方法，则继续处理。

8.2.2 头部检查

如果 UAS 不理解请求中的头字段（即，头字段未在本规范或任何支持的扩展中定义），服务器必须忽略该头字段并继续处理消息。 UAS 应该忽略处理请求所不需要的任何格式错误的标头字段。

8.2.2.1 To 和 Request-URI

To 标头字段标识由 From 字段中标识的用户指定的请求的原始接收者。由于呼叫转移或其他代理操作，原始接收者可能是也可能不是处理请求的 UAS。 当 To 头域不是 UAS 的身份时，UAS 可以应用它希望确定是否接受请求的任何策略。 但是，建议 UAS 接受请求，即使它们无法识别 To 标头字段中的 URI 方案（例如，tel: URI），或者如果 To 标头字段未解决此问题的已知或当前用户 无人机系统。 另一方面，如果 UAS 决定拒绝请求，它应该生成一个带有 403（禁止）状态码的响应，并将其传递给服务器事务以进行传输。

但是，Request-URI 标识要处理请求的 UAS。 如果 Request-URI 使用 UAS 不支持的方案，它应该以 416（不支持的 URI 方案）响应拒绝请求。 如果 Request-URI 没有标识 UAS 愿意接受请求的地址，它应该以 404（未找到）响应拒绝该请求。 通常，使用 REGISTER 方法将其记录地址绑定到特定联系地址的 UA 将看到 Request-URI 等于该联系地址的请求。 接收到的 Request-URI 的其他潜在来源包括 UA 发送的建立或刷新对话的请求和响应的 Contact 头字段。

8.2.2.2 合并请求

如果请求在 To 头域中没有标签，UAS 核心必须检查请求是否与正在进行的事务相匹配。 如果 From 标签、Call-ID 和 CSeq 与正在进行的事务相关联的完全匹配，但请求与该事务不匹配（基于第 17.2.3 节中的匹配规则），UAS 核心应该生成 482（循环 Detected) 响应并将其传递给服务器事务。

同一个请求不止一次到达 UAS，遵循不同的路径，很可能是由于分叉。 UAS 处理收到的第一个此类请求，并以 482（检测到循环）响应其余请求。

8.2.2.3 要求

假设 UAS 确定它是处理请求的正确元素，它会检查 Require 头字段（如果存在）。

UAC 使用 Require 头字段来告诉 UAS 关于 UAC 期望 UAS 支持以正确处理请求的 SIP 扩展。 其格式在第 20.32 节中描述。 如果 UAS 不理解 Require 头字段中列出的选项标签，它必须通过生成状态码 420（错误扩展）的响应来响应。UAS 必须添加一个 Unsupported 头字段，并在其中列出它不理解的那些在请求的 Require 头字段中的选项。

请注意，在 SIP CANCEL 请求或为非 2xx 响应发送的 ACK 请求中不得使用 Require 和 Proxy-Require。 如果这些请求中存在这些头字段，则必须忽略它们。

对 2xx 响应的 ACK 请求必须仅包含初始请求中存在的那些 Require 和 Proxy-Require 值。

例子：

```
UAC->UAS:   INVITE sip:watson@bell-telephone.com SIP/2.0
            Require: 100rel

UAS->UAC:   SIP/2.0 420 Bad Extension
            Unsupported: 100rel
```

此行为确保客户端-服务器交互在双方都理解所有选项时不会延迟进行，并且仅在不理解选项时减慢（如上例所示）。 对于匹配良好的客户端-服务器对，交互进行得很快，节省了协商机制通常需要的往返行程。 此外，当客户端需要服务器不理解的功能时，它还可以消除歧义。 某些功能，例如呼叫处理字段，仅对终端系统感兴趣。

8.2.3 内容处理

假设 UAS 了解客户端所需的任何扩展，UAS 会检查消息的正文以及描述它的标头字段。如果有任何 body 的类型（由 Content-Type 表示）、语言（由 Content-Language 表示）或编码（由 Content-Encoding 表示）不被理解，并且该 body 部分不是可选的（如 Content-Disposition 头域），UAS 必须以 415（不支持的媒体类型）响应拒绝请求。如果请求包含 UAS 不支持的类型的主体，响应必须包含一个 Accept 头字段，列出它理解的所有主体的类型。如果请求包含 UAS 不理解的内容编码，则响应必须包含一个 Accept-Encoding 头字段，列出 UAS 理解的编码。如果请求包含 UAS 不理解的语言的内容，则响应必须包含一个 Accept-Language 头字段，指示 UAS 可以理解的语言。除了这些检查之外，正文处理还取决于方法和类型。有关处理特定于内容的标头字段的更多信息，请参阅第 7.4 节以及第 20.11 至 20.15 节。

8.2.4 应用扩展

希望在生成响应时应用某些扩展的 UAS 不得这样做，除非在请求的 Supported 头字段中指示了对该扩展的支持。 如果不支持所需的扩展，服务器应该只依赖基线 SIP 和客户端支持

的任何其他扩展。 在极少数情况下，如果没有扩展，服务器无法处理请求，服务器可以发送 421（需要扩展）响应。 此响应表明如果不支持特定扩展，则无法生成正确的响应。 所需的扩展必须包含在响应的 Require 头字段中。 不推荐这种行为，因为它通常会破坏互操作性。

应用于非 421 响应的任何扩展必须列在响应中包含的 Require 头字段中。 当然，服务器不得应用未在请求的 Supported 标头字段中列出的扩展。 因此，响应中的 Require 标头字段将只包含标准跟踪 RFC 中定义的选项标签。

8.2.5 处理请求

假设前面小节中的所有检查都通过了，UAS 处理就变成了方法特定的。 第 10 节涵盖 REGISTER 请求，第 11 节涵盖 OPTIONS 请求，第 13 节涵盖 INVITE 请求，第 15 节涵盖 BYE 请求。

8.2.6 生成响应

当 UAS 希望构建对请求的响应时，它遵循以下小节中详述的一般程序。 可能还需要特定于相关响应代码的其他行为（本节未详细说明）。

一旦完成了与创建响应相关的所有过程，UAS 会将响应返回给它从其接收请求的服务器事务。

8.2.6.1 发送临时响应

用于生成响应的一个很大程度上非特定于方法的准则是 UAS 不应该为非邀请请求发出临时响应。 相反，UAS 应该尽快生成对非邀请请求的最终响应。

当生成 100（尝试）响应时，请求中存在的任何时间戳头字段必须复制到此 100（尝试）响应中。 如果生成响应有延迟，UAS 应该在响应中的 Timestamp 值中添加一个延迟值。 该值必须包含发送响应和接收请求之间的时间差，以秒为单位。

8.2.6.2 标题和标签

响应的 From 字段必须等于请求的 From 头字段。 响应的 Call-ID 头域必须等于请求的 Call-ID 头域。 响应的 CSeq 头域必须等于请求的 CSeq 域。 响应中的 Via 头域值必须等于请求中的 Via 头域值，并且必须保持相同的顺序。

如果请求在请求中包含 To 标记，则响应中的 To 标头字段必须等于请求的标头字段。 但是，如果请求中的 To 头域不包含标签，则响应中 To 头域中的 URI 必须等于 To 头域中的 URI； 此外，UAS 必须在响应的 To 头域中添加一个标签（除了 100（尝试）响应，其

中可能存在标签）。 这用于识别正在响应的 UAS，可能导致对话 ID 的组成部分。 相同的标签必须用于对该请求的所有响应，包括最终的和临时的（同样除了 100（尝试））。 生成标签的过程在第 19.3 节中定义。

8.2.7 无状态 UAS 行为

无状态 UAS 是不维护事务状态的 UAS。它正常回复请求，但在发送响应后丢弃通常由 UAS 保留的任何状态。 如果无状态 UAS 接收到请求的重传，它会重新生成响应并重新发送它，就像它正在回复请求的第一个实例一样。 如果请求相同，UAS 不能是无状态的，除非该方法的请求处理总是会产生相同的响应。 例如，这排除了无国籍注册商。 无状态 UAS 不使用事务层； 它们直接从传输层接收请求，并将响应直接发送到传输层。

无状态 UAS 角色主要用于处理发出质询响应的未经身份验证的请求。 如果未经身份验证的请求被有状态地处理，那么未经身份验证的请求的恶意泛滥可能会产生大量事务状态，这可能会减慢或完全停止 UAS 中的呼叫处理，从而有效地产生拒绝服务的情况； 有关详细信息，请参阅第 26.1.5 节。

无状态 UAS 最重要的行为如下：

o 无状态 UAS 不得发送临时 (1xx) 响应。

o 无状态的 UAS 不得重传响应。

o 无状态的 UAS 必须忽略 ACK 请求。

o 无状态的 UAS 必须忽略 CANCEL 请求。

o 必须以无状态的方式为响应生成标头标签 - 以一致的方式为相同的请求生成相同的标签。有关标签构造的信息，请参见第 19.3 节。

在所有其他方面，无状态 UAS 的行为方式与有状态 UAS 相同。 对于每个新请求，UAS 可以在有状态或无状态模式下运行。

8.3 重定向服务器

在某些架构中，可能希望通过依赖重定向来减少负责路由请求的代理服务器上的处理负载，并提高信令路径的稳健性。

重定向允许服务器在对客户端的响应中将请求的路由信息推回，从而使自己脱离此事务的进一步消息传递的循环，同时仍有助于定位请求的目标。 当请求的发起者收到重定向时，它将根据收到的 URI 发送一个新的请求。 通过将 URI 从网络的核心传播到其边缘，重定向允许相当大的网络可扩展性。

重定向服务器在逻辑上由服务器事务层和有权访问某种位置服务的事务用户组成(有关注册商和位置服务的更多信息，请参见第 10 节）。 该位置服务实际上是一个数据库，其中包含单个 URI 和一组一个或多个替代位置之间的映射，在这些位置可以找到该 URI 的目标。

重定向服务器不发出任何它自己的 SIP 请求。 在收到除 CANCEL 以外的请求后，服务器要么拒绝该请求，要么从位置服务收集替代位置列表并返回 3xx 类的最终响应。 对于格式良好的 CANCEL 请求，它应该返回 2xx 响应。 此响应结束 SIP 事务。 重定向服务器维护整个 SIP 事务的事务状态。 检测重定向服务器之间的转发循环是客户端的责任。

当重定向服务器对请求返回 3xx 响应时，它会将(一个或多个)替代位置列表填充到 Contact 标头字段中。 还可以提供联系人头字段值的“过期”参数来指示联系人数据的生命周期。

Contact 标头字段包含 URI，它们提供了新的位置或用户名来尝试，或者可以简单地指定额外的传输参数。 301（永久移动）或 302（临时移动）响应也可能给出初始请求所针对的相同位置和用户名，但指定其他传输参数，例如要尝试的不同服务器或多播地址，或更改 SIP 传输 从 UDP 到 TCP，反之亦然。

但是，重定向服务器不得将请求重定向到与 Request-URI 中的 URI 相同的 URI； 相反，如果 URI 不指向它自己，服务器可以将请求代理到目标 URI，或者可以用 404 拒绝它。

如果客户端正在使用出站代理，并且该代理实际上重定向请求，则可能会出现无限重定向循环。

请注意，Contact 标头字段值也可以引用与最初调用的资源不同的资源。例如，连接到 PSTN 网关的 SIP 呼叫可能需要传递特殊的信息通知，例如“您拨打的号码已更改”。

Contact 响应头字段可以包含任何合适的 URI，指示可以到达被叫方的位置，不限于 SIP URI。例如，它可以包含电话、传真或 irc（如果已定义）或 mailto: (RFC 2368 [32]) URL 的 URI。第 26.4.4 节讨论了将 SIPS URI 重定向到非 SIPS URI 的含义和限制。

Contact 标头字段值的“expires”参数指示 URI 的有效时间。 该参数的值是一个表示秒数的数字。 如果没有提供这个参数，Expires 头域的值决定了 URI 的有效期。 格式错误的值应该被视为等同于 3600。

这提供了与 RFC 2543 的适度向后兼容性，允许在此标头字段中使用绝对时间。 如果接收到绝对时间，将被视为格式错误，然后默认为 3600。

重定向服务器必须忽略无法理解的特性（包括无法识别的标头字段、Require 中的任何未知选项标签，甚至方法名称）并继续重定向相关请求。

9 取消请求

上一节讨论了为所有方法的请求生成请求和处理响应的一般 UA 行为。 在本节中，我们将

讨论一种通用方法，称为 CANCEL。

CANCEL 请求，顾名思义，用于取消客户端之前发送的请求。 具体来说，它要求 UAS 停止处理请求并对该请求生成错误响应。 CANCEL 对 UAS 已经给出最终响应的请求没有影响。因此，它对 CANCEL 请求最有用，因为它可能需要服务器很长时间才能响应。 因此，CANCEL 最适合 INVITE 请求，因为它可能需要很长时间才能生成响应。 在这种用法中，接收到 INVITE 的 CANCEL 请求但尚未发送最终响应的 UAS 将“停止振铃”，然后以特定的错误响应（487）响应 INVITE。

CANCEL 请求可以由代理和用户代理客户端构建和发送。第 15 节讨论了 UAC 在什么条件下会取消邀请请求，第 16.10 节讨论了 CANCEL 的代理使用。

有状态代理响应 CANCEL，而不是简单地转发将从下游元素接收到的响应。 出于这个原因，CANCEL 被称为“逐跳”请求，因为它在每个有状态代理跃点处得到响应。

9.1 客户行为

不应发送 CANCEL 请求来取消 INVITE 以外的请求。

由于 INVITE 以外的请求会立即得到响应，因此为非 INVITE 请求发送 CANCEL 总是会产生竞争条件。

以下过程用于构造 CANCEL 请求。 CANCEL 请求中的 Request-URI、Call-ID、To、CSeq 的数字部分和 From 头字段必须与被取消请求中的相同，包括标签。 由客户端构造的 CANCEL 必须只有一个 Via 头字段值与被取消请求中的顶部 Via 值匹配。对这些标头字段使用相同的值允许 CANCEL 与它取消的请求相匹配（第 9.2 节说明了这种匹配是如何发生的）。 但是，CSeq 头字段的方法部分必须具有 CANCEL 值。 这使得它可以作为交易本身被识别和处理（参见第 17 节）。

如果被取消的请求包含 Route 头字段，则 CANCEL 请求必须包含该 Route 头字段的值。

这是必要的，以便无状态代理能够正确路由 CANCEL 请求。

CANCEL 请求不得包含任何 Require 或 Proxy-Require 头字段。

一旦 CANCEL 被构造，客户端应该检查它是否已经收到任何响应（临时的或最终的）被取消的请求（这里称为“原始请求”）。

如果没有收到临时响应，则不得发送 CANCEL 请求； 相反，客户端必须在发送请求之前等待临时响应的到来。 如果原始请求已生成最终响应，则不应发送 CANCEL，因为它是有效的空操作，因为 CANCEL 对已经生成最终响应的请求没有影响。当客户端决定发送 CANCEL 时，它会为 CANCEL 创建一个客户端事务，并将 CANCEL 请求连同目标地址、端口和传输一起传递给它。 CANCEL 的目的地址、端口和传输必须与用于发送原始请求的相同。

如果允许在收到前一个请求的响应之前发送 CANCEL，则服务器可以在原始请求之前收到 CANCEL。

请注意，与原始请求对应的事务和 CANCEL 事务都将独立完成。 但是，取消请求的 UAC 不能依赖于接收原始请求的 487（请求终止）响应，因为符合 RFC 2543 的 UAS 不会生成这样的响应。 如果原始请求在 64*T1 秒内没有最终响应（T1 在第 17.1.1.1 节中定义），则客户端应考虑原始事务已取消并应销毁处理原始请求的客户端事务。

9.2 服务器行为

CANCEL 方法请求服务器端的 TU 取消一个挂起的事务。 TU 通过接受 CANCEL 请求来确定要取消的事务，然后假设请求方法不是 CANCEL 或 ACK，并应用第 17.2.3 节的事务匹配过程。 匹配交易是要取消的交易。

在服务器上对 CANCEL 请求的处理取决于服务器的类型。 无状态代理会转发它，有状态代理可能会响应它并生成它自己的一些 CANCEL 请求，而 UAS 会响应它。 有关 CANCEL 的代理处理，请参见第 16.10 节。

UAS 首先根据第 8.2 节中描述的通用 UAS 处理来处理 CANCEL 请求。但是，由于 CANCEL 请求是逐跳的并且无法重新提交，因此服务器无法对它们进行质询，以便在 Authorization 标头字段中获取正确的凭据。 另请注意，CANCEL 请求不包含 Require 标头字段。

如果 UAS 按照上述过程没有找到 CANCEL 的匹配交易，它应该以 481（Call Leg/Transaction doesn't Exist）响应 CANCEL。 如果原始请求的事务仍然存在，UAS 在接收到 CANCEL 请求时的行为取决于它是否已经发送了原始请求的最终响应。 如果有，CANCEL 请求对原始请求的处理没有影响，对任何会话状态没有影响，对原始请求生成的响应没有影响。 如果 UAS 尚未对原始请求发出最终响应，则其行为取决于原始请求的方法。 如果原始请求是 INVITE，UAS 应该立即用 487（请求终止）响应 INVITE。 CANCEL 请求对使用本规范中定义的任何其他方法处理事务没有影响。

无论原始请求的方法如何，只要 CANCEL 匹配现有事务，UAS 就会以 200（OK）响应自行回答 CANCEL 请求。 该响应是按照第 8.2.6 节中描述的过程构建的，注意对 CANCEL 的响应的 To 标记和对原始请求的响应中的 To 标记应该相同。 对 CANCEL 的响应被传递给服务器事务进行传输。

10 注册

10.1 概述

SIP 提供发现功能。 如果一个用户想要发起与另一个用户的会话，SIP 必须发现可以访问目

标用户的当前主机。 这个发现过程通常由诸如代理服务器和重定向服务器之类的 SIP 网络元素完成，它们负责接收请求，根据对用户位置的了解确定将其发送到哪里，然后将其发送到那里。 为此，SIP 网络元素咨询称为位置服务的抽象服务，该服务为特定域提供地址绑定。 这些地址绑定将传入的 SIP 或 SIPS URI（例如 sip:bob@biloxi.com）映射到一个或多个以某种方式“更接近”所需用户的 URI，例如 sip:bob@engineering.biloxi.com . 最终，代理将咨询定位服务，该服务将接收到的 URI 映射到所需接收者当前所在的用户代理。

注册在定位服务中为特定域创建绑定，将记录地址 URI 与一个或多个联系人地址相关联。因此，当该域的代理接收到其 Request-URI 与记录地址匹配的请求时，代理会将请求转发到注册到该记录地址的联系人地址。 通常，只有当对该记录地址的请求将被路由到该域时，在该域的位置服务中注册该记录地址才有意义。 在大多数情况下，这意味着注册的域需要与记录地址的 URI 中的域相匹配。

可以通过多种方式建立定位服务的内容。 一种方式是行政上的。 在上面的示例中，通过访问公司数据库，已知 Bob 是工程部门的成员。 但是，SIP 为 UA 提供了一种显式创建绑定的机制。 这种机制称为注册。

注册需要向称为注册商的特殊类型的 UAS 发送注册请求。 注册商充当域定位服务的前端，根据 REGISTER 请求的内容读取和写入映射。 然后通常由负责为该域路由请求的代理服务器咨询该位置服务。

图 2 给出了整个注册过程的说明。请注意，注册器和代理服务器是可以由网络中的单个设备扮演的逻辑角色； 为清楚起见，在本图中将两者分开。 另请注意，如果两者是独立的元素，UA 可能会通过代理服务器发送请求以到达注册商。

SIP 不要求使用特定的机制来实现位置服务。 唯一的要求是某个域的注册器必须能够读取和写入数据到位置服务，并且该域的代理或重定向服务器必须能够读取相同的数据。 注册商可能与同一域的特定 SIP 代理服务器位于同一位置。

10.2 构造注册请求

REGISTER 请求添加、删除和查询绑定。 REGISTER 请求可以在记录地址和一个或多个联系人地址之间添加新的绑定。 代表特定记录地址的注册可由适当授权的第三方执行。 客户端还可以删除以前的绑定或查询以确定当前为记录地址设置了哪些绑定。

除非另有说明，REGISTER 请求的构造和客户端发送 REGISTER 请求的行为与第 8.1 节和第 17.1 节中描述的一般 UAC 行为相同。

REGISTER 请求不会建立对话。 UAC 可以根据第 8.1 节中描述的预先存在的路由集在 REGISTER 请求中包含 Route 头字段。 Record-Route 头域在 REGISTER 请求或响应中没有意义，如果存在则必须被忽略。 特别是，UAC 绝不能基于对 REGISTER 请求的任何响应中是否存在 Record-Route 头字段来创建新的路由集。

以下头字段，除了联系人，必须包含在注册请求中。 可以包含一个 Contact 头域：

Request-URI：Request-URI 命名了要注册的位置服务的域（例如，“sip:chicago.com”）。 SIP URI 的“userinfo”和“@”组件不能出现。

To：To 头域包含要创建、查询或修改其注册的记录的地址。 To 标头字段和 Request-URI 字段通常不同，因为前者包含用户名。 此地址记录必须是 SIP URI 或 SIPS URI。

From：From 头字段包含负责注册的人员的记录地址。 除非请求是第三方注册，否则该值与 To 标头字段相同。

Call-ID：来自 UAC 的所有注册都应该使用相同的 Call-ID 头字段值，用于发送到特定注册商的注册。

如果同一个客户端使用不同的 Call-ID 值，注册器就无法检测到延迟的 REGISTER 请求是否可能乱序到达。

CSeq：CSeq 值保证 REGISTER 请求的正确排序。对于每个具有相同 Call-ID 的 REGISTER 请求，UA 必须将 CSeq 值加一。

Contact：注册请求可以包含一个联系人头字段，该字段具有零个或多个包含地址绑定的值。

UA 不得发送新的注册（即，包含新的 Contact 头字段值，而不是重传），直到它们收到注册商对前一个注册器的最终响应或前一个 REGISTER 请求已超时。

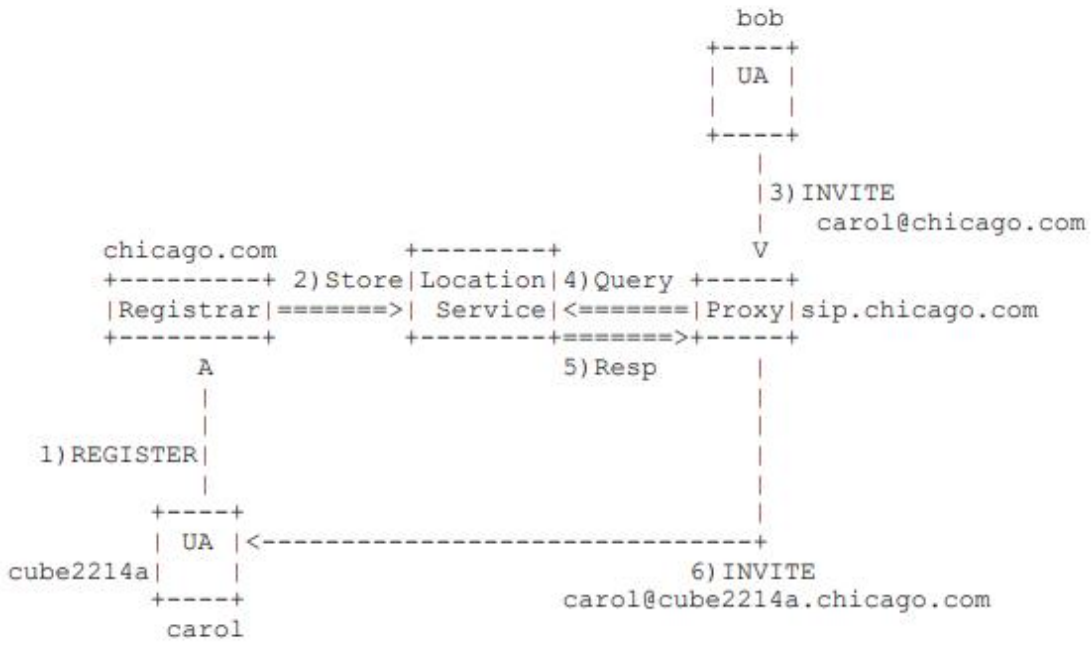

Figure 2: REGISTER example

以下 Contact 标头参数在 REGISTER 请求中具有特殊含义：

action：RFC 2543 中的“action”参数已被弃用。 UAC 不应该使用“action”参数。

expires：“expires”参数表示 UA 希望绑定有效的时间。 该值是一个表示秒数的数字。 如果未提供此参数，则使用 Expires 标头字段的值。 实现可以将大于 2**32-1（4294967295 秒或 136 年）的值视为等同于 2**32-1。 格式错误的值应该被视为等同于 3600。

10.2.1 添加绑定

发送给注册商的 REGISTER 请求包括联系人地址，SIP 请求记录地址应转发到该地址。 记录地址包含在 REGISTER 请求的 To 头字段中。

请求的 Contact 标头字段值通常由标识特定 SIP 端点的 SIP 或 SIPS URI（例如，“sip:carol@cube2214a.chicago.com”）组成，但它们可以使用任何 URI 方案。 例如，SIP UA 可以选择将电话号码（使用 tel URL，RFC 2806 [9]）或电子邮件地址（使用 mailto URL，RFC 2368 [32]）注册为记录地址的联系人。

例如，记录地址为“sip:carol@chicago.com”的 Carol 将在域 chicago.com 的 SIP 注册商处注册。 然后，chicago.com 域中的代理服务器将使用她的注册将 Carol 的记录地址的请求路由到她的 SIP 端点。

一旦客户端在注册商处建立了绑定，它可以根据需要发送包含新绑定或对现有绑定进行修改的后续注册。 对 REGISTER 请求的 2xx 响应将在 Contact 头字段中包含已在此注册器处为此记录地址注册的绑定的完整列表。

如果 REGISTER 请求的 To 头域中的记录地址是 SIPS URI，那么请求中的任何 Contact 头域值也应该是 SIPS URI。 只有当联系地址所代表的资源的安全性通过其他方式得到保证时，客户端才应在 SIPS 记录地址下注册非 SIPS URI。 这可能适用于调用除 SIP 之外的协议的 URI，或由除 TLS 之外的协议保护的 SIP 设备。

注册不需要更新所有绑定。 通常，UA 只更新自己的联系地址。

10.2.1.1 设置联系人地址过期时间

当客户端发送 REGISTER 请求时，它可能会建议一个过期间隔，指示客户端希望注册有效多长时间。 （如第 10.3 节所述，注册商根据其本地策略选择实际时间间隔。）

客户端可以通过两种方式建议绑定的到期间隔：通过 Expires 标头字段或“expires”Contact 标头参数。 当在单个 REGISTER 请求中给出多个绑定时，后者允许在每个绑定的基础上建议到期间隔，而前者建议所有不包含“expires”参数的 Contact 标头字段值的到期间隔。

如果 REGISTER 中没有表示建议的过期时间的机制，则客户端表明它希望服务器选择。

10.2.1.2 联系地址之间的偏好

如果在 REGISTER 请求中发送了多个联系人，则注册 UA 打算将这些联系人头字段值中的所有 URI 与 To 字段中存在的记录地址相关联。 可以使用 Contact 标头字段中的“q”参数对该列表进行优先级排序。 “q”参数表示与此地址记录的其他绑定相比，特定联系人头字段值的相对偏好。 第 16.6 节描述了代理服务器如何使用此首选项指示。

10.2.2 移除绑定

注册是软状态并过期，除非刷新，但也可以显式删除。 客户端可以尝试影响注册商选择的过期间隔，如第 10.2.1 节所述。 UA 通过在 REGISTER 请求中为该联系地址指定“0”的到期间隔来请求立即删除绑定。 UA 应该支持这种机制，以便绑定可以在它们的过期间隔过去之前被删除。

“*”的 REGISTER-specific Contact 头字段值适用于所有注册，但不得使用它，除非 Expires 头字段的值为“0”。

使用“*”Contact 标头字段值允许注册 UA 删除与记录地址关联的所有绑定，而无需知道它们的精确值。

10.2.3 获取绑定

对任何 REGISTER 请求的成功响应都包含现有绑定的完整列表，无论请求是否包含 Contact 标头字段。 如果 REGISTER 请求中不存在 Contact 标头字段，则绑定列表保持不变。

10.2.4 刷新绑定

每个 UA 负责刷新它之前建立的绑定。 UA 不应该刷新其他 UA 设置的绑定。

来自注册商的 200 (OK) 响应包含枚举所有当前绑定的联系人字段列表。 UA 使用第 19.1.4 节中的比较规则比较每个联系地址以查看它是否创建了联系地址。 如果是，它根据 expires 参数更新过期时间间隔，如果不存在，则根据 Expires 字段值更新过期时间间隔。 然后，UA 在到期间隔过去之前为其每个绑定发出 REGISTER 请求。 它可以将多个更新组合成一个 REGISTER 请求。

UA 应该在单个引导周期内对所有注册使用相同的 Call-ID。 注册刷新应该发送到与原始注册相同的网络地址，除非重定向。

10.2.5 设置内部时钟

如果 REGISTER 请求的响应包含 Date 头字段，客户端可以使用此头字段来了解当前时间，

以便设置任何内部时钟。

10.2.6 发现注册商

UA 可以使用三种方式来确定向其发送注册的地址：通过配置、使用记录地址和多播。 可以以超出本规范范围的方式为 UA 配置注册地址。 如果没有配置注册器地址，UA 应该使用记录地址的主机部分作为 Request-URI 并使用正常的 SIP 服务器定位机制 [4] 在那里处理请求。 例如，用户“sip:carol@chicago.com”的 UA 将 REGISTER 请求发送到“sip:chicago.com”。

最后，可以将 UA 配置为使用多播。 多播注册地址为众所周知的“所有 SIP 服务器”多播地址“sip.mcast.net”（IPv4 为 224.0.1.75）。 没有分配众所周知的 IPv6 多播地址； 这种分配将在需要时单独记录。 SIP UA 可以监听该地址并使用它来了解其他本地用户的位置（参见 [33]）； 但是，他们没有回应请求。

多播注册在某些环境中可能是不合适的，例如，如果多个企业共享同一个局域网。

10.2.7 发送请求

一旦构建了 REGISTER 方法并确定了消息的目的地，UAC 就会按照第 8.1.2 节中描述的过程将 REGISTER 移交给事务层。

如果事务层因为 REGISTER 未产生响应而返回超时错误，则 UAC 不应立即重新尝试向同一注册器注册。

立即重新尝试也可能超时。 为导致超时得到纠正的条件等待一些合理的时间间隔可以减少网络上不必要的负载。 没有规定具体的时间间隔。

10.2.8 错误响应

如果 UA 收到 423（Interval Too Brief）响应，它可以在使 REGISTER 请求中所有联系地址的过期间隔等于或大于 423（ 间隔太简短）响应。

10.3 处理 REGISTER 请求

注册商是一个 UAS，它响应 REGISTER 请求并维护其管理域内的代理服务器和重定向服务器可访问的绑定列表。注册商根据第 8.2 节和第 17.2 节处理请求，但它只接受 REGISTER 请求。 注册商不得生成 6xx 响应。

注册商可以酌情重定向 REGISTER 请求。 一种常见的用法是注册器在多播接口上侦听以将多播 REGISTER 请求重定向到其自己的单播接口，并带有 302（临时移动）响应。

如果 Record-Route 头域包含在 REGISTER 请求中，注册器必须忽略它。 注册商不得在对 REGISTER 请求的任何响应中包含 Record-Route 头字段。

注册商可能会收到一个请求，该请求遍历代理，该代理将 REGISTER 视为未知请求并添加了 Record-Route 头字段值。

注册商必须知道（例如，通过配置）它维护绑定的域集。 REGISTER 请求必须由注册商按照接收顺序进行处理。 REGISTER 请求也必须以原子方式处理，这意味着特定的 REGISTER 请求要么被完全处理，要么根本不被处理。 每个 REGISTER 消息必须独立于任何其他注册或绑定更改进行处理。

收到 REGISTER 请求时，注册商会执行以下步骤：

1. 注册商检查 Request-URI 以确定它是否有权访问 Request-URI 中标识的域的绑定。 如果不是，并且如果服务器还充当代理服务器，则服务器应该按照第 16 节中描述的代理消息的一般行为将请求转发到寻址域。

2. 为保证注册商支持任何必要的扩展，注册商必须处理 Require 头字段值，如第 8.2.2 节中针对 UAS 所述。

3. 注册商应该验证 UAC。 SIP 用户代理的身份验证机制在第 22 节中描述。注册行为绝不会覆盖 SIP 的通用身份验证框架。 如果没有可用的身份验证机制，注册商可以将 From 地址作为请求发起者的断言身份。

4. 注册商应该确定经过身份验证的用户是否有权修改此地址记录的注册。 例如，注册商可能会查阅授权数据库，该数据库将用户名映射到该用户有权修改绑定的记录地址列表。 如果经过身份验证的用户无权修改绑定，注册商必须返回 403（禁止）并跳过其余步骤。

在支持第三方注册的架构中，一个实体可能负责更新与多个记录地址相关的注册。

5. 注册器从请求的 To 头域中提取地址记录。 如果记录地址对请求 URI 中的域无效，注册商必须发送 404（未找到）响应并跳过其余步骤。 然后必须将 URI 转换为规范形式。 为此，必须删除所有 URI 参数（包括用户参数），并且必须将任何转义字符转换为其未转义形式。 结果用作绑定列表的索引。

6. 注册器检查请求中是否包含 Contact 头域。 如果没有，它会跳到最后一步。 如果存在 Contact 标头字段，则注册商检查是否存在一个包含特殊值“*”和 Expires 字段的 Contact 字段值。 如果请求有额外的联系人字段或过期时间不是零，则请求无效，服务器必须返回 400（无效请求）并跳过其余步骤。 如果不是，注册器检查呼叫 ID 是否与为每个绑定存储的值一致。 如果不是，它必须删除绑定。 如果它同意，它必须仅当请求中的 Cseq 高于为该绑定存储的值时才删除该绑定。 否则，更新必须中止并且请求失败。

7. 注册商现在依次处理联系人头字段中的每个联系人地址。 对于每个地址，它确定到期间隔如下：

- 如果字段值具有“expires”参数，则该值必须作为请求的过期时间。

- 如果没有这样的参数，但请求有一个 Expires 头字段，则该值必须作为请求的过期时间。

- 如果两者都没有，本地配置的默认值必须作为请求的过期时间。

注册商可以选择一个小于请求的过期间隔的过期时间。 当且仅当请求的到期间隔大于零且小于一小时且小于注册商配置的最小值时，注册商可以拒绝注册并返回 423（Interval Too Brief）响应。 此响应必须包含一个 Min-Expires 标头字段，该字段说明注册商愿意遵守的最小到期间隔。 然后它会跳过剩余的步骤。

允许注册商设置注册间隔可以保护它免受过度频繁的注册刷新，同时限制它需要维护的状态并降低注册过时的可能性。 注册的过期时间间隔经常用于创建服务。 一个示例是跟随我服务，其中用户可能只能在终端上短暂可用。 因此，注册商应接受简短的注册； 仅当间隔太短以至于刷新会降低注册服务商的性能时，才应拒绝请求。

对于每个地址，注册商然后使用 URI 比较规则搜索当前绑定列表。 如果绑定不存在，则暂时添加。 如果绑定确实存在，注册商会检查 Call-ID 值。 如果现有绑定中的 Call-ID 值与请求中的 Call-ID 值不同，则必须在到期时间为零时删除绑定，否则更新。 如果它们相同，则注册器将比较 CSeq 值。 如果该值高于现有绑定的值，则必须按上述方式更新或删除该绑定。 如果不是，更新必须中止并且请求失败。

该算法可确保忽略来自同一 UA 的无序请求。

每个绑定记录记录请求中的 Call-ID 和 Cseq 值。

当且仅当所有绑定更新和添加都成功时，必须提交绑定更新（即，使代理或重定向服务器可见）。 如果其中任何一个失败（例如，因为后端数据库提交失败），请求必须失败并返回 500（服务器错误）响应，并且必须删除所有暂定绑定更新。

8. 注册商返回 200（OK）响应。 响应必须包含枚举所有当前绑定的联系人头字段值。 每个联系人值必须具有一个“过期”参数，指示注册商选择的过期间隔。 响应应该包含一个 Date 头域。

11 查询能力

SIP 方法 OPTIONS 允许 UA 查询另一个 UA 或代理服务器的能力。 这允许客户端发现有关支持的方法、内容类型、扩展、编解码器等的信息，而无需“响铃”另一方。 例如，在客户端将 Require 头字段插入到列出不确定目标 UAS 支持的选项的 INVITE 之前，客户端可以使用 OPTIONS 查询目标 UAS 以查看此选项是否在 Supported 头字段中返回。 所有 UA 必须支持 OPTIONS 方法。

OPTIONS 请求的目标由 Request-URI 标识，它可以标识另一个 UA 或 SIP 服务器。 如果 OPTIONS 被寻址到代理服务器，则设置 Request-URI 时不包含用户部分，类似于为 REGISTER 请求设置 Request-URI 的方式。

或者，接收到 Max-Forwards 头字段值为 0 的 OPTIONS 请求的服务器可以响应该请求，而不管 Request-URI 是什么。

这种行为在 HTTP/1.1 中很常见。 此行为可用作“跟踪路由”功能，通过发送一系列具有递增 Max-Forwards 值的 OPTIONS 请求来检查各个跃点服务器的功能。

与一般 UA 行为的情况一样，如果 OPTIONS 没有产生响应，事务层可以返回超时错误。 这可能表明目标无法到达，因此不可用。

OPTIONS 请求可以作为已建立对话的一部分发送，以向对等方查询可以在对话稍后使用的能力。

11.1 OPTIONS 请求的构造

OPTIONS 请求是使用 SIP 请求的标准规则构建的，如第 8.1.1 节所述。

Contact 头域可能出现在 OPTIONS 中。

应该包含一个 Accept 头字段来指示 UAC 希望在响应中接收的消息正文的类型。 通常，这设置为用于描述 UA 的媒体功能的格式，例如 SDP (application/sdp)。

假设对 OPTIONS 请求的响应范围为原始请求中的 Request-URI。但是，只有当 OPTIONS 作为已建立对话的一部分发送时，才能保证生成 OPTIONS 响应的服务器将接收到未来的请求。

示例 OPTIONS 请求：

```
OPTIONS sip:carol@chicago.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
Max-Forwards: 70
To: <sip:carol@chicago.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:alice@pc33.atlanta.com>
Accept: application/sdp
Content-Length: 0
```

11.2 OPTIONS 请求的处理

对 OPTIONS 的响应是使用第 8.2.6 节中讨论的 SIP 响应的标准规则构建的。 选择的响应代码必须与如果请求是 INVITE 时选择的响应代码相同。 也就是说，如果 UAS 准备好接受呼叫，将返回 200（OK），如果 UAS 忙，将返回 486（Busy Here）等。这允许使用 OPTIONS

请求来确定基本 UAS 的状态，它可以指示 UAS 是否会接受 INVITE 请求。

在对话中接收到的 OPTIONS 请求会生成一个 200（OK）响应，该响应与在对话外部构建的响应相同，并且对对话没有任何影响。

由于 OPTIONS 和 INVITE 请求的代理处理不同，因此使用 OPTIONS 具有局限性。 虽然分叉的 INVITE 可能会导致返回多个 200（OK）响应，但分叉的 OPTIONS 只会导致单个 200（OK）响应，因为它是由使用非 INVITE 处理的代理处理的。 有关规范性细节，请参见第 16.7 节。

如果对 OPTIONS 的响应由代理服务器生成，则代理返回 200（OK），列出服务器的功能。响应不包含消息正文。

Allow、Accept、Accept-Encoding、Accept-Language 和 Supported 头字段应该出现在对 OPTIONS 请求的 200（OK）响应中。 如果响应是由代理生成的，则应该省略 Allow 头字段，因为它是模棱两可的，因为代理是方法不可知的。 联系头字段可能出现在 200（OK）响应中，并且与 3xx 响应具有相同的语义。 也就是说，他们可能会列出一组替代名称和联系用户的方法。 可能存在警告头字段。

可以发送消息体，其类型由 OPTIONS 请求中的 Accept 头字段确定（如果 Accept 头字段不存在，则 application/sdp 是默认值）。 如果类型包括可以描述媒体功能的类型，UAS 应该在响应中包含一个主体用于该目的。 在 [13] 中描述了在 application/sdp 的情况下构建这种主体的详细信息。

由 UAS 生成的示例 OPTIONS 响应（对应于第 11.1 节中的请求）：

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
 ;received=192.0.2.4
To: <sip:carol@chicago.com>;tag=93810874
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:carol@chicago.com>
Contact: <mailto:carol@chicago.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Accept: application/sdp
Accept-Encoding: gzip
Accept-Language: en
Supported: foo
Content-Type: application/sdp
Content-Length: 274

(SDP not shown)
```

12 对话

用户代理的一个关键概念是对话。 对话代表两个用户代理之间持续一段时间的对等 SIP 关系。 该对话有助于用户代理之间的消息排序以及它们之间的请求的正确路由。 对话表示解

释 SIP 消息的上下文。 第 8 节讨论了对话外请求和响应的独立于方法的 UA 处理。 本节讨论如何使用这些请求和响应来构建对话，以及如何在对话中发送后续请求和响应。

每个 UA 都使用一个对话 ID 来标识一个对话，该对话 ID 由一个 Call-ID 值、一个本地标记和一个远程标记组成。 对话中涉及的每个 UA 的对话 ID 都不相同。 具体而言，一个 UA 的本地标签与对端 UA 的远程标签相同。 标签是不透明的标记，有助于生成唯一的对话 ID。

对话 ID 还与所有响应以及在“收件人”字段中包含标签的任何请求相关联。 计算消息的对话 ID 的规则取决于 SIP 元素是 UAC 还是 UAS。 对于 UAC，对话 ID 的 Call-ID 值设置为消息的 Call-ID，远程标签设置为消息 To 字段中的标签，本地标签设置为消息中的标签 消息的 From 字段（这些规则适用于请求和响应）。 正如人们对 UAS 所期望的那样，对话 ID 的 Call-ID 值设置为消息的 Call-ID，远程标记设置为消息的 From 字段中的标记，本地标记设置为 到消息的收件人字段中的标签。

对话包含对话中进一步消息传输所需的某些状态。 此状态由对话 ID、本地序列号（用于将请求从 UA 到其对等方的顺序）、远程序列号（用于从其对等方到 UA 的请求排序）、本地 URI、远程 URI、 远程目标、一个称为“安全”的布尔标志和一个路由集，它是一个有序的 URI 列表。路由集是向对等方发送请求需要遍历的服务器列表。对话也可以处于“早期”状态，当它使用临时响应创建时会发生这种情况，然后在 2xx 最终响应到达时转换到“已确认”状态。 对于其他响应，或者如果该对话根本没有响应，则早期对话终止。

12.1 创建对话

对话是通过使用特定方法生成对请求的非失败响应来创建的。在本规范中，只有带有 To 标签的 2xx 和 101-199 响应（请求为 INVITE）会建立对话。 由对请求的非最终响应建立的对话处于“早期”状态，称为早期对话。 扩展可以定义创建对话框的其他方式。 第 13 节给出了特定于 INVITE 方法的更多细节。 在这里，我们描述了创建不依赖于方法的对话状态的过程。

UA 必须为对话 ID 组件分配值，如下所述。

12.1.1 UAS 行为

当 UAS 使用建立对话的响应响应请求时（例如 2xx 到 INVITE），UAS 必须将请求中的所有 Record-Route 头字段值复制到响应中（包括 URI、URI 参数和任何 Record-Route 头域参数，无论它们是 UAS 已知的还是未知的）并且必须保持这些值的顺序。 UAS 必须在响应中添加一个 Contact 头域。 Contact 头域包含一个地址，UAS 希望在对话中的后续请求被联系到该地址（在 INVITE 的情况下，它包括 2xx 响应的 ACK）。通常，此 URI 的主机部分是主机的 IP 地址或 FQDN。 Contact 头域中提供的 URI 必须是 SIP 或 SIPS URI。如果发起对话的请求在 Request-URI 或顶部 Record-Route 标头字段值中包含 SIPS URI（如果有）或 Contact 标头字段（如果没有 Record-Route 标头字段），则 Contact 标头响应中的字段必须是一个 SIPS URI。 URI 应该具有全局范围（也就是说，相同的 URI 可以在此对话框之

外的消息中使用）。同样的，INVITE 的 Contact 头域中的 URI 的范围也不限于这个对话。因此，即使在此对话之外，它也可以用于发送给 UAC 的消息中。

UAS 然后构建对话的状态。 此状态必须在对话期间保持。

如果请求通过 TLS 到达，并且 Request-URI 包含 SIPS URI，则“安全”标志设置为 TRUE。

路由集必须设置为请求中 Record-Route 头字段中的 URI 列表，按顺序获取并保留所有 URI 参数。 如果请求中不存在 Record-Route 头字段，则路由集必须设置为空集。 此路由集即使为空，也会覆盖此对话框中未来请求的任何预先存在的路由集。 远程目标必须设置为来自请求的 Contact 头字段的 URI。

远程序列号必须设置为请求的 Cseq 头字段中的序列号值。本地序列号必须为空。对话 ID 的呼叫标识符组件必须设置为请求中 Call-ID 的值。对话 ID 的本地标记组件必须设置为请求响应中 To 字段中的标记（始终包含一个标记），并且对话 ID 的远程标记组件必须设置为来自 From 请求中的字段。 UAS 必须准备好接收在 From 字段中没有标签的请求，在这种情况下，标签被认为具有 null 值。

这是为了保持与 RFC 2543 的向后兼容性，RFC 2543 没有强制使用 From 标签。

远程 URI 必须设置为 From 字段中的 URI，本地 URI 必须设置为 To 字段中的 URI。

12.1.2 UAC 行为

当 UAC 发送可以建立对话的请求（例如 INVITE）时，它必须在 Contact 头字段中提供具有全局范围的 SIP 或 SIPS URI（即，可以在此对话之外的消息中使用相同的 SIP URI） 请求。如果请求具有 Request-URI 或带有 SIPS URI 的最顶层 Route 头字段值，则 Contact 头字段必须包含 SIPS URI。

当 UAC 接收到建立对话的响应时，它会构建对话的状态。 此状态必须在对话期间保持。

如果请求是通过 TLS 发送的，并且 Request-URI 包含 SIPS URI，则“安全”标志设置为 TRUE。

路由集必须设置为响应中 Record-Route 头字段中的 URI 列表，以相反的顺序获取并保留所有 URI 参数。 如果响应中不存在 Record-Route 头字段，则路由集必须设置为空集。 此路由集即使为空，也会覆盖此对话框中未来请求的任何预先存在的路由集。 远程目标必须设置为响应的 Contact 头字段中的 URI。

本地序列号必须设置为请求的 Cseq 头字段中的序列号值。 远程序列号必须为空（当远程 UA 在对话中发送请求时建立）。 对话 ID 的呼叫标识符组件必须设置为请求中 Call-ID 的值。 对话 ID 的本地标记组件必须设置为请求中 From 字段中的标记，并且对话 ID 的远程标记组件必须设置为响应的 To 字段中的标记。 UAC 必须准备好接收在 To 字段中没有标签的响应，在这种情况下，标签被认为具有 null 值。

这是为了保持与 RFC 2543 的向后兼容性，RFC 2543 没有强制使用 To 标签。

远程 URI 必须设置为 To 字段中的 URI，本地 URI 必须设置为 From 字段中的 URI。

12.2 对话中的请求

一旦在两个 UA 之间建立了对话，它们中的任何一个都可以根据需要在对话中启动新事务。发送请求的 UA 将担任交易的 UAC 角色。 收到请求的 UA 将扮演 UAS 角色。 请注意，这些角色可能与在建立对话的事务期间持有的 UA 不同。

对话中的请求可能包含 Record-Route 和 Contact 头字段。 但是，这些请求不会导致对话的路由集被修改，尽管它们可能会修改远程目标 URI。 具体来说，不是目标刷新请求的请求不会修改对话的远程目标 URI，而目标刷新请求的请求会。 对于使用 INVITE 建立的对话，唯一定义的目标刷新请求是 re-INVITE（参见第 14 节）。 其他扩展可以为以其他方式建立的对话定义不同的目标刷新请求。

请注意，ACK 不是目标刷新请求。

目标刷新请求仅更新对话的远程目标 URI，而不是由 Record-Route 形成的路由集。 更新后者将引入与 RFC 2543 兼容系统的严重向后兼容性问题。

12.2.1 UAC 行为

12.2.1.1 生成请求

对话中的请求是通过使用存储为对话一部分的状态的许多组件来构造的。

请求的 To 字段中的 URI 必须设置为来自对话状态的远程 URI。 请求的 To 头域中的标签必须设置为对话 ID 的远程标签。 请求的 From URI 必须设置为对话状态的本地 URI。 请求的 From 头域中的标签必须设置为对话 ID 的本地标签。如果远程或本地标记的值为空，则标记参数必须分别从 To 或 From 头字段中省略。

在后续请求中使用原始请求中的 To 和 From 字段中的 URI 是为了与 RFC 2543 向后兼容，RFC 2543 使用 URI 进行对话识别。 在本规范中，只有标签用于对话识别。 预计在本规范的后续修订版中将不推荐在中间对话请求中强制反映原始 To 和 From URI。

请求的 Call-ID 必须设置为对话的 Call-ID。 对话中的请求必须在每个方向上包含严格单调递增和连续的 Cseq 序列号（加一）（当然除了 ACK 和 CANCEL，它们的编号等于被确认或取消的请求）。 因此，如果本地序列号不为空，则本地序列号的值必须加一，并且这个值必须放在 Cseq 头域中。 如果本地序列号为空，则必须使用第 8.1.1.5 节的指南选择初始值。 CSeq 头域值中的方法域必须与请求的方法相匹配。

使用 32 位的长度，客户端可以在一次调用中每秒生成一个请求，持续大约 136 年，然后才需要回绕。 选择序列号的初始值，以便同一调用中的后续请求不会回绕。 非零初始值允许客户端使用基于时间的初始序列号。 例如，客户端可以选择 32 位秒时钟的 31 个最高有效位作为初始序列号。

UAC 使用远程目标和路由集来构建请求的 Request-URI 和 Route 头字段。

如果路由集为空，UAC 必须将远程目标 URI 放入 Request-URI。 UAC 不得在请求中添加 Route 头字段。

如果路由集不为空，并且路由集中的第一个 URI 包含 lr 参数（参见第 19.1.1 节），UAC 必须将远程目标 URI 放入 Request-URI 并且必须包含一个包含 route 按顺序设置值，包括所有参数。

如果路由集不为空，并且它的第一个 URI 不包含 lr 参数，则 UAC 必须将路由集中的第一个 URI 放入 Request-URI，剥离 Request-URI 中不允许的任何参数。 UAC 必须添加一个 Route 头字段，其中包含按顺序排列的其余路由集值，包括所有参数。 然后，UAC 必须将远程目标 URI 作为最后一个值放入 Route 头字段中。

例如，如果远程目标是 sip:user@remoteua 并且路由集包含：

```
<sip:proxy1>,<sip:proxy2>,<sip:proxy3;lr>,<sip:proxy4>
```

请求将由以下 Request-URI 和 Route 头字段组成：

```
METHOD sip:proxy1
Route: <sip:proxy2>,<sip:proxy3;lr>,<sip:proxy4>,<sip:user@remoteua>
```

如果路由集的第一个 URI 不包含 lr 参数，则指示的代理不理解本文档中描述的路由机制，并将按照 RFC 2543 中的规定行事，将 Request-URI 替换为第一个 Route 头字段值 转发消息时接收。 将 Request-URI 放在 Route 标头字段的末尾可以通过严格路由器保留该 Request-URI 中的信息（当请求到达松散路由器时，它将返回到 Request-URI）。

UAC 应该在对话中的任何目标刷新请求中包含一个 Contact 头字段，除非需要更改它，否则 URI 应该与对话中先前请求中使用的相同。如果“安全”标志为真，则该 URI 必须是 SIPS URI。 如第 12.2.2 节所述，目标刷新请求中的 Contact 标头字段会更新远程目标 URI。 如果在对话期间其地址发生变化，这允许 UA 提供新的联系地址。

但是，不是目标刷新请求的请求不会影响对话的远程目标 URI。

请求的其余部分如第 8.1.1 节所述形成。

一旦构建了请求，服务器的地址就会被计算出来并发送请求，对对话之外的请求使用相同的过程（第 8.1.2 节）。

8.1.2 节中的过程通常会导致请求被发送到由最顶层 Route 头域值或 Request-URI 指示的地址，如果没有 Route 头域存在的话。 受限于某些限制，它们允许将请求发送到备用地址（例如路由集中未表示的默认出站代理）。

12.2.1.2 处理响应

UAC 将从事务层接收对请求的响应。 如果客户端事务返回超时，这将被视为 408（请求超时）响应。

对于在对话中发送的请求接收 3xx 响应的 UAC 的行为与在对话外发送的请求相同。 此行为在第 8.1.3.4 节中描述。

但是请注意，当 UAC 尝试替代位置时，它仍然使用为对话设置的路由来构建请求的 Route 标头。

当 UAC 收到对目标刷新请求的 2xx 响应时，它必须将对话的远程目标 URI 替换为该响应中 Contact 头字段中的 URI（如果存在）。

如果对话中请求的响应是 481（呼叫/事务不存在）或 408（请求超时），UAC 应该终止对话。 如果根本没有收到请求的响应，UAC 也应该终止对话（客户端事务将通知 TU 超时。）

对于 INVITE 发起的对话，终止对话包括发送 BYE。

12.2.2 UAS 行为

与任何其他请求一样，在对话中发送的请求是原子的。 如果 UAS 接受特定请求，则执行与其相关的所有状态更改。 如果请求被拒绝，则不会执行任何状态更改。

请注意，某些请求（例如 INVITE）会影响多个状态。

UAS 将接收来自事务层的请求。 如果请求在 To 标头字段中有标签，UAS 核心计算与请求对应的对话标识符，并将其与现有对话进行比较。 如果有匹配，这是一个中间对话请求。 在这种情况下，UAS 首先对对话之外的请求应用相同的处理规则，这将在第 8.2 节中讨论。

如果请求在 To 标头字段中有标签，但对话标识符与任何现有对话不匹配，则 UAS 可能已经崩溃并重新启动，或者它可能收到了对不同（可能失败）UAS 的请求（UAS 可以 构造 To 标记，以便 UAS 可以识别该标记用于为其提供恢复的 UAS）。 另一种可能性是传入的请求只是被错误路由了。 基于 To 标签，UAS 可以接受或拒绝请求。 接受对可接受的 To 标记的请求提供了稳健性，因此即使在崩溃时对话也可以持续存在。 希望支持此功能的 UA 必须考虑一些问题，例如即使在重新启动时也选择单调增加的 Cseq 序列号、重建路由集以及接受超出范围的 RTP 时间戳和序列号。

如果 UAS 由于不希望重新创建对话而希望拒绝请求，它必须以 481（呼叫/事务不存在）状态码响应请求并将其传递给服务器事务。

可以在对话中接收不会以任何方式改变对话状态的请求（例如，OPTIONS 请求）。 就像在对话之外收到它们一样处理它们。

如果远程序列号为空，则必须将其设置为请求中 Cseq 头字段值中的序列号值。 如果远程序列号不为空，但请求的序列号低于远程序列号，则请求是无序的，必须以 500（服务器内部错误）响应拒绝。 如果远程序号不为空，并且请求的序号大于远程序号，则请求是有序的。 Cseq 序列号可能比远程序列号高一个以上。 这不是错误情况，UAS 应该准备好接收和处理 CSeq 值比先前接收到的请求高一个以上的请求。 然后，UAS 必须将远程序列号设置为请求中 Cseq 头字段值中的序列号值。

如果代理质疑 UAC 生成的请求，则 UAC 必须重新提交带有凭据的请求。 重新提交的请求将有一个新的 Cseq 编号。 UAS 永远不会看到第一个请求，因此，它会注意到 Cseq 编号空间中的间隙。 这样的差距并不代表任何错误情况。

当 UAS 接收到目标刷新请求时，它必须将对话的远程目标 URI 替换为该请求中 Contact 头字段中的 URI（如果存在）。

12.3 终止对话

与该方法无关，如果对话之外的请求生成非 2xx 最终响应，则通过对该请求的临时响应创建的任何早期对话都将终止。 终止确认对话的机制是特定于方法的。 在本规范中，BYE 方法终止一个会话和与之关联的对话。 详见第 15 节。

13 启动会话

13.1 概述

当用户代理客户端希望启动会话（例如，音频、视频或游戏）时，它会制定一个 INVITE 请求。 INVITE 请求要求服务器建立会话。 该请求可能由代理转发，最终到达一个或多个可能接受邀请的 UAS。这些 UAS 将经常需要询问用户是否接受邀请。一段时间后，这些 UAS 可以通过发送 2xx 响应来接受邀请（意味着要建立会话）。 如果邀请未被接受，则根据拒绝原因发送 3xx、4xx、5xx 或 6xx 响应。 在发送最终响应之前，UAS 还可以发送临时响应（1xx）以通知 UAC 联系被叫用户的进度。

在可能收到一个或多个临时响应后，UAC 将获得一个或多个 2xx 响应或一个非 2xx 最终响应。 由于接收对 INVITE 的最终响应可能需要很长时间，因此 INVITE 事务的可靠性机制与其他请求（如 OPTIONS）的可靠性机制不同。 一旦收到最终响应，UAC 需要为它收到的每个最终响应发送一个 ACK。 发送此 ACK 的过程取决于响应的类型。 对于 300 到 699 之间的最终响应，ACK 处理在事务层完成并遵循一组规则（参见第 17 节）。 对于 2xx 响应，ACK 由 UAC 核心生成。

对 INVITE 的 2xx 响应建立会话，并且它还在发出 INVITE 的 UA 和生成 2xx 响应的 UA 之间创建对话。 因此，当从不同的远程 UA 接收到多个 2xx 响应时（因为 INVITE 分叉），

每个 2xx 都会建立不同的对话。 所有这些对话框都是同一个调用的一部分。

本节提供有关使用 INVITE 建立会话的详细信息。 支持 INVITE 的 UA 还必须支持 ACK、CANCEL 和 BYE。

13.2 UAC 处理

13.2.1 创建初始邀请

由于初始 INVITE 表示对话之外的请求，因此其构造遵循第 8.1.1 节的过程。 INVITE 的具体情况需要额外处理。

一个允许头字段（第 20.5 节）应该出现在 INVITE 中。 它指示在对话期间，在发送 INVITE 的 UA 上，可以在对话中调用哪些方法。 例如，一个能够在对话 [34] 中接收 INFO 请求的 UA 应该包括一个列出 INFO 方法的 Allow 头字段。

支持的头字段（第 20.37 节）应该出现在 INVITE 中。 它列举了 UAC 理解的所有扩展。

一个 Accept (Section 20.1) 头域可能出现在 INVITE 中。它指示 UA 在收到的响应中以及在 INVITE 建立的对话中发送给它的任何后续请求中，哪些 Content-Types 是可接受的。Accept 标头字段对于指示对各种会话描述格式的支持特别有用。

UAC 可以添加一个 Expires 头域（第 20.19 节）来限制邀请的有效性。如果到了 Expires 头域中指示的时间并且没有收到对 INVITE 的最终答复，则 UAC 核心应该根据第 9 节为 INVITE 生成一个 CANCEL 请求。

UAC 可能还会发现添加主题（第 20.36 节）、组织（第 20.25 节）和用户代理（第 20.41 节）头字段等有用。 它们都包含与邀请相关的信息。

UAC 可以选择将消息正文添加到 INVITE。 第 8.1.1.10 节处理如何构建用于描述消息正文的头字段——Content-Type 等。

包含会话描述的消息正文有特殊规则 - 它们对应的 Content-Disposition 是“会话”。 SIP 使用提议/应答模型，其中一个 UA 发送会话描述，称为提议，其中包含会话的建议描述。报价指示所需的通信方式（音频、视频、游戏）、这些方式的参数（例如编解码器类型）以及从应答者接收媒体的地址。另一个 UA 用另一个会话描述进行响应，称为应答，它指示接受哪些通信方式、适用于这些方式的参数以及从提供者接收媒体的地址。提议/答案交换在对话的上下文中，因此如果 SIP INVITE 导致多个对话，则每个对话都是单独的提议/答案交换。报价/答案模型定义了何时可以提出报价和答案的限制（例如，您不能在报价进行时提出新报价）。这会限制提议和答案在 SIP 消息中的显示位置。在本规范中，offer 和 answer 只能出现在 INVITE 请求和响应以及 ACK 中。提议和答案的使用受到进一步限制。对于初始 INVITE 交易，规则是：

o 初始提议必须在 INVITE 中，或者如果没有，则在从 UAS 发回 UAC 的第一个可靠的非故障消息中。 在本规范中，这是最终的 2xx 响应。

o 如果初始报价在 INVITE 中，则答案必须在从 UAS 返回到 UAC 的可靠非失败消息中，该消息与该 INVITE 相关。 对于本规范，这只是对该邀请的最终 2xx 响应。 同样的确切答案也可以放在答案之前发送的任何临时回复中。 UAC 必须将它收到的第一个会话描述视为答案，并且必须忽略后续对初始 INVITE 的响应中的任何会话描述。

o 如果初始提议是在从 UAS 返回给 UAC 的第一个可靠的非失败消息中，则答案必须在该消息的确认中（在本规范中，ACK 表示 2xx 响应）。

o 在发送或接收到第一个报价的答复后，UAC 可以根据为该方法指定的规则在请求中生成后续报价，但前提是它已收到对任何先前报价的答复，并且未发送任何未发送的报价 没有得到答案。

o 一旦 UAS 发送或接收到对初始报价的答复，它不得在对初始邀请的任何响应中生成后续报价。 这意味着仅基于此规范的 UAS 在完成初始交易之前永远无法生成后续报价。

具体来说，上述规则为仅符合本规范的 UA 指定了两个交换 - 报价在 INVITE 中，答案在 2xx 中（也可能在 1xx 中，具有相同的值），或者报价在 2xx，答案在 ACK 中。 所有支持 INVITE 的用户代理必须支持这两个交换。

会话描述协议 (SDP) (RFC 2327 [1]) 必须被所有用户代理支持，作为描述会话的一种手段，并且它用于构建提议和答案必须遵循 [13] 中定义的程序。

刚刚描述的 offer-answer 模型的限制仅适用于 Content-Disposition 头字段值为“会话”的主体。 因此，INVITE 和 ACK 都可能包含正文消息（例如，INVITE 携带照片（Content-Disposition:render），ACK 携带会话描述（Content-Disposition:session））。

如果缺少 Content-Disposition 标头字段，则 Content-Type application/sdp 的主体暗示处置“会话”，而其他内容类型暗示“渲染”。

一旦创建了 INVITE，UAC 就会遵循为在对话之外发送请求而定义的过程（第 8 节）。 这导致构建客户端事务，该事务最终将发送请求并将响应传递给 UAC。

13.2.2 处理邀请响应

一旦 INVITE 被传递给 INVITE 客户端事务，UAC 就会等待 INVITE 的响应。如果 INVITE 客户端事务返回超时而不是响应，则 TU 就像收到 408（请求超时）响应一样，如第 8.1.3 节所述。

13.2.2.1 1xx 响应

在接收到一个或多个最终响应之前，可能会到达零个、一个或多个临时响应。 INVITE 请求的临时响应可以创建“早期对话”。 如果临时响应在 To 字段中有标签，并且响应的对话 ID 与现有对话不匹配，则使用第 12.1.2 节中定义的过程构建一个。

仅当 UAC 需要在初始 INVITE 事务完成之前向对话内的对等方发送请求时，才需要早期对话。 只要对话处于早期状态，临时响应中的标头字段就适用（例如，临时响应中的 Allow 标头字段包含在对话处于早期状态时可以在对话中使用的方法）。

13.2.2.2 3xx 响应

一个 3xx 响应可能包含一个或多个 Contact 标头字段值，这些值提供被调用者可能可到达的新地址。 根据 3xx 响应的状态码（参见第 21.3 节），UAC 可以选择尝试这些新地址。

13.2.2.3 4xx、5xx 和 6xx 响应

对于 INVITE，可能会收到一个非 2xx 的最终响应。 4xx、5xx 和 6xx 响应可能包含一个 Contact 头字段值，指示可以找到有关错误的附加信息的位置。 随后的最终响应（仅在错误条件下到达）必须被忽略。

收到非 2xx 最终响应后，所有早期对话都被视为终止。

在收到非 2xx 最终响应后，UAC 核心认为 INVITE 事务已完成。 INVITE 客户端事务处理响应的 ACK 生成（参见第 17 节）。

13.2.2.4 2xx 响应

由于分叉代理，单个 INVITE 请求的多个 2xx 响应可能会到达 UAC。 每个响应由 To 标头字段中的标记参数区分，每个响应代表一个不同的对话，具有不同的对话标识符。

如果 2xx 响应中的对话标识符与现有对话的对话标识符匹配，则对话必须转换为“已确认”状态，并且必须根据 2xx 响应使用第 12.2 节的过程重新计算为对话设置的路由 .1.2. 否则，必须使用第 12.1.2 节的过程构建一个处于“已确认”状态的新对话。

请注意，唯一重新计算的状态是路由集。 不会重新计算对话中发送的其他状态，例如最高序列号（远程和本地）。 仅重新计算路由集以实现向后兼容性。 RFC 2543 没有强制在 1xx 中镜像 Record-Route 标头字段，只有 2xx。 但是，我们无法更新对话的整个状态，因为对话中的请求可能已经在早期对话中发送，例如修改了序列号。

UAC 核心必须为从事务层接收到的每个 2xx 生成一个 ACK 请求。 除了 Cseq 和与身份验

证相关的头字段外，ACK 的头字段的构造方式与对话中发送的任何请求的相同（参见第 12 节）。 CSeq 头域的序列号必须与被确认的 INVITE 相同，但 CSeq 方法必须是 ACK。 ACK 必须包含与 INVITE 相同的凭据。 如果 2xx 包含一个提议（基于上述规则），则 ACK 必须在其正文中携带一个答案。 如果 2xx 响应中的提议不可接受，UAC 核心必须在 ACK 中生成一个有效的答案，然后立即发送一个 BYE。

一旦构造了 ACK，就使用 [4] 的过程来确定目标地址、端口和传输。 但是，请求被直接传递给传输层进行传输，而不是客户端事务。 这是因为 UAC 核心处理 ACK 的重传，而不是事务层。 每次重传触发 ACK 的 2xx 最终响应到达时，必须将 ACK 传递给客户端传输。

UAC 核心认为 INVITE 事务在收到第一个 2xx 响应后 64*T1 秒完成。 此时，所有尚未转换为已建立对话的早期对话都将终止。 一旦 UAC 核心认为 INVITE 事务已完成，就不会再有新的 2xx 响应到达。

如果在确认对 INVITE 的任何 2xx 响应后，UAC 不想继续该对话，则 UAC 必须通过发送 BYE 请求来终止对话，如第 15 节所述。

13.3 UAS 处理

13.3.1 INVITE 的处理

UAS 核心将接收来自事务层的 INVITE 请求。 它首先执行第 8.2 节的请求处理过程，该过程适用于对话内部和外部的请求。

假设这些处理状态在没有产生响应的情况下完成，UAS 核心执行额外的处理步骤：

1. 如果请求是一个包含 Expires 头域的 INVITE，UAS 核心为头域值中指示的秒数设置一个计时器。 当计时器触发时，邀请被视为过期。 如果邀请在 UAS 生成最终响应之前到期，则应生成 487（请求终止）响应。

2. 如果请求是中间对话请求，则首先应用第 12.2.2 节中描述的方法无关处理。 它还可能修改会话； 第 14 节提供了详细信息。

3. 如果请求在 To 头域中有标签，但对话标识符与任何现有对话都不匹配，则 UAS 可能已崩溃并重新启动，或者可能已收到对不同（可能失败）UAS 的请求。 第 12.2.2 节提供了在这种情况下实现稳健行为的指南。

从这里开始的处理假设 INVITE 在对话之外，因此是为了建立一个新的会话。

INVITE 可能包含会话描述，在这种情况下，UAS 将被呈现该会话的提议。 即使 INVITE 在对话之外，用户也可能已经是该会话的参与者。 当多个其他参与者邀请用户参加同一个多播会议时，可能会发生这种情况。 如果需要，UAS 可以使用会话描述中的标识符来检测这种重复。 例如，SDP 在 origin (o) 字段中包含会话 id 和版本号。 如果用户已经是会话的

成员，并且会话描述中包含的会话参数没有改变，UAS 可以静默接受 INVITE（即发送 2xx 响应而不提示用户）。

如果 INVITE 不包含会话描述，则 UAS 被要求参与会话，并且 UAC 已要求 UAS 提供会话的提议。 它必须在它的第一个非故障可靠消息中向 UAC 提供报价。 在本规范中，这是对 INVITE 的 2xx 响应。

UAS 可以指示进度、接受、重定向或拒绝邀请。 在所有这些情况下，它都会使用第 8.2.6 节中描述的程序来制定响应。

13.3.1.1 进展

如果 UAS 不能立即应答邀请，它可以选择向 UAC 指示某种进展（例如，电话正在响铃的指示）。 这是通过 101 到 199 之间的临时响应来完成的。这些临时响应建立了早期对话，因此除了第 8.2.6 节的程序之外，还遵循第 12.1.1 节的程序。 UAS 可以发送任意数量的临时响应。 这些中的每一个都必须指示相同的对话 ID。 但是，这些将无法可靠地交付。

如果 UAS 希望延长一段时间来回复 INVITE，则需要请求“延期”以防止代理取消交易。 当事务中的响应之间存在 3 分钟的间隔时，代理可以选择取消事务。 为了防止取消，UAS 必须每分钟发送一个非 100 的临时响应，以处理丢失临时响应的可能性。

当用户被搁置时，或者当与允许在不接听电话的情况下进行通信的 PSTN 系统互通时，INVITE 事务可以持续很长时间。 后者在交互式语音响应 (IVR) 系统中很常见。

13.3.1.2 邀请被重定向

如果 UAS 决定重定向呼叫，则发送 3xx 响应。 一个 300（多项选择）、301（永久移动）或 302（临时移动）响应应该包含一个包含一个或多个要尝试的新地址的 URI 的 Contact 头字段。 响应被传递给 INVITE 服务器事务，该事务将处理其重传。

13.3.1.3 邀请被拒绝

当被调用者当前不愿意或不能在该端系统接听额外的呼叫时，会发生一种常见的情况。 在这种情况下应该返回 486（这里忙）。 如果 UAS 知道没有其他终端系统能够接受此呼叫，则应发送 600（到处忙）响应。 但是，一般而言，UAS 不太可能知道这一点，因此通常不会使用此响应。 响应被传递给 INVITE 服务器事务，该事务将处理其重传。

拒绝包含在 INVITE 中的提议的 UAS 应该返回 488（此处不可接受）响应。 这样的响应应该包含一个警告头字段值，解释为什么拒绝报价。

13.3.1.4 接受邀请

UAS 核心生成 2xx 响应。 该响应建立了一个对话，因此除了第 8.2.6 节的过程之外，还遵循第 12.1.1 节的过程。

对 INVITE 的 2xx 响应应该包含 Allow 头域和 Supported 头域，并且可以包含 Accept 头域。 包括这些头字段允许 UAC 确定 UAS 在呼叫期间支持的功能和扩展，而无需探测。

如果 INVITE 请求包含要约，并且 UAS 尚未发送应答，则 2xx 必须包含应答。 如果 INVITE 不包含要约，如果 UAS 尚未发送要约，则 2xx 必须包含要约。

一旦构建了响应，它就会被传递给 INVITE 服务器事务。 但是请注意，INVITE 服务器事务将在收到此最终响应并将其传递给传输时立即销毁。 因此，需要周期性地将响应直接传递给传输，直到 ACK 到达。 2xx 响应以从 T1 秒开始的间隔传递给传输，每次重传时加倍，直到达到 T2 秒（T1 和 T2 在第 17 节中定义）。 当收到响应的 ACK 请求时，响应重传停止。 这与用于发送响应的任何传输协议无关。

由于 2xx 是端到端重传的，因此 UAS 和 UAC 之间可能存在 UDP 跃点。 为了确保跨这些跃点的可靠传递，即使 UAS 的传输是可靠的，也会定期重传响应。

如果服务器在 64*T1 秒内重新发送 2xx 响应而没有收到 ACK，则确认对话，但应终止会话。如第 15 节所述，这是通过 BYE 完成的。

14 修改现有会话

一个成功的 INVITE 请求（见第 13 节）建立了两个用户代理之间的对话和一个使用 offer-answer 模型的会话。 第 12 节解释了如何使用目标刷新请求修改现有对话（例如，更改对话的远程目标 URI）。 本节介绍如何修改实际会话。 此修改可能涉及更改地址或端口、添加媒体流、删除媒体流等。 这是通过在建立会话的同一对话中发送新的 INVITE 请求来完成的。 在现有对话中发送的 INVITE 请求称为重新邀请。

注意，单次 re-INVITE 可以同时修改对话和会话参数。

调用者或被调用者都可以修改现有会话。

UA 在检测媒体故障时的行为是本地策略的问题。 但是，不建议自动生成 re-INVITE 或 BYE 以避免在出现拥塞时使网络充满流量。 在任何情况下，如果这些消息是自动发送的，它们应该在某个随机间隔之后发送。

请注意，上面的段落是指自动生成的 BYE 和 re-INVITE。如果用户在媒体故障时挂断，UA 会像往常一样发送 BYE 请求。

14.1 UAC 行为

适用于 INVITE 中的会话描述（第 13.2.1 节）的相同提议-应答模型适用于重新邀请。 因此，例如，想要添加媒体流的 UAC 将创建一个包含此媒体流的新报价，并将其在 INVITE 请求中发送给其对等方。 重要的是要注意发送的是会话的完整描述，而不仅仅是更改。 这支持各种元素中的无状态会话处理，并支持故障转移和恢复功能。 当然，UAC 可以发送没有会话描述的 re-INVITE，在这种情况下，对 re-INVITE 的第一个可靠的非失败响应将包含提议（在本规范中，即 2xx 响应）。

如果会话描述格式具有版本号的能力，提供者应该指出会话描述的版本已经改变。

re-INVITE 的 To、From、Call-ID、CSeq 和 Request-URI 的设置遵循与现有对话中的常规请求相同的规则，如第 12 节所述。

UAC 可以选择不向 re-INVITE 添加 Alert-Info 头字段或带有 Content-Disposition “alert”的正文，因为 UAS 通常不会在收到 re-INVITE 时提醒用户。

与可以分叉的 INVITE 不同，re-INVITE 永远不会分叉，因此只会产生一个最终响应。 re-INVITE 永远不会分叉的原因是 Request-URI 将目标标识为与其建立对话的 UA 实例，而不是标识用户的记录地址。

请注意，当另一个 INVITE 事务在任一方向进行时，UAC 不得在对话中发起新的 INVITE 事务。

1.如果有一个正在进行的 INVITE 客户端事务，TU 必须等到事务达到完成或终止状态后才发起新的 INVITE。

2.如果有一个正在进行的 INVITE 服务器事务，TU 必须等到事务达到确认或终止状态后才发起新的 INVITE。

但是，UA 可以在 INVITE 事务进行时发起常规事务。 当常规事务正在进行时，UA 也可以发起 INVITE 事务。

如果 UA 收到对 re-INVITE 的非 2xx 最终响应，则会话参数必须保持不变，就好像没有发出 re-INVITE 一样。 请注意，如第 12.2.1.2 节所述，如果非 2xx 最终响应是 481（呼叫/事务不存在）或 408（请求超时），或者根本没有收到 re-INVITE 响应 （即 INVITE 客户端事务返回超时），UAC 将终止对话。

如果 UAC 接收到对 re-INVITE 的 491 响应，它应该启动一个定时器，其值 T 选择如下：

1. 如果 UAC 是对话 ID 的 Call-ID 的所有者（意味着它生成了该值），则 T 在 2.1 到 4 秒之间随机选择一个值，以 10 毫秒为单位。

2. 如果 UAC 不是对话 ID 的 Call-ID 的所有者，则 T 具有随机选择的值，介于 0 和 2 秒之间，以 10 毫秒为单位。

当计时器触发时，UAC 应该再次尝试重新邀请，如果它仍然希望该会话修改发生。 例如，如果呼叫已被 BYE 挂断，则不会发生 re-INVITE。

发送 re-INVITE 和为 re-INVITE 的 2xx 响应生成 ACK 的规则与初始 INVITE 的规则相同（第 13.2.1 节）。

14.2 UAS 行为

第 13.3.1 节描述了用于区分传入的 re-INVITE 和传入的初始 INVITE 以及处理现有对话的 re-INVITE 的过程。

在同一对话中，在向具有较低 Cseq 序列号的第一个 INVITE 发送最终响应之前接收第二个 INVITE 的 UAS 必须向第二个 INVITE 返回 500（服务器内部错误）响应，并且必须包含 Retry-After 头字段 随机选择的值介于 0 到 10 秒之间。

一个 UAS 在一个对话中收到一个 INVITE 而它在那个对话上发送的一个 INVITE 正在进行中，必须返回一个 491（请求未决）响应接收到的 INVITE。

如果 UA 收到对现有对话的 re-INVITE，它必须检查会话描述中的任何版本标识符，或者，如果没有版本标识符，则检查会话描述的内容以查看它是否已更改。 如果会话描述发生了变化，UAS 必须相应地调整会话参数，可能在请求用户确认之后。

会话描述的版本控制可用于适应会议新成员的能力、添加或删除媒体或从单播会议更改为多播会议。

如果新的会话描述不可接受，UAS 可以通过返回 488（此处不可接受）响应来拒绝它。 这个响应应该包含一个警告头域。

如果 UAS 生成 2xx 响应并且从未收到 ACK，它应该生成 BYE 来终止对话。

UAS 可以选择不为 re-INVITE 生成 180（振铃）响应，因为 UAC 通常不会将此信息呈现给用户。 出于同样的原因，UAS 可以选择不使用 Alert-Info 头域或带有 Content-Disposition “alert”的主体来响应 re-INVITE。

UAS 在 2xx 中提供要约（因为 INVITE 不包含要约）应该构造要约，就好像 UAS 正在发出全新的呼叫一样，受发送更新现有会话的要约的约束，如中所述 [13] 在 SDP 的情况下。具体来说，这意味着它应该包括 UA 愿意支持的尽可能多的媒体格式和媒体类型。 UAS 必须确保会话描述在媒体格式、传输或其他需要对等方支持的参数中与其先前的会话描述重叠。这是为了避免对等方拒绝会话描述。 然而，如果 UAC 不能接受，UAC 应该生成一个带有有效会话描述的答案，然后发送 BYE 来终止会话。

15 终止会话

本节介绍终止 SIP 建立的会话的过程。会话的状态和对话的状态是密切相关的。当使用 INVITE 启动会话时，来自不同 UAS 的每个 1xx 或 2xx 响应都会创建一个对话，如果该响应完成了提议/答案交换，它还会创建一个会话。结果，每个会话都与单个对话框“关联”- 导致其创建的那个。如果初始 INVITE 生成非 2xx 最终响应，则终止通过响应请求创建的所有会话（如果有）和所有对话（如果有）。通过完成事务，非 2xx 的最终响应还可以防止由于邀请而创建进一步的会话。 BYE 请求用于终止特定会话或尝试的会话。在这种情况下，特定会话是与对话另一侧的对等 UA 的会话。当在对话上收到 BYE 时，与该对话相关的任何会话都应该终止。 UA 不能在对话之外发送 BYE。调用方的 UA 可以为已确认或早期的对话发送 BYE，被叫方的 UA 可以在已确认的对话上发送 BYE，但不能在早期的对话中发送 BYE。

但是，被调用方的 UA 不得在确认对话上发送 BYE，直到它收到 2xx 响应的 ACK 或服务器事务超时。如果没有 SIP 扩展定义了与对话相关的其他应用层状态，BYE 也会终止对话。

对 INVITE 的非 2xx 最终响应对对话和会话的影响使得 CANCEL 的使用具有吸引力。 CANCEL 尝试强制对 INVITE 做出非 2xx 响应（特别是 487）。 因此，如果 UAC 希望完全放弃其呼叫尝试，它可以发送 CANCEL。 如果 INVITE 导致对 INVITE 的最终响应为 2xx，这意味着 UAS 在 CANCEL 进行时接受了邀请。 UAC 可以继续任何 2xx 响应建立的会话，或者可以用 BYE 终止它们。

“挂断”的概念在 SIP 中没有得到很好的定义。 它特定于特定的但常见的用户界面。 通常，当用户挂机时，它表示希望终止建立会话的尝试，并终止任何已经创建的会话。 对于调用者的 UA，如果初始 INVITE 没有生成最终响应，这将意味着 CANCEL 请求，并且在最终响应之后对所有已确认的对话进行 BYE。 对于被叫方的 UA，它通常意味着 BYE； 想必，当用户拿起电话时，产生了一个 2xx，所以挂断会导致收到 ACK 后的 BYE。 这并不意味着用户在收到 ACK 之前不能挂机，只是意味着他的手机中的软件需要保持一段时间的状态才能正确清理。 如果特定的 UI 允许用户在接听电话之前拒绝来电，则 403（禁止）是一种很好的表达方式。 根据上述规则，不能发送 BYE。

15.1 用 BYE 请求终止会话

15.1.1 UAC 行为

BYE 请求的构造与对话中的任何其他请求一样，如第 12 节所述。

一旦构造了 BYE，UAC 核心就会创建一个新的非 INVITE 客户端事务，并将 BYE 请求传递给它。 一旦 BYE 请求被传递给客户端事务，UAC 必须认为会话终止（并因此停止发送或侦听媒体）。 如果 BYE 的响应是 481（呼叫/事务不存在）或 408（请求超时）或根本没有收到 BYE 的响应（即客户端事务返回超时），UAC 必须认为会话和对话已终止。

15.1.2 UAS 行为

UAS 首先根据第 8.2 节中描述的通用 UAS 处理来处理 BYE 请求。 接收 BYE 请求的 UAS 核心检查它是否与现有对话匹配。 如果 BYE 与现有对话不匹配，UAS 核心应该生成 481（呼叫/事务不存在）响应并将其传递给服务器事务。

此规则意味着 UAC 发送的不带标签的 BYE 将被拒绝。 这是对 RFC 2543 的更改，后者允许不带标签的 BYE。

接收到现有对话的 BYE 请求的 UAS 核心必须遵循第 12.2.2 节的过程来处理请求。 一旦完成，UAS 应该终止会话（并因此停止发送和侦听媒体）。 它可以选择不参与的唯一情况是多播会话，即使对话中的其他参与者已经终止了对会话的参与，也可以参与。 无论它是否结束参与会话，UAS 核心必须生成一个 2xx 对 BYE 的响应，并且必须将其传递给服务器事务以进行传输。

UAS 必须仍然响应为该对话接收到的任何未决请求。 建议对那些挂起的请求生成 487（请求终止）响应。

16 代理行为

16.1 概述

SIP 代理是将 SIP 请求路由到用户代理服务器并将 SIP 响应路由到用户代理客户端的元素。一个请求在到达 UAS 的过程中可能会经过多个代理。 每个人都会做出路由决策，在将请求转发到下一个元素之前修改请求。 响应将通过请求以相反顺序遍历的同一组代理进行路由。

作为代理是 SIP 元素的逻辑角色。 当请求到达时，一个可以扮演代理角色的元素首先决定它是否需要自己响应请求。 例如，请求可能格式错误，或者元素在充当代理之前可能需要来自客户端的凭据。 元素可以使用任何适当的错误代码进行响应。 当直接响应请求时，该元素扮演 UAS 的角色，并且必须按照第 8.2 节中的描述进行行为。

对于每个新请求，代理可以在有状态或无状态模式下运行。 当无状态时，代理充当简单的转发元素。 它将每个请求向下游转发到单个元素，该元素通过根据请求做出目标和路由决策来确定。 它只是转发它在上游收到的每个响应。 一旦消息被转发，无状态代理就会丢弃有关消息的信息。 有状态代理会记住有关每个传入请求的信息（特别是事务状态）以及作为处理传入请求的结果而发送的任何请求。 它使用此信息来影响与该请求关联的未来消息的处理。 有状态代理可以选择“分叉”一个请求，将其路由到多个目的地。 任何转发到多个位置的请求都必须有状态地处理。

在某些情况下，代理可以使用有状态传输（例如 TCP）转发请求而不是事务状态。 例如，代理可以无状态地将来自一个 TCP 连接的请求转发到另一个事务，只要它在消息中放置足够的信息以能够将响应转发到请求到达的同一连接。 在不同类型的传输之间转发的请求，其中代理的 TU 必须在确保其中一种传输上的可靠交付方面发挥积极作用，必须以事务状态转发。

有状态代理可以在处理请求期间的任何时候转换到无状态操作，只要它没有做任何会阻止它最初无状态的事情（例如，分叉或生成 100 响应）。 当执行这样的转换时，所有状态都被简单地丢弃。 代理不应发起 CANCEL 请求。

对请求进行无状态或有状态操作时所涉及的大部分处理是相同的。 接下来的几个小节是从有状态代理的角度编写的。 最后一部分列出了无状态代理行为不同的地方。

16.2 有状态代理

当有状态时，代理纯粹是一个 SIP 事务处理引擎。 它的行为在这里根据第 17 节中定义的服务器和客户端事务建模。有状态代理有一个服务器事务与一个或多个客户端事务相关联，该事务由更高层的代理处理组件（见图 3），称为代理核心 . 传入请求由服务器事务处理。来自服务器事务的请求被传递到代理核心。 代理核心确定将请求路由到哪里，选择一个或多个下一跳位置。 每个下一跳位置的传出请求由其自己的关联客户端事务处理。 代理核心收集来自客户端事务的响应，并使用它们向服务器事务发送响应。

有状态代理为收到的每个新请求创建一个新的服务器事务。 然后，根据第 17 节，该请求

的任何重传都将由该服务器事务处理。代理核心必须像 UAS 一样在该服务器事务上发送立即临时（例如 100 Trying），如第 8.2.6 节所述。 因此，有状态代理不应该对非邀请请求生成 100 个（尝试）响应。

这是代理行为的模型，而不是软件的模型。 实现可以自由地采用任何复制此模型定义的外部行为的方法。

对于所有新请求，包括任何具有未知方法的请求，打算代理请求的元素必须：

1. 验证请求（第 16.3 节）

2.预处理路由信息（16.4 节）

3. 确定请求的目标（第 16.5 节）

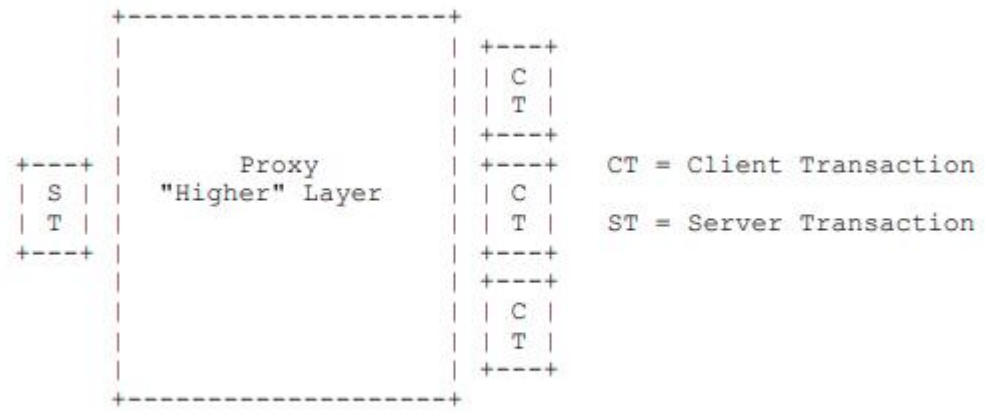

Figure 3: Stateful Proxy Model

4. 将请求转发到每个目标（第 16.6 节）

5. 处理所有响应（第 16.7 节）

16.3 请求验证

在元素可以代理请求之前，它必须验证消息的有效性。 有效的消息必须通过以下检查：

1. Reasonable Syntax
2. URI scheme
3. Max-Forwards
4. (Optional) Loop Detection
5. Proxy-Require
6. Proxy-Authorization

如果这些检查中的任何一个失败，该元素必须充当用户代理服务器（参见第 8.2 节）并以错误代码响应。

请注意，不需要代理来检测合并的请求，并且不得将合并的请求视为错误条件。 接收请求的端点将解决合并，如第 8.2.2.2 节所述。

1.合理的语法检查

请求的格式必须足够好，以便通过服务器事务处理。 这些请求验证步骤的其余部分或请求转发部分涉及的任何组件都必须格式正确。 转发消息时，任何其他组件，无论格式是否正确，都应该被忽略并保持不变。 例如，一个元素不会因为格式错误的 Date 标头字段而拒绝请求。 同样，代理在转发请求之前不会删除格式错误的 Date 标头字段。

该协议旨在扩展。 未来的扩展可能会随时定义新的方法和标头字段。 元素不得拒绝代理请求，因为它包含它不知道的方法或标头字段。

2.URI scheme 检查

如果 Request-URI 有一个代理不理解其方案的 URI，代理应该以 416（不支持的 URI 方案）响应拒绝该请求。

3. Max-Forwards 检查

Max-Forwards 标头字段（第 20.22 节）用于限制 SIP 请求可以遍历的元素数量。

如果请求不包含 Max-Forwards 头字段，则通过此检查。

如果请求包含字段值大于零的 Max-Forwards 头字段，则检查通过。

如果请求包含字段值为零 (0) 的 Max-Forwards 头字段，则元素不得转发请求。 如果请求是针对 OPTIONS 的，则元素可以充当最终接收者并根据第 11 节做出响应。否则，元素必须返回 483（跳数过多）响应。

4.可选的环路检测检查

一个元素可以在转发请求之前检查转发循环。 如果请求包含带有 sent-by 值的 Via 头字段，该值等于代理放置在先前请求中的值，则该请求之前已被此元素转发。 请求已经循环或合法地通过元素螺旋。为了确定请求是否已经循环，元素可以对该消息执行 16.6 节的步骤 8 中描述的分支参数计算，并将其与在该 Via 头字段中接收到的参数进行比较。 如果参数匹配，则请求已循环。 如果它们不同，则请求呈螺旋式上升，处理将继续。 如果检测到循环，元素可以返回 482（检测到循环）响应。

5. Proxy-Require 检查

该协议的未来扩展可能会引入需要代理进行特殊处理的功能。 端点将在使用这些功能的请求中包含一个 Proxy-Require 标头字段，告诉代理不要处理请求，除非该功能被理解。

如果请求包含一个 Proxy-Require 头字段（第 20.29 节）和一个或多个此元素无法理解的选项标签，则该元素必须返回 420（错误扩展）响应。 响应必须包含一个 Unsupported（第 20.40 节）头字段，列出元素不理解的选项标签。

6. Proxy-Authorization 检查

如果一个元素在转发请求之前需要凭证，则必须按照第 22.3 节所述检查请求。 该部分还定义了检查失败时元素必须执行的操作。

16.4 路由信息预处理

代理必须检查请求的 Request-URI。 如果请求的 Request-URI 包含此代理先前放置在 Record-Route 头字段中的值（参见第 16.6 节第 4 项），则代理必须用 Route 头字段中的最后一个值替换请求中的 Request-URI ，并从 Route 标头字段中删除该值。 然后代理必须继续进行，就好像它收到了这个修改后的请求一样。

只有当向代理（可能是端点）发送请求的元素是严格的路由器时，才会发生这种情况。 接收时的这种重写对于启用与这些元素的向后兼容性是必要的。 它还允许遵循本规范的元素通过严格路由代理保留 Request-URI（参见第 12.2.1.1 节）。

此要求不强制代理保持状态以检测其先前放置在 Record-Route 标头字段中的 URI。 相反，代理只需要在这些 URI 中放置足够的信息，以便在它们以后出现时将它们识别为它提供的值。

如果 Request-URI 包含一个 maddr 参数，代理必须检查它的值是否在代理配置为负责的地址或域集合中。 如果 Request-URI 有一个带有代理负责的值的 maddr 参数，并且使用 Request-URI 中指示的端口和传输（显式或默认）接收请求，则代理必须剥离 maddr 和任何非 -默认端口或传输参数并继续处理，就好像请求中不存在这些值一样。

请求可能会通过与代理匹配的 maddr 到达，但在与 URI 中指示的端口或传输不同的端口或传输上。 这样的请求需要使用指定的端口和传输转发到代理。

如果 Route 头字段中的第一个值指示此代理，则代理必须从请求中删除该值。

16.5 确定请求目标

接下来，代理计算请求的目标。 这组目标要么由请求的内容预先确定，要么从抽象的位置服务中获得。 集合中的每个目标都表示为一个 URI。

如果请求的 Request-URI 包含 maddr 参数，则 Request-URI 必须作为唯一的目标 URI 放入目标集中，并且代理必须继续执行第 16.6 节。

如果 Request-URI 的域指示该元素不负责的域，则 Request-URI 必须作为唯一目标放入目标集中，并且该元素必须继续执行请求转发任务（第 16.6 节）。

在许多情况下，代理可能会收到对它不负责的域的请求。 处理传出调用的防火墙代理（HTTP 代理处理传出请求的方式）是可能发生这种情况的一个示例。

如果请求的目标集没有如上所述预先确定，这意味着该元素负责 Request-URI 中的域，并且该元素可以使用它希望确定将请求发送到何处的任何机制。 这些机制中的任何一个都可以建模为访问抽象的位置服务。 这可能包括从 SIP 注册器创建的位置服务获取信息、读取数据库、咨询存在服务器、利用其他协议或简单地对请求 URI 执行算法替换。 当访问由注册商构建的位置服务时，Request-URI 必须首先按照第 10.3 节中的描述进行规范化，然后才能用作索引。 这些机制的输出用于构建目标集。

如果 Request-URI 没有为代理提供足够的信息来确定目标集，它应该返回一个 485 (Ambiguous) 响应。 这个响应应该包含一个包含要尝试的新地址的 URI 的 Contact 头字段。 例如，邀请 sip:John.Smith@company.com 在其位置服务列出多个 John Smiths 的代理中可能不明确。 有关详细信息，请参阅第 21.4.23 节。

元素的请求或当前环境中的任何信息或关于元素的当前环境的任何信息都可以用于构建目标集。 例如，可以根据内容或标题字段和主体的存在、请求到达的时间、请求到达的接口、先前请求的失败，甚至元素的当前利用率级别来构建不同的集合。

由于通过这些服务定位潜在目标，它们的 URI 被添加到目标集中。 目标只能在目标集中放置一次。 如果目标 URI 已经存在于集合中（基于 URI 类型的相等定义），则不得再次添加它。

如果原始请求的 Request-URI 未指示此代理负责的资源，则代理不得将其他目标添加到目标集。

如果代理负责该 URI，则代理只能在转发期间更改请求的 Request-URI。 如果代理不负责该 URI，则它不会在 3xx 或 416 响应上递归，如下所述。

如果原始请求的 Request-URI 指示该代理负责的资源，则代理可以在开始请求转发后继续将目标添加到集合中。 它可以使用在该处理过程中获得的任何信息来确定新的目标。 例如，代理可以选择将在重定向响应 (3xx) 中获得的联系人合并到目标集中。 如果代理在构建目标集时使用动态信息源（例如，如果它咨询 SIP 注册器），它应该在处理请求期间监视该源。 新位置应在可用时添加到目标集中。 如上所述，任何给定的 URI 不得多次添加到集合中。

仅允许将 URI 添加到集合中一次可以减少不必要的网络流量，并且在合并来自重定向请求的联系人的情况下可以防止无限递归。

例如，一个普通的位置服务是一个“无操作”，其中目标 URI 等于传入的请求 URI。 该请求被发送到特定的下一跳代理以进行进一步处理。 在第 16.6 节第 6 项的请求转发期间，下

一跳的标识（表示为 SIP 或 SIPS URI）作为最顶部的 Route 头字段值插入到请求中。

如果 Request-URI 指示此代理上的资源不存在，则代理必须返回 404（未找到）响应。

如果在应用上述所有操作后目标集仍然为空，则代理必须返回一个错误响应，它应该是 480（临时不可用）响应。

16.6 请求转发

一旦目标集非空，代理就可以开始转发请求。 有状态代理可以以任何顺序处理该集合。 它可以串行处理多个目标，允许每个客户端事务在开始下一个事务之前完成。 它可以与每个目标并行启动客户端事务。 它也可以任意将集合分成组，串行处理组，并行处理每个组中的目标。

一种常见的排序机制是使用从 Contact 头字段获得的目标的 qvalue 参数（参见第 20.10 节）。 目标从最高 qvalue 到最低处理。 可以并行处理具有相等 qvalue 的目标。

有状态代理必须有一种机制来在收到响应时维护目标集，并将每个转发请求的响应与原始请求相关联。 就该模型而言，该机制是代理层在转发第一个请求之前创建的“响应上下文”。

对于每个目标，代理按照以下步骤转发请求：

1. 复制收到的请求

2.更新 Request-URI

3.更新 Max-Forwards 头域

4. 可选地添加一个 Record-route 头域值

5. 可选择添加额外的标题字段

6.后处理路由信息

7. 确定下一跳地址、端口和传输

8.添加一个 Via 头域值

9.必要时添加 Content-Length 头域

10.转发新请求

11.设置定时器 C

下面详细介绍了这些步骤中的每一个：

1.复制请求

代理从接收到的请求的副本开始。 副本必须最初包含来自接收到的请求的所有头字段。 不得删除下文所述处理中未详细说明的字段。 副本应该像接收到的请求一样保持头部字段的顺序。 代理不得使用公共字段名称重新排序字段值（参见第 7.3.1 节）。 代理不得添加、修改或删除消息正文。

实际的实现不需要执行复制； 主要要求是每个下一跳的处理都以相同的请求开始。

2.Request-URI

副本起始行中的 Request-URI 必须替换为此目标的 URI。如果 URI 包含 Request-URI 中不允许的任何参数，则必须删除它们。

这是代理角色的本质。 这是代理将请求路由到其目的地的机制。

在某些情况下，接收到的 Request-URI 被放入目标集中而不被修改。 对于那个目标，上面的替换实际上是一个空操作。

3. Max-Forwards

如果副本包含 Max-Forwards 头字段，则代理必须将其值减一 (1)。

如果副本不包含 Max-Forwards 头字段，则代理必须添加一个字段值，该字段值应为 70。

一些现有的 UA 不会在请求中提供 Max-Forwards 头字段。

4. Record-Route

如果此代理希望在此请求创建的对话中保留在未来请求的路径上（假设该请求创建了一个对话），它必须在任何现有 Record-Route 标头字段值之前将 Record-Route 标头字段值插入副本中 ，即使已经存在 Route 标头字段。

建立对话的请求可能包含预加载的 Route 头字段。

如果这个请求已经是对话的一部分，如果代理希望保留在对话中未来请求的路径上，它应该插入一个 Record-Route 头字段值。 在第 12 节中描述的正常端点操作中，这些 Record-Route 头字段值不会对端点使用的路由集产生任何影响。

如果代理选择不将 Record-Route 标头字段值插入到已经是对话一部分的请求中，它将保留在路径上。 但是，当失败的端点重新构建对话时，它将从路径中删除。

代理可以将 Record-Route 头域值插入到任何请求中。 如果请求未启动对话，端点将忽略该值。 有关端点如何使用 Record-Route 标头字段值来构造 Route 标头字段的详细信息，请参阅第 12 节。

请求路径中的每个代理都选择是否独立添加 Record-Route 标头字段值 - 请求中 Record-Route 标头字段的存在并不强制此代理添加值。

放置在 Record-Route 头字段值中的 URI 必须是 SIP 或 SIPS URI。这个 URI 必须包含一个 lr 参数（见第 19.1.1 节）。 对于请求转发到的每个目的地，此 URI 可能不同。 URI 不应该包含传输参数，除非代理知道（例如在专用网络中）将在后续请求路径中的下一个下游元素支持该传输。

此代理提供的 URI 将被其他一些元素用于做出路由决策。 通常，此代理无法知道该元素的功能，因此它必须将自己限制为 SIP 实现的强制元素：SIP URI 以及 TCP 或 UDP 传输。

当 [4] 的服务器定位程序应用于它时，放置在 Record-Route 头字段中的 URI 必须解析为插入它的元素（或合适的替代），以便后续请求到达相同的 SIP 元素。 如果 Request-URI 包含一个 SIPS URI，或者最上面的 Route 头域值（在第 6 条的后处理之后）包含一个 SIPS URI，那么放入 Record-Route 头域的 URI 必须是一个 SIPS URI。 此外，如果没有通过 TLS 接收到请求，则代理必须插入 Record-Route 头字段。 以类似的方式，代理通过 TLS 接收请求，但在 Request-URI 或最顶层 Route 标头字段值中生成没有 SIPS URI 的请求（在项目符号 6 的后处理之后），必须插入 Record-Route 标头 不是 SIPS URI 的字段。

在整个对话过程中，安全边界上的代理必须保持在边界上。

如果放置在 Record-Route 标头字段中的 URI 在响应中返回时需要重写，则 URI 必须足够不同，以便在那时定位。（请求可能会通过此代理盘旋，从而导致添加多个 Record-Route 标头字段值）。 第 16.7 节的第 8 项建议了一种使 URI 足够不同的机制。

代理可以在 Record-Route 头域值中包含参数。 这些将在对请求的某些响应中得到回应，例如对 INVITE 的 200（OK）响应。 这样的参数对于在消息中而不是代理中保持状态可能很有用。

如果代理需要在任何类型的对话的路径中（例如跨越防火墙），它应该使用它不理解的方法为每个请求添加一个 Record-Route 头字段值，因为该方法可能具有对话语义。

代理放入 Record-Route 头字段的 URI 仅在其发生的事务创建的任何对话的生命周期内有效。 例如，对话状态代理可以在对话终止后拒绝接受 Request-URI 中具有该值的未来请求。当然，非对话状态代理不知道对话何时终止，但它们可以在值中编码足够的信息以将其与未来请求的对话标识符进行比较，并可以拒绝与该信息不匹配的请求。 端点不得使用从提供它的对话框之外的 Record-Route 头字段获得的 URI。 有关端点使用 Record-Route 标头字段的更多信息，请参阅第 12 节。

某些服务可能需要记录路由，其中代理需要观察对话中的所有消息。 但是，它会减慢处理

速度并损害可扩展性，因此代理应该仅在特定服务需要时记录路由。

Record-Route 过程设计用于任何发起对话的 SIP 请求。INVITE 是本规范中唯一的此类请求，但协议的扩展可以定义其他请求。

5.添加额外的标题字段

此时代理可以将任何其他适当的头字段添加到副本中。

6.后处理路由信息

代理可能有一个本地策略，该策略要求请求在传递到目的地之前访问一组特定的代理。 代理必须确保所有这些代理都是松散的路由器。 通常，只有在代理位于同一管理域内时，才能确定地知道这一点。 这组代理由一组 URI 表示（每个都包含 lr 参数）。 该集合必须在任何现有值（如果存在）之前推送到副本的 Route 头字段中。 如果 Route 标头字段不存在，则必须添加它，包含该 URI 列表。

如果代理有一个本地策略要求请求访问一个特定的代理，则将 Route 值推送到 Route 标头字段的替代方法是绕过下面第 10 项的转发逻辑，而只是将请求发送到该地址， 该特定代理的端口和传输。 如果请求具有 Route 头字段，则不得使用此替代方案，除非知道下一跳代理是松散路由器。 否则，可以使用这种方法，但首选上面的路由插入机制，因为它的健壮性、灵活性、通用性和操作的一致性。 此外，如果 Request-URI 包含 SIPS URI，则必须使用 TLS 与该代理进行通信。

如果副本包含 Route 头字段，则代理必须检查 URI 的第一个值。如果该 URI 不包含 lr 参数，则代理必须按如下方式修改副本：

- 代理必须将 Request-URI 作为最后一个值放入 Route 头字段。

- 代理必须然后将第一个 Route 头字段值放入 Request-URI 并从 Route 头字段中删除该值。

将 Request-URI 附加到 Route 标头字段是用于通过严格路由元素传递 Request-URI 中的信息的机制的一部分。 将第一个 Route 标头字段值“弹出”到 Request-URI 中，按照严格路由元素期望接收它的方式格式化消息（在 Request-URI 中使用自己的 URI，在第一个 Route 标头字段中访问下一个位置 价值）。

7. 确定下一跳地址、端口和传输

代理可以有一个本地策略来将请求发送到特定的 IP 地址、端口和传输，独立于 Route 和 Request-URI 的值。如果代理不确定 IP 地址、端口和传输是否对应于松散路由器的服务器，则不得使用此类策略。 但是，不推荐这种通过特定下一跳发送请求的机制； 相反，如上所述，应为此目的使用 Route 标头字段。

在没有这种覆盖机制的情况下，代理将应用 [4] 中列出的过程，如下所示来确定将请求发送到何处。 如果代理重新格式化了请求以发送到上述步骤 6 中描述的严格路由元素，则代理必须将这些过程应用于请求的 Request-URI。 否则，代理必须将过程应用于 Route 头字段中的第一个值（如果存在），否则为 Request-URI。 这些过程将产生一组有序的（地址、端口、传输）元组。 与哪个 URI 被用作 [4] 的过程的输入无关，如果 Request-URI 指定一个 SIPS 资源，则代理必须遵循 [4] 的过程，就好像输入 URI 是一个 SIPS URI。

如 [4] 中所述，代理必须尝试将消息传递到该集合中的第一个元组，并按顺序处理该集合，直到传递尝试成功。

对于每个尝试的元组，代理必须根据元组对消息进行适当的格式化，并使用步骤 8 到 10 中详述的新客户端事务发送请求。

由于每次尝试都使用一个新的客户端事务，它代表一个新的分支。 因此，在步骤 8 中插入的 Via 头字段提供的分支参数对于每次尝试都必须不同。

如果客户端事务从其状态机报告发送请求失败或超时，则代理继续到该有序集中的下一个地址。 如果有序集合耗尽，则请求无法转发到目标集合中的该元素。 代理不需要在响应上下文中放置任何东西，但在其他方面就像目标集的这个元素返回 408（请求超时）最终响应一样。

8.添加一个 Via 头域值

代理必须在现有的 Via 头域值之前插入一个 Via 头域值到副本中。 该值的构造遵循第 8.1.1.7 节的相同准则。 这意味着代理将计算其自己的分支参数，该参数对于该分支将是全局唯一的，并包含必要的魔术 cookie。 请注意，这意味着对于通过代理的螺旋式或循环式请求的不同实例，分支参数将是不同的。

选择检测循环的代理对其用于构建分支参数的值有额外的限制。 选择检测循环的代理应该创建一个分支参数，该参数可被实现分为两部分。 如上所述，第一部分必须满足第 8.1.1.7 节的约束。 第二个用于执行循环检测并将循环与螺旋区分开来。

循环检测是通过验证当请求返回到代理时，那些对请求处理有影响的字段没有改变来执行的。放置在分支参数这部分的值应该反映所有这些字段（包括任何 Route、Proxy-Require 和 Proxy-Authorization 头字段）。这是为了确保如果请求被路由回代理并且其中一个字段发生更改，它将被视为螺旋而不是循环（参见第 16.3 节）。创建此值的常用方法是计算 To 标记、From 标记、Call-ID 标头字段、接收到的请求的 Request-URI（转换前）、最上面的 Via 标头和来自的序列号的加密哈希除了可能存在的任何 Proxy-Require 和 Proxy-Authorization 头字段之外，CSeq 头字段。用于计算散列的算法取决于实现，但以十六进制表示的 MD5 (RFC 1321 [35]) 是一个合理的选择。 （令牌不允许使用 Base64。）

如果代理希望检测循环，它提供的“分支”参数必须依赖于影响请求处理的所有信息，包括传入的 Request-URI 和任何影响请求准入或路由的头字段。 这对于区分循环请求和返回此服务器之前路由参数已更改的请求是必要的。

请求方法不得包含在分支参数的计算中。 特别是，CANCEL 和 ACK 请求（对于非 2xx 响应）必须具有与它们取消或确认的相应请求相同的分支值。 分支参数用于在处理这些请求的服务器上关联这些请求（参见第 17.2.3 和 9.2 节）。

9.必要时添加 Content-Length 头域

如果请求将使用基于流的传输发送到下一跳，并且副本不包含 Content-Length 头字段，则代理必须插入一个具有正确值的请求正文（参见第 20.14 节）。

10.转发请求

有状态代理必须为该请求创建一个新的客户端事务，如第 17.1 节所述，并指示事务使用步骤 7 中确定的地址、端口和传输来发送请求。

11.设置定时器 C

为了处理 INVITE 请求永远不会产生最终响应的情况，TU 使用称为定时器 C 的定时器。当 INVITE 请求被代理时，必须为每个客户端事务设置定时器 C。计时器必须大于 3 分钟。第 16.7 节第 2 节讨论如何使用临时响应更新此计时器，第 16.8 节讨论触发时的处理。

16.7 响应处理

当元素接收到响应时，它首先尝试定位与响应匹配的客户端事务（第 17.1.3 节）。 如果没有找到，元素必须将响应（即使它是信息响应）作为无状态代理（如下所述）处理。 如果找到匹配项，则将响应交给客户端事务。

转发未找到客户端事务（或更一般地说，任何有关已发送相关请求的知识）的响应提高了健壮性。 特别是，它确保正确转发对 INVITE 请求的“延迟”2xx 响应。

当客户端事务将响应传递给代理层时，必须进行以下处理：

1. 找到合适的响应上下文

2. 更新计时器 C 以获得临时响应

3.去掉最上面的 Via

4.将响应添加到响应上下文

5. 检查是否应立即转发此响应

6.必要时，从响应上下文中选择最佳的最终响应

如果在与响应上下文关联的每个客户端事务终止后没有转发最终响应，则代理必须从它迄今为止看到的那些中选择并转发“最佳”响应。

必须对转发的每个响应执行以下处理。 很可能会转发对每个请求的多个响应：至少每个临时响应和一个最终响应。

7. 必要时聚合授权头字段值

8.可选地重写 Record-Route 头域值

9.转发回复

10. 生成任何必要的 CANCEL 请求

上述每个步骤的详细信息如下：

1. 查找上下文

代理使用第 16.6 节中描述的密钥来定位它在转发原始请求之前创建的“响应上下文”。 其余处理步骤在此上下文中进行。

2. 更新计时器 C 以获得临时响应

对于 INVITE 事务，如果响应是状态码为 101 到 199 的临时响应（即，除了 100 之外的任何值），代理必须为该客户端事务重置计时器 C。 计时器可以重置为不同的值，但该值必须大于 3 分钟。

3. Via

代理从响应中删除最顶层的 Via 头字段值。

如果响应中没有保留 Via 头字段值，则响应是针对此元素的，并且不得转发。 本节中描述的其余处理不在此消息上执行，而是遵循第 8.1.3 节中描述的 UAC 处理规则（传输层处理已经发生）。

例如，当元素生成第 10 节中描述的 CANCEL 请求时，就会发生这种情况。

4.添加对上下文的响应

收到的最终响应存储在响应上下文中，直到在与此上下文关联的服务器事务上生成最终响应。该响应可能是要在该服务器事务上返回的最佳最终响应的候选者。 即使未选择此响应，也可能需要来自此响应的信息来形成最佳响应。

如果代理选择通过将 3xx 响应中的任何联系人添加到目标集来递归它们，则它必须在将响

应添加到响应上下文之前将它们从响应中删除。 但是，如果原始请求的 Request-URI 是 SIPS URI，则代理不应递归到非 SIPS URI。 如果代理在 3xx 响应中对所有联系人进行递归，则代理不应将生成的非接触式响应添加到响应上下文中。

在将响应添加到响应上下文之前删除联系人可以防止上游的下一个元素重试该代理已经尝试过的位置。

3xx 响应可能包含 SIP、SIPS 和非 SIP URI 的混合。 代理可以选择在 SIP 和 SIPS URI 上进行递归，并将其余部分放入要返回的响应上下文中，可能在最终响应中。

如果代理收到一个请求的 416（不支持的 URI 方案）响应，该请求的 Request-URI 方案不是 SIP，但最初收到的请求中的方案是 SIP 或 SIPS（即代理将方案从 SIP 或 SIPS 更改为代理请求时的其他内容），代理应该向目标集添加一个新的 URI。 此 URI 应该是刚刚尝试过的非 SIP URI 的 SIP URI 版本。 对于 tel URL，这是通过将 tel URL 的电话用户部分放入 SIP URI 的用户部分，并将主机部分设置为发送先前请求的域来实现的。 有关从 tel URL 形成 SIP URI 的更多详细信息，请参阅第 19.1.6 节。

与 3xx 响应一样，如果代理通过尝试 SIP 或 SIPS URI 在 416 上“递归”，则不应将 416 响应添加到响应上下文中。

5.检查转发响应

在服务器事务上发送最终响应之前，必须立即转发以下响应：

- 100(Trying) 以外的任何临时响应

- 任何 2xx 响应

如果收到 6xx 响应，它不会立即转发，但有状态代理应该取消所有客户端挂起的事务，如第 10 节所述，并且不得在此上下文中创建任何新分支。

这是对 RFC 2543 的更改，RFC 2543 要求代理立即转发 6xx 响应。 对于 INVITE 事务，这种方法的问题是 2xx 响应可能会到达另一个分支，在这种情况下，代理必须转发 2xx。 结果是 UAC 可以收到一个 6xx 响应，然后是一个 2xx 响应，这是绝对不允许发生的。 根据新规则，在收到 6xx 后，代理将发出 CANCEL 请求，这通常会导致所有未完成的客户端事务的 487 响应，然后在此时将 6xx 转发到上游。

在服务器事务上发送最终响应后，必须立即转发以下响应：

- 对 INVITE 请求的任何 2xx 响应

有状态代理不得立即转发任何其他响应。 特别是，有状态代理不得转发任何 100（尝试）响应。 如步骤“将响应添加到上下文”中所述，已收集那些作为稍后作为“最佳”响应转发的候选响应。

任何选择用于立即转发的响应必须按照“聚合授权头字段值”到“记录路由”步骤中的描述进行处理。

此步骤与下一步相结合，确保有状态代理将准确地转发一个对非邀请请求的最终响应，以及准确地转发一个非 2xx 响应或一个或多个 2xx 响应到邀请请求。

6. 选择最佳响应

如果上述规则没有立即转发最终响应并且此响应上下文中的所有客户端事务都已终止，则有状态代理必须向响应上下文的服务器事务发送最终响应。

有状态代理必须在响应上下文中接收和存储的响应中选择“最佳”最终响应。

如果上下文中没有最终响应，则代理必须向服务器事务发送 408（请求超时）响应。

否则，代理必须转发来自存储在响应上下文中的响应的响应。 如果上下文中存在任何 6xx 类响应，它必须从 6xx 类响应中进行选择。 如果不存在 6xx 类响应，代理应该从存储在响应上下文中的最低响应类中进行选择。 代理可以选择该类中的任何响应。 如果选择了 4xx 类，代理应该优先考虑提供影响重新提交此请求的信息的响应，例如 401、407、415、420 和 484。

接收到 503（服务不可用）响应的代理不应将其转发到上游，除非它可以确定它可能代理的任何后续请求也会生成 503。换句话说，转发 503 意味着代理知道它无法为任何请求提供服务 ，而不仅仅是生成 503 的请求中的 Request-URI 的那个。如果收到的唯一响应是 503，则代理应该生成 500 响应并将该响应转发到上游。

转发的响应必须按照“Aggregate Authorization Header Field Values”到“Record-Route”步骤中的描述进行处理。

例如，如果代理将请求转发到 4 个位置，并收到 503、407、501 和 404 响应，它可能会选择转发 407（需要代理身份验证）响应。

1xx 和 2xx 响应可能参与对话的建立。 当请求不包含 To 标记时，UAC 使用响应中的 To 标记来区分对对话创建请求的多个响应。 如果请求不包含标签，则代理不得在 1xx 或 2xx 响应的 To 头字段中插入标签。 代理不得修改 1xx 或 2xx 响应的 To 头字段中的标签。

由于代理可能不会在对不包含标签的请求的 1xx 响应的 To 头字段中插入标签，因此它不能自行发出非 100 临时响应。但是，它可以将请求分支到与代理共享相同元素的 UAS。这个 UAS 可以返回它自己的临时响应，进入与请求发起者的早期对话。 UAS 不必是来自代理的谨慎过程。 它可以是在与代理相同的代码空间中实现的虚拟 UAS。

3-6xx 响应逐跳传递。 当发出 3-6xx 响应时，该元素实际上充当 UAS，发出自己的响应，通常基于从下游元素接收到的响应。 当简单地将 3-6xx 响应转发给不包含 To 标记的请求

时，元素应该保留 To 标记。

代理不得修改对包含 To 标记的请求的任何转发响应中的 To 标记。

如果代理替换转发的 3-6xx 响应中的 To 标记，对上游元素没有影响，但保留原始标记可能有助于调试。

当代理从多个响应中聚合信息时，从其中选择一个 To 标记是任意的，生成一个新的 To 标记可能会使调试更容易。 例如，当组合 401（未经授权）和 407（需要代理身份验证）质询，或组合来自未加密和未经身份验证的 3xx 响应的联系人值时，就会发生这种情况。

7. 聚合授权标头字段值（Aggregate Authorization Header Field Values）

如果选择的响应是 401（未授权）或 407（需要代理身份验证），则代理必须从收到的所有其他 401（未授权）和 407（需要代理身份验证）响应中收集任何 WWW-Authenticate 和 Proxy-Authenticate 头字段值，以便 远在此响应上下文中，并将它们添加到此响应中而无需在转发之前进行修改。 产生的 401（未授权）或 407（需要代理验证）响应可能有多个 WWW-Authenticate AND Proxy-Authenticate 头字段值。

这是必要的，因为请求转发到的任何或所有目的地都可能已请求凭据。 客户端需要在重试请求时接收所有这些挑战并为每个挑战提供凭据。 第 26 节提供了这种行为的动机。

8. Record-Route

如果选择的响应包含由该代理最初提供的 Record-Route 头字段值，则代理可以选择在转发响应之前重写该值。 这允许代理为自己的下一个上游和下游元素提供不同的 URI。 代理可以出于任何原因选择使用此机制。 例如，它对多宿主主机很有用。

如果代理通过 TLS 接收到请求，并通过非 TLS 连接将其发送出去，则代理必须将 Record-Route 头字段中的 URI 重写为 SIPS URI。 如果代理通过非 TLS 连接接收请求，并通过 TLS 将其发送出去，则代理必须将 Record-Route 头字段中的 URI 重写为 SIP URI。

代理提供的新 URI 必须满足对放置在请求中 Record-Route 头字段中的 URI 的相同约束（参见第 16.6 节的步骤 4），并进行以下修改：

URI 不应该包含传输参数，除非代理知道将在后续请求路径中的下一个上游（与下游相反）元素支持该传输。

当代理确实决定修改响应中的 Record-Route 标头字段时，它执行的操作之一是定位它已插入的 Record-Route 值。 如果请求呈螺旋式上升，并且代理在螺旋式的每次迭代中都插入了一个 Record-Route 值，那么在响应中定位正确的值（必须是反向的正确迭代）是很棘手的。 上述规则建议希望重写 Record-Route 头字段值的代理将足够不同的 URI 插入 Record-Route 头字段，以便可以选择正确的进行重写。 实现此目的的推荐机制是代理将代理实例的唯一标识符附加到 URI 的用户部分。

当响应到达时，代理修改第一个标识符与代理实例匹配的 Record-Route。 修改会生成一个 URI，而不会将此数据附加到 URI 的用户部分。 在下一次迭代中，相同的算法（使用参数查找最顶层的 Record-Route 头字段值）将正确提取该代理插入的下一个 Record-Route 头字段值。

并非对代理添加 Record-Route 标头字段值的请求的每个响应都将包含 Record-Route 标头字段。 如果响应确实包含 Record-Route 头字段，它将包含代理添加的值。

9. Forward response

在执行步骤“聚合授权头字段值”到“记录路由”中描述的处理之后，代理可以对选定的响应执行任何特定于功能的操作。代理不得添加、修改或删除消息正文。除非另有说明，否则代理不得删除除第 16.7 节第 3 项中讨论的 Via 头字段值之外的任何头字段值。特别是，代理不得删除它可能已添加到下一个 Via 头字段的任何“接收”参数处理与此响应关联的请求时的值。代理必须将响应传递给与响应上下文关联的服务器事务。这将导致响应被发送到现在在最上面的 Via 头字段值中指示的位置。如果服务器事务不再可用于处理传输，则元素必须通过将响应发送到服务器传输来无状态地转发响应。服务器事务可能指示发送响应失败或在其状态机中发出超时信号。这些错误将被记录以用于适当的诊断目的，但协议不需要代理采取任何补救措施。

代理必须维护响应上下文，直到其所有关联事务都已终止，即使在转发最终响应之后也是如此。

10. Generate CANCELs

如果转发的响应是最终响应，则代理必须为与此响应上下文关联的所有未决客户端事务生成 CANCEL 请求。 当代理接收到 6xx 响应时，它还应该为与此响应上下文关联的所有未决客户端事务生成 CANCEL 请求。 待处理的客户端事务是已收到临时响应但没有最终响应（处于进行中状态）并且尚未为其生成关联的 CANCEL 的事务。 生成 CANCEL 请求在第 9.1 节中描述。

在转发最终响应时取消挂起的客户端事务的要求并不能保证端点不会收到对 INVITE 的多个 200（OK）响应。 在发送和处理 CANCEL 请求之前，可能会在多个分支上生成 200（OK）响应。 此外，可以合理地预期未来的扩展可能会覆盖此要求以发出 CANCEL 请求。

16.8 处理定时器 C

如果计时器 C 应该触发，代理必须要么用它选择的任何值重置计时器，要么终止客户端事务。 如果客户端事务已收到临时响应，则代理必须生成与该事务匹配的 CANCEL 请求。 如果客户端事务没有收到临时响应，代理必须表现得好像事务收到了 408（请求超时）响应。

允许代理重置计时器允许代理在计时器触发时根据当前条件（例如利用率）动态延长事务的生命周期。

16.9 处理传输错误

如果传输层在尝试转发请求时通知代理错误（参见第 18.4 节），代理必须表现得好像转发的请求收到了 503（服务不可用）响应。

如果代理在转发响应时收到错误通知，它会丢弃响应。 由于此通知，代理不应取消与此响应上下文关联的任何未完成的客户端事务。

如果代理取消其未完成的客户端事务，单个恶意或行为不端的客户端可能会通过其 Via 头字段导致所有事务失败。

16.10 取消处理

有状态的代理可以对它在任何时候生成的任何其他请求生成一个 CANCEL（受制于收到对该请求的临时响应，如第 9.1 节所述）。 代理必须在收到匹配的 CANCEL 请求时取消与响应上下文关联的任何未决客户端事务。

有状态代理可以根据 INVITE 的 Expires 标头字段中指定的时间段为挂起的 INVITE 客户端事务生成 CANCEL 请求。 但是，这通常是不必要的，因为所涉及的端点将负责发出事务结束的信号。

虽然 CANCEL 请求由其自己的服务器事务在有状态代理中处理，但不会为其创建新的响应上下文。 相反，代理层在其现有响应上下文中搜索处理与此 CANCEL 关联的请求的服务器事务。 如果找到匹配的响应上下文，元素必须立即向 CANCEL 请求返回 200（OK）响应。在这种情况下，该元素充当第 8.2 节中定义的用户代理服务器。 此外，该元素必须为上下文中的所有未决客户端事务生成 CANCEL 请求，如第 16.7 节步骤 10 中所述。

如果未找到响应上下文，则元素不知道应用 CANCEL 的请求。它必须无状态地转发 CANCEL 请求（它之前可能已经无状态地转发了关联的请求）。

16.11 无状态代理

当无状态行为时，代理是一个简单的消息转发器。 无状态行为时执行的大部分处理与有状态行为时相同。 此处详细说明了差异。

无状态代理没有任何事务的概念，或用于描述有状态代理行为的响应上下文。 相反，无状态代理直接从传输层获取消息，包括请求和响应（参见第 18 节）。 因此，无状态代理不会自行重传消息。 但是，它们确实会转发它们收到的所有重传（它们没有能力将重传与原始消息区分开来）。 此外，当无状态处理请求时，元素不得生成自己的 100（尝试中）或任何其他临时响应。

无状态代理必须验证第 16.3 节中描述的请求。

无状态代理必须遵循第 16.4 到 16.5 节中描述的请求处理步骤，但以下例外：

o 无状态代理必须从目标集中选择一个且只有一个目标。 此选择必须仅依赖于消息中的字段和服务器的时不变属性。 特别是，每次处理重传请求时，必须将其转发到相同的目的地。此外，CANCEL 和非路由 ACK 请求必须生成与其关联的 INVITE 相同的选择。

无状态代理必须遵循第 16.6 节中描述的请求处理步骤，但以下例外：

o 跨空间和时间的唯一分支 ID 的要求也适用于无状态代理。 但是，无状态代理不能简单地使用随机数生成器来计算分支 ID 的第一个组件，如第 16.6 节第 8 节所述。这是因为请求的重传需要具有相同的值，而无状态代理无法分辨 从原始请求重新传输。 因此，每次转发重传请求时，使其唯一的分支参数组件必须相同。 因此，对于无状态代理，必须将分支参数计算为消息参数的组合函数，这些参数在重传时是不变的。

无状态代理可以使用它喜欢的任何技术来保证其分支 ID 在事务中的唯一性。 但是，建议使用以下程序。 代理检查接收到的请求的最上面的 Via 头字段中的分支 ID。 如果它以魔术 cookie 开头，则传出请求的分支 ID 的第一个组成部分被计算为接收到的分支 ID 的哈希值。 否则，分支 ID 的第一个组件被计算为最顶层 Via、To 头字段中的标签、From 头字段中的标签、Call-ID 头字段、CSeq 编号（但不是方法）的哈希 ，以及收到的请求中的 Request-URI。 这些字段之一将始终在两个不同的事务中有所不同。

o 第 16.6 节中指定的所有其他消息转换必须导致重传请求的相同转换。 特别是，如果代理插入 Record-Route 值或将 URI 推送到 Route 头字段中，它必须在请求的重传中放置相同的值。 至于 Via 分支参数，这意味着转换必须基于请求的时不变配置或重传不变属性。

o 无状态代理确定将请求转发到何处，如第 16.6 节第 10 节中对有状态代理的描述。请求直接发送到传输层，而不是通过客户端事务。

由于无状态代理必须将重新传输的请求转发到相同的目的地并向每个请求添加相同的分支参数，因此它只能使用来自消息本身的信息和时不变的配置数据进行这些计算。 如果配置状态不是时不变的（例如，如果路由表被更新），任何可能受更改影响的请求可能不会在等于更改之前或之后的事务超时窗口的时间间隔内无状态转发。 在该时间间隔内处理受影响请求的方法是实施决策。 一个常见的解决方案是有状态地转发它们的事务。

无状态代理不得对 CANCEL 请求执行特殊处理。 它们与任何其他请求一样由上述规则处理。特别是，无状态代理将相同的 Route 标头字段处理应用于 CANCEL 请求，它应用于任何其他请求。

第 16.7 节中描述的响应处理不适用于无状态行为的代理。 当响应到达无状态代理时，代理必须检查第一个（最顶层）Via 头字段值中的 sent-by 值。 如果该地址与代理匹配，（它等于该代理已插入先前请求的值）代理必须从响应中删除该头字段值并将结果转发到下一个 Via 头字段值中指示的位置。 代理不得添加、修改或删除消息正文。 除非另有说明，否则代理不得删除任何其他头字段值。 如果地址与代理不匹配，则消息必须被静默丢弃。

16.12 代理路由处理总结

相反，在没有本地策略的情况下，代理对包含 Route 头字段的请求执行的处理可以总结为以下步骤。

1. 代理将检查 Request-URI。 如果它指示此代理拥有的资源，则代理将用运行位置服务的结果替换它。 否则，代理不会更改 Request-URI。

2. 代理将检查最顶层 Route 头字段值中的 URI。 如果它指示此代理，则代理将其从 Route 头字段中删除（已到达此路由节点）。

3. 如果没有 Route 头域，proxy 会将请求转发到 Route 头域值中的 URI 或 Request-URI 中指示的资源。 代理通过将 [4] 中的过程应用于该 URI 来确定转发请求时要使用的地址、端口和传输。

如果在请求的路径上没有遇到严格路由元素，Request-URI 将始终指示请求的目标。

16.12.1 示例

16.12.1.1 基本 SIP 梯形

这个场景是基本的 SIP 梯形，U1 -> P1 -> P2 -> U2，两个代理记录路由。 这是流程。

U1 发送：

```
INVITE sip:callee@domain.com SIP/2.0
Contact: sip:caller@u1.example.com
```

到 P1。 P1 是出站代理。 P1 不对 domain.com 负责，因此它在 DNS 中查找它并将其发送到那里。 它还添加了一个 Record-Route 标头字段值：

```
INVITE sip:callee@domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p1.example.com;lr>
```

P2 得到这个。 它负责 domain.com，因此它运行位置服务并重写 Request-URI。 它还添加了一个 Record-Route 标头字段值。 没有 Route 头字段，因此它解析新的 Request-URI 以确定将请求发送到哪里：

```
INVITE sip:callee@u2.domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p2.domain.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

u2.domain.com 的被调用者得到这个并以 200 OK 响应：

```
SIP/2.0 200 OK
Contact: sip:callee@u2.domain.com
Record-Route: <sip:p2.domain.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

u2 的被调用者还将其对话状态的远程目标 URI 设置为 sip:caller@u1.example.com 并将其路由设置为：

```
(<sip:p2.domain.com;lr>,<sip:p1.example.com;lr>)
```

正常情况下，这由 P2 转发到 P1 到 U1。 现在，U1 将其对话状态的远程目标 URI 设置为 sip:callee@u2.domain.com 并将其路由设置为：

```
(<sip:p1.example.com;lr>,<sip:p2.domain.com;lr>)
```

由于所有路由集元素都包含 lr 参数，因此 U1 构造了以下 BYE 请求：

```
BYE sip:callee@u2.domain.com SIP/2.0
Route: <sip:p1.example.com;lr>,<sip:p2.domain.com;lr>
```

与任何其他元素（包括代理）一样，它使用 DNS 解析最顶层 Route 标头字段值中的 URI，以确定将请求发送到何处。 这转到 P1。 P1 注意到它不对 Request-URI 中指示的资源负责，因此它不会更改它。 它确实看到它是 Route 标头字段中的第一个值，因此它删除了该值，并将请求转发给 P2：

```
BYE sip:callee@u2.domain.com SIP/2.0
Route: <sip:p2.domain.com;lr>
```

P2 还注意到它不负责 Request-URI 指示的资源（它负责 domain.com，而不是 u2.domain.com），因此它没有更改它。 它确实在第一个 Route 标头字段值中看到了自己，因此它会删除它并根据针对 Request-URI 的 DNS 查找将以下内容转发到 u2.domain.com：

```
BYE sip:callee@u2.domain.com SIP/2.0
```

16.12.1.2 遍历严格路由代理

在这种情况下，会在四个代理之间建立一个对话，每个代理都会添加 Record-Route 标头字段值。 第三个代理实现了 RFC 2543 中指定的严格路由程序，许多工作正在进行中。

U1->P1->P2->P3->P4->U2

到达 U2 的 INVITE 包含：

```
INVITE sip:callee@u2.domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p4.domain.com;lr>
Record-Route: <sip:p3.middle.com>
Record-Route: <sip:p2.example.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

U2 以 200 OK 响应。 之后，U2 根据第一个 Route 头域的值向 P4 发送如下 BYE 请求。

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p4.domain.com;lr>
Route: <sip:p3.middle.com>
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
```

P4 不对 Request-URI 中指示的资源负责，因此它将不理会它。 它注意到它是第一个 Route 标头字段值中的元素，因此将其删除。 然后它根据 sip:p3.middle.com 的第一个 Route 头字段值准备发送请求，但它注意到此 URI 不包含 lr 参数，因此在发送之前，它将请求重新格式化为：

```
BYE sip:p3.middle.com SIP/2.0
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

P3 是一个严格的路由器，因此它将以下内容转发给 P2：

```
BYE sip:p2.example.com;lr SIP/2.0
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

P2 看到 request-URI 是它放入 Record-Route 头字段的值，因此在进一步处理之前，它将请求重写为：

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p1.example.com;lr>
```

P2 不对 u1.example.com 负责，所以它根据 Route 头域值的解析向 P1 发送请求。

P1 在最顶层的 Route 标头字段值中注意到自己，因此将其删除，结果是：

```
BYE sip:caller@u1.example.com SIP/2.0
```

由于 P1 不对 u1.example.com 负责，也没有 Route 头域，所以 P1 会根据 Request-URI 将请求转发到 u1.example.com。

16.12.1.3 重写记录路由头字段值

在这种情况下，U1 和 U2 位于不同的私有命名空间中，它们通过代理 P1 进入对话，代理 P1 充当命名空间之间的网关。

U1->P1->U2

U1 发送：

```
INVITE sip:callee@gateway.leftprivatespace.com SIP/2.0
Contact: <sip:caller@u1.leftprivatespace.com>
```

P1 使用其位置服务并向 U2 发送以下信息：

```
INVITE sip:callee@rightprivatespace.com SIP/2.0
Contact: <sip:caller@u1.leftprivatespace.com>
Record-Route: <sip:gateway.rightprivatespace.com;lr>
```

U2 将此 200 (OK) 发送回 P1：

```
SIP/2.0 200 OK
Contact: <sip:callee@u2.rightprivatespace.com>
Record-Route: <sip:gateway.rightprivatespace.com;lr>
```

P1 重写其 Record-Route 标头参数以提供 U1 会发现有用的值，并将以下内容发送到 U1：

```
SIP/2.0 200 OK
Contact: <sip:callee@u2.rightprivatespace.com>
Record-Route: <sip:gateway.leftprivatespace.com;lr>
```

随后，U1 向 P1 发送以下 BYE 请求：

```
BYE sip:callee@u2.rightprivatespace.com SIP/2.0
Route: <sip:gateway.leftprivatespace.com;lr>
```

其中 P1 转发给 U2 为：

```
BYE sip:callee@u2.rightprivatespace.com SIP/2.0
```

17 事务

SIP 是一种事务性协议：组件之间的交互发生在一系列独立的消息交换中。具体而言，SIP 事务由单个请求和对该请求的任何响应组成，其中包括零个或多个临时响应和一个或多个最终响应。 在请求是 INVITE 的事务（称为 INVITE 事务）的情况下，仅当最终响应不是 2xx 响应时，事务还包括 ACK。 如果响应是 2xx，则 ACK 不被视为事务的一部分。

这种分离的原因在于将所有 200 (OK) 响应传递给 UAC 的 INVITE 的重要性。 为了将它们全部传送给 UAC，UAS 独自负责重新传输它们（见第 13.3.1.4 节），而 UAC 独自负责用 ACK 确认它们（见第 13.2.2.4 节）。 由于此 ACK 仅由 UAC 重新传输，因此它实际上被视为它自己的事务。

事务有客户端和服务器端。 客户端称为客户端事务，服务器端称为服务器事务。 客户端事务发送请求，服务器事务发送响应。 客户端和服务器事务是嵌入在任意数量的元素中的逻辑功能。 具体来说，它们存在于用户代理和有状态代理服务器中。 考虑第 4 节中的示例。在此示例中，UAC 执行客户端事务，其出站代理执行服务器事务。 出站代理还执行客户端事务，该事务将请求发送到入站代理中的服务器事务。 该代理还执行客户端事务，然后将请求发送到 UAS 中的服务器事务。 如图 4 所示。

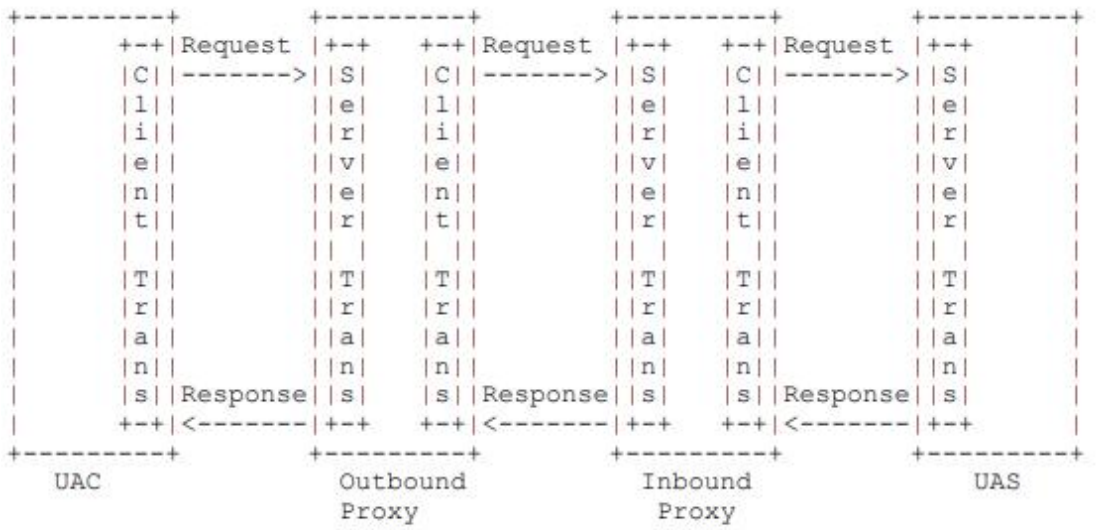

Figure 4: Transaction relationships

无状态代理不包含客户端或服务器事务。 事务存在于一侧的 UA 或有状态代理与另一侧的 UA 或有状态代理之间。 就 SIP 事务而言，无状态代理实际上是透明的。 客户端事务的目的是接收来自嵌入客户端的元素（将此元素称为“事务用户”或 TU；它可以是 UA 或有状态代理）的请求，并将请求可靠地传递给 服务器事务。

客户端事务还负责接收响应并将其传递给 TU，过滤掉任何响应重传或不允许的响应（例如对 ACK 的响应）。 此外，在 INVITE 请求的情况下，客户端事务负责为任何接受 2xx 响应的最终响应生成 ACK 请求。

同样，服务器事务的目的是接收来自传输层的请求并将它们传递给 TU。 服务器事务过滤来自网络的任何请求重传。 服务器事务接受来自 TU 的响应并将它们传递到传输层以通过网络传输。 在 INVITE 事务的情况下，它会吸收除 2xx 响应之外的任何最终响应的 ACK 请求。

2xx 响应及其 ACK 接受特殊处理。 此响应仅由 UAS 重传，其 ACK 仅由 UAC 生成。 需要这种端到端处理，以便呼叫者知道已接受呼叫的整个用户集。 由于这种特殊处理，2xx 响应的重传由 UA 核心处理，而不是事务层。 类似地，2xx 的 ACK 生成由 UA 核心处理。 路径上的每个代理仅将每个 2xx 响应转发给 INVITE 及其相应的 ACK。

17.1 客户事务

客户端事务通过维护状态机来提供其功能。

TU 通过一个简单的接口与客户端事务进行通信。 当 TU 希望启动一个新事务时，它会创建一个客户端事务，并将 SIP 请求传递给它以发送，以及一个 IP 地址、端口和传输到它的发送对象。 客户端事务开始执行其状态机。 有效响应从客户端事务向上传递到 TU。

有两种类型的客户端事务状态机，这取决于 TU 传递的请求的方法。 一个处理 INVITE 请求的客户端事务。这种类型的机器称为 INVITE 客户端事务。另一种类型处理除 INVITE 和

ACK 之外的所有请求的客户端事务。 这称为非邀请客户事务。 ACK 没有客户端事务。 如果 TU 希望发送一个 ACK，它会直接将一个 ACK 传递给传输层进行传输。

INVITE 事务不同于其他方法，因为它的持续时间较长。 通常，需要人工输入才能响应邀请。预计发送响应的长时间延迟支持三次握手。 另一方面，其他方法的请求有望快速完成。 由于非 INVITE 事务依赖于双向握手，TU 应该立即响应非 INVITE 请求。

17.1.1 INVITE 客户事务

17.1.1.1 INVITE 事务概述

INVITE 事务由三次握手组成。客户端事务发送 INVITE，服务器事务发送响应，客户端事务发送 ACK。对于不可靠的传输（例如 UDP），客户端事务以从 T1 秒开始的间隔重新传输请求，并在每次重新传输后加倍。 T1 是往返时间 (RTT) 的估计值，默认为 500 毫秒。此处描述的几乎所有事务计时器都随 T1 缩放，并且更改 T1 会调整它们的值。请求不会通过可靠传输重新传输。收到 1xx 响应后，任何重传都会完全停止，客户端等待进一步的响应。服务器事务可以发送额外的 1xx 响应，服务器事务不能可靠地传输这些响应。最终，服务器事务决定发送最终响应。对于不可靠的传输，该响应会定期重新传输，而对于可靠的传输，它会发送一次。对于在客户端事务中接收到的每个最终响应，客户端事务都会发送一个 ACK ，其目的是终止响应的重新传输。

17.1.1.2 形式描述

INVITE 客户端事务的状态机如图 5 所示。当 TU 通过 INVITE 请求启动新的客户端事务时，必须进入初始状态“调用”。 客户端事务必须将请求传递给传输层进行传输（参见第 18 节）。如果使用了不可靠的传输，客户端事务必须以 T1 的值启动计时器 A。 如果使用可靠传输，客户端事务不应启动计时器 A（计时器 A 控制请求重传）。 对于任何传输，客户端事务必须启动定时器 B，其值为 64*T1 秒（定时器 B 控制事务超时）。

当计时器 A 触发时，客户端事务必须通过将请求传递给传输层来重新传输请求，并且必须将计时器重置为 2*T1 的值。 在事务层上下文中重传的正式定义是把之前发送到传输层的消息再次传递给传输层。

当计时器 A 在 2*T1 秒后触发时，请求必须再次重传（假设客户端事务仍处于此状态）。这个过程必须继续，以便在每次传输后以双倍的间隔重新传输请求。 这些重传应该只在客户端事务处于“调用”状态时进行。

T1 的默认值为 500 毫秒。 T1 是客户端和服务器事务之间 RTT 的估计值。 元素可以（尽管不推荐）在不允许一般 Internet 连接的封闭专用网络中使用较小的 T1 值。 T1 可以选择更大，如果事先知道（例如在高延迟接入链路上）RTT 更大，则建议这样做。 无论 T1 的值是多少，都必须使用本节中描述的重传指数退避。

如果定时器 B 触发时客户端事务仍处于“调用”状态，客户端事务应该通知 TU 发生超时。客户端事务不得生成 ACK。 64*T1 的值等于在传输不可靠的情况下发送七个请求所需的时间量。

如果客户端事务在“调用”状态下收到临时响应，它会转换到“进行中”状态。 在“Proceeding”状态下，客户端事务不应再重传请求。 此外，临时响应必须传递给 TU。 任何进一步的临时响应必须在“Proceeding”状态下传递给 TU。

当处于“Calling”或“Proceeding”状态时，接收到状态码为 300-699 的响应必须导致客户端事务转换为“Completed”。客户端事务必须将接收到的响应传递给 TU，并且客户端事务必须生成一个 ACK 请求，即使传输是可靠的（从响应构造 ACK 的指南在第 17.1.1.3 节中给出），然后传递向传输层发送 ACK 以进行传输。 ACK 必须发送到与发送原始请求相同的地址、端口和传输。当客户端事务进入“已完成”状态时，它应该启动计时器 D，对于不可靠传输，其值至少为 32 秒，对于可靠传输，其值为零秒。计时器 D 反映了在使用不可靠传输时服务器事务可以保持在“已完成”状态的时间量。这等于 INVITE 服务器事务中的 Timer H，其默认值为 64*T1。但是，客户端事务不知道服务器事务正在使用的 T1 的值，因此使用绝对最小值 32 秒而不是基于 T1 的 Timer D。

在“已完成”状态下接收到的最终响应的任何重传必须导致 ACK 被重新传递给传输层以进行重传，但新接收到的响应不得传递给 TU。 响应的重传被定义为根据第 17.1.3 节的规则匹配相同客户端事务的任何响应。

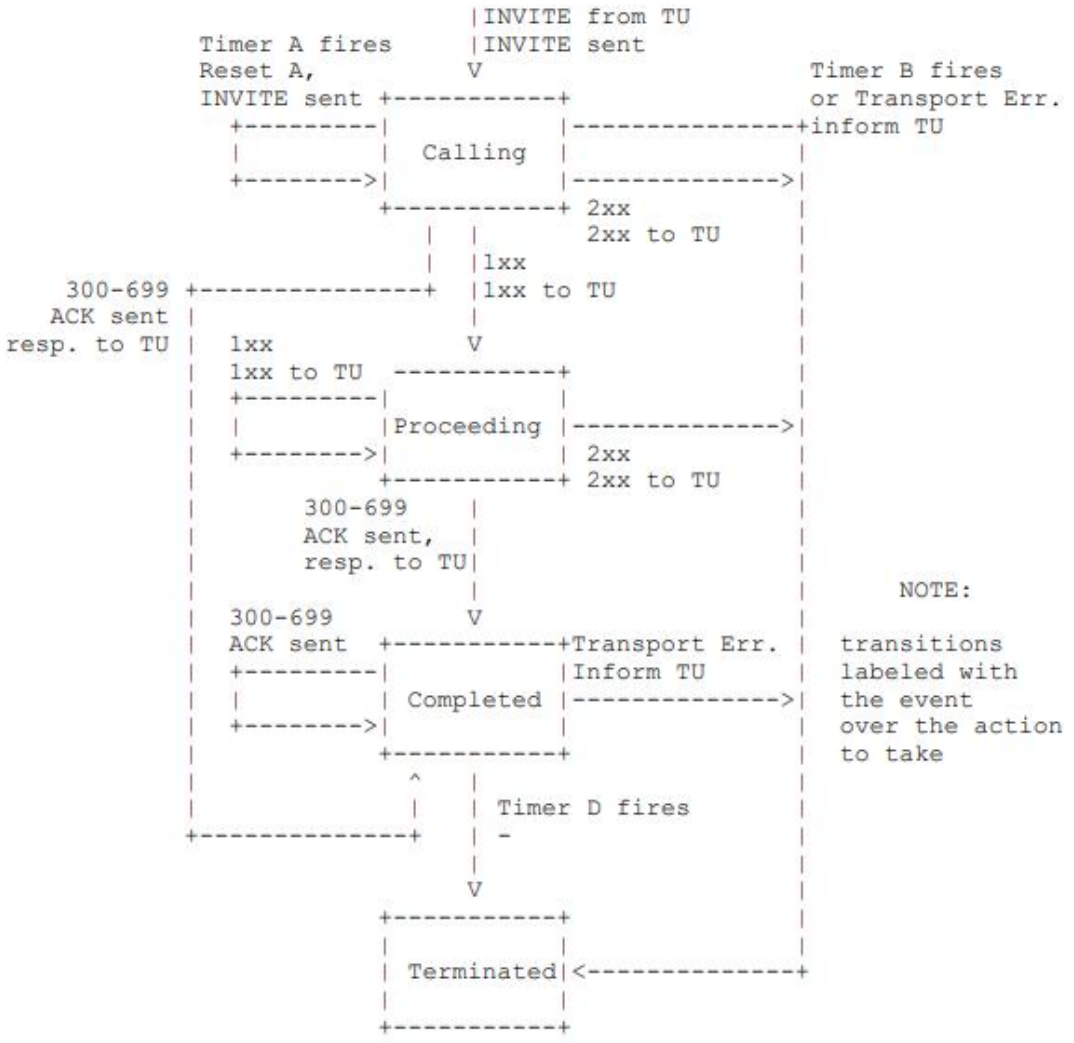

Figure 5: INVITE client transaction

如果在客户端事务处于“已完成”状态时定时器 D 触发，客户端事务必须移动到终止状态。

当处于“Calling”或“Proceeding”状态时，接收到 2xx 响应必须导致客户端事务进入“Terminated”状态，并且响应必须向上传递给 TU。 此响应的处理取决于 TU 是代理核心还是 UAC 核心。 UAC 核心将处理此响应的 ACK 生成，而代理核心将始终向上游转发 200（OK）。 代理和 UAC 之间对 200（OK）的不同处理是它的处理不在事务层中进行的原因。

客户端事务必须在它进入“终止”状态的那一刻被销毁。 这实际上是保证正确操作所必需的。 原因是对 INVITE 的 2xx 响应被区别对待； 每一个都由代理转发，并且 UAC 中的 ACK 处理是不同的。 因此，每个 2xx 都需要传递到代理核心（以便可以转发）和 UAC 核心（以便可以确认）。 没有事务层处理发生。 每当传输接收到响应时，如果传输层没有找到匹配的客户端事务（使用第 17.1.3 节的规则），则响应直接传递给核心。 由于匹配的客户端事务被第一个 2xx 销毁，后续的 2xx 将找不到匹配项，因此被传递给核心。

17.1.1.3 ACK 请求的构造

本节指定在客户端事务中发送的 ACK 请求的构造。 为 2xx 生成 ACK 的 UAC 核心必须遵循第 13 节中描述的规则。

客户端事务构造的 ACK 请求必须包含 Call-ID、From 和 Request-URI 的值，这些值等于客户端事务传递给传输的请求中那些头字段的值（称为“原始 要求”）。 ACK 中的 To 头域必须等于被确认的响应中的 To 头域，因此通常与原始请求中的 To 头域不同，增加了 tag 参数。 ACK 必须包含一个 Via 头域，并且这必须等于原始请求的顶部 Via 头域。 ACK 中的 CSeq 头域必须包含与原始请求中存在的序列号相同的值，但方法参数必须等于“ACK”。

如果响应被确认的 INVITE 请求具有 Route 头字段，则这些头字段必须出现在 ACK 中。 这是为了确保 ACK 可以通过任何下游无状态代理正确路由。

尽管任何请求都可能包含正文，但 ACK 中的正文是特殊的，因为如果正文不被理解，则无法拒绝请求。 因此，不建议将非 2xx 的正文放置在 ACK 中，但如果这样做，则正文类型将限制为 INVITE 中出现的任何类型，假设对 INVITE 的响应不是 415。如果是，则正文 ACK 可以是 415 中的 Accept 头字段中列出的任何类型。

例如，考虑以下请求：

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKkjshdyff
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=88sja8x
Max-Forwards: 70
Call-ID: 987asjd97y7atg
CSeq: 986759 INVITE
```

对此请求的非 2xx 最终响应的 ACK 请求如下所示：

```
ACK sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKkjshdyff
To: Bob <sip:bob@biloxi.com>;tag=99sa0xk
From: Alice <sip:alice@atlanta.com>;tag=88sja8x
Max-Forwards: 70
Call-ID: 987asjd97y7atg
CSeq: 986759 ACK
```

17.1.2 Non-INVITE 客户事务

17.1.2.1 Non-INVITE 事务概述

非 INVITE 事务不使用 ACK。 它们是简单的请求-响应交互。 对于不可靠的传输，请求以从 T1 开始并加倍直到到达 T2 的间隔重新传输。 如果接收到临时响应，则继续对不可靠传输进行重传，但间隔为 T2。 服务器事务仅在接收到请求的重新传输时才重新传输它发送的最后一个响应，该响应可以是临时响应或最终响应。 这就是为什么即使在临时响应之后仍需要继续请求重传的原因； 他们将确保可靠地交付最终响应。

与 INVITE 事务不同，非 INVITE 事务对 2xx 响应没有特殊处理。 结果是只有一个对非 INVITE 的 2xx 响应被传送到 UAC。

17.1.2.2 形式描述

非 INVITE 客户端事务的状态机如图 6 所示。它与 INVITE 的状态机非常相似。

当 TU 通过请求启动新的客户端事务时，进入“尝试”状态。 当进入这个状态时，客户端事务应该设置定时器 F 在 64*T1 秒内触发。 请求必须被传递到传输层进行传输。 如果使用了不可靠的传输，客户端事务必须设置定时器 E 在 T1 秒内触发。 如果定时器 E 仍处于此状态时触发，则定时器复位，但这次的值为 MIN(2*T1, T2)。 当计时器再次触发时，它被重置为 MIN(4*T1, T2)。该过程继续进行，以便以指数增加的间隔发生重传，该间隔以 T2 为上限。 T2 的默认值为 4 秒，它表示非 INVITE 服务器事务在不立即响应的情况下响应请求所需的时间。 对于 T1 和 T2 的默认值，这导致间隔为 500 ms、1 s、2 s、4 s、4 s、4 s 等。

如果 Timer F 在客户端事务仍处于“尝试”状态时触发，客户端事务应该通知 TU 超时，然后它应该进入“终止”状态。 如果在“尝试”状态下接收到临时响应，则必须将响应传递给 TU，然后客户端事务应该移动到“正在进行”状态。 如果在“尝试”状态下收到最终响应（状态代码 200-699），则必须将响应传递给 TU，并且客户端事务必须转换到“已完成”状态。

如果 Timer E 在“Proceeding”状态下触发，请求必须传递给传输层进行重传，并且 Timer E 必须重置为 T2 秒的值。 如果计时器 F 在“Proceeding”状态下触发，则必须通知 TU 超时，并且客户端事务必须转换到终止状态。 如果在“Proceeding”状态下接收到最终响应（状态代码 200-699），则必须将响应传递给 TU，并且客户端事务必须转换为“Completed”状态。

一旦客户端事务进入“已完成”状态，它必须将 Timer K 设置为在 T4 秒内触发不可靠传输，并在零秒内触发可靠传输。 “已完成”状态的存在是为了缓冲可能接收到的任何额外的响应重传（这就是为什么客户端事务仅针对不可靠的传输保留在那里的原因）。 T4 表示网络将花费的时间来清除客户端和服务器事务之间的消息。T4 的默认值为 5s。使用第 17.1.3 节中指定的规则，当响应与同一事务匹配时，响应是重传。 如果计时器 K 在此状态下触发，客户端事务必须转换到“终止”状态。

一旦事务处于终止状态，它必须立即销毁。

17.1.3 匹配对客户端事务的响应

当客户端中的传输层收到响应时，它必须确定哪个客户端事务将处理该响应，以便进行第 17.1.1 和 17.1.2 节的处理。 顶部 Via 头字段中的分支参数用于此目的。 响应在两种情况下匹配客户端事务：

1. 如果响应的顶部 Via 头域中的 branch 参数的值与创建事务的请求的顶部 Via 头域中的 branch 参数的值相同。

2.如果 CSeq 头域中的 method 参数匹配创建事务的请求的方法。 需要该方法，因为 CANCEL 请求构成不同的事务，但共享相同的分支参数值。

如果通过多播发送请求，它可能会从不同的服务器生成多个响应。 这些响应在最顶部的 Via 中都将具有相同的分支参数，但在 To 标记中有所不同。 根据上述规则，接收到的第一个响应将被使用，其他响应将被视为重传。 这不是错误； 多播 SIP 仅提供基本的“类似单跳发现”服务，仅限于处理单个响应。 有关详细信息，请参阅第 18.1.1 节。

17.1.4 处理传输错误

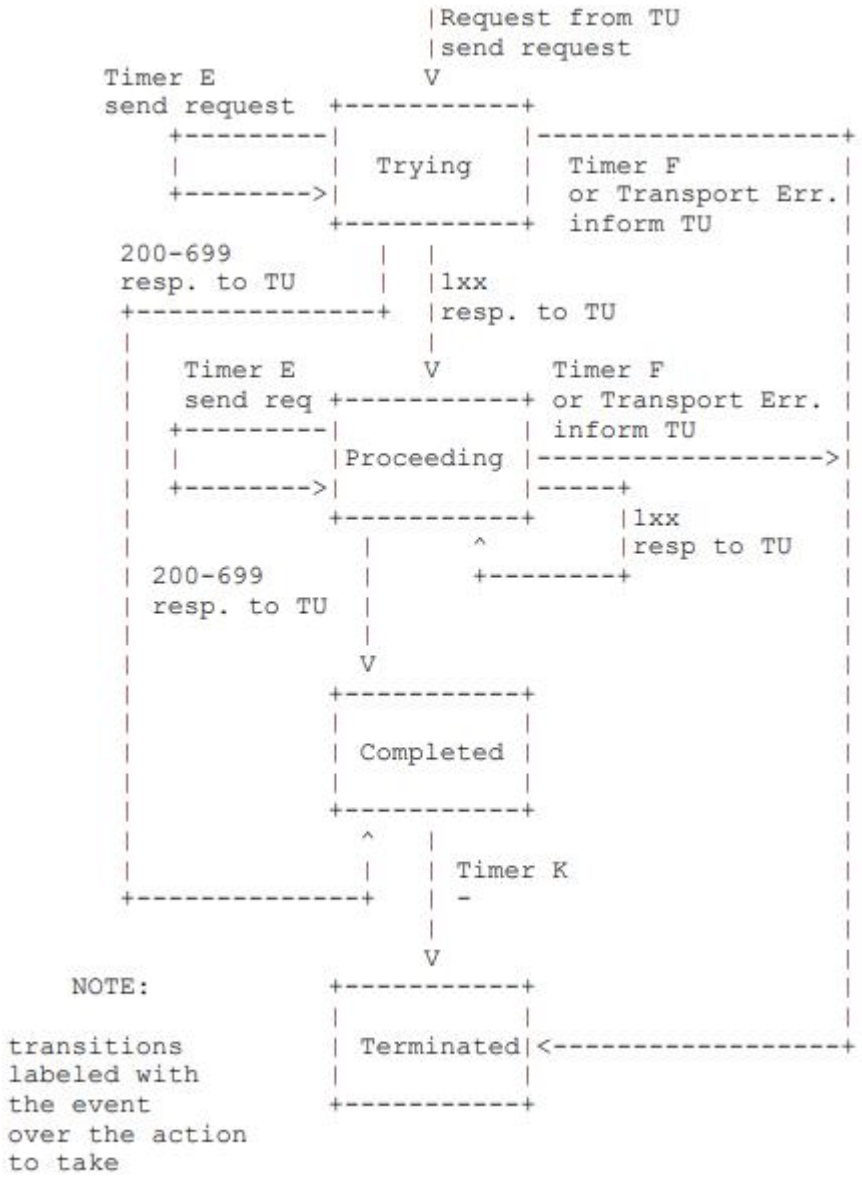

Figure 6: non-INVITE client transaction

当客户端事务向传输层发送请求以进行发送时，如果传输层指示失败，则遵循以下过程。

客户端事务应该通知 TU 传输失败已经发生，并且客户端事务应该直接转换到“终止”状态。 TU 将处理[4] 中描述的故障转移机制。

17.2 服务器事务

服务器事务负责向 TU 传递请求并可靠地传输响应。 它通过状态机完成此操作。 服务器事务由核心在收到请求时创建，并且需要对该请求进行事务处理（并非总是如此）。

与客户端事务一样，状态机取决于接收到的请求是否是 INVITE 请求。

17.2.1 INVITE 服务器事务

INVITE 服务器事务的状态图如图 7 所示。

当为请求构建服务器事务时，它进入“Proceeding”状态。 服务器事务必须生成 100（尝试）响应，除非它知道 TU 将在 200 毫秒内生成临时或最终响应，在这种情况下它可能会生成 100（尝试）响应。 需要此临时响应来快速终止请求重传，以避免网络拥塞。 100（尝试）响应是根据第 8.2.6 节中的过程构建的，除了在响应的 To 头字段中插入标签（当请求中不存在标签时）从 MAY 降级为 SHOULD NOT。 请求必须传递给 TU。

TU 将任意数量的临时响应传递给服务器事务。 只要服务器事务处于“Proceeding”状态，这些中的每一个都必须传递给传输层进行传输。 它们不会被事务层可靠地发送（它们不会被它重新传输），并且不会导致服务器事务的状态发生变化。 如果在“Proceeding”状态下接收到请求重传，则必须将最近从 TU 接收的临时响应传递给传输层进行重传。 如果请求根据第 17.2.3 节的规则匹配相同的服务器事务，则该请求是重传。

如果在“Proceeding”状态下，TU 将 2xx 响应传递给服务器事务，则服务器事务必须将此响应传递给传输层进行传输。 它不会被服务器事务重传； 2xx 响应的重传由 TU 处理。 然后服务器事务必须转换到“终止”状态。

当处于“Proceeding”状态时，如果 TU 将状态码从 300 到 699 的响应传递给服务器事务，则响应必须传递给传输层进行传输，并且状态机必须进入“Completed”状态。 对于不可靠的传输，计时器 G 设置为在 T1 秒内触发，而不是为可靠传输设置触发。

这与 RFC 2543 有所不同，在 RFC 2543 中，响应总是被重新传输，即使通过可靠的传输也是如此。

当进入“完成”状态时，定时器 H 必须设置为在 64*T1 秒内触发所有传输。 计时器 H 确定服务器事务何时放弃重新传输响应。 它的值被选择为等于 Timer B，即客户端事务将继续重试发送请求的时间量。 如果定时器 G 触发，响应将再次传递给传输层进行重传，并且定时器 G 设置为在 MIN(2*T1, T2) 秒内触发。 从那时起，当定时器 G 触发时，响应将再次传递给传输器以进行传输，并且定时器 G 重置为两倍的值，除非该值超过 T2，在这种情况下，它会重置为 T2 的值。 这与非 INVITE 客户端事务的“正在尝试”状态中的请求的

重新传输行为相同。 此外，当处于“完成”状态时，如果接收到请求重传，服务器应该将响应传递给传输以进行重传。

如果在服务器事务处于“已完成”状态时收到 ACK，则服务器事务必须转换到“已确认”状态。由于计时器 G 在此状态下被忽略，响应的任何重传都将停止。

如果计时器 H 在“已完成”状态下触发，则意味着从未收到 ACK。 在这种情况下，服务器事务必须转换到“终止”状态，并且必须向 TU 指示事务已发生故障。

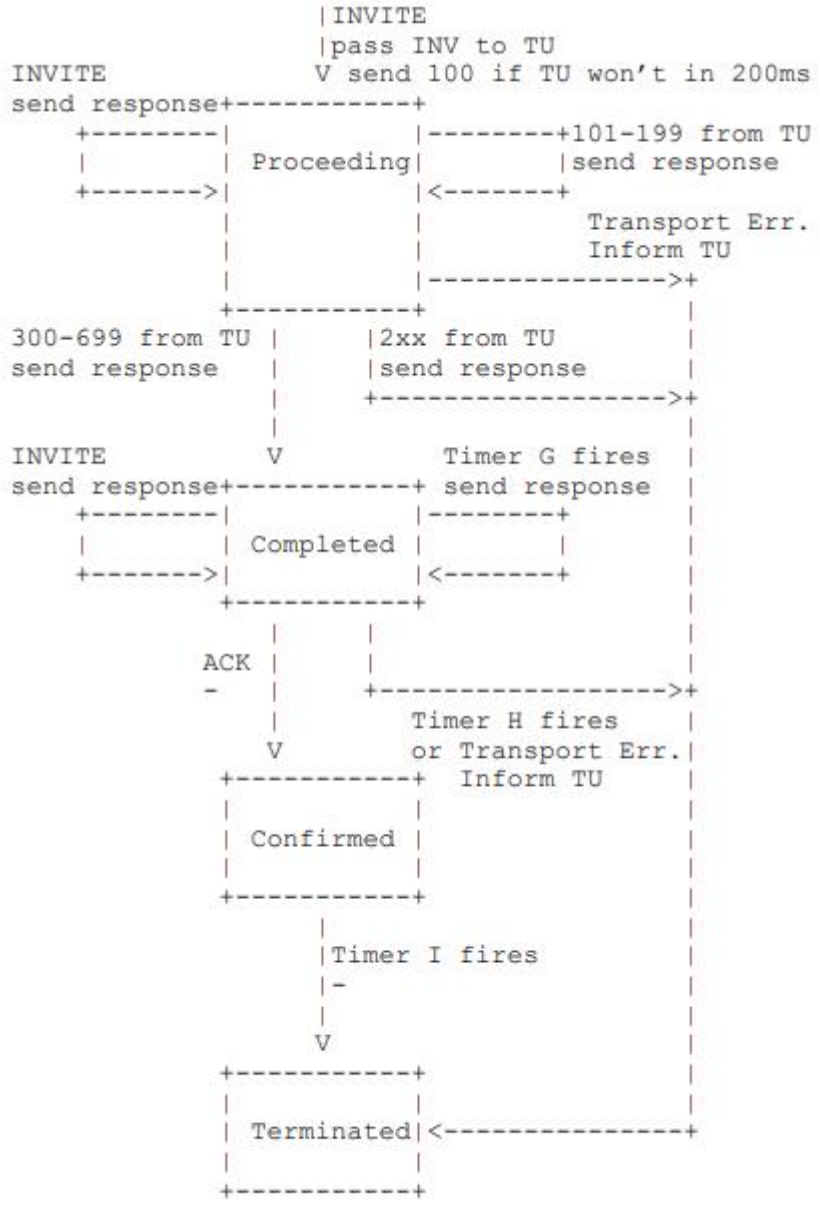

Figure 7: INVITE server transaction

“已确认”状态的目的是吸收任何到达的附加 ACK 消息，这些消息是由最终响应的重传触发的。 当进入此状态时，计时器 I 设置为在 T4 秒内触发不可靠传输，以及零秒用于可靠传输。 一旦定时器 I 触发，服务器必须转换到“终止”状态。

一旦事务处于“终止”状态，它必须立即销毁。 与客户端事务一样，这需要确保对 INVITE 的

2xx 响应的可靠性。

17.2.2 Non-INVITE 服务器事务

非 INVITE 服务器事务的状态机如图 8 所示。

状态机在“尝试”状态下初始化，并在初始化时传递除 INVITE 或 ACK 之外的请求。 这个请求被传递给 TU。 一旦处于“尝试”状态，任何进一步的请求重传都将被丢弃。 如果使用第 17.2.3 节中指定的规则，请求与相同的服务器事务匹配，则该请求是重传。

当处于“尝试”状态时，如果 TU 将临时响应传递给服务器事务，则服务器事务必须进入“进行中”状态。 响应必须传递给传输层进行传输。 在“Proceeding”状态下从 TU 接收到的任何进一步的临时响应必须传递给传输层进行传输。 如果在“Proceeding”状态下接收到请求的重传，则必须将最近发送的临时响应传递给传输层进行重传。 如果 TU 在“Proceeding”状态下将最终响应（状态代码 200-699）传递给服务器，则事务必须进入“Completed”状态，并且必须将响应传递给传输层进行传输。

当服务器事务进入“已完成”状态时，它必须将 Timer J 设置为在 64*T1 秒内触发不可靠传输，以及零秒内触发可靠传输。 当处于“已完成”状态时，服务器事务必须将最终响应传递给传输层，以便在收到重传请求时进行重传。 TU 传递给服务器事务的任何其他最终响应必须在处于“已完成”状态时被丢弃。 服务器事务保持在这个状态直到定时器 J 触发，此时它必须转换到“终止”状态。

服务器事务必须在它进入“终止”状态的那一刻被销毁。

17.2.3 将请求与服务器事务匹配

当服务器从网络接收到请求时，它必须与现有事务匹配。 这是通过以下方式完成的。

检查请求的最顶部 Via 头字段中的分支参数。 如果它存在并且以魔法 cookie“z9hG4bK”开头，则该请求是由符合本规范的客户端事务生成的。 因此，分支参数在该客户端发送的所有事务中都是唯一的。 如果满足以下条件，则请求与事务匹配：

1.请求中的分支参数等于创建事务的请求的顶部 Via 头字段中的参数，并且

2. 请求顶部 Via 中的 sent-by 值等于创建事务的请求中的值，并且

3.请求的方法与创建事务的方法匹配，除了 ACK，创建事务的请求的方法是 INVITE。

此匹配规则同样适用于 INVITE 和非 INVITE 事务。

sent-by 值用作匹配过程的一部分，因为可能存在来自不同客户端的分支参数的意外或恶意复制。

如果顶部 Via 头字段中的分支参数不存在，或者不包含魔术 cookie，则使用以下过程。 这些存在是为了处理与 RFC 2543 兼容实现的向后兼容性。

如果 Request-URI、To tag、From tag、Call-ID、CSeq 和 top Via 头字段与创建事务的 INVITE 请求匹配，则 INVITE 请求匹配事务。在这种情况下，INVITE 是对创建事务的原始请求的重传。如果 Request-URI、From 标记、Call-ID、CSeq 号（不是方法）和 top Via 头字段与创建事务的 INVITE 请求的内容以及 ACK 的 To 标记匹配，则 ACK 请求匹配事务匹配服务器事务发送的响应的 To 标记。匹配是根据为每个标头字段定义的匹配规则完成的。在 ACK 匹配过程的 To 标头字段中包含标记有助于区分 2xx 的 ACK 与代理上其他响应的 ACK 的歧义，该代理可能已经转发了两个响应（这可能发生在不寻常的情况下。特别是，当代理分叉请求时，然后崩溃，响应可能会被传递到另一个代理，最终可能会向上游转发多个响应）。与先前 ACK 匹配的 INVITE 事务匹配的 ACK 请求被视为该先前 ACK 的重传。

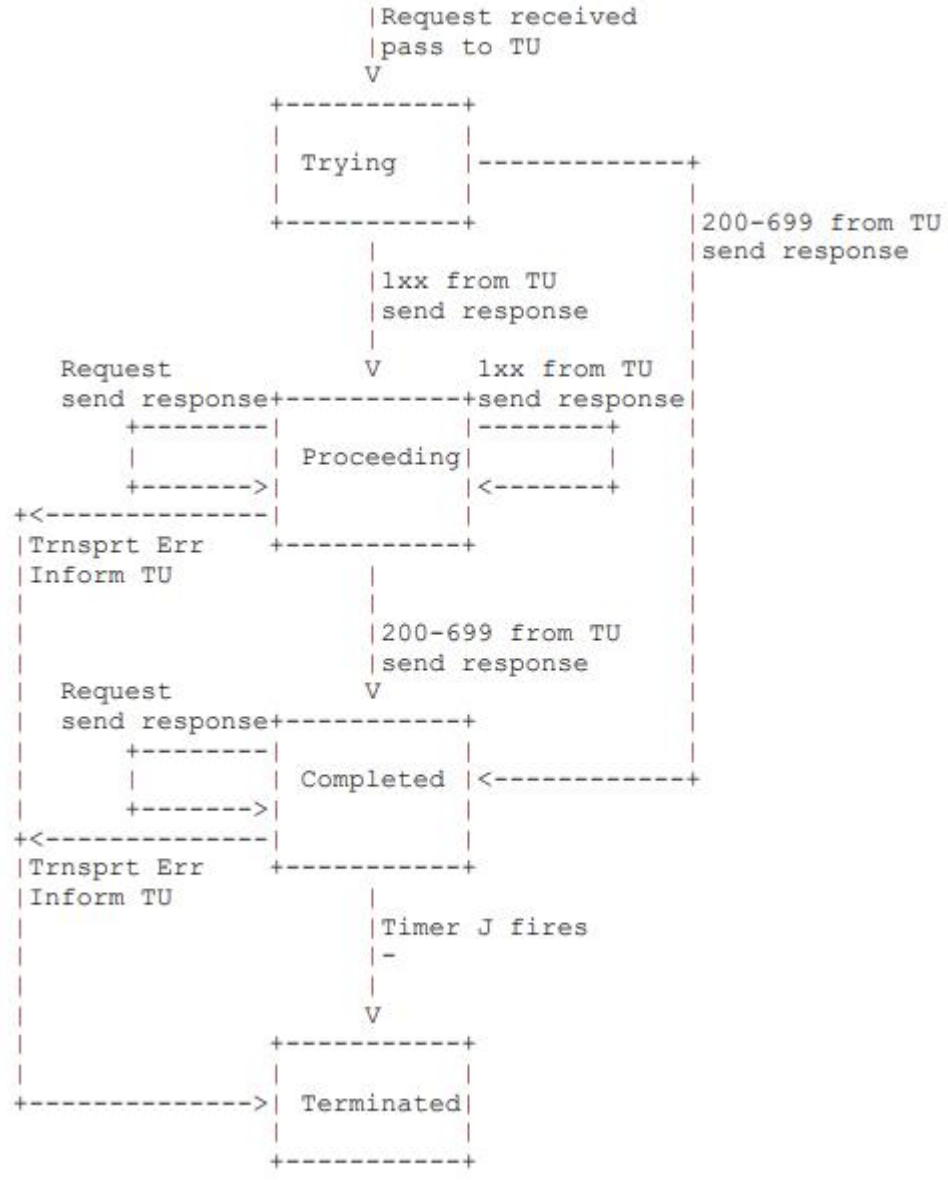

Figure 8: non-INVITE server transaction

对于所有其他请求方法，如果 Request-URI、To 标记、From 标记、Call-ID、CSeq（包括方法）和 top Via 头字段与创建事务的请求匹配，则请求与事务匹配 . 匹配是根据为每个标头字段定义的匹配规则完成的。 当非 INVITE 请求与现有事务匹配时，它是创建该事务的请求的重新传输。

因为匹配规则包括 Request-URI，所以服务器无法匹配对事务的响应。 当 TU 将响应传递给服务器事务时，它必须将其传递给响应所针对的特定服务器事务。

17.2.4 处理传输错误

当服务器事务向传输层发送要发送的响应时，如果传输层指示失败，则遵循以下过程。

首先，遵循 [4] 中的过程，尝试将响应传递给备份。 如果这些都失败了，根据 [4] 中失败的定义，服务器事务应该通知 TU 发生了失败，并且应该转换到终止状态。

18 传输

传输层负责通过网络传输实际传输请求和响应。 这包括在面向连接的传输的情况下确定用于请求或响应的连接。

传输层负责管理 TCP 和 SCTP 等传输协议的持久连接，或基于这些协议的 TLS，包括对传输层开放的协议。这包括客户端或服务器传输打开的连接，以便在客户端和服务器传输功能之间共享连接。这些连接由连接远端的地址、端口和传输协议形成的元组索引。当传输层打开连接时，此索引设置为目标 IP、端口和传输。当传输层接受连接时，此索引设置为源 IP 地址、端口号和传输。需要注意的是，由于源端口通常是临时的，但无法知道它是临时的还是通过[4]中的过程选择的，传输层接受的连接经常不会被重用。结果是使用面向连接的传输的“对等”关系中的两个代理经常使用两个连接，一个用于在每个方向上启动的事务。

建议在通过该连接发送或接收最后一条消息后，连接保持打开一段实现定义的持续时间。这个持续时间应该至少等于元素将事务从实例化到终止状态所需的最长时间。 这是为了使事务很可能在启动它们的同一连接上完成（例如，请求、响应，在 INVITE 的情况下，非 2xx 响应的 ACK）。 这通常意味着至少 64*T1（有关 T1 的定义，请参见第 17.1.1.1 节）。 但是，例如，在具有使用较大计时器 C 值的 TU 的元素中（第 16.6 节的第 11 条），它可能会更大。

所有 SIP 元素必须实现 UDP 和 TCP。 SIP 元素可以实现其他协议。

使 UA 强制使用 TCP 是对 RFC 2543 的重大更改。它是出于处理较大消息的需要而产生的，这些消息必须使用 TCP，如下所述。 因此，即使一个元素从不发送大消息，它也可能会收到一个并且需要能够处理它们。

18.1 客户端

18.1.1 发送请求

传输层的客户端负责发送请求和接收响应。 传输层的用户将请求、IP 地址、端口、传输以及可能的多播目标的 TTL 传递给客户端传输。

如果请求在路径 MTU 的 200 字节内，或者大于 1300 字节且路径 MTU 未知，则必须使用 RFC 2914 [43] 拥塞控制传输协议（例如 TCP）发送请求。如果这导致传输协议与顶部 Via 中指示的传输协议发生变化，则顶部 Via 中的值必须更改。 这可以防止 UDP 上的消息碎片化，并为较大的消息提供拥塞控制。 但是，实现必须能够处理最大数据报包大小的消息。对于 UDP，此大小为 65,535 字节，包括 IP 和 UDP 标头。

消息大小和 MTU 之间的 200 字节“缓冲区”适应了 SIP 中的响应可能大于请求的事实。例如，这是由于在对 INVITE 的响应中添加了 Record-Route 标头字段值。 使用额外的缓冲区，响应可能比请求大 170 字节左右，并且仍然不会在 IPv4 上被分段（大约 30 字节被 IP/UDP 消耗，假设没有 IPSec）。 基于 1500 字节以太网 MTU 的假设，当路径 MTU 未知时选择 1300。

如果一个元素由于这些消息大小限制而通过 TCP 发送请求，并且该请求将通过 UDP 发送，如果尝试建立连接会生成不支持的 ICMP 协议或导致 TCP 重置，则该元素 应该使用 UDP 重试请求。 这只是为了提供与不支持 TCP 的 RFC 2543 兼容实现的向后兼容性。 预计此行为将在本规范的未来修订版中被弃用。

向多播地址发送请求的客户端必须将“maddr”参数添加到包含目标多播地址的 Via 头字段值中，对于 IPv4，应该添加值为 1 的“ttl”参数。 IPv6 多播的用法 本规范中没有定义，当需要时将成为未来标准化的主题。

这些规则导致 SIP 中的多播有目的地限制。 它的主要功能是提供“类似单跳发现”的服务，将请求传递给一组同类服务器，只需要处理其中任何一个的响应。 此功能对于注册最有用。事实上，根据 17.1.3 节中的事务处理规则，客户端事务将接受第一个响应，并将任何其他响应视为重传，因为它们都包含相同的 Via 分支标识符。

在发送请求之前，客户端传输必须将“sent-by”字段的值插入到 Via 头字段中。 此字段包含 IP 地址或主机名和端口。 建议使用 FQDN。 该字段用于在特定条件下发送响应，如下所述。 如果端口不存在，则默认值取决于传输。 UDP、TCP 和 SCTP 为 5060，TLS 为 5061。

对于可靠的传输，响应通常在收到请求的连接上发送。 因此，客户端传输必须准备好在用于发送请求的同一连接上接收响应。 在错误情况下，服务器可能会尝试打开一个新连接来发送响应。 为了处理这种情况，传输层还必须准备好在发送请求的源 IP 地址和“sent-by”字段中的端口号上接收传入连接。 它还必须准备好接收任何地址和端口上的传入连接，这些地址和端口将由服务器根据 [4] 的第 5 节中描述的过程选择。

对于不可靠的单播传输，客户端传输必须准备好接收来自发送请求的源 IP 地址的响应（因为响应被发送回源地址）和“sent-by”字段中的端口号。 此外，与可靠传输一样，在某些情况下，响应将被发送到其他地方。 客户端必须准备好接收服务器根据 [4] 第 5 节中描述的过程选择的任何地址和端口上的响应。

对于多播，客户端传输必须准备好在请求发送到的同一多播组和端口上接收响应（也就是说，它需要是向其发送请求的多播组的成员。）

如果请求的目的地是一个 IP 地址、端口和现有连接已打开的传输，则建议使用此连接发送请求，但可以打开和使用另一个连接。

如果使用多播发送请求，则将其发送到传输用户提供的组地址、端口和 TTL。 如果使用单播不可靠传输发送请求，则将其发送到传输用户提供的 IP 地址和端口。

18.1.2 接收响应

当收到响应时，客户端传输检查顶部的 Via 头字段值。 如果该标头字段值中的“sent-by”参数的值与客户端传输配置为插入请求中的值不对应，则必须静默丢弃响应。

如果存在任何客户端事务，客户端传输使用第 17.1.3 节的匹配过程来尝试将响应与现有事务匹配。 如果匹配，则必须将响应传递给该事务。 否则，必须将响应传递给核心（无论是无状态代理、有状态代理还是 UA）以进行进一步处理。 这些“杂散”响应的处理取决于核心（例如，代理将转发它们，而 UA 将丢弃它们）。

18.2 服务器

18.2.1 接收请求

服务器应该准备好接收任何 IP 地址、端口和传输组合的请求，这些请求可能是 DNS 查找 SIP 或 SIPS URI [4] 的结果，该 URI 是为了与该服务器通信而分发的。在此上下文中，“分发”包括将 URI 放置在 REGISTER 请求或重定向响应中的 Contact 头字段中，或者放置在请求或响应中的 Record-Route 头字段中。 URI 也可以通过将其放置在网页或名片上来“分发”。还建议服务器在所有公共接口上侦听默认 SIP 端口（TCP 和 UDP 为 5060，TLS over TCP 为 5061）上的请求。典型的例外是专用网络，或者当多个服务器实例在同一主机上运行时。对于服务器侦听 UDP 的任何端口和接口，它必须在相同的端口和接口上侦听 TCP。这是因为如果消息太大，可能需要使用 TCP 而不是 UDP 发送消息。结果，反之则不成立。服务器不需要在特定地址和端口上侦听 UDP，因为它正在侦听 TCP 的相同地址和端口。当然，服务器需要在特定地址和端口上侦听 UDP 可能还有其他原因。

当服务器传输通过任何传输接收请求时，它必须检查顶部 Via 头字段值中的“sent-by”参数的值。 如果“sent-by”参数的主机部分包含一个域名，或者如果它包含一个与数据包源地址不同的 IP 地址，则服务器必须在该 Via 头字段值中添加一个“received”参数。 该参数必须

包含接收数据包的源地址。 这是为了帮助服务器传输层发送响应，因为它必须发送到请求来自的源 IP 地址。

考虑服务器传输接收到的请求，部分看起来像：

```
INVITE sip:bob@Biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060
```

使用源 IP 地址 192.0.2.4 接收请求。 在向上传递请求之前，传输添加了一个“received”参数，因此请求部分看起来像：

```
INVITE sip:bob@Biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;received=192.0.2.4
```

接下来，服务器传输尝试将请求与服务器事务匹配。 它使用第 17.2.3 节中描述的匹配规则来做到这一点。 如果找到匹配的服务器事务，则将请求传递给该事务进行处理。 如果未找到匹配项，则将请求传递给核心，核心可能决定为该请求构建新的服务器事务。 请注意，当 UAS 核心向 INVITE 发送 2xx 响应时，服务器事务被破坏。 这意味着当 ACK 到达时，将没有匹配的服务器事务，并且根据此规则，将 ACK 传递给 UAS 核心，在那里进行处理。

18.2.2 发送响应

服务器传输使用顶部 Via 头字段的值来确定将响应发送到何处。 它必须遵循以下过程：

o 如果“sent-protocol”是一个可靠的传输协议，例如 TCP 或 SCTP，或基于这些的 TLS，则必须使用现有连接将响应发送到创建事务的原始请求的源，如果该连接仍然打开 . 这需要服务器传输维护服务器事务和传输连接之间的关联。 如果该连接不再打开，服务器应该打开一个到“received”参数中的 IP 地址的连接，如果存在，使用“sent-by”值中的端口，或者该传输的默认端口，如果没有 指定端口。 如果该连接尝试失败，服务器应该使用 [4] 中的过程用于服务器以确定打开连接并发送响应的 IP 地址和端口。

o 否则，如果 Via 头字段值包含“maddr”参数，则必须使用“sent-by”中指示的端口将响应转发到其中列出的地址，如果不存在则使用端口 5060。 如果地址是多播地址，则应使用“ttl”参数中指示的 TTL 发送响应，如果该参数不存在，则 TTL 为 1。

o 否则（对于不可靠的单播传输），如果顶部 Via 具有“received”参数，则必须使用“sent-by”值中指示的端口或使用端口将响应发送到“received”参数中的地址 如果没有明确指定，则为 5060。 如果失败，例如，引发 ICMP“端口不可达”响应，则 [4] 的第 5 节的过程应该用于确定将响应发送到哪里。

o 否则，如果它没有被标记为接收者，则必须使用 [4] 的第 5 节中的过程将响应发送到由“sent-by”值指示的地址。

18.3 Framing

在面向消息的传输（例如 UDP）的情况下，如果消息具有 Content-Length 头字段，则假定消息正文包含那么多字节。如果传输包中有超出正文末尾的附加字节，则必须丢弃它们。如

果传输数据包在消息正文结束之前结束，则认为这是一个错误。 如果消息是响应，则必须将其丢弃。 如果消息是请求，则元素应该生成 400（错误请求）响应。 如果消息没有 Content-Length 头字段，则假定消息正文在传输包的末尾结束。

在面向流的传输（如 TCP）的情况下，Content-Length 头字段指示主体的大小。 Content-Length 头域必须与面向流的传输一起使用。

18.3 框架

错误处理与消息是请求还是响应无关。

如果传输用户要求通过不可靠的传输发送消息，结果是 ICMP 错误，则行为取决于 ICMP 错误的类型。 主机、网络、端口或协议不可达错误，或参数问题错误应该导致传输层通知传输用户发送失败。 源猝灭和 TTL 超出 ICMP 错误应该被忽略。

如果传输用户要求通过可靠传输发送请求，结果是连接失败，则传输层应该通知传输用户发送失败。

19 通用消息组件

SIP 消息的某些组成部分出现在 SIP 消息中的不同位置（有时也出现在它们之外），值得单独讨论。

19.1 SIP 和 SIPS 统一资源指标

SIP 或 SIPS URI 标识通信资源。 与所有 URI 一样，SIP 和 SIPS URI 可以放置在网页、电子邮件消息或印刷文献中。 它们包含足够的信息来启动和维护与资源的通信会话。

通信资源的示例包括：

- 在线服务的用户
- 出现在多线电话上
- 消息系统上的邮箱
- 网关服务处的 PSTN 号码
- 组织中的一个组（例如“销售”或“帮助台”）

SIPS URI 指定安全地联系资源。 这尤其意味着 TLS 将在 UAC 和拥有 URI 的域之间使用。

从那里，安全通信用于到达用户，其中特定的安全机制取决于域的策略。 如果希望安全地与该资源通信，则只需更改方案即可将 SIP URI 描述的任何资源“升级”为 SIPS URI。

19.1.1 SIP 和 SIPS URI 组件

“sip:”和“sips:”方案遵循 RFC 2396 [5] 中的指南。 它们使用类似于 mailto URL 的形式，允许指定 SIP 请求标头字段和 SIP 消息正文。 这使得通过在网页或电子邮件消息中使用 URI 来指定会话的主题、媒体类型或紧迫性成为可能。 第 25 节介绍了 SIP 或 SIPS URI 的正式语法。在 SIP URI 的情况下，它的一般形式是：

```
sip:user:password@host:port;uri-parameters?headers
```

SIPS URI 的格式是相同的，只是方案是“sips”而不是 sip。 这些标记，以及它们扩展中的一些标记，具有以下含义：

user：被寻址主机上特定资源的标识符。 在此上下文中，术语“主机”通常指的是域。 URI 的“用户信息”由此用户字段、密码字段和它们后面的 @ 符号组成。 URI 的 userinfo 部分是可选的，当目标主机没有用户概念或主机本身是被识别的资源时，可能不存在。 如果 @ 符号出现在 SIP 或 SIPS URI 中，则用户字段不得为空。

如果被寻址的主机可以处理电话号码，例如，互联网电话网关，则在 RFC 2806 [9] 中定义的电话用户字段可以用于填充用户字段。 在第 19.1.2 节中描述的 SIP 和 SIPS URI 中编码电话用户字段有特殊的转义规则。

password：与用户关联的密码。 虽然 SIP 和 SIPS URI 语法允许存在此字段，但不推荐使用它，因为在几乎所有使用它的情况下，以明文（例如 URI）形式传递身份验证信息已被证明是一种安全风险 . 例如，在此字段中传输 PIN 码会公开 PIN。

请注意，密码字段只是用户部分的扩展。 不希望对该字段的密码部分赋予特殊意义的实现可以简单地将“用户：密码”视为单个字符串。

host：提供 SIP 资源的主机。 主机部分包含完全限定域名或数字 IPv4 或 IPv6 地址。 建议尽可能使用完全限定域名表格。

port：发送请求的端口号。

URI parameters：影响从 URI 构造的请求的参数。

URI 参数添加在主机端口组件之后，并用分号分隔。

URI 参数采用以下形式：

```
parameter-name "=" parameter-value
```

即使在一个 URI 中可以包含任意数量的 URI 参数，任何给定的参数名称也不能出现超过一次。

这种可扩展的机制包括 transport、maddr、ttl、user、method 和 lr 参数。

传输参数确定用于发送 SIP 消息的传输机制，如 [4] 中所指定。 SIP 可以使用任何网络传输协议。 为 UDP (RFC 768 [14])、TCP (RFC 761 [15]) 和 SCTP (RFC 2960 [16]) 定义了参数名称。 对于 SIPS URI，传输参数必须指示可靠传输。

maddr 参数指示该用户要联系的服务器地址，覆盖从主机字段派生的任何地址。 当 maddr 参数存在时，URI 的端口和传输组件适用于 maddr 参数值中指示的地址。 [4] 描述了对传输、maddr 和主机端口的正确解释，以便获得发送请求的目标地址、端口和传输。

maddr 字段已被用作松散源路由的一种简单形式。 它允许 URI 指定必须在到达目的地的途中经过的代理。 强烈建议不要继续以这种方式使用 maddr 参数（启用它的机制已弃用）。实现应该使用本文档中描述的路由机制，必要时建立一个预先存在的路由集（参见第 8.1.1.1 节）。 这提供了一个完整的 URI 来描述要遍历的节点。

ttl 参数确定 UDP 多播数据包的生存时间值，并且必须仅在 maddr 是多播地址且传输协议为 UDP 时使用。 例如，要使用 ttl 为 15 的多播到 239.255.255.1 指定对 alice@atlanta.com 的调用，将使用以下 URI：

```
sip:alice@atlanta.com;maddr=239.255.255.1;ttl=15
```

有效电话用户字符串集是有效用户字符串的子集。用户 URI 参数的存在是为了将电话号码与恰好看起来像电话号码的用户名区分开来。 如果用户字符串包含格式化为电话用户的电话号码，则用户参数值“电话”应该存在。 即使没有此参数，如果本地对用户名的名称空间的限制允许，SIP 和 SIPS URI 的接收者也可以将 pre-@ 部分解释为电话号码。

从 URI 构造的 SIP 请求的方法可以用 method 参数指定。

lr 参数，当存在时，表示负责此资源的元素实现了本文档中指定的路由机制。 此参数将在 URI 代理中使用到 Record-Route 标头字段值中，并且可能出现在预先存在的路由集中的 URI 中。

此参数用于实现与实现 RFC 2543 的严格路由机制的系统的向后兼容性，并且 rfc2543bis 草案达到 bis-05。准备基于不包含此参数的 URI 发送请求的元素可以假定接收元素实现严格路由并重新格式化消息以保留 Request-URI 中的信息。

此参数用于实现与实现 RFC 2543 的严格路由机制的系统的向后兼容性，并且 rfc2543bis 草案达到 bis-05。准备基于不包含此参数的 URI 发送请求的元素可以假定接收元素实现严格路由并重新格式化消息以保留 Request-URI 中的信息。由于 uri-parameter 机制是可扩展的，SIP 元素必须 默默地忽略他们不理解的任何 uri 参数。

Headers：要包含在从 URI 构造的请求中的标头字段。

SIP 请求中的标头字段可以用“？”指定 URI 中的机制。 标头名称和值以与号分隔的 hname

= hvalue 对进行编码。 特殊的 hname “body”表示关联的 hvalue 是 SIP 请求的消息体。

表 1 根据 URI 出现的上下文总结了 SIP 和 SIPS URI 组件的使用。 外部列描述了出现在 SIP 消息之外的任何地方的 URI，例如在网页或名片上。 标有“m”的条目是强制性的，标有“o”的是可选的，标有“-”的是不允许的。 处理 URI 的元素应该忽略任何不允许的组件（如果它们存在）。 第二列表示可选元素的默认值（如果它不存在）。 “--”表示该元素不是可选的，或者没有默认值。

Contact 头字段中的 URI 有不同的限制，具体取决于头字段出现的上下文。 一组适用于建立和维护对话的消息（INVITE 及其 200（OK）响应）。 另一个适用于注册和重定向消息（REGISTER，它的 200（OK）响应，以及对任何方法的 3xx 类响应）。

19.1.2 字符转义要求

	default	Req.-URI	To	From	reg./redir. Contact	dialog Contact/ R-R/Route	external
user	--	o	o	o	o	o	o
password	--	o	o	o	o	o	o
host	--	m	m	m	m	m	m
port	(1)	o	-	-	o	o	o
user-param	ip	o	o	o	o	o	o
method	INVITE	-	-	-	-	-	o
maddr-param	--	o	-	-	o	o	o
ttl-param	1	o	-	-	o	-	o
transp.-param	(2)	o	-	-	o	o	o
lr-param	--	o	-	-	-	o	o
other-param	--	o	o	o	o	o	o
headers	--	-	-	-	o	-	o

(1)：默认端口值取决于传输和方案。 sip 的默认值为 5060：使用 UDP、TCP 或 SCTP。 sip: 使用 TLS over TCP 和 sips: over TCP 的默认值为 5061。

(2)：默认传输取决于方案。 对于 sip:，它是 UDP。 对于 sips:，它是 TCP。

表 1：SIP 头字段值、Request-URI 和引用的 URI 组件的使用和默认值。

SIP 在定义必须在 SIP URI 中转义的字符集时遵循 RFC 2396 [5] 的要求和指南，并使用其“%”HEX HEX 机制进行转义。 来自 RFC 2396 [5]：

在任何给定的 URI 组件中实际保留的字符集由该组件定义。 一般来说，如果 URI 的语义发生变化，如果字符被其转义的 US-ASCII 编码 [5] 替换，则该字符被保留。 排除的 US-ASCII 字符 (RFC 2396 [5])，例如空格和控制字符以及用作 URI 分隔符的字符，也必须进行转义。 URI 不得包含未转义的空格和控制字符。

对于每个组件，有效的 BNF 扩展集准确定义了哪些字符可能未转义。 所有其他字符必须转义。

例如，“@”不在用户组件的字符集中，因此用户“j@s0n”必须至少有@符号编码，如“j%40s0n”。

扩展第 25 节中的 hname 和 hvalue 标记表明，标题字段名称和值中的所有 URI 保留字符必须被转义。

用户组件的电话用户子集有特殊的转义考虑。 在电话用户的 RFC 2806 [9] 描述中未保留的字符集包含各种语法元素中的许多字符，这些字符在 SIP URI 中使用时需要转义。 任何出现在电话用户中但未出现在用户规则的 BNF 扩展中的字符都必须转义。

请注意，在 SIP 或 SIPS URI 的主机组件中不允许字符转义（% 字符在其扩展中无效）。 随着国际化域名要求的最终确定，这可能会在未来发生变化。 当前的实现不得试图通过将主机组件中接收到的转义字符视为字面上等同于它们的未转义对应物来提高鲁棒性。 满足 IDN 要求所需的行为可能会有很大不同。

19.1.3 示例 SIP 和 SIPS URI

```
sip:alice@atlanta.com
sip:alice:secretword@atlanta.com;transport=tcp
sips:alice@atlanta.com?subject=project%20x&priority=urgent
sip:+1-212-555-1212:1234@gateway.com;user=phone
sips:1212@gateway.com
sip:alice@192.0.2.4
sip:atlanta.com;method=REGISTER?to=alice%40atlanta.com
sip:alice;day=tuesday@atlanta.com
```

上面最后一个示例 URI 的用户字段值为“alice;day=tuesday”。上面定义的转义规则允许分号在此字段中未转义。 出于本协议的目的，该字段是不透明的。 该值的结构仅对负责资源的 SIP 元素有用。

19.1.4 URI 比较

本规范中的某些操作需要确定两个 SIP 或 SIPS URI 是否等效。 在本规范中，注册商需要在注册请求中比较联系人 URI 中的绑定（参见第 10.3 节）。 根据以下规则比较 SIP 和 SIPS URI 是否相等：

o SIP 和 SIPS URI 从不等同。

o SIP 和 SIPS URI 的用户信息的比较区分大小写。 这包括包含密码或格式化为电话订户的用户信息。 URI 的所有其他组件的比较不区分大小写，除非另有明确定义。

o 在比较 SIP 和 SIPS URI 时，参数和标头字段的顺序并不重要。

o “保留”集（参见 RFC 2396 [5]）中的字符以外的字符等同于它们的“%”HEX HEX“编码。

o 作为主机名的 DNS 查找结果的 IP 地址与该主机名不匹配。

o 要使两个 URI 相等，用户、密码、主机和端口组件必须匹配。

省略用户组件的 URI 与包含一个的 URI 不匹配。 省略密码组件的 URI 与包含密码组件的 URI 不匹配。

省略任何具有默认值的组件的 URI 不会与显式包含该组件及其默认值的 URI 匹配。 例如，省略可选端口组件的 URI 不会匹配显式声明端口 5060 的 URI。传输参数、ttl 参数、用户参数和方法组件也是如此。

将 sip:user@host 定义为不等同于 sip:user@host:5060 是对 RFC 2543 的更改。从 URI 派生地址时，需要从等效 URI 中获取等效地址。 URI sip:user@host:5060 将始终解析为端口 5060。URI sip:user@host 可以通过 [4] 中详述的 DNS SRV 机制解析为其他端口。

o URI uri-parameter 组件比较如下：

- 出现在两个 URI 中的任何 uri 参数必须匹配。

- 仅出现在一个 URI 中的用户、ttl 或方法 uri 参数永远不会匹配，即使它包含默认值。

- 包含 maddr 参数的 URI 与不包含 maddr 参数的 URI 不匹配。

- 在比较 URI 时，将忽略仅出现在一个 URI 中的所有其他 uri 参数。

o URI 标头组件永远不会被忽略。 任何存在的标头组件都必须存在于两个 URI 中并且匹配 URI 以匹配。 第 20 节中为每个头字段定义了匹配规则。

以下每个集合中的 URI 是等效的：

```
sip:%61lice@atlanta.com;transport=TCP
sip:alice@AtLanTa.CoM;Transport=tcp

sip:carol@chicago.com
sip:carol@chicago.com;newparam=5
sip:carol@chicago.com;security=on

sip:biloxi.com;transport=tcp;method=REGISTER?to=sip:bob%40biloxi.com
sip:biloxi.com;method=REGISTER;transport=tcp?to=sip:bob%40biloxi.com

sip:alice@atlanta.com?subject=project%20x&priority=urgent
sip:alice@atlanta.com?priority=urgent&subject=project%20x

The URIs within each of the following sets are not equivalent:

SIP:ALICE@AtLanTa.CoM;Transport=udp            (different usernames)
sip:alice@AtLanTa.CoM;Transport=UDP

sip:bob@biloxi.com                  (can resolve to different ports)
sip:bob@biloxi.com:5060

sip:bob@biloxi.com              (can resolve to different transports)
sip:bob@biloxi.com;transport=udp

sip:bob@biloxi.com     (can resolve to different port and transports)
sip:bob@biloxi.com:6000;transport=tcp

sip:carol@chicago.com                    (different header component)
sip:carol@chicago.com?Subject=next%20meeting

sip:bob@phone21.boxesbybob.com   (even though that's what
sip:bob@192.0.2.4                 phone21.boxesbybob.com resolves to)
```

请注意，相等是不传递的：

o sip:carol@chicago.com 和 sip:carol@chicago.com;security=on 是等价的

o sip:carol@chicago.com 和 sip:carol@chicago.com;security=off 是等价的

o sip:carol@chicago.com;security=on 和 sip:carol@chicago.com;security=off 不等价

19.1.5 从 URI 形成请求

直接从 URI 形成请求时，实现需要小心。 来自名片、网页，甚至来自协议内部来源（例如已注册联系人）的 URI 可能包含不适当的标头字段或正文部分。

实现必须在形成的请求的 Request-URI 中包含任何提供的传输、maddr、ttl 或用户参数。 如果 URI 包含方法参数，则其值必须用作请求的方法。 方法参数不得放在 Request-URI 中。未知的 URI 参数必须放在消息的 Request-URI 中。

实现应该将 URI 中的任何标头或正文部分的存在视为希望将它们包含在消息中，并选择在每个组件的基础上接受请求。

实现不应该尊重这些明显危险的标头字段：From、Call-ID、CSeq、Via 和 Record-Route。

实现不应该尊重任何请求的 Route 头字段值，以免在恶意攻击中被用作不知情的代理。

实现不应该尊重包含可能导致其错误地宣传其位置或功能的标头字段的请求。 其中包括：Accept、Accept-Encoding、Accept-Language、Allow、Contact（在其对话框使用中）、Organization、Supported 和 User-Agent。

实现应该验证任何请求的描述性标头字段的准确性，包括：Content-Disposition、Content-Encoding、Content-Language、Content-Length、Content-Type、Date、Mime-Version 和 Timestamp。

如果从给定 URI 构造消息形成的请求不是有效的 SIP 请求，则 URI 无效。 实现绝不能继续传输请求。 相反，它应该在它发生的上下文中由于一个无效的 URI 而采取行动。

构造的请求可能在许多方面无效。 这些包括但不限于标头字段中的语法错误、URI 参数的无效组合或消息正文的错误描述。

发送从给定 URI 形成的请求可能需要实现不可用的功能。 例如，URI 可能指示使用未实现的传输或扩展。 实现应该拒绝发送这些请求，而不是修改它们以匹配它们的能力。 实现不得发送需要其不支持的扩展的请求。

例如，可以通过存在 Require 标头参数或具有未知或明确不支持的值的方法 URI 参数来形成此类请求。

19.1.6 关联 SIP URI 和电话 URL

当 tel URL (RFC 2806 [9]) 被转换为 SIP 或 SIPS URI 时，tel URL 的整个电话用户部分，包括任何参数，都被放入 SIP 或 SIPS URI 的 userinfo 部分。

```
Thus, tel:+358-555-1234567;postd=pp22 becomes

   sip:+358-555-1234567;postd=pp22@foo.com;user=phone

or
   sips:+358-555-1234567;postd=pp22@foo.com;user=phone

not
   sip:+358-555-1234567@foo.com;postd=pp22;user=phone

or

   sips:+358-555-1234567@foo.com;postd=pp22;user=phone
```

通常，以这种方式转换为 SIP 或 SIPS URI 的等效“电话”URL 可能不会生成等效的 SIP 或 SIPS URI。 SIP 和 SIPS URI 的用户信息作为区分大小写的字符串进行比较。 tel URL 的不区分大小写部分的变化和 tel URL 参数的重新排序不会影响 tel URL 等价性，但会影响由它们形成的 SIP URI 的等价性。

例如，

```
tel:+358-555-1234567;postd=pp22
tel:+358-555-1234567;POSTD=PP22
```

是等价的，而

```
sip:+358-555-1234567;postd=pp22@foo.com;user=phone
sip:+358-555-1234567;POSTD=PP22@foo.com;user=phone
```

不是。

同样地，

```
tel:+358-555-1234567;postd=pp22;isub=1411
tel:+358-555-1234567;isub=1411;postd=pp22
```

是等价的，而

```
sip:+358-555-1234567;postd=pp22;isub=1411@foo.com;user=phone
sip:+358-555-1234567;isub=1411;postd=pp22@foo.com;user=phone
```

不是。

为了缓解这个问题，构建电话订户字段以放置在 SIP 或 SIPS URI 的 userinfo 部分中的元素应该将电话订户的任何不区分大小写的部分折叠为小写，并按参数名称对电话订户参数进行词法排序， 除了 isdn-subaddress 和 post-dial，它们首先出现并按此顺序。 （tel URL 的所有组件，除了 future-extension 参数外，都被定义为不区分大小写。）

```
Following this suggestion, both

   tel:+358-555-1234567;postd=pp22
   tel:+358-555-1234567;POSTD=PP22

   become

     sip:+358-555-1234567;postd=pp22@foo.com;user=phone

and both

     tel:+358-555-1234567;tsp=a.b;phone-context=5
     tel:+358-555-1234567;phone-context=5;tsp=a.b

   become

     sip:+358-555-1234567;phone-context=5;tsp=a.b@foo.com;user=phone
```

19.2 选项标签

选项标签是用于在 SIP 中指定新选项（扩展）的唯一标识符。这些标签用于 Require (Section 20.32)、 Proxy-Require (Section 20.29)、 Supported (Section 20.37) 和 Unsupported (Section 20.40) 头字段。 请注意，这些选项以 option-tag = token 形式作为参数出现在这些标头字段中（有关 token 的定义，请参见第 25 节）。

选项标签在标准跟踪 RFC 中定义。 这是对过去做法的改变，旨在确保持续的多供应商互操作性（参见第 20.32 节和第 20.37 节中的讨论）。 选项标签的 IANA 注册表用于确保轻松参考。

19.3 标签

“tag”参数用于 SIP 消息的 To 和 From 报头字段。 它用作识别对话的通用机制，它是 Call-ID 和两个标签的组合，一个来自对话中的每个参与者。当 UA 在对话之外发送请求时，它只包含一个 From 标签，提供对话 ID 的“一半”。 对话是从响应中完成的，每个响应都在 To 标头字段中贡献后半部分。 SIP 请求的分叉意味着可以从单个请求建立多个对话。这也解释了对双边对话标识符的需要； 如果没有接收者的贡献，发起者无法消除从单个请求建立的多个对话的歧义。

当 UA 生成标签以插入请求或响应时，它必须是全局唯一的，并且具有至少 32 位随机性的密码随机性。 此选择要求的一个属性是，UA 会将不同的标记放入 INVITE 的 From 标头中，而不是将其放入对同一 INVITE 的响应的 To 标头中。 这是为了让 UA 邀请自己加入会话，这是 PSTN 网关中呼叫“发夹”的常见情况。 类似地，不同呼叫的两个 INVITE 将具有不同的 From 标签，不同呼叫的两个响应将具有不同的 To 标签。

除了全局唯一性的要求之外，生成标签的算法是特定于实现的。 标签在容错系统中很有帮助，在该系统中，对话将在故障后在备用服务器上恢复。 UAS 可以选择标签，以便备份可以将请求识别为故障服务器上对话的一部分，因此确定它应该尝试恢复对话和与之相关的任何其他状态。

20 头字段

头字段的一般语法在第 7.3 节中介绍。 本节列出了完整的标题字段集以及有关语法、含义和用法的注释。 在本节中，我们使用 [HX.Y] 来指代当前 HTTP/1.1 规范 RFC 2616 [8] 的第 X.Y 节。 给出了每个标题字段的示例。

表 2 和表 3 总结了与方法和代理处理有关的头字段信息。

“where”列描述了可以使用头域的请求和响应类型。 此列中的值为：

R: header 字段只能出现在请求中；

r：标头字段只能出现在响应中；

2xx、4xx 等：数值或范围表示可以使用头域的响应码；

c: header 字段从请求复制到响应。

“where”列中的空条目表示标头字段可能存在于所有请求和响应中。

"proxy" 列描述了代理可能对标头字段执行的操作：

a：如果不存在，代理可以添加或连接头字段。

m：代理可以修改现有的头字段值。

d：代理可以删除一个头域值。

r：代理必须能够读取头域，因此该头域不能被加密。

接下来的六列与方法中是否存在标头字段有关：

c：有条件的； 对标头字段的要求取决于消息的上下文。

m：头域为必填项。

m*：应该发送标头字段，但客户端/服务器需要准备好接收没有该标头字段的消息。

o：头域是可选的。

t：应该发送头域，但客户端/服务器需要准备好接收没有该头域的消息。

如果使用基于流的协议（例如 TCP）作为传输，则必须发送头字段。

*：如果消息体不为空，则需要标头字段。 有关详细信息，请参阅第 20.14、20.15 和 7.4 节。

-：头字段不适用。

“可选”意味着一个元素可以在请求或响应中包含头域，如果请求或响应中存在，UA 可以忽略头域（该规则的例外是 20.32 中讨论的 Require 头域）。 “强制”头域必须出现在请求中，并且必须被接收请求的 UAS 理解。响应中必须存在强制响应头字段，并且处理响应的 UAC 必须理解该头字段。 “不适用”意味着请求中不能出现头部字段。 如果一个错误被放置在请求中，它必须被接收请求的 UAS 忽略。 类似地，一个标有“不适用”的响应头域意味着 UAS 不能在响应中放置头域，并且 UAC 必须忽略响应中的头域。

UA 应该忽略无法理解的扩展头参数。

还定义了一些常见标头字段名称的紧凑形式，以便在整体消息大小成为问题时使用。

Contact、From 和 To 标头字段包含一个 URI。如果 URI 包含逗号、问号或分号，则 URI 必须用尖括号（< 和 >）括起来。 任何 URI 参数都包含在这些括号内。 如果 URI 未包含在尖括号中，则任何以分号分隔的参数都是标头参数，而不是 URI 参数。

20.1 Accept

Accept 头域遵循 [H14.1] 中定义的语法。 语义也是相同的，除了如果没有 Accept 头字段

存在，服务器应该假设默认值 application/sdp。

空的 Accept 标头字段表示不接受任何格式。

例子：

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Accept	R		-	o	-	o	m*	o
Accept	2xx		-	-	-	o	m*	o
Accept	415		-	c	-	c	c	c
Accept-Encoding	R		-	o	-	o	o	o
Accept-Encoding	2xx		-	-	-	o	m*	o
Accept-Encoding	415		-	c	-	c	c	c
Accept-Language	R		-	o	-	o	o	o
Accept-Language	2xx		-	-	-	o	m*	o
Accept-Language	415		-	c	-	c	c	c
Alert-Info	R	ar	-	-	-	o	-	-
Alert-Info	180	ar	-	-	-	o	-	-
Allow	R		-	o	-	o	o	o
Allow	2xx		-	o	-	m*	m*	o
Allow	r		-	o	-	o	o	o
Allow	405		-	m	-	m	m	m
Authentication-Info	2xx		-	o	-	o	o	o
Authorization	R		o	o	o	o	o	o
Call-ID	c	r	m	m	m	m	m	m
Call-Info		ar	-	-	-	o	o	o
Contact	R		o	-	-	m	o	o
Contact	1xx		-	-	-	o	-	-
Contact	2xx		-	-	-	m	o	o
Contact	3xx	d	-	o	-	o	o	o
Contact	485		-	o	-	o	o	o
Content-Disposition			o	o	-	o	o	o
Content-Encoding			o	o	-	o	o	o
Content-Language			o	o	-	o	o	o
Content-Length		ar	t	t	t	t	t	t
Content-Type			*	*	-	*	*	*
CSeq	c	r	m	m	m	m	m	m
Date		a	o	o	o	o	o	o
Error-Info	300-699	a	-	o	o	o	o	o
Expires			-	-	-	o	-	o
From	c	r	m	m	m	m	m	m
In-Reply-To	R		-	-	-	o	-	-
Max-Forwards	R	amr	m	m	m	m	m	m
Min-Expires	423		-	-	-	-	-	m
MIME-Version			o	o	-	o	o	o
Organization		ar	-	-	-	o	o	o

Table 2: Summary of header fields, A--O

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Priority	R	ar	-	-	-	o	-	-
Proxy-Authenticate	407	ar	-	m	-	m	m	m
Proxy-Authenticate	401	ar	-	o	o	o	o	o
Proxy-Authorization	R	dr	o	o	-	o	o	o
Proxy-Require	R	ar	-	o	-	o	o	o
Record-Route	R	ar	o	o	o	o	o	-
Record-Route	2xx,18x	mr	-	o	o	o	o	-
Reply-To			-	-	-	o	-	-
Require		ar	-	c	-	c	c	c
Retry-After	404,413,480,486		-	o	o	o	o	o
	500,503		-	o	o	o	o	o
	600,603		-	o	o	o	o	o
Route	R	adr	c	c	c	c	c	c
Server	r		-	o	o	o	o	o
Subject	R		-	-	-	o	-	-
Supported	R		-	o	o	m*	o	o
Supported	2xx		-	o	o	m*	m*	o
Timestamp			o	o	o	o	o	o
To	c(1)	r	m	m	m	m	m	m
Unsupported	420		-	m	-	m	m	m
User-Agent			o	o	o	o	o	o
Via	R	amr	m	m	m	m	m	m
Via	rc	dr	m	m	m	m	m	m
Warning	r		-	o	o	o	o	o
WWW-Authenticate	401	ar	-	m	-	m	m	m
WWW-Authenticate	407	ar	-	o	-	o	o	o

Table 3: Summary of header fields, P--Z; (1): copied with possible addition of tag

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

20.2 Accept-Encoding

Accept-Encoding 头域与 Accept 类似，但限制了响应中可接受的内容编码 [H3.5]。 见 [H14.3]。 SIP 中的语义与 [H14.3] 中定义的语义相同。

一个空的 Accept-Encoding 头域是允许的。 相当于 Accept-Encoding：identity，即只允许 identity encoding，意思是不编码，是允许的。

如果没有 Accept-Encoding 头域，服务器应该假设一个默认值 identity。

这与 HTTP 定义略有不同，后者表明当不存在时，可以使用任何编码，但首选标识编码。

例子：

```
Accept-Encoding: gzip
```

20.3 Accept-Language

Accept-Language 头域在请求中用于指示原因短语、会话描述或作为响应中的消息体携带的状态响应的首选语言。 如果没有 Accept-Language 头域，服务器应该假设所有语言都可以被客户端接受。

Accept-Language 头域遵循 [H14.4] 中定义的语法。 基于“q”参数排序语言的规则也适用于 SIP。

例子：

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

20.4 Alert-Info

当出现在 INVITE 请求中时，Alert-Info 头域指定 UAS 的替代铃声。当出现在 180（Ringing）响应中时，Alert-Info 头域指定 UAC 的备用回铃音。 一个典型的用法是代理插入这个头域来提供一个独特的环特性。

Alert-Info 头域会引入安全风险。 这些风险和处理它们的方法在第 20.9 节中讨论，该节讨论了 Call-Info 头字段，因为风险是相同的。

此外，用户应该能够有选择地禁用此功能。

这有助于防止不受信任的元素使用此标头字段可能导致的中断。

例子：

```
Alert-Info: <http://www.example.com/sounds/moo.wav>
```

20.5 Allow

Allow 标头字段列出了生成消息的 UA 支持的方法集。

UA 理解的所有方法，包括 ACK 和 CANCEL，必须包含在 Allow 头字段中的方法列表中（如果存在）。 不能将 Allow 头字段的缺失解释为意味着发送消息的 UA 不支持任何方法。 相反，这意味着 UA 没有提供任何关于它支持哪些方法的信息。

在对 OPTIONS 以外的方法的响应中提供 Allow 标头字段可减少所需消息的数量。

例子：

```
Allow: INVITE, ACK, OPTIONS, CANCEL, BYE
```

20.6 Authentication-Info

Authentication-Info 标头字段提供与 HTTP Digest 的相互身份验证。 UAS 可以在 2xx 响应中包含此头字段，该请求已使用基于 Authorization 头字段的摘要成功进行身份验证。

语法和语义遵循 RFC 2617 [17] 中的规定。

例子：

```
Authentication-Info: nextnonce="47364c23432d2e131a5fb210812c"
```

20.7 Authorization

Authorization 头域包含一个 UA 的认证凭证。第 22.2 节概述了 Authorization 头字段的使用，第 22.4 节描述了与 HTTP 身份验证一起使用时的语法和语义。

此标头字段与 Proxy-Authorization 一起打破了有关多个标头字段值的一般规则。 虽然不是逗号分隔的列表，但该头域名称可能出现多次，并且不得使用第 7.3 节中描述的常规规则将其组合成单个头行。

在下面的示例中，Digest 参数周围没有引号：

```
Authorization: Digest username="Alice", realm="atlanta.com",
 nonce="84a4cc6f3082121f32b42a2187831a9e",
 response="7587245234b3434cc3412213e5f113a5432"
```

20.8 Call-ID

Call-ID 头域唯一地标识一个特定的邀请或特定客户端的所有注册。 单个多媒体会议可能会产生多个具有不同 Call-ID 的呼叫，例如，如果用户多次邀请单个人参加同一个（长时间运行的）会议。 呼叫 ID 区分大小写，并且简单地逐字节进行比较。

Call-ID 头域的紧凑形式是 i。

例子：

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@biloxi.com
i:f81d4fae-7dec-11d0-a765-00a0c91e6bf6@192.0.2.4
```

20.9 Call-Info

Call-Info 标头字段提供有关调用者或被调用者的附加信息，具体取决于它是在请求还是响应中找到。 URI 的用途由“用途”参数描述。 “icon”参数指定适合作为调用者或被调用者的图标表示的图像。 “info”参数一般描述调用者或被调用者，例如，通过网页。 “card”参数提供名片，例如，采用 vCard [36] 或 LDIF [37] 格式。 可以使用 IANA 和第 27 节中的程序注册其他令牌。

使用 Call-Info 标头字段可能会带来安全风险。 如果被调用者获取恶意调用者提供的 URI，则被调用者可能会面临显示不当或冒犯性内容、危险或非法内容等的风险。 因此，如果 UA 可以验证发起头字段的元素的真实性并信任该元素，则建议 UA 仅呈现 Call-Info 头字段中的信息。 这不必是对等 UA； 代理可以将此标头字段插入请求中。

例子：

```
Call-Info: <http://wwww.example.com/alice/photo.jpg> ;purpose=icon,
  <http://www.example.com/alice/> ;purpose=info
```

20.10 Contact

Contact 标头字段值提供了一个 URI，其含义取决于它所在的请求或响应的类型。

Contact 标头字段值可以包含显示名称、带有 URI 参数的 URI 和标头参数。

本文档定义了联系参数“q”和“expires”。 这些参数仅在联系人出现在 REGISTER 请求或响应中或 3xx 响应中时使用。 其他参数可以在其他规范中定义。

当标头字段值包含显示名称时，包含所有 URI 参数的 URI 包含在“<”和“>”中。 如果不存在“<”和“>”，则 URI 之后的所有参数都是标头参数，而不是 URI 参数。 如果需要更大的字符集，则显示名称可以是标记或带引号的字符串。

即使“display-name”为空，如果“addr-spec”包含逗号、分号或问号，则必须使用“name-addr”形式。 显示名称和“<”之间可能有也可能没有 LWS。

这些用于解析显示名称、URI 和 URI 参数以及标头参数的规则也适用于标头字段 To 和 From。

Contact 头域的作用类似于 HTTP 中的 Location 头域。 但是，HTTP 标头字段只允许一个地址，不加引号。 由于 URI 可以包含逗号和分号作为保留字符，因此它们可能分别被误认为是标题或参数分隔符。

Contact 标头字段的紧凑形式是 m（表示“已移动”）。

例子：

```
Contact: "Mr. Watson" <sip:watson@worcester.bell-telephone.com>
   ;q=0.7; expires=3600,
   "Mr. Watson" <mailto:watson@bell-telephone.com> ;q=0.1
m: <sips:bob@192.0.2.4>;expires=60
```

20.11 Content-Disposition

Content-Disposition 头域描述了消息体，或者对于多部分消息，消息体部分如何被 UAC 或 UAS 解释。 这个 SIP 头域扩展了 MIME Content-Type (RFC 2183 [18])。

Content-Disposition 标头的几个新“处置类型”由 SIP 定义。 值“session”表示主体部分描述了一个会话，用于呼叫或早期（预呼叫）媒体。 值“render”表示应该向用户显示或以其他方式呈现身体部分。 请注意，使用值“render”而不是“inline”是为了避免 MIME 正文显示为整

个消息呈现的一部分的含义（因为 SIP 消息的 MIME 正文通常不显示给用户）。 为了向后兼容，如果缺少 Content-Disposition 标头字段，服务器应该假设 Content-Type application/sdp 的主体是处置“会话”，而其他内容类型是“渲染”。

处置类型“图标”表示正文部分包含适合作为调用者或被调用者的图标表示的图像，当接收到消息时，用户代理可以以信息方式呈现该图像，或者在发生对话时持续呈现。 值“alert”表示正文部分包含信息，例如音频剪辑，应该由用户代理呈现，以提醒用户收到请求，通常是启动对话的请求； 例如，在发送 180 Ringing 临时响应后，该警报主体可以呈现为电话铃声。

任何具有向用户呈现内容的“处置类型”的 MIME 主体仅应在消息经过正确身份验证后进行处理。

处理参数，handling-param，描述了如果 UAS 接收到它不理解其内容类型或处置类型的消息体，它应该如何反应。 该参数定义了“可选”和“必需”的值。 如果缺少处理参数，则应假定值为“必需”。 处理参数在 RFC 3204 [19] 中描述。

如果缺少此标头字段，则 MIME 类型确定默认内容处置。 如果没有，则假定为“渲染”。

例子：

```
Content-Disposition: session
```

20.12 Content-Encoding

Content-Encoding 标头字段用作“媒体类型”的修饰符。 当存在时，它的值指示已将哪些附加内容编码应用于实体主体，因此必须应用哪些解码机制才能获得 Content-Type 标头字段引用的媒体类型。 Content-Encoding 主要用于允许在不丢失其底层媒体类型标识的情况下压缩正文。

如果多个编码已应用于实体主体，则内容编码必须按照它们应用的顺序列出。

所有内容编码值都不区分大小写。 IANA 充当内容编码价值令牌的注册机构。 有关内容编码语法的定义，请参见 [H3.5]。

客户端可以将内容编码应用于请求中的正文。服务器可以将内容编码应用于响应中的主体。服务器必须只使用请求的 Accept-Encoding 头域中列出的编码。

Content-Encoding 头域的紧凑形式是 e。
例子：

```
Content-Encoding: gzip
e: tar
```

20.13 Content-Language

```
See [H14.12]. Example:

   Content-Language: fr
```

20.14 Content-Length

Content-Length 标头字段指示发送给接收者的消息体的大小，以十进制的八位字节数表示。 应用程序应该使用该字段来指示要传输的消息体的大小，无论实体的媒体类型如何。 如果使用基于流的协议（例如 TCP）作为传输，则必须使用头字段。

消息正文的大小不包括分隔标头字段和正文的 CRLF。 任何大于或等于零的 Content-Length 都是有效值。 如果消息中没有正文，则 Content-Length 标头字段值必须设置为零。

省略 Content-Length 的能力简化了动态生成响应的类 cgi 脚本的创建。

头域的紧凑形式是 l。

例子：

```
Content-Length: 349
l: 173
```

20.15 Content-Type

Content-Type 头域表示发送给接收者的消息体的媒体类型。 “媒体类型”元素在 [H3.7] 中定义。 如果正文不为空，则必须存在 Content-Type 标头字段。 如果 body 为空，并且存在 Content-Type 头字段，则表明特定类型的 body 长度为零（例如，空的音频文件）。

头域的紧凑形式是 c。

例子：

```
Content-Type: application/sdp
c: text/html; charset=ISO-8859-4
```

20.16 CSeq

请求中的 Cseq 头字段包含一个十进制序列号和请求方法。 序列号必须可以表示为 32 位无符号整数。 CSeq 的方法部分区分大小写。 CSeq 头字段用于对对话中的事务进行排序，提供一种唯一标识事务的方法，并区分新请求和请求重传。 如果序列号和请求方法相同，则认为两个 Cseq 头字段相等。 例子：

```
CSeq: 4711 INVITE
```

20.17 Date

日期标题字段包含日期和时间。 与 HTTP/1.1 不同，SIP 仅支持最新的 RFC 1123 [20] 日期格式。 与 [H3.3] 中一样，SIP 将 SIP-date 中的时区限制为“GMT”，而 RFC 1123 允许任何时区。 RFC 1123 日期区分大小写。

Date 标头字段反映了请求或响应首次发送的时间。

没有电池支持时钟的简单终端系统可以使用 Date 头字段来获取当前时间的概念。 但是，在其 GMT 形式中，它要求客户知道他们与 GMT 的偏移量。

例子：

```
Date: Sat, 13 Nov 2010 23:29:00 GMT
```

20.18 Error-Info

Error-Info 标头字段提供指向有关错误状态响应的附加信息的指针。

SIP UAC 具有用户界面功能，从 PC 软件客户端上的弹出窗口和音频到“黑色”电话或通过网关连接的端点上的纯音频。 Error-Info 标头字段允许发送两者，而不是强制生成错误的服务器在发送带有详细原因短语的错误状态代码和播放录音之间进行选择。 然后，UAC 可以选择将哪个错误指示符呈现给调用者。

UAC 可以将 Error-Info 头域中的 SIP 或 SIPS URI 视为重定向中的联系人并生成新的 INVITE，从而建立记录的通知会话。 可以向用户呈现非 SIP URI。

例子：

```
SIP/2.0 404 The number you have dialed is not in service
Error-Info: <sip:not-in-service-recording@atlanta.com>
```

20.19 Expires

Expires 标头字段给出消息（或内容）过期的相对时间。

这的确切含义取决于方法。

INVITE 中的到期时间不会影响邀请可能导致的实际会话的持续时间。 然而，会话描述协议可以提供表达会话持续时间的时间限制的能力。

该字段的值是 0 到 (2**32)-1 之间的整数秒数（十进制），从收到请求开始计算。

例子：

```
Expires: 5
```

20.20 From

From 头域表示请求的发起者。 这可能与对话的发起者不同。 被调用者向调用者发送的请求使用 From 头字段中的被调用者地址。

可选的“显示名称”旨在由人类用户界面呈现。 如果客户端的身份要保持隐藏，系统应该使用显示名称“匿名”。 即使“display-name”为空，如果“addr-spec”包含逗号、问号或分号，则必须使用“name-addr”形式。 语法问题在第 7.3.1 节中讨论。

如果它们的 URI 匹配并且它们的参数匹配，则两个 From 头字段是等效的。 为了比较的目的，忽略一个头字段中不存在于另一个头字段中的扩展参数。 这意味着显示名称和尖括号的存在与否不会影响匹配。

有关解析显示名称、URI 和 URI 参数以及标头字段参数的规则，请参阅第 20.10 节。

From 头域的紧凑形式是 f。

例子：

```
From: "A. G. Bell" <sip:agb@bell-telephone.com> ;tag=a48s
From: sip:+12125551212@server.phone2net.com;tag=887s
f: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

20.21 In-Reply-To

In-Reply-To 标头字段枚举了此调用引用或返回的 Call-ID。 这些 Call-ID 可能已被客户端缓存，然后包含在返回调用中的此标头字段中。

这允许自动呼叫分配系统将返回呼叫路由到第一个呼叫的发起者。 这也允许被呼叫者过滤呼叫，以便只接受他们发起的呼叫的返回呼叫。 此字段不能替代请求身份验证。

例子：

```
In-Reply-To: 70710@saturn.bell-tel.com, 17320@saturn.bell-tel.com
```

20.22 Max-Forwards

Max-Forwards 标头字段必须与任何 SIP 方法一起使用，以限制可以将请求转发到下一个下游服务器的代理或网关的数量。 当客户端尝试跟踪看起来失败或在中间链中循环的请求链

时，这也很有用。

Max-Forwards 值是一个 0-255 范围内的整数，表示允许转发此请求消息的剩余次数。 每个转发请求的服务器都会减少此计数。 推荐的初始值为 70。

此标头字段应由无法以其他方式保证循环检测的元素插入。 例如，B2BUA 应该插入一个 Max-Forwards 头字段。

例子：

```
Max-Forwards: 6
```

20.23 Min-Expires

Min-Expires 标头字段传达由该服务器管理的软状态元素支持的最小刷新间隔。 这包括由注册商存储的联系人头字段。标头字段包含从 0 到 (2**32)-1 的十进制整数秒数。第 10.2.8、10.3 和 21.4.17 节描述了 423（Interval Too Brief）响应中头字段的使用。

例子：

```
Min-Expires: 60
```

20.24 MIME-Version

```
See [H19.4.1].

Example:

   MIME-Version: 1.0
```

20.25 Organization

组织标头字段传达发出请求或响应的 SIP 元素所属的组织的名称。

客户端软件可以使用该字段来过滤呼叫。

```
Example:

   Organization: Boxes by Bob
```

20.26 Priority

Priority 标头字段指示客户端感知到的请求的紧迫性。 Priority 标头字段描述了 SIP 请求对接收人或其代理应具有的优先级。 例如，它可能会被考虑到有关呼叫路由和接受的决策中。对于这些决定，不包含 Priority 头域的消息应该被视为指定了“正常”的优先级。 Priority 头

字段不影响通信资源的使用，例如路由器中的数据包转发优先级或对 PSTN 网关中电路的访问。 标头字段可以具有值“非紧急”、“正常”、“紧急”和“紧急”，但可以在别处定义附加值。建议仅在生命、肢体或财产面临迫在眉睫的危险时才使用“紧急”的值。 否则，没有为此头字段定义的语义。

这些是 RFC 2076 [38] 的值，加上“紧急”。

```
Examples:

   Subject: A tornado is heading our way!
   Priority: emergency

or

   Subject: Weekend plans
   Priority: non-urgent
```

20.27 Proxy-Authenticate

Proxy-Authenticate 标头字段值包含身份验证质询。

这个头域的使用在[H14.33]中定义。 有关其用法的更多详细信息，请参见第 22.3 节。

```
Example:

    Proxy-Authenticate: Digest realm="atlanta.com",
     domain="sip:ssl.carrier.com", qop="auth",
     nonce="f84f1cec41e6cbe5aea9c8e88d359",
     opaque="", stale=FALSE, algorithm=MD5
```

20.28 Proxy-Authorization

Proxy-Authorization 头字段允许客户端向需要身份验证的代理标识自己（或其用户）。 Proxy-Authorization 字段值由包含用户代理的身份验证信息的凭据组成，用于代理和/或请求资源的领域。

见第 22.3 节关于这个头域的用法的定义。

此标头字段与授权一起打破了有关多个标头字段名称的一般规则。 虽然不是逗号分隔的列表，但该标题字段名称可能出现多次，并且不得使用第 7.3.1 节中描述的常规规则组合成单个标题行。

```
Example:

Proxy-Authorization: Digest username="Alice", realm="atlanta.com",
   nonce="c60f3082ee1212b402a21831ae",
   response="245f23415f11432b3434341c022"
```

20.29 Proxy-Require

Proxy-Require 头域用于指示代理必须支持的代理敏感特性。 有关此消息的机制和使用示例的更多详细信息，请参见第 20.32 节。

```
Example:

   Proxy-Require: foo
```

20.30 Record-Route

Record-Route 标头字段由代理插入请求中，以强制对话中的未来请求通过代理进行路由。

第 16.12.1 节描述了它与 Route 头字段一起使用的示例。

```
Example:

   Record-Route: <sip:server10.biloxi.com;lr>,
                 <sip:bigbox3.site3.atlanta.com;lr>
```

20.31 Reply-To

Reply-To 头域包含一个逻辑返回 URI，它可能与 From 头域不同。 例如，URI 可以用于返回未接来电或未建立的会话。 如果用户希望保持匿名，则头字段应该从请求中省略或以不泄露任何私人信息的方式填充。

即使“display-name”为空，如果“addr-spec”包含逗号、问号或分号，则必须使用“name-addr”形式。 语法问题在第 7.3.1 节中讨论。

```
Example:

   Reply-To: Bob <sip:bob@biloxi.com>
```

20.32 Require

UAC 使用 Require 头字段来告诉 UAS 有关 UAC 期望 UAS 支持以处理请求的选项。 虽然是一个可选的头域，但如果它存在 Require 则不能被忽略。

Require 头域包含一个选项标签列表，在第 19.2 节中描述。 每个选项标签定义一个必须被理解为处理请求的 SIP 扩展。 通常，这用于指示需要理解一组特定的扩展头字段。 符合本规范的 UAC 必须仅包含对应于标准跟踪 RFC 的选项标签。

```
Example:

   Require: 100rel
```

20.33 Retry-After

Retry-After 标头字段可与 500（服务器内部错误）或 503（服务不可用）响应一起使用，以指示服务预计对请求客户端不可用的时间，并与 404（未找到）、413（ 请求实体太大）、480（暂时不可用）、486（此处忙）、600（忙）或 603（拒绝）响应以指示被叫方预计何时再次可用。 此字段的值是响应时间之后的正整数秒数（十进制）。

可选注释可用于指示有关回调时间的附加信息。 一个可选的“持续时间”参数表示从可用的初始时间开始，被叫方可以到达多长时间。 如果没有给出持续时间参数，则假定服务无限期可用。

```
Examples:

   Retry-After: 18000;duration=3600
   Retry-After: 120 (I'm in a meeting)
```

20.34 Route

Route 标头字段用于强制通过列出的代理集路由请求。 Route 头域的使用示例在第 16.12.1 节中。

```
Example:

   Route: <sip:bigbox3.site3.atlanta.com;lr>,
          <sip:server10.biloxi.com;lr>
```

20.35 Server

Server 头字段包含有关 UAS 用于处理请求的软件的信息。

揭示服务器的特定软件版本可能会使服务器更容易受到已知包含安全漏洞的软件的攻击。实现者应该使 Server 头域成为一个可配置的选项。

```
Example:

   Server: HomeServer v2
```

20.36 Subject

Subject 标头字段提供摘要或指示呼叫的性质，允许在无需解析会话描述的情况下进行呼叫过滤。 会话描述不必使用与邀请相同的主题指示。

Subject 标头字段的紧凑形式是 s。

```
Example:

    Subject: Need more boxes
    s: Tech Support
```

20.37 Supported

Supported 头域列举了 UAC 或 UAS 支持的所有扩展。

Supported 头字段包含选项标签列表，如第 19.2 节所述，UAC 或 UAS 可以理解这些标签。符合本规范的 UA 必须仅包含对应于标准跟踪 RFC 的选项标签。 如果为空，则表示不支持任何扩展。

Supported 头域的紧凑形式是 k。

```
Example:

    Supported: 100rel
```

20.38 Timestamp

Timestamp 头域描述了 UAC 何时向 UAS 发送请求。

有关如何生成对包含标头字段的请求的响应的详细信息，请参阅第 8.2.6 节。 尽管此处没有定义使用标头的规范行为，但它允许扩展或 SIP 应用程序获得 RTT 估计。

```
Example:

    Timestamp: 54
```

20.39 To

To 标头字段指定请求的逻辑接收者。

可选的“显示名称”旨在由人类用户界面呈现。 “tag”参数用作对话识别的一般机制。

有关“tag”参数的详细信息，请参见第 19.3 节。

相等的 To 头字段的比较与 From 头字段的比较相同。 有关解析显示名称、URI 和 URI 参数以及标头字段参数的规则，请参阅第 20.10 节。

To 头域的紧凑形式是 t。

以下是有效 To 标头字段的示例：

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

20.40 Unsupported

Unsupported 头域列出了 UAS 不支持的特性。 有关动机，请参见第 20.32 节。

```
Example:

   Unsupported: foo
```

20.41 User-Agent

User-Agent 标头字段包含有关发起请求的 UAC 的信息。 这个头域的语义在[H14.43]中定义。

揭示用户代理的特定软件版本可能会使用户代理更容易受到针对已知包含安全漏洞的软件的攻击。 实施者应该使 User-Agent 头域成为一个可配置的选项。

```
Example:

   User-Agent: Softphone Beta1.5
```

20.42 Via

Via 标头字段指示请求到目前为止所采用的路径，并指示在路由响应中应遵循的路径。 Via 头字段值中的分支 ID 参数用作事务标识符，并被代理用于检测循环。

Via 头字段值包含用于发送消息的传输协议、客户端的主机名或网络地址，以及可能希望接收响应的端口号。 Via 头域值还可以包含“maddr”、“ttl”、“received”和“branch”等参数，其含义和使用在其他章节中描述。 对于符合本规范的实现，分支参数的值必须以魔术 cookie“z9hG4bK”开头，如第 8.1.1.7 节所述。

这里定义的传输协议是“UDP”、“TCP”、“TLS”和“SCTP”。 “TLS”表示基于 TCP 的 TLS。 当请求发送到 SIPS URI 时，协议仍指示“SIP”，传输协议为 TLS。

```
Via: SIP/2.0/UDP erlang.bell-telephone.com:5060;branch=z9hG4bK87asdks7
Via: SIP/2.0/UDP 192.0.2.1:5060 ;received=192.0.2.207
     ;branch=z9hG4bK77asjd
```

Via 头域的紧凑形式是 v。

在此示例中，消息来自具有两个地址 192.0.2.1 和 192.0.2.207 的多宿主主机。 发件人猜错了将使用哪个网络接口。 Erlang.bell-telephone.com 注意到了不匹配，并在前一跳的 Via 头字段值中添加了一个参数，其中包含数据包实际来自的地址。

主机或网络地址和端口号不需要遵循 SIP URI 语法。 具体来说，允许“：”或“/”两侧的 LWS，

如下所示：

```
Via: SIP / 2.0 / UDP first.example.com: 4000;ttl=16
  ;maddr=224.2.0.1 ;branch=z9hG4bKa7c6a8dlze.1
```

尽管本规范要求所有请求中都存在分支参数，但标头字段的 BNF 表明它是可选的。 这允许与不必插入分支参数的 RFC 2543 元素进行互操作。

如果两个 Via 头域的 send-protocol 和 sent-by 域相等，它们的参数集相同，并且所有参数的值相等，则它们相等。

20.43 Warning

警告头字段用于携带有关响应状态的附加信息。 警告标头字段值与响应一起发送，并包含三位警告代码、主机名和警告文本。

“警告文本”应该是最有可能被接收响应的人类用户理解的自然语言。 该决定可以基于任何可用的知识，例如用户的位置、请求中的 Accept-Language 字段或响应中的 Content-Language 字段。 默认语言是 i-default [21]。

下面列出了当前定义的“警告代码”，并附有推荐的英文警告文本及其含义说明。 这些警告描述了会话描述引起的故障。 以“3”开头的警告代码的第一位表示特定于 SIP 的警告。 警告 300 到 329 保留用于指示会话描述中关键字的问题，330 到 339 是与会话描述中请求的基本网络服务相关的警告，370 到 379 是与会话描述中请求的定量 QoS 参数相关的警告，390 通过 399 是不属于上述类别之一的杂项警告。

300 不兼容的网络协议：会话描述中包含的一个或多个网络协议不可用。

301 不兼容的网络地址格式：会话描述中包含的一种或多种网络地址格式不可用。

302 不兼容的传输协议：会话描述中描述的一种或多种传输协议不可用。

303 不兼容的带宽单位：不理解会话描述中包含的一个或多个带宽测量单位。

304 媒体类型不可用：会话描述中包含的一种或多种媒体类型不可用。

305 不兼容的媒体格式：会话描述中包含的一种或多种媒体格式不可用。

306 属性不理解：不支持会话描述中的一个或多个媒体属性。

307 会话描述参数不理解：不理解上面列出的参数以外的参数。

330 组播不可用：用户所在站点不支持组播。

331 单播不可用：用户所在的站点不支持单播通信（通常是由于防火墙的存在）。

370 带宽不足：会话描述中指定的或媒体定义的带宽超过了已知可用的带宽。

399 杂项警告：警告文本可以包括要呈现给人类用户或记录的任意信息。 收到此警告的系统不得采取任何自动操作。

1xx 和 2xx 已被 HTTP/1.1 占用。

可以通过 IANA 定义额外的“警告代码”，如第 27.2 节中所定义。

```
Examples:

   Warning: 307 isi.edu "Session parameter 'foo' not understood"
   Warning: 301 isi.edu "Incompatible network address type 'E.164'"
```

20.44 WWW-Authenticate

WWW-Authenticate 标头字段值包含身份验证质询。 有关其用法的更多详细信息，请参见第 22.2 节。

```
Example:

   WWW-Authenticate: Digest realm="atlanta.com",
     domain="sip:boxesbybob.com", qop="auth",
     nonce="f84f1cec41e6cbe5aea9c8e88d359",
     opaque="", stale=FALSE, algorithm=MD5
```

21 响应代码

响应代码与 HTTP/1.1 响应代码一致并对其进行了扩展。 并不是所有的 HTTP/1.1 响应代码都是合适的，这里只给出合适的。 不应使用其他 HTTP/1.1 响应代码。 此外，SIP 定义了一个新类 6xx。

21.1 临时 1xx

临时响应，也称为信息响应，表明所联系的服务器正在执行一些进一步的操作，并且还没有确定的响应。 如果服务器预计需要超过 200 毫秒才能获得最终响应，则它会发送 1xx 响应。 请注意，1xx 响应的传输不可靠。 它们永远不会导致客户端发送 ACK。 临时 (1xx) 响应可能包含消息正文，包括会话描述。

21.1.1 100 Trying

此响应表明下一跳服务器已收到请求，并且正在代表此调用采取一些未指定的操作（例如，正在查询数据库）。 与所有其他临时响应一样，此响应会停止 UAC 重新传输 INVITE。 100（尝试）响应与其他临时响应不同，因为它永远不会由有状态代理向上游转发。

21.1.2 180 Ringing

收到 INVITE 的 UA 正在尝试提醒用户。 此响应可用于启动本地回铃。

21.1.3 181 Call Is Being Forwarded

服务器可以使用此状态码来指示呼叫正被转发到一组不同的目的地。

21.1.4 182 Queued

被叫方暂时不可用，但服务器已决定将呼叫排队而不是拒绝它。 当被调用者可用时，它将返回适当的最终状态响应。 原因短语可以提供有关呼叫状态的更多详细信息，例如，“5 个呼叫排队；预期等待时间为 15 分钟”。 服务器可以发出几个 182 (Queued) 响应来更新呼叫者关于排队呼叫的状态。

21.1.5 183 Session Progress

183（会话进度）响应用于传达有关未分类的呼叫进度的信息。 Reason-Phrase、标头字段或消息正文可用于传达有关呼叫进度的更多详细信息。

21.2 成功 2xx

请求成功。

21.2.1 200 OK

请求已成功。 响应返回的信息取决于请求中使用的方法。

21.3 重定向 3xx

3xx 响应提供有关用户的新位置或可能能够满足呼叫的替代服务的信息。

21.3.1 300 Multiple Choices

请求中的地址解析为多个选项，每个选项都有自己的特定位置，用户（或 UA）可以选择首选通信端点并将其请求重定向到该位置。

如果接受请求头字段允许，响应可以包含一个消息体，其中包含资源特征和位置列表，用户或 UA 可以从中选择最合适的一个。 但是，尚未为此邮件正文定义任何 MIME 类型。

选择也应该被列为联系人字段（第 20.10 节）。 与 HTTP 不同，SIP 响应可以包含多个联系人字段或联系人字段中的地址列表。 UA 可以使用 Contact 头域的值进行自动重定向，也可以要求用户确认选择。 但是，本规范没有为这种自动选择定义任何标准。

如果可以在几个不同的位置到达被调用者并且服务器不能或不喜欢代理请求，则此状态响应是适当的。

21.3.2 301 Moved Permanently

在 Request-URI 中的地址上找不到用户，请求客户端应该在 Contact 头域（第 20.10 节）给出的新地址处重试。 请求者应该使用这个新值更新任何本地目录、地址簿和用户位置缓存，并将未来的请求重定向到列出的地址。

21.3.3 302 Moved Temporarily

请求客户端应该在 Contact 头域（第 20.10 节）给出的新地址重试请求。 新请求的 Request-URI 使用响应中的 Contact 标头字段的值。

Contact URI 的有效期可以通过 Expires（第 20.19 节）头域或 Contact 头域中的 expires 参数来指示。 代理和 UA 都可以在过期期间缓存这个 URI。 如果没有明确的过期时间，则该地址仅对递归有效一次，并且不得为将来的事务缓存。

如果从 Contact 头域缓存的 URI 失败，来自重定向请求的 Request-URI 可能会再次尝试一次。

临时 URI 可能在到期时间之前就已过期，并且新的临时 URI 可能可用。

21.3.4 305 Use Proxy

请求的资源必须通过 Contact 字段给出的代理访问。 Contact 字段给出代理的 URI。 接收者应该通过代理重复这个单一的请求。 305（使用代理）响应必须仅由 UAS 生成。

21.3.5 380 Alternative Service

呼叫不成功，但可以提供替代服务。

替代服务在响应的消息正文中进行了描述。 这些机构的格式在这里没有定义，可能是未来标准化的主题。

21.4 请求失败 4xx

4xx 响应是来自特定服务器的明确失败响应。 客户端不应该在没有修改的情况下重试相同的请求（例如，添加适当的授权）。 但是，对不同服务器的相同请求可能会成功。

21.4.1 400 Bad Request

由于语法错误，无法理解该请求。Reason-Phrase 应该更详细地识别语法问题，例如，“Missing Call-ID header field”。

21.4.2 401 Unauthorized

该请求需要用户身份验证。 此响应由 UAS 和注册商发出，而 407（需要代理身份验证）由代理服务器使用。

21.4.3 402 Payment Required

保留供将来使用。

21.4.4 403 Forbidden

服务器理解请求，但拒绝执行。 授权将无济于事，并且不应重复请求。

21.4.5 404 Not Found

服务器有明确的信息表明用户不存在于 Request-URI 中指定的域中。 如果 Request-URI 中的域与请求接收者处理的任何域都不匹配，也会返回此状态。

21.4.6 405 Method Not Allowed

Request-Line 中指定的方法是可以理解的，但对于 Request-URI 标识的地址是不允许的。

响应必须包含一个 Allow 头字段，该字段包含指定地址的有效方法列表。

21.4.7 406 Not Acceptable

请求标识的资源只能生成响应实体，其内容特征根据请求中发送的 Accept 头字段不可接受。

21.4.8 407 Proxy Authentication Required

此代码类似于 401（未授权），但表明客户端必须首先通过代理验证自己。 SIP 访问认证在第 26 节和第 22.3 节中解释。

此状态代码可用于访问通信通道（例如，电话网关）而不是被叫方需要身份验证的应用程序。

21.4.9 408 Request Timeout

服务器无法在适当的时间内产生响应，例如，如果它无法及时确定用户的位置。 客户端可以在以后的任何时间重复请求而无需修改。

21.4.10 410 Gone

请求的资源在服务器上不再可用，并且不知道转发地址。 预计这种情况将被视为永久性的。如果服务器不知道或无法确定条件是否是永久的，则应该使用状态代码 404（未找到）。

21.4.11 413 Request Entity Too Large

服务器拒绝处理请求，因为请求实体主体大于服务器愿意或能够处理的。 服务器可以关闭连接以阻止客户端继续请求。

如果条件是临时的，服务器应该包含一个 Retry-After 头域来指示它是临时的，并且在什么时间之后客户端可以重试。

21.4.12 414 Request-URI Too Long

服务器拒绝为请求提供服务，因为 Request-URI 比服务器愿意解释的要长。

21.4.13 415 Unsupported Media Type

服务器拒绝为请求提供服务，因为请求的消息正文采用了服务器不支持所请求方法的格式。服务器必须根据内容的具体问题，使用 Accept、Accept-Encoding 或 Accept-Language 头字段返回可接受格式的列表。 该响应的 UAC 处理在第 8.1.3.5 节中描述。

21.4.14 416 Unsupported URI Scheme

服务器无法处理请求，因为 Request-URI 中的 URI 方案对服务器来说是未知的。 客户端对此响应的处理在第 8.1.3.5 节中描述。

21.4.15 420 Bad Extension

服务器不理解 Proxy-Require（第 20.29 节）或 Require（第 20.32 节）头字段中指定的协议扩展。 服务器必须在响应的 Unsupported 头字段中包含不支持的扩展列表。 该响应的 UAC 处理在第 8.1.3.5 节中描述。

21.4.16 421 Extension Required

UAS 需要一个特定的扩展来处理请求，但是这个扩展没有列在请求的 Supported 头字段中。带有此状态码的响应必须包含一个列出所需扩展的 Require 头字段。

UAS 不应该使用这个响应，除非它真的不能为客户端提供任何有用的服务。 相反，如果在 Supported 头字段中没有列出所需的扩展，服务器应该使用基线 SIP 功能和客户端支持的任何扩展来处理请求。

21.4.17 423 Interval Too Brief

服务端拒绝请求，因为请求刷新的资源过期时间太短。 注册商可以使用此响应来拒绝其 Contact 标头字段过期时间太短的注册。 此响应的使用和相关的 Min-Expires 头字段在第 10.2.8、10.3 和 20.23 节中描述。

21.4.18 480 Temporarily Unavailable

被叫终端系统联系成功，但被叫当前不可用（例如，未登录、已登录但处于无法与被叫通信的状态，或已激活“请勿打扰”功能）。 响应可能会在 Retry-After 头字段中指示更好的调用时间。 用户也可能在其他地方可用（此服务器不知道）。 原因短语应该指出被调用者不可用的更精确原因。 该值应该由 UA 设置。 状态 486（此处忙）可用于更准确地指示呼叫失败的特定原因。

此状态也由识别由 Request-URI 标识的用户的重定向或代理服务器返回，但当前没有该用户的有效转发位置。

21.4.19 481 Call/Transaction Does Not Exist

此状态表明 UAS 收到了与任何现有对话或事务不匹配的请求。

21.4.20 482 Loop Detected

服务器检测到一个循环（第 16.3 节第 4 项）。

21.4.21 483 Too Many Hops

服务器收到一个请求，其中包含一个值为 0 的 Max-Forwards（第 20.22 节）标头字段。

21.4.22 484 Address Incomplete

服务器收到一个带有不完整 Request-URI 的请求。 应在原因短语中提供附加信息。

此状态代码允许重叠拨号。 使用重叠拨号，客户端不知道拨号字符串的长度。 它发送长度增加的字符串，提示用户进行更多输入，直到它不再收到 484（地址不完整）状态响应。

21.4.23 485 Ambiguous

Request-URI 不明确。 响应可能包含联系人头字段中可能的明确地址的列表。 揭示替代方案可能会侵犯用户或组织的隐私。 必须可以配置服务器以响应状态 404（未找到）或禁止列出模糊请求 URI 的可能选择列表。

使用 Request-URI sip:lee@example.com 对请求的示例响应：

```
SIP/2.0 485 Ambiguous
Contact: Carol Lee <sip:carol.lee@example.com>
Contact: Ping Lee <sip:p.lee@example.com>
Contact: Lee M. Foote <sips:lee.foote@example.com>
```

一些电子邮件和语音邮件系统提供此功能。 由于语义不同，因此使用了与 3xx 分开的状态代码：对于 300，假设提供的选择将到达同一个人或服务。 虽然自动选择或顺序搜索对于 3xx 响应有意义，但对于 485（不明确）响应则需要用户干预。

21.4.24 486 Busy Here

被呼叫方的终端系统已成功联系，但被呼叫方目前不愿意或无法在该终端系统上接听额外的呼叫。 响应可能会在 Retry-After 头字段中指示更好的调用时间。 用户也可以在别处可用，例如通过语音邮件服务。 如果客户端知道没有其他终端系统将能够接受此呼叫，则应该使用状态 600（无处不在）。

21.4.25 487 Request Terminated

请求被 BYE 或 CANCEL 请求终止。 对于 CANCEL 请求本身，永远不会返回此响应。

21.4.26 488 Not Acceptable Here

响应与 606（不可接受）含义相同，但仅适用于 Request-URI 寻址的特定资源，并且请求可

能在其他地方成功。

包含媒体能力描述的消息体可能出现在响应中，根据 INVITE（或 application/sdp，如果不存在）中的 Accept 头字段格式化，与 200（OK）中的消息体相同 响应 OPTIONS 请求。

21.4.27 491 Request Pending

该请求由在同一对话中具有待处理请求的 UAS 接收。 第 14.2 节描述了如何解决这种“眩光”情况。

21.4.28 493 Undecipherable

请求由 UAS 接收，其中包含加密的 MIME 正文，收件人不拥有或不会提供适当的解密密钥。 此响应可能有一个包含适当公钥的单个正文，该公钥应用于加密发送到此 UA 的 MIME 正文。 此响应代码的使用详情可在第 23.2 节中找到。

21.5 服务器故障 5xx

5xx 响应是服务器本身出错时给出的失败响应。

21.5.1 500 Server Internal Error

服务器遇到了阻止它完成请求的意外情况。 客户端可以显示特定的错误情况，并可以在几秒钟后重试请求。

如果条件是临时的，服务器可以指示客户端何时可以使用 Retry-After 头字段重试请求。

21.5.2 501 Not Implemented

服务器不支持完成请求所需的功能。 当 UAS 无法识别请求方法并且无法为任何用户支持它时，这是适当的响应。 （代理转发所有请求而不管方法。）

请注意，当服务器识别出请求方法时会发送 405（不允许方法），但不允许或不支持该方法。

21.5.3 502 Bad Gateway

服务器在充当网关或代理时，在尝试满足请求时从其访问的下游服务器接收到无效响应。

21.5.4 503 Service Unavailable

由于服务器临时过载或维护，服务器暂时无法处理请求。 服务器可以在 Retry-After 头域中指示客户端何时重试请求。 如果没有给出 Retry-After，客户端必须像收到 500（服务器内部错误）响应一样行事。

接收到 503（服务不可用）的客户端（代理或 UAC）应该尝试将请求转发到备用服务器。 如果存在，它不应该在 Retry-After 头字段中指定的持续时间内将任何其他请求转发到该服务器。

服务器可以拒绝连接或丢弃请求，而不是响应 503（服务不可用）。

21.5.5 504 Server Time-out

服务器在尝试处理请求时没有收到来自它访问的外部服务器的及时响应。 如果上游服务器在 Expires 标头字段中指定的期限内没有响应，则应使用 408（请求超时）。

21.5.6 505 Version Not Supported

服务器不支持或拒绝支持请求中使用的 SIP 协议版本。 服务器表示它无法或不愿意使用与客户端相同的主要版本完成请求，除了此错误消息。

21.5.7 513 Message Too Large

由于消息长度超出其能力，服务器无法处理该请求。

21.6 全局故障 6xx

6xx 响应表明服务器拥有关于特定用户的明确信息，而不仅仅是 Request-URI 中指示的特定实例。

21.6.1 600 Busy Everywhere

被叫方终端系统已成功联系，但被叫方正忙，此时不想接听电话。 响应可能会在 Retry-After 头字段中指示更好的调用时间。 如果被呼叫者不想透露拒绝呼叫的原因，则被呼叫者使用状态码 603（拒绝）。 仅当客户端知道没有其他端点（例如语音邮件系统）将回答该请求时，才会返回此状态响应。 否则，应返回 486（此处忙）。

21.6.2 603 Decline

被调用者的机器已成功联系，但用户明确不希望或不能参与。 响应可能会在 Retry-After 头字段中指示更好的调用时间。 仅当客户端知道没有其他端点将响应该请求时，才会返回此状态响应。

21.6.3 604 Does Not Exist Anywhere

服务器拥有 Request-URI 中所指示的用户在任何地方都不存在的权威信息。

21.6.4 606 Not Acceptable

已成功联系用户代理，但会话描述的某些方面（例如请求的媒体、带宽或寻址方式）不可接受。

606（不可接受）响应意味着用户希望进行通信，但不能充分支持所描述的会话。 606（不可接受）响应可以在警告头字段中包含原因列表，描述为什么所描述的会话不被支持。 警告原因代码在第 20.43 节中列出。

包含媒体能力描述的消息体可能出现在响应中，根据 INVITE（或 application/sdp，如果不存在）中的 Accept 头字段格式化，与 200（OK）中的消息体相同 响应 OPTIONS 请求。

希望不需要频繁协商，当邀请新用户加入已经存在的会议时，协商可能无法进行。 由邀请发起者决定是否对 606（不可接受）响应采取行动。

仅当客户端知道没有其他端点将响应该请求时，才会返回此状态响应。

22 HTTP 认证的使用

SIP 提供了一种无状态、基于质询的身份验证机制，该机制基于 HTTP 中的身份验证。 任何时候代理服务器或 UA 收到请求（第 22.1 节中给出的例外情况除外），它可以质询请求的发起者以提供其身份的保证。 一旦确定了发起者，请求的接收者应该确定这个用户是否被授权提出有问题的请求。 本文档中不推荐或讨论授权系统。

本节中描述的“摘要”身份验证机制仅提供消息身份验证和重放保护，没有消息完整性或机密性。 需要采取超出 Digest 提供的保护措施来防止主动攻击者修改 SIP 请求和响应。

请注意，由于其安全性较弱，“基本”身份验证的使用已被弃用。 服务器不得使用“基本”授权方案接受凭据，服务器也不得使用“基本”进行质询。 这是对 RFC 2543 的更改。

22.1 框架

SIP 身份验证的框架与 HTTP (RFC 2617 [17]) 的框架非常相似。 特别是，auth-scheme、auth-param、challenge、realm、realm-value 和 credentials 的 BNF 是相同的（尽管不允许使用“Basic”作为方案）。 在 SIP 中，UAS 使用 401（未授权）响应来挑战 UAC 的身份。 此外，注册商和重定向服务器可以使用 401（未授权）响应进行身份验证，但代理不得，而是可以使用 407（需要代理身份验证）响应。 在各种消息中包含 Proxy-Authenticate、Proxy-Authorization、WWW-Authenticate 和 Authorization 的要求与 RFC 2617 [17] 中描述的要求相同。

由于 SIP 没有规范根 URL 的概念，因此保护空间的概念在 SIP 中的解释不同。 领域字符串单独定义了保护域。 这是对 RFC 2543 的更改，在 RFC 2543 中，Request-URI 和领域共同定义了保护域。

先前的保护域定义引起了一些混乱，因为 UAC 发送的 Request-URI 和挑战服务器接收到的 Request-URI 可能不同，实际上 Request-URI 的最终形式可能不为 UAC 所知。 UAC。 此外，之前的定义依赖于 Request-URI 中是否存在 SIP URI，并且似乎排除了替代 URI 方案（例如，tel URL）。

将对接收到的请求进行身份验证的用户代理或代理服务器的操作员必须遵守以下准则来为其服务器创建领域字符串：

o 领域字符串必须是全局唯一的。 建议领域字符串包含主机名或域名，遵循 RFC 2617 [17] 第 3.2.1 节中的建议。

o 领域字符串应该呈现一个人类可读的标识符，可以呈现给用户。

For example:

```
INVITE sip:bob@biloxi.com SIP/2.0
Authorization: Digest realm="biloxi.com", <...>
```

一般而言，SIP 认证对于特定领域（保护域）是有意义的。 因此，对于 Digest 身份验证，每个这样的保护域都有自己的一组用户名和密码。 如果服务器不需要对特定请求进行身份验证，它可以接受默认用户名“anonymous”，它没有密码（密码为“”）。 类似地，代表许多用户的 UAC，例如 PSTN 网关，可能有自己的设备特定用户名和密码，而不是特定用户的帐户，用于他们的领域。

虽然服务器可以合法地挑战大多数 SIP 请求，但本文档定义的两个请求需要对身份验证进行特殊处理：ACK 和 CANCEL。

在使用响应来携带用于计算 nonces 的值的身份验证方案（例如 Digest）下，任何不接受响应的请求都会出现一些问题，包括 ACK。 出于这个原因，服务器接受的 INVITE 中的任何凭据都必须被该服务器接受以用于 ACK。 创建 ACK 消息的 UAC 将复制所有出现在 ACK 对应的 INVITE 中的 Authorization 和 Proxy-Authorization 头字段值。 服务器不得尝试挑战

ACK。

尽管 CANCEL 方法确实接受了响应（2xx），但服务器不得尝试挑战 CANCEL 请求，因为这些请求无法重新提交。 通常，如果 CANCEL 请求来自发送被取消请求的同一跃点，则服务器应该接受它（前提是如第 26.2.1 节所述的某种传输或网络层安全关联已到位）。

当 UAC 接收到质询时，如果 UAC 设备不知道，它应该向用户呈现质询中“领域”参数的内容（出现在 WWW-Authenticate 头字段或 Proxy-Authenticate 头字段中） 相关领域的凭证。使用其领域的凭据预配置 UA 的服务提供商应该意识到，当在预配置的设备上受到挑战时，用户将没有机会为该领域提供自己的凭据。

最后，请注意，即使 UAC 可以找到与正确领域相关联的凭据，也可能存在这些凭据可能不再有效或者挑战服务器出于任何原因将不接受这些凭据（尤其是当“匿名”与 未提交密码）。在这种情况下，服务器可能会重复其质询，或者它可能会以 403 Forbidden 响应。 UAC 不得使用刚刚被拒绝的凭据重新尝试请求（尽管如果 nonce 过时，请求可能会被重试）。

22.2 用户对用户认证

当 UAS 收到来自 UAC 的请求时，UAS 可以在处理请求之前验证发起者。 如果请求中没有提供凭据（在授权头字段中），UAS 可以通过拒绝带有 401（未授权）状态代码的请求来挑战发起者提供凭据。

WWW-Authenticate response-header 字段必须包含在 401（未授权）响应消息中。 该字段值由至少一个指示身份验证方案和适用于该领域的参数的质询组成。

401 质询中的 WWW-Authenticate 标头字段的示例是：

```
WWW-Authenticate: Digest
        realm="biloxi.com",
        qop="auth,auth-int",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

当发起 UAC 收到 401（未授权）时，它应该（如果能够）使用正确的凭据重新发起请求。在继续之前，UAC 可能需要发起用户的输入。 一旦提供了身份验证凭据（直接由用户提供，或在内部密钥环中发现），UA 应该缓存 To 标头字段和“领域”的给定值的凭据，并尝试在下一次重用这些值 请求该目的地。 UA 可以以任何他们想要的方式缓存凭证。

如果找不到域的凭据，UAC 可以尝试使用用户名“anonymous”且没有密码（密码“”）重试请求。

一旦找到凭证，任何希望通过 UAS 或注册商验证自己的 UA——通常但不一定，在收到 401（未授权）响应后——可以通过在请求中包含 Authorization 头字段来实现。 Authorization 字段值由凭证组成，其中包含所请求资源领域的 UA 身份验证信息以及支持身份验证和重放保护所需的参数。

Authorization 头域的一个例子是：

```
Authorization: Digest username="bob",
        realm="biloxi.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        uri="sip:bob@biloxi.com",
        qop=auth,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

当 UAC 在收到 401（未授权）或 407（需要代理身份验证）响应后重新提交带有其凭据的请求时，它必须像发送更新请求时一样增加 CSeq 头字段值。

22.3 代理到用户认证

类似地，当 UAC 向代理服务器发送请求时，代理服务器可以在处理请求之前对发起者进行身份验证。 如果请求中没有提供凭据（在 Proxy-Authorization 标头字段中），则代理可以通过拒绝具有 407（需要代理身份验证）状态代码的请求来挑战发起者提供凭据。 代理必须使用适用于所请求资源的代理的 Proxy-Authenticate 头字段值填充 407（需要代理身份验证）消息。

Proxy-Authenticate 和 Proxy-Authorization 的使用与 [17] 中描述的并行，有一个区别。 代理不得向 Proxy-Authorization 标头字段添加值。 所有 407（需要代理验证）响应必须按照任何其他响应的程序向上游转发到 UAC。 UAC 负责添加 Proxy-Authorization 头字段值，该值包含已请求身份验证的代理领域的凭据。

如果代理要重新提交添加 Proxy-Authorization 标头字段值的请求，则需要在新请求中增加 CSeq。 但是，这会导致提交原始请求的 UAC 丢弃来自 UAS 的响应，因为 Cseq 值会不同。

当发起方 UAC 收到 407（需要代理身份验证）时，它应该（如果能够）使用正确的凭据重新发起请求。 它应该遵循与上面给出的响应 401 相同的“领域”参数的显示过程。

如果找不到域的凭据，UAC 可以尝试使用用户名“anonymous”且没有密码（密码“”）重试请求。

UAC 还应该缓存在重新发起的请求中使用的凭证。

推荐使用以下规则进行代理凭据缓存：

如果 UA 在对具有特定 Call-ID 的请求的 401/407 响应中接收到 Proxy-Authenticate 头字段值，则它应该在包含相同 Call-ID 的所有后续请求中包含该领域的凭据。 这些凭据不得跨对话缓存； 但是，如果 UA 配置了其本地出站代理的领域，当存在一个时，UA 可以跨对话缓存该领域的凭据。 请注意，这确实意味着对话中的未来请求可能包含沿 Route 标头路径的任何代理都不需要的凭据。

任何希望向代理服务器验证自己的 UA——通常，但不一定，在收到 407（需要代理验证）响应后——可以通过在请求中包含 Proxy-Authorization 头字段值来实现。

Proxy-Authorization 请求头字段允许客户端向需要身份验证的代理标识自己（或其用户）。 Proxy-Authorization 标头字段值由证书组成，其中包含 UA 的身份验证信息，用于所请求资源的代理和/或领域。

Proxy-Authorization 标头字段值仅适用于其领域在“领域”参数中标识的代理（此代理之前可能已要求使用 Proxy-Authenticate 字段进行身份验证）。 当在一个链中使用多个代理时，Proxy-Authorization 头字段值不得被其领域与该值中指定的“领域”参数不匹配的任何代理使用。

请注意，如果在 Proxy-Authorization 头字段中使用了不支持领域的身份验证方案，则代理服务器必须尝试解析所有 Proxy-Authorization 头字段值以确定其中一个是否具有代理服务器认为有效的值 证书。 因为这在大型网络中可能非常耗时，所以代理服务器应该使用支持 Proxy-Authorization 标头字段中的领域的身份验证方案。

如果请求被分叉（如第 16.7 节所述），则各种代理服务器和/或 UA 可能希望挑战 UAC。在这种情况下，分叉代理服务器负责将这些挑战聚合为单个响应。 在对分叉请求的响应中收到的每个 WWW-Authenticate 和 Proxy-Authenticate 值必须放入由分叉代理发送给 UA 的单个响应中； 这些标头字段值的顺序并不重要。

当代理服务器发出质询以响应请求时，它不会代理请求，直到 UAC 使用有效凭据重试请求。分叉代理可以同时将请求转发到需要身份验证的多个代理服务器，每个代理服务器依次不会转发请求，直到始发 UAC 在其各自领域中对其自身进行身份验证。 如果 UAC 不为每个质询提供凭据，则发出质询的代理服务器不会将请求转发到目标用户可能位于的 UA，因此，分叉的优点在很大程度上丧失了。

当重新提交其请求以响应包含多个质询的 401（未授权）或 407（需要代理身份验证）时，UAC 可以为每个 WWW-Authenticate 值包含一个 Authorization 值，并为每个 Proxy-Authenticate 值包含一个 Proxy-Authorization 值 UAC 希望提供凭证。 如上所述，请求中的多个凭证应该由“领域”参数来区分。

与同一领域关联的多个质询可能出现在同一 401（未经授权）或 407（需要代理身份验证）中。 例如，当分叉请求到达同一管理域中使用公共领域的多个代理时，可能会发生这种情况。 因此，当它重试请求时，UAC 可以在 Authorization 或 Proxy-Authorization 头字段中提供具有相同“领域”参数值的多个凭证。 相同的凭证应该用于相同的领域。

22.4 摘要式认证方案

本节描述将 HTTP Digest 身份验证方案应用于 SIP 所需的修改和说明。 SIP 方案的使用几乎与 HTTP [17] 完全相同。

由于 RFC 2543 基于 RFC 2069 [39] 中定义的 HTTP Digest，支持 RFC 2617 的 SIP 服务器必须确保它们向后兼容 RFC 2069。这种向后兼容的过程在 RFC 2617 中指定。但是请注意，SIP 服务器 不得接受或请求基本身份验证。

Digest 认证的规则遵循 [17] 中定义的规则，除了以下区别外，将“HTTP/1.1”替换为“SIP/2.0”：

1.挑战中包含的 URI 具有以下 BNF：

```
URI  =  SIP-URI / SIPS-URI
```

2. RFC 2617 中的 BNF 有一个错误，即 HTTP Digest 认证的 Authorization 头域的 'uri' 参数没有用引号引起来。 （RFC 2617 第 3.5 节中的示例是正确的。）对于 SIP，'uri' 必须用引号引起来。

3. digest-uri-value 的 BNF 为：

```
digest-uri-value  =  Request-URI ; as defined in Section 25
```

4. 基于 Etag 选择随机数的示例过程不适用于 SIP。

5. RFC 2617 [17] 中关于缓存操作的文本不适用于 SIP。

6. RFC 2617 [17] 要求服务器检查请求行中的 URI 和授权头字段中包含的 URI 是否指向同一资源。 在 SIP 上下文中，这两个 URI 可能指代不同的用户，因为在某些代理上进行转发。 因此，在 SIP 中，服务器可以检查 Authorization 头字段值中的 Request-URI 是否对应于服务器愿意接受转发或直接请求的用户，但如果这两个字段不匹配，则不一定是失败 相等的。

7. 作为对 Digest 认证方案中消息完整性保证的 A2 值计算的澄清，实现者应该假设，当 entity-body 为空时（即，当 SIP 消息没有正文时），实体的哈希 -body 解析为空字符串的 MD5 哈希，或者：

```
   H(entity-body) = MD5("") =
"d41d8cd98f00b204e9800998ecf8427e"
```

8. RFC 2617 指出，如果没有发送 qop 指令，则不得在授权（以及扩展的 Proxy-Authorization）头字段中发送 cnonce 值。 因此，任何依赖于 cnonce 的算法（包括“MD5-Sess”）都需要发送 qop 指令。 为了向后兼容 RFC 2069，在 RFC 2617 中使用“qop”参数是可选的； 由于 RFC 2543 基于 RFC 2069，遗憾的是“qop”参数必须为客户端和服务器接收可选。 然而，服务器必须总是在 WWW-Authenticate 和 Proxy-Authenticate 头域值中发送一个“qop”参数。 如果客户端在质询头字段中收到“qop”参数，它必须在任何生成的授权头字段中发送“qop”参数。

RFC 2543 不允许使用 Authentication-Info 标头字段（它有效地使用了 RFC 2069）。 但是，我们现在允许使用此标头字段，因为它提供对正文的完整性检查并提供相互身份验证。 RFC 2617 [17] 使用请求中的 qop 属性定义了向后兼容的机制。 服务器必须使用这些机制来确定客户端是否支持 RFC 2617 中未在 RFC 2069 中指定的新机制。

23 S/MIME

SIP 消息携带 MIME 主体，MIME 标准包括保护 MIME 内容以确保完整性和机密性的机制

（包括“multipart/signed”和“application/pkcs7-mime”MIME 类型，参见 RFC 1847 [22]、RFC 2630 [23] 和 RFC 2633 [24]）。 然而，实施者应该注意，可能有很少的网络中介（不是典型的代理服务器）依赖于查看或修改 SIP 消息（尤其是 SDP）的正文，并且安全的 MIME 可能会阻止这些中介的运行。

这尤其适用于某些类型的防火墙。

用于加密 RFC 2543 中描述的 SIP 消息的标头字段和正文的 PGP 机制已被弃用。

23.1 S/MIME 证书

为 S/MIME 目的而用于识别最终用户的证书在一个重要方面不同于服务器使用的证书 - 这些证书不是断言持有者的身份对应于特定的主机名，而是断言持有者 由最终用户地址标识。 该地址由 SIP 或 SIPS URI 的“userinfo”“@”和“domainname”部分的串联组成（换句话说，“bob@biloxi.com”形式的电子邮件地址），最常见的对应于 用户的记录地址。

这些证书还与用于签署或加密 SIP 消息正文的密钥相关联。 正文使用发送者的私钥进行签名（发送者可能在适当的情况下将其公钥包含在消息中），但正文使用预期接收者的公钥加密。 显然，发送者必须预先知道接收者的公钥才能加密消息体。 公钥可以存储在虚拟密钥环上的 UA 中。

每个支持 S/MIME 的用户代理必须包含一个专门用于最终用户证书的密钥环。 此密钥环应在记录地址和相应证书之间映射。 随着时间的推移，用户在使用相同的记录地址填充信令的原始 URI（From 头字段）时应该使用相同的证书。

任何依赖于最终用户证书存在的机制都受到严重限制，因为如今几乎没有统一的权威机构为最终用户应用程序提供证书。 但是，用户应该从已知的公共证书颁发机构获取证书。 作为替代方案，用户可以创建自签名证书。 26.4.2 节进一步探讨了自签名证书的含义。 实现还可以在所有 SIP 实体之间存在先前信任关系的部署中使用预配置的证书。

除了获取最终用户证书的问题之外，很少有著名的分发最终用户证书的集中式目录。 但是，证书的持有者应该在任何适当的公共目录中发布他们的证书。 类似地，UAC 应该支持一种机制，用于导入（手动或自动）在与 SIP 请求的目标 URI 对应的公共目录中发现的证书。

23.2 S/MIME 密钥交换

SIP 本身也可以用作以下列方式分发公钥的一种手段。

每当 CMS SignedData 消息用于 SIP 的 S/MIME 时，它必须包含带有验证签名所必需的公钥的证书。

当 UAC 发送包含启动对话的 S/MIME 正文的请求，或在对话上下文之外发送非 INVITE 请求时，UAC 应该将正文构造为 S/MIME 'multipart/signed' CMS SignedData 正文 . 如果所需

的 CMS 服务是 EnvelopedData（并且目标用户的公钥已知），UAC 应该发送封装在 SignedData 消息中的 EnvelopedData 消息。

当 UAS 接收到包含 S/MIME CMS 正文的请求时，UAS 应该首先验证证书，如果可能的话，使用证书颁发机构的任何可用根证书。 UAS 还应该确定证书的主题（对于 S/MIME，SubjectAltName 将包含适当的身份）并将此值与请求的 From 头字段进行比较。如果证书不能被验证，因为它是自签名的，或者由未知的权威机构签名，或者如果它是可验证的但它的主题不对应于请求的 From 头域，UAS 必须通知它的用户状态证书（包括证书的主题、其签名者和任何关键指纹信息）并在继续之前请求明确的许可。如果证书成功验证并且证书的主题对应于 SIP 请求的 From 头字段，或者如果用户（通知后）明确授权使用证书，UAS 应该将此证书添加到本地密钥环，由证书持有人的记录地址索引。

当一个 UAS 发送一个包含 S/MIME 主体的响应来回答对话中的第一个请求，或者一个对对话上下文之外的非邀请请求的响应时，UAS 应该将主体构造为一个 S/MIME 'multipart /signed' CMS SignedData 正文。 如果所需的 CMS 服务是 EnvelopedData，UAS 应该发送封装在 SignedData 消息中的 EnvelopedData 消息。

当 UAC 接收到包含 S/MIME CMS 主体的响应时，UAC 应该首先使用任何适当的根证书验证证书，如果可能的话。 UAC 还应该确定证书的主题并将该值与响应的 To 字段进行比较；尽管两者很可能不同，但这并不一定表明存在安全漏洞。如果证书由于自签名或由未知权威机构签名而无法验证，则 UAC 必须通知其用户证书的状态（包括证书的主题、其签名者和任何密钥指纹信息）和在继续之前请求明确的许可。如果证书验证成功，并且证书的主题对应于响应中的 To 头字段，或者如果用户（通知后）明确授权使用证书，UAC 应该将此证书添加到本地密钥环，由证书持有人的记录地址索引。如果 UAC 在之前的任何交易中都没有向 UAS 传输自己的证书，它应该使用 CMS SignedData 主体来进行下一个请求或响应。

在未来的情况下，当 UA 接收到包含对应于其密钥环中的值的 From 头字段的请求或响应时，UA 应该将这些消息中提供的证书与其密钥环中的现有证书进行比较。 如果存在差异，UA 必须通知其用户证书的更改（最好是表明这是潜在的安全漏洞）并在继续处理信令之前获得用户的许可。 如果用户授权这个证书，它应该和这个地址记录的任何先前值一起添加到密钥环中。

但是请注意，当使用自签名证书或由不知名的权威机构签名的证书时，这种密钥交换机制并不能保证密钥的安全交换——它容易受到众所周知的攻击。 然而，在作者看来，它提供的安全性众所周知总比没有好。 它实际上可以与广泛使用的 SSH 应用程序相媲美。 第 26.4.2 节更详细地探讨了这些限制。

如果 UA 接收到一个 S/MIME 正文，该正文已使用接收者未知的公钥加密，它必须以 493（不可解密）响应拒绝该请求。 此响应应该包含响应者的有效证书（如果可能，对应于在被拒绝请求的 To 头字段中给出的任何记录地址）在 MIME 正文中，带有“certs-only”“smime-type”参数。

在没有任何证书的情况下发送的 493（不可破译）表明响应者不能或不会使用 S/MIME 加密消息，尽管它们可能仍支持 S/MIME 签名。

请注意，如果用户代理接收包含非可选的 S/MIME 正文的请求（带有“必需”的 Content-Disposition 标头“处理”参数），如果 MIME 类型为 415 Unsupported Media Type 响应，则用户代理必须拒绝该请求 不明白。 当发送 S/MIME 时接收到此类响应的用户代理应该通知其用户远程设备不支持 S/MIME，并且如果合适，它可以随后在没有 S/MIME 的情况下重新发送请求； 但是，这个 415 响应可能构成降级攻击。

如果用户代理在请求中发送 S/MIME 正文，但收到包含不安全的 MIME 正文的响应，则 UAC 应该通知其用户会话无法安全。 但是，如果支持 S/MIME 的用户代理接收到带有不安全正文的请求，它不应该以安全正文响应，但如果它期望来自发件人的 S/MIME（例如，因为发件人的 From 标头字段值对应 到它的钥匙串上的一个身份），UAS 应该通知它的用户会话不能被保护。

当异常证书管理事件发生时，前面文本中出现的许多情况要求用户通知。 用户可能会问他们在这些情况下应该做什么。 首先，证书中的意外更改或预期安全时缺乏安全性是需要谨慎的原因，但不一定表明攻击正在进行中。 用户可能会中止任何连接尝试或拒绝他们收到的连接请求； 用电话术语来说，他们可以挂断电话并回拨。 用户可能希望找到一种替代方式来联系对方并确认他们的密钥已合法更改。 请注意，用户有时会被迫更改他们的证书，例如当他们怀疑他们的私钥的机密性已被泄露时。 当他们的私钥不再是私有的时，用户必须合法地生成一个新密钥，并与任何持有旧密钥的用户重新建立信任。

最后，如果在对话过程中，UA 在 CMS SignedData 消息中接收到与先前在对话期间交换的证书不对应的证书，则 UA 必须通知其用户该更改，最好以表明这是 潜在的安全漏洞。

23.3 保护 MIME 主体

SIP 感兴趣的安全 MIME 主体有两种类型：这些主体的使用应遵循 S/MIME 规范 [24] 并有一些变化。

o “multipart/signed”必须仅与 CMS 分离签名一起使用。

这允许与非 S/MIME 兼容的收件人向后兼容。

o S/MIME 正文应该有一个 Content-Disposition 标头字段，并且“处理”参数的值应该是“必需的”。

o 如果 UAC 在其密钥环上没有与它想要向其发送请求的记录地址关联的证书，则它不能发送加密的“application/pkcs7-mime”MIME 消息。 UAC 可以发送初始请求，例如带有 CMS 分离签名的 OPTIONS 消息，以请求远程端的证书（签名应该在第 23.4 节中描述的类型的“消息/sip”主体上）。

请注意，未来 S/MIME 的标准化工作可能会定义非基于证书的密钥。

o S/MIME 主体的发送者应该使用“SMIMECapabilities”（参见 [24] 的第 2.5.2 节）属性来

表达他们的能力和进一步通信的偏好。 请特别注意，发送者可以使用“preferSignedData”功能来鼓励接收者使用 CMS SignedData 消息进行响应（例如，当发送如上所述的 OPTIONS 请求时）。

o S/MIME 实现必须至少支持 SHA1 作为数字签名算法，以及 3DES 作为加密算法。 可能支持所有其他签名和加密算法。 实现可以通过“SMIMECapabilities”属性协商对这些算法的支持。

o SIP 消息中的每个 S/MIME 正文应该只使用一个证书进行签名。 如果 UA 接收到具有多个签名的消息，则最外层的签名应被视为此正文的单个证书。 不应使用并行签名。

以下是 SIP 消息中加密的 S/MIME SDP 正文的示例：

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
     name=smime.p7m
Content-Disposition: attachment; filename=smime.p7m
   handling=required

**********************************************************
* Content-Type: application/sdp                          *
*                                                        *
* v=0                                                    *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com    *
* s=-                                                    *
* t=0 0                                                  *
* c=IN IP4 pc33.atlanta.com                              *
* m=audio 3456 RTP/AVP 0 1 3 99                          *
* a=rtpmap:0 PCMU/8000                                   *
**********************************************************
```

23.4 使用 S/MIME 的 SIP 标头隐私和完整性：隧道 SIP

作为为 SIP 标头字段提供某种程度的端到端身份验证、完整性或机密性的一种方式，S/MIME 可以将整个 SIP 消息封装在“message/sip”类型的 MIME 主体中，然后在这些主体中应用 MIME 安全性。 与典型的 SIP 机构相同。 这些封装的 SIP 请求和响应不构成单独的对话或事务，它们是用于验证完整性或提供附加信息的“外部”消息的副本。

如果 UAS 接收到包含隧道“消息/sip”S/MIME 主体的请求，它应该在响应中包含具有相同 smime 类型的隧道“消息/sip”主体。

任何传统的 MIME 正文（例如 SDP）都应该附加到“内部”消息，以便它们也可以从 S/MIME 安全性中受益。 请注意，如果还应在请求中传输任何不安全的 MIME 类型，则“消息/sip”主体可以作为 MIME“多部分/混合”主体的一部分发送。

23.4.1 SIP 头的完整性和保密性

当使用 S/MIME 完整性或机密性机制时，“内部”消息中的值与“外部”消息中的值之间可能存在差异。 本节中给出了处理本文档中描述的所有标题字段的任何此类差异的规则。

请注意，出于宽松时间戳的目的，所有通过“message/sip”隧道传输的 SIP 消息都应该在“内部”和“外部”报头中包含一个 Date 报头。

23.4.1.1 完整性

每当执行完整性检查时，应使用 20 中描述的 SIP 的比较规则，通过将签名正文中的头字段的值与“外部”消息中的值匹配来确定头字段的完整性。

可以被代理服务器合法修改的头部字段有：Request-URI、Via、Record-Route、Route、Max-Forwards 和 Proxy-Authorization。 如果这些头字段不是完整的端到端，实现不应该认为这是违反安全性的。 对本文档中定义的任何其他头字段的更改构成完整性违规； 必须通知用户存在差异。

23.4.1.2 保密

当消息被加密时，头部字段可能包含在“外部”消息中不存在的加密正文中。

某些标头字段必须始终具有纯文本版本，因为它们是请求和响应中必需的标头字段 - 这些包括：

收件人、发件人、呼叫 ID、CSeq、联系人。 虽然为 Call-ID、CSeq 或 Contact 提供加密替代方案可能没有用，但允许为“外部”To 或 From 中的信息提供替代方案。 请注意，加密正文中的值不用于识别事务或对话的目的——它们只是提供信息。 如果加密正文中的 From 标头字段与“外部”消息中的值不同，则加密正文中的值应该显示给用户，但不得在任何未来消息的“外部”标头字段中使用。

首先，用户代理将希望加密具有端到端语义的标头字段，包括：主题、回复、组织、接受、接受编码、接受语言、警报信息、错误信息、身份验证 -Info、Expires、In-Reply-To、Require、Supported、Unsupported、Retry-After、User-Agent、Server 和 Warning。 如果这些头字段中的任何一个出现在加密正文中，则应使用它们而不是任何“外部”头字段，无论这需要向用户显示头字段值还是在 UA 中设置内部状态。 然而，它们不应该用在任何未来消息的“外部”标题中。

如果存在，“内部”和“外部”标题中的日期标题字段必须始终相同。

由于 MIME 主体附加到“内部”消息，因此实现通常会加密 MIME 特定的头字段，包括：MIME-Version、Content-Type、Content-Length、Content-Language、Content-Encoding 和

Content-Disposition。 “外部”消息将具有 S/MIME 正文的正确 MIME 标头字段。 这些标头字段（以及它们前面的任何 MIME 主体）应被视为在 SIP 消息中接收的普通 MIME 标头字段和主体。

加密以下标头字段并不是特别有用：Min-Expires、Timestamp、Authorization、Priority 和 WWW-Authenticate。 此类别还包括那些可由代理服务器更改的标头字段（在上一节中描述）。 如果它们未包含在“外部”消息中，则 UA 不应将它们包含在“内部”消息中。 在加密正文中接收任何这些头字段的 UA 应该忽略加密值。

请注意，对 SIP 的扩展可能会定义额外的标头字段； 这些扩展的作者应该描述这些头域的完整性和机密性属性。 如果 SIP UA 遇到具有完整性违规的未知头字段，它必须忽略该头字段。

23.4.2 隧道完整性和认证

如果发送方希望保护的标头字段被复制到使用 CMS 分离签名签名的“消息/sip”MIME 主体中，则 S/MIME 主体中的隧道 SIP 消息可以提供 SIP 标头字段的完整性。

如果“消息/sip”正文至少包含基本的对话标识符（To、From、Call-ID、CSeq），那么签名的 MIME 正文可以提供有限的身份验证。 至少，如果用于签署正文的证书对于接收者来说是未知的并且无法验证，则该签名可用于确定对话中稍后的请求是由发起对话的同一证书持有者传输的。 如果签名的 MIME 正文的接收者有更强的动机来信任证书（他们能够验证它，他们从受信任的存储库中获取它，或者他们经常使用它），那么签名可以被视为一个更强的断言 证书主体的身份。

为了消除对整个头部字段的加减可能造成的混淆，发送者应该从签名正文中复制请求中的所有头部字段。 任何需要完整性保护的消息体必须附加到“内部”消息。

如果 Date 标头出现在带有签名正文的消息中，则接收者应该将标头字段值与其自己的内部时钟（如果适用）进行比较。 如果检测到显着的时间差异（大约一个小时或更长时间），用户代理应该提醒用户注意异常，并注意这是潜在的安全漏洞。

如果消息的接收者检测到消息中的完整性违规，如果消息是请求，则可以使用 403（禁止）响应拒绝该消息，或者可以终止任何现有的对话。 UA 应该将这种情况通知用户，并要求明确指导如何进行。

以下是使用隧道“消息/sip”主体的示例：

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42
Content-Length: 568

--boundary42
Content-Type: message/sip

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <bob@biloxi.com>
From: Alice <alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 here.com
s=Session SDP
c=IN IP4 pc33.atlanta.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
   handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42-
```

23.4.3 隧道加密

也可能希望使用这种机制来加密 CMS EnvelopedData 消息 S/MIME 正文中的“message/sip”MIME 正文，但实际上，大多数标头字段至少对网络有一些用途； 使用 S/MIME 进行加密的一般用途是保护 SDP 等消息体，而不是消息头。 一些信息头字段，例如主题或组织，可能会保证端到端的安全性。 未来 SIP 应用程序定义的标头也可能需要混淆。

加密报头字段的另一个可能应用是选择性匿名。 可以使用不包含个人信息的 From 标头字段构建请求（例如，sip:anonymous@anonymizer.invalid）。 但是，第二个 From 头字段包含原始记录的真实地址，可以在“message/sip”MIME 正文中加密，其中它仅对对话的端点可见。

请注意，如果此机制用于匿名，则消息的接收者将不再将 From 标头字段用作其证书钥匙串的索引，以检索与发送者关联的正确 S/MIME 密钥。 消息必须首先被解密，并且“内部”From 头字段必须用作索引。

为了提供端到端的完整性，加密的“message/sip”MIME 主体应该由发送者签名。 这将创建一个“multipart/signed”MIME 主体，其中包含一个加密主体和一个签名，两者都是“application/pkcs7-mime”类型。

在以下示例中，对于加密和签名的消息，星号 ("*") 中的文本已加密：

```
  INVITE sip:bob@biloxi.com SIP/2.0
  Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
  To: Bob <sip:bob@biloxi.com>
  From: Anonymous <sip:anonymous@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 314159 INVITE
  Max-Forwards: 70
  Date: Thu, 21 Feb 2002 13:02:03 GMT
  Contact: <sip:pc33.atlanta.com>
  Content-Type: multipart/signed;
    protocol="application/pkcs7-signature";
    micalg=sha1; boundary=boundary42
  Content-Length: 568

  --boundary42
  Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
       name=smime.p7m
  Content-Transfer-Encoding: base64
  Content-Disposition: attachment; filename=smime.p7m
     handling=required
  Content-Length: 231

**************************************************************
* Content-Type: message/sip                                  *
*                                                            *
* INVITE sip:bob@biloxi.com SIP/2.0                          *
* Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8    *
* To: Bob <bob@biloxi.com>                                   *
* From: Alice <alice@atlanta.com>;tag=1928301774             *
* Call-ID: a84b4c76e66710                                    *
* CSeq: 314159 INVITE                                        *
* Max-Forwards: 70                                           *
* Date: Thu, 21 Feb 2002 13:02:03 GMT                        *
* Contact: <sip:alice@pc33.atlanta.com>                      *
*                                                            *
* Content-Type: application/sdp                              *
*                                                            *
* v=0                                                        *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com        *
* s=Session SDP                                              *
* t=0 0                                                      *
* c=IN IP4 pc33.atlanta.com                                  *
* m=audio 3456 RTP/AVP 0 1 3 99                              *
* a=rtpmap:0 PCMU/8000                                       *
**************************************************************
--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
   handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42-
```

24 例子

在以下示例中，为简洁起见，我们经常省略消息正文以及相应的 Content-Length 和 Content-Type 标头字段。

24.1 注册

Bob 在启动时注册。 消息流如图 9 所示。请注意，为了简单起见，通常没有显示注册所需的身份验证。

Figure 9: SIP Registration Example

```
F1 REGISTER Bob -> Registrar

    REGISTER sip:registrar.biloxi.com SIP/2.0
    Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
    Max-Forwards: 70
    To: Bob <sip:bob@biloxi.com>
    From: Bob <sip:bob@biloxi.com>;tag=456248
    Call-ID: 843817637684230@998sdasdh09
    CSeq: 1826 REGISTER
    Contact: <sip:bob@192.0.2.4>
    Expires: 7200
    Content-Length: 0
```

注册将在两个小时后到期。 注册商以 200 OK 响应：

```
F2 200 OK Registrar -> Bob

     SIP/2.0 200 OK
     Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
      ;received=192.0.2.4
     To: Bob <sip:bob@biloxi.com>;tag=2493k59kd
     From: Bob <sip:bob@biloxi.com>;tag=456248
     Call-ID: 843817637684230@998sdasdh09
     CSeq: 1826 REGISTER
     Contact: <sip:bob@192.0.2.4>
     Expires: 7200
     Content-Length: 0
```

24.2 会话设置

此示例包含第 4 节中示例会话设置的完整详细信息。消息流如图 1 所示。请注意，这些流显示了所需的最小标头字段集 - 通常会存在一些其他标头字段，例如 Allow 和 Supported 。

F1 INVITE Alice -> atlanta.com proxy

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

F2 100 Trying atlanta.com proxy -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Content-Length: 0
```

F3 INVITE atlanta.com proxy -> biloxi.com proxy

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 69
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

F4 100 Trying biloxi.com proxy -> atlanta.com proxy

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Content-Length: 0
```

F5 INVITE biloxi.com proxy -> Bob

```
INVITE sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c2ff8.1
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 68
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

F6 180 Ringing Bob -> biloxi.com proxy

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c2ff8.1
 ;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0
```

F7 180 Ringing biloxi.com proxy -> atlanta.com proxy

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0
```

F8 180 Ringing atlanta.com proxy -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0
```

F9 200 OK Bob -> biloxi.com proxy

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c2ff8.1
 ;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

F10 200 OK biloxi.com proxy -> atlanta.com proxy

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

```
F11 200 OK atlanta.com proxy -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)


F12 ACK Alice -> Bob

ACK sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds9
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 ACK
Content-Length: 0
```

Alice 和 Bob 之间的媒体会话现已建立。

鲍勃先挂断了电话。 请注意，Bob 的 SIP 电话维护自己的 Cseq 编号空间，在本示例中，该编号空间以 231 开头。由于 Bob 发出请求，因此 To 和 From URI 和标签已交换。

```
F13 BYE Bob -> Alice

BYE sip:alice@pc33.atlanta.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
Max-Forwards: 70
From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
To: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 231 BYE
Content-Length: 0


F14 200 OK Alice -> Bob

SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
To: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 231 BYE
Content-Length: 0
```

SIP 呼叫流程文档 [40] 包含更多 SIP 消息示例。

25 SIP 协议的增强 BNF

本文档中指定的所有机制都在散文和 RFC 2234 [10] 中定义的增强巴库斯-瑙尔形式 (BNF) 中进行了描述。 RFC 2234 的第 6.1 节定义了一组本规范使用的核心规则，此处不再赘述。 实施者需要熟悉 RFC 2234 的符号和内容才能理解该规范。 某些基本规则是大写的，例如 SP、LWS、HTAB、CRLF、DIGIT、ALPHA 等。定义中使用尖括号来阐明规则名称的使用。

方括号的使用在语法上是多余的。 它用作特定参数可选使用的语义提示。

25.1 基本规则

在本规范中使用以下规则来描述基本的解析结构。 US-ASCII 编码字符集由 ANSI X3.4-1986 定义。

```
alphanum  =  ALPHA / DIGIT
```

一些规则从 RFC 2396 [5] 中合并而来，但经过更新以使其符合 RFC 2234 [10]。 这些包括：

```
reserved    =  ";" / "/" / "?" / ":" / "@" / "&" / "=" / "+"
               / "$" / ","
unreserved  =  alphanum / mark
mark        =  "-" / "_" / "." / "!" / "~" / "*" / "'"
               / "(" / ")"
escaped     =  "%" HEXDIG HEXDIG
```

如果续行以空格或水平制表符开头，则 SIP 标头字段值可以折叠到多行。 所有线性空白，包括折叠，都具有与 SP 相同的语义。 在解释字段值或向下游转发消息之前，接收者可以用单个 SP 替换任何线性空白。 这旨在与 RFC 2616 [8] 中描述的 HTTP/1.1 完全相同。 当线性空白是可选的时，通常在标记和分隔符之间使用 SWS 构造。

```
LWS  =  [*WSP CRLF] 1*WSP ; linear whitespace
SWS  =  [LWS] ; sep whitespace
```

为了将标头名称与值的其余部分分开，使用冒号，根据上述规则，冒号允许前面有空格，但不能换行，后面有空格，包括换行符。 HCOLON 定义了这个结构。

```
HCOLON  =  *( SP / HTAB ) ":" SWS
```

TEXT-UTF8 规则仅用于不打算由消息解析器解释的描述性字段内容和值。 *TEXT-UTF8 的单词包含来自 UTF-8 字符集 (RFC 2279 [7]) 的字符。 TEXT-UTF8-TRIM 规则用于描述性字段内容，这些内容是 n 个带引号的字符串，其中前导和尾随 LWS 没有意义。 在这方面，SIP 与使用 ISO 8859-1 字符集的 HTTP 不同。

```
TEXT-UTF8-TRIM  =  1*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
```

TEXT-UTF8-TRIM 的定义中允许 CRLF 仅作为头字段延续的一部分。 在解释 TEXT-UTF8-TRIM 值之前，预计折叠 LWS 将替换为单个 SP。

十六进制数字字符用于多个协议元素。 某些元素（身份验证）强制十六进制字母为小写。

```
LHEX  =  DIGIT / %x61-66 ;lowercase a-f
```

许多 SIP 标头字段值由由 LWS 或特殊字符分隔的单词组成。 除非另有说明，否则标记不区分大小写。 这些特殊字符必须在带引号的字符串中才能在参数值中使用。 Call-ID 中使用了单词构造，以允许使用大多数分隔符。

```
token        =  1*(alphanum / "-" / "." / "!" / "%" / "*"
                / "_" / "+" / "`" / "'" / "~" )
separators   =  "(" / ")" / "<" / ">" / "@" /
                "," / ";" / ":" / "\" / DQUOTE /
                "/" / "[" / "]" / "?" / "=" /
                "{" / "}" / SP / HTAB
word         =  1*(alphanum / "-" / "." / "!" / "%" / "*" /
                "_" / "+" / "`" / "'" / "~" /
                "(" / ")" / "<" / ">" /
                ":" / "\" / DQUOTE /
                "/" / "[" / "]" / "?" /
                "{" / "}" )
```

当使用标记或元素之间使用分隔符时，通常允许在这些字符之前或之后使用空格：

```
STAR    =  SWS "*" SWS ; asterisk
SLASH   =  SWS "/" SWS ; slash
EQUAL   =  SWS "=" SWS ; equal
LPAREN  =  SWS "(" SWS ; left parenthesis
RPAREN  =  SWS ")" SWS ; right parenthesis
RAQUOT  =  ">" SWS ; right angle quote
LAQUOT  =  SWS "<"; left angle quote
COMMA   =  SWS "," SWS ; comma
SEMI    =  SWS ";" SWS ; semicolon
COLON   =  SWS ":" SWS ; colon
LDQUOT  =  SWS DQUOTE; open double quotation mark
RDQUOT  =  DQUOTE SWS ; close double quotation mark
```

通过用括号将注释文本括起来，可以将注释包含在某些 SIP 标头字段中。 仅允许在包含“comment”作为其字段值定义的一部分的字段中使用注释。 在所有其他字段中，括号被视为字段值的一部分。

```
comment  =  LPAREN *(ctext / quoted-pair / comment) RPAREN
ctext    =  %x21-27 / %x2A-5B / %x5D-7E / UTF8-NONASCII
            / LWS
```

ctext 包括除左右括号和反斜杠之外的所有字符。 如果使用双引号括起来，则将文本字符串解析为单个单词。 在带引号的字符串中，引号 (") 和反斜杠 (\) 需要进行转义。

```
quoted-string  =  SWS DQUOTE *(qdtext / quoted-pair ) DQUOTE
qdtext         =  LWS / %x21 / %x23-5B / %x5D-7E
                  / UTF8-NONASCII
```

反斜杠字符 ("\") 只能在带引号的字符串和注释结构中用作单字符引用机制。 与 HTTP/1.1 不同，字符 CR 和 LF 不能通过这种机制进行转义，以避免与行折叠和标题分隔冲突。

```
quoted-pair  =  "\" (%x00-09 / %x0B-0C
                / %x0E-7F)

SIP-URI          =  "sip:" [ userinfo ] hostport
                    uri-parameters [ headers ]
SIPS-URI         =  "sips:" [ userinfo ] hostport
                    uri-parameters [ headers ]
userinfo         =  ( user / telephone-subscriber ) [ ":" password ] "@"
user             =  1*( unreserved / escaped / user-unreserved )
user-unreserved  =  "&" / "=" / "+" / "$" / "," / ";" / "?" / "/"
password         =  *( unreserved / escaped /
                    "&" / "=" / "+" / "$" / "," )
hostport         =  host [ ":" port ]
host             =  hostname / IPv4address / IPv6reference
hostname         =  *( domainlabel "." ) toplabel [ "." ]
domainlabel      =  alphanum
                    / alphanum *( alphanum / "-" ) alphanum
toplabel         =  ALPHA / ALPHA *( alphanum / "-" ) alphanum

IPv4address    =  1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
IPv6reference  =  "[" IPv6address "]"
IPv6address    =  hexpart [ ":" IPv4address ]
hexpart        =  hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]
hexseq         =  hex4 *( ":" hex4)
hex4           =  1*4HEXDIG
port           =  1*DIGIT
```

电话用户的 BNF 可以在 RFC 2806 [9] 中找到。 但是请注意，任何允许在 SIP URI 的用户部分中不允许的字符都必须转义。

```
uri-parameters    =  *( ";" uri-parameter)
uri-parameter     =  transport-param / user-param / method-param
                     / ttl-param / maddr-param / lr-param / other-param
transport-param   =  "transport="
                     ( "udp" / "tcp" / "sctp" / "tls"
                     / other-transport)
other-transport   =  token
user-param        =  "user=" ( "phone" / "ip" / other-user)
other-user        =  token
method-param      =  "method=" Method
ttl-param         =  "ttl=" ttl
maddr-param       =  "maddr=" host
lr-param          =  "lr"
other-param       =  pname [ "=" pvalue ]
pname             =  1*paramchar
pvalue            =  1*paramchar
paramchar         =  param-unreserved / unreserved / escaped
param-unreserved  =  "[" / "]" / "/" / ":" / "&" / "+" / "$"

headers          =  "?" header *( "&" header )
header           =  hname "=" hvalue
hname            =  1*( hnv-unreserved / unreserved / escaped )
hvalue           =  *( hnv-unreserved / unreserved / escaped )
hnv-unreserved   =  "[" / "]" / "/" / "?" / ":" / "+" / "$"

SIP-message    =  Request / Response
Request        =  Request-Line
                  *( message-header )
                  CRLF
                  [ message-body ]
Request-Line   =  Method SP Request-URI SP SIP-Version CRLF
Request-URI    =  SIP-URI / SIPS-URI / absoluteURI
absoluteURI    =  scheme ":" ( hier-part / opaque-part )
hier-part      =  ( net-path / abs-path ) [ "?" query ]
net-path       =  "//" authority [ abs-path ]
abs-path       =  "/" path-segments
```

```
opaque-part     =  uric-no-slash *uric
uric            =  reserved / unreserved / escaped
uric-no-slash   =  unreserved / escaped / ";" / "?" / ":" / "@"
                   / "&" / "=" / "+" / "$" / ","
path-segments   =  segment *( "/" segment )
segment         =  *pchar *( ";" param )
param           =  *pchar
pchar           =  unreserved / escaped /
                   ":" / "@" / "&" / "=" / "+" / "$" / ","
scheme          =  ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
authority       =  srvr / reg-name
srvr            =  [ [ userinfo "@" ] hostport ]
reg-name        =  1*( unreserved / escaped / "$" / ","
                   / ";" / ":" / "@" / "&" / "=" / "+" )
query           =  *uric
SIP-Version     =  "SIP" "/" 1*DIGIT "." 1*DIGIT

message-header  =  (Accept
                /  Accept-Encoding
                /  Accept-Language
                /  Alert-Info
                /  Allow
                /  Authentication-Info
                /  Authorization
                /  Call-ID
                /  Call-Info
                /  Contact
                /  Content-Disposition
                /  Content-Encoding
                /  Content-Language
                /  Content-Length
                /  Content-Type
                /  CSeq
                /  Date
                /  Error-Info
                /  Expires
                /  From
                /  In-Reply-To
                /  Max-Forwards
                /  MIME-Version
                /  Min-Expires
                /  Organization
                /  Priority
                /  Proxy-Authenticate
                /  Proxy-Authorization
                /  Proxy-Require
                /  Record-Route
                /  Reply-To
```

```
                 /  Require
                 /  Retry-After
                 /  Route
                 /  Server
                 /  Subject
                 /  Supported
                 /  Timestamp
                 /  To
                 /  Unsupported
                 /  User-Agent
                 /  Via
                 /  Warning
                 /  WWW-Authenticate
                 /  extension-header) CRLF

INVITEm           =  %x49.4E.56.49.54.45 ; INVITE in caps
ACKm              =  %x41.43.4B ; ACK in caps
OPTIONSm          =  %x4F.50.54.49.4F.4E.53 ; OPTIONS in caps
BYEm              =  %x42.59.45 ; BYE in caps
CANCELm           =  %x43.41.4E.43.45.4C ; CANCEL in caps
REGISTERm         =  %x52.45.47.49.53.54.45.52 ; REGISTER in caps
Method            =  INVITEm / ACKm / OPTIONSm / BYEm
                     / CANCELm / REGISTERm
                     / extension-method
extension-method  =  token
Response          =  Status-Line
                     *( message-header )
                     CRLF
                     [ message-body ]

Status-Line     =  SIP-Version SP Status-Code SP Reason-Phrase CRLF
Status-Code     =  Informational
                /  Redirection
                /  Success
                /  Client-Error
                /  Server-Error
                /  Global-Failure
                /  extension-code
extension-code  =  3DIGIT
Reason-Phrase   =  *(reserved / unreserved / escaped
                   / UTF8-NONASCII / UTF8-CONT / SP / HTAB)

Informational  =  "100"  ;  Trying
               /  "180"  ;  Ringing
               /  "181"  ;  Call Is Being Forwarded
               /  "182"  ;  Queued
               /  "183"  ;  Session Progress
```

```
Success    =  "200"   ;  OK

Redirection  =  "300"   ;  Multiple Choices
             /   "301"   ;  Moved Permanently
             /   "302"   ;  Moved Temporarily
             /   "305"   ;  Use Proxy
             /   "380"   ;  Alternative Service

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
              /   "410"   ;  Gone
              /   "413"   ;  Request Entity Too Large
              /   "414"   ;  Request-URI Too Large
              /   "415"   ;  Unsupported Media Type
              /   "416"   ;  Unsupported URI Scheme
              /   "420"   ;  Bad Extension
              /   "421"   ;  Extension Required
              /   "423"   ;  Interval Too Brief
              /   "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
              /   "487"   ;  Request Terminated
              /   "488"   ;  Not Acceptable Here
              /   "491"   ;  Request Pending
              /   "493"   ;  Undecipherable

Server-Error  =  "500"   ;  Internal Server Error
              /   "501"   ;  Not Implemented
              /   "502"   ;  Bad Gateway
              /   "503"   ;  Service Unavailable
              /   "504"   ;  Server Time-out
              /   "505"   ;  SIP Version not supported
              /   "513"   ;  Message Too Large
```

```
Global-Failure  =  "600"  ;  Busy Everywhere
                /  "603"  ;  Decline
                /  "604"  ;  Does not exist anywhere
                /  "606"  ;  Not Acceptable

Accept          =  "Accept" HCOLON
                    [ accept-range *(COMMA accept-range) ]
accept-range    =  media-range *(SEMI accept-param)
media-range     =  ( "*/*"
                   / ( m-type SLASH "*" )
                   / ( m-type SLASH m-subtype )
                   ) *( SEMI m-parameter )
accept-param    =  ("q" EQUAL qvalue) / generic-param
qvalue          =  ( "0" [ "." 0*3DIGIT ] )
                   / ( "1" [ "." 0*3("0") ] )
generic-param   =  token [ EQUAL gen-value ]
gen-value       =  token / host / quoted-string

Accept-Encoding  =  "Accept-Encoding" HCOLON
                     [ encoding *(COMMA encoding) ]
encoding         =  codings *(SEMI accept-param)
codings          =  content-coding / "*"
content-coding   =  token

Accept-Language  =  "Accept-Language" HCOLON
                     [ language *(COMMA language) ]
language         =  language-range *(SEMI accept-param)
language-range   =  ( ( 1*8ALPHA *( "-" 1*8ALPHA ) ) / "*" )

Alert-Info   =  "Alert-Info" HCOLON alert-param *(COMMA alert-param)
alert-param  =  LAQUOT absoluteURI RAQUOT *( SEMI generic-param )

Allow  =  "Allow" HCOLON [Method *(COMMA Method)]

Authorization     =  "Authorization" HCOLON credentials
credentials       =  ("Digest" LWS digest-response)
                     / other-response
digest-response   =  dig-resp *(COMMA dig-resp)
dig-resp          =  username / realm / nonce / digest-uri
                      / dresponse / algorithm / cnonce
                      / opaque / message-qop
                      / nonce-count / auth-param
username          =  "username" EQUAL username-value
username-value    =  quoted-string
digest-uri        =  "uri" EQUAL LDQUOT digest-uri-value RDQUOT
digest-uri-value  =  rquest-uri ; Equal to request-uri as specified
                     by HTTP/1.1
message-qop       =  "qop" EQUAL qop-value
```

```
cnonce              =  "cnonce" EQUAL cnonce-value
cnonce-value        =  nonce-value
nonce-count         =  "nc" EQUAL nc-value
nc-value            =  8LHEX
dresponse           =  "response" EQUAL request-digest
request-digest      =  LDQUOT 32LHEX RDQUOT
auth-param          =  auth-param-name EQUAL
                       ( token / quoted-string )
auth-param-name     =  token
other-response      =  auth-scheme LWS auth-param
                       *(COMMA auth-param)
auth-scheme         =  token

Authentication-Info  =  "Authentication-Info" HCOLON ainfo
                        *(COMMA ainfo)
ainfo                =  nextnonce / message-qop
                         / response-auth / cnonce
                         / nonce-count
nextnonce            =  "nextnonce" EQUAL nonce-value
response-auth        =  "rspauth" EQUAL response-digest
response-digest      =  LDQUOT *LHEX RDQUOT

Call-ID  =  ( "Call-ID" / "i" ) HCOLON callid
callid   =  word [ "@" word ]

Call-Info   =  "Call-Info" HCOLON info *(COMMA info)
info        =  LAQUOT absoluteURI RAQUOT *( SEMI info-param)
info-param  =  ( "purpose" EQUAL ( "icon" / "info"
               / "card" / token ) ) / generic-param

Contact        =  ("Contact" / "m" ) HCOLON
                  ( STAR / (contact-param *(COMMA contact-param)))
contact-param  =  (name-addr / addr-spec) *(SEMI contact-params)
name-addr      =  [ display-name ] LAQUOT addr-spec RAQUOT
addr-spec      =  SIP-URI / SIPS-URI / absoluteURI
display-name   =  *(token LWS)/ quoted-string

contact-params     =  c-p-q / c-p-expires
                      / contact-extension
c-p-q              =  "q" EQUAL qvalue
c-p-expires        =  "expires" EQUAL delta-seconds
contact-extension  =  generic-param
delta-seconds      =  1*DIGIT

Content-Disposition   =  "Content-Disposition" HCOLON
                         disp-type *( SEMI disp-param )
disp-type             =  "render" / "session" / "icon" / "alert"
                         / disp-extension-token
```

```
disp-param            =  handling-param / generic-param
handling-param        =  "handling" EQUAL
                         ( "optional" / "required"
                         / other-handling )
other-handling        =  token
disp-extension-token  =  token

Content-Encoding  =  ( "Content-Encoding" / "e" ) HCOLON
                     content-coding *(COMMA content-coding)

Content-Language  =  "Content-Language" HCOLON
                     language-tag *(COMMA language-tag)
language-tag      =  primary-tag *( "-" subtag )
primary-tag       =  1*8ALPHA
subtag            =  1*8ALPHA

Content-Length  =  ( "Content-Length" / "l" ) HCOLON 1*DIGIT
Content-Type    =  ( "Content-Type" / "c" ) HCOLON media-type
media-type      =  m-type SLASH m-subtype *(SEMI m-parameter)
m-type          =  discrete-type / composite-type
discrete-type   =  "text" / "image" / "audio" / "video"
                   / "application" / extension-token
composite-type  =  "message" / "multipart" / extension-token
extension-token =  ietf-token / x-token
ietf-token      =  token
x-token         =  "x-" token
m-subtype       =  extension-token / iana-token
iana-token      =  token
m-parameter     =  m-attribute EQUAL m-value
m-attribute     =  token
m-value         =  token / quoted-string

CSeq  =  "CSeq" HCOLON 1*DIGIT LWS Method

Date          =  "Date" HCOLON SIP-date
SIP-date      =  rfc1123-date
rfc1123-date  =  wkday "," SP date1 SP time SP "GMT"
date1         =  2DIGIT SP month SP 4DIGIT
                 ; day month year (e.g., 02 Jun 1982)
time          =  2DIGIT ":" 2DIGIT ":" 2DIGIT
                 ; 00:00:00 - 23:59:59
wkday         =  "Mon" / "Tue" / "Wed"
                 / "Thu" / "Fri" / "Sat" / "Sun"
month         =  "Jan" / "Feb" / "Mar" / "Apr"
                 / "May" / "Jun" / "Jul" / "Aug"
                 / "Sep" / "Oct" / "Nov" / "Dec"

Error-Info  =  "Error-Info" HCOLON error-uri *(COMMA error-uri)
```

```
error-uri    =  LAQUOT absoluteURI RAQUOT *( SEMI generic-param )

Expires      =  "Expires" HCOLON delta-seconds
From         =  ( "From" / "f" ) HCOLON from-spec
from-spec    =  ( name-addr / addr-spec )
                *( SEMI from-param )
from-param   =  tag-param / generic-param
tag-param    =  "tag" EQUAL token

In-Reply-To  =  "In-Reply-To" HCOLON callid *(COMMA callid)

Max-Forwards  =  "Max-Forwards" HCOLON 1*DIGIT

MIME-Version  =  "MIME-Version" HCOLON 1*DIGIT "." 1*DIGIT

Min-Expires  =  "Min-Expires" HCOLON delta-seconds

Organization  =  "Organization" HCOLON [TEXT-UTF8-TRIM]

Priority         =  "Priority" HCOLON priority-value
priority-value   =  "emergency" / "urgent" / "normal"
                    / "non-urgent" / other-priority
other-priority   =  token

Proxy-Authenticate  =  "Proxy-Authenticate" HCOLON challenge
challenge           =  ("Digest" LWS digest-cln *(COMMA digest-cln))
                       / other-challenge
other-challenge     =  auth-scheme LWS auth-param
                       *(COMMA auth-param)
digest-cln          =  realm / domain / nonce
                        / opaque / stale / algorithm
                        / qop-options / auth-param
realm               =  "realm" EQUAL realm-value
realm-value         =  quoted-string
domain              =  "domain" EQUAL LDQUOT URI
                       *( 1*SP URI ) RDQUOT
URI                 =  absoluteURI / abs-path
nonce               =  "nonce" EQUAL nonce-value
nonce-value         =  quoted-string
opaque              =  "opaque" EQUAL quoted-string
stale               =  "stale" EQUAL ( "true" / "false" )
algorithm           =  "algorithm" EQUAL ( "MD5" / "MD5-sess"
                       / token )
qop-options         =  "qop" EQUAL LDQUOT qop-value
                       *("," qop-value) RDQUOT
qop-value           =  "auth" / "auth-int" / token

Proxy-Authorization  =  "Proxy-Authorization" HCOLON credentials
```

```
Proxy-Require  =  "Proxy-Require" HCOLON option-tag
                  *(COMMA option-tag)
option-tag     =  token

Record-Route  =  "Record-Route" HCOLON rec-route *(COMMA rec-route)
rec-route     =  name-addr *( SEMI rr-param )
rr-param      =  generic-param

Reply-To      =  "Reply-To" HCOLON rplyto-spec
rplyto-spec   =  ( name-addr / addr-spec )
                 *( SEMI rplyto-param )
rplyto-param  =  generic-param
Require       =  "Require" HCOLON option-tag *(COMMA option-tag)

Retry-After  =  "Retry-After" HCOLON delta-seconds
                [ comment ] *( SEMI retry-param )

retry-param  =  ("duration" EQUAL delta-seconds)
                / generic-param

Route        =  "Route" HCOLON route-param *(COMMA route-param)
route-param  =  name-addr *( SEMI rr-param )

Server           =  "Server" HCOLON server-val *(LWS server-val)
server-val       =  product / comment
product          =  token [SLASH product-version]
product-version  =  token

Subject  =  ( "Subject" / "s" ) HCOLON [TEXT-UTF8-TRIM]

Supported  =  ( "Supported" / "k" ) HCOLON
              [option-tag *(COMMA option-tag)]

Timestamp  =  "Timestamp" HCOLON 1*(DIGIT)
              [ "." *(DIGIT) ] [ LWS delay ]
delay      =  *(DIGIT) [ "." *(DIGIT) ]

To        =  ( "To" / "t" ) HCOLON ( name-addr
             / addr-spec ) *( SEMI to-param )
to-param  =  tag-param / generic-param

Unsupported  =  "Unsupported" HCOLON option-tag *(COMMA option-tag)
User-Agent   =  "User-Agent" HCOLON server-val *(LWS server-val)
```

```
Via               =  ( "Via" / "v" ) HCOLON via-parm *(COMMA via-parm)
via-parm          =  sent-protocol LWS sent-by *( SEMI via-params )
via-params        =  via-ttl / via-maddr
                     / via-received / via-branch
                     / via-extension
via-ttl           =  "ttl" EQUAL ttl
via-maddr         =  "maddr" EQUAL host
via-received      =  "received" EQUAL (IPv4address / IPv6address)
via-branch        =  "branch" EQUAL token
via-extension     =  generic-param
sent-protocol     =  protocol-name SLASH protocol-version
                     SLASH transport
protocol-name     =  "SIP" / token
protocol-version  =  token
transport         =  "UDP" / "TCP" / "TLS" / "SCTP"
                     / other-transport
sent-by           =  host [ COLON port ]
ttl               =  1*3DIGIT ; 0 to 255

Warning        =  "Warning" HCOLON warning-value *(COMMA warning-value)
warning-value  =  warn-code SP warn-agent SP warn-text
warn-code      =  3DIGIT
warn-agent     =  hostport / pseudonym
                  ;  the name or pseudonym of the server adding
                  ;  the Warning header, for use in debugging
warn-text      =  quoted-string
pseudonym      =  token

WWW-Authenticate  =  "WWW-Authenticate" HCOLON challenge

extension-header  =  header-name HCOLON header-value
header-name       =  token
header-value      =  *(TEXT-UTF8char / UTF8-CONT / LWS)
message-body  =  *OCTET
```

26 安全注意事项：威胁模型和安全使用建议

SIP 不是一个容易保护的协议。 它对中介的使用、它的多方面信任关系、它在完全不信任的元素之间的预期使用以及它的用户对用户操作使得安全性绝非易事。 今天需要在各种环境和用途中无需广泛协调即可部署的安全解决方案。 为了满足这些不同的需求，将需要适用于 SIP 不同方面和用途的几种不同机制。

请注意，SIP 信令本身的安全性与与 SIP 一起使用的协议的安全性（例如 RTP）或 SIP 可能承载的任何特定主体的安全性无关（尽管 MIME 安全性在保护 SIP 方面起着重要作用）。与会话关联的任何媒体都可以独立于任何关联的 SIP 信令进行端到端加密。 媒体加密不在本文档的范围内。

下面的考虑首先检查了一组广泛识别 SIP 安全需求的经典威胁模型。 然后详细介绍了解决这些威胁所需的一组安全服务，然后解释了可用于提供这些服务的几种安全机制。 接下来，列举了对 SIP 实施者的要求，以及可以使用这些安全机制来提高 SIP 安全性的示例性部署。

一些关于隐私的说明结束了本节。

26.1 攻击和威胁模型

本节详细介绍了大多数 SIP 部署应该常见的一些威胁。特意选择了这些威胁来说明 SIP 所需的每项安全服务。

以下示例绝不提供针对 SIP 的威胁的详尽列表； 相反，这些是“经典”威胁，表明需要特定的安全服务，这些服务可以潜在地防止所有类别的威胁。

这些攻击假定攻击者可以潜在地读取网络上的任何数据包的环境 - 预计 SIP 将经常在公共 Internet 上使用。 网络上的攻击者可能能够修改数据包（可能在某些受感染的中间人处）。攻击者可能希望窃取服务、窃听通信或中断会话。

26.1.1 注册劫持

SIP 注册机制允许用户代理向注册器标识自己作为用户（由记录地址指定）所在的设备。 注册商评估在 REGISTER 消息的 From 头字段中声明的身份，以确定此请求是否可以修改与 To 头字段中的记录地址相关联的联系地址。 虽然这两个字段通常相同，但在许多有效的部署中，第三方可以代表用户注册联系人。

然而，SIP 请求的 From 标头字段可以由 UA 的所有者任意修改，这为恶意注册打开了大门。例如，成功冒充被授权更改与记录地址关联的联系人的攻击者可以取消注册所有现有联系人的 URI，然后将他们自己的设备注册为适当的联系人地址，从而将所有请求定向到 受影响的用户访问攻击者的设备。

这种威胁属于依赖于请求发起者缺乏加密保证的威胁家族。 任何代表有价值服务的 SIP UAS（例如，将 SIP 请求与传统电话呼叫互通的网关）都可能希望通过对其接收的请求进行身份验证来控制对其资源的访问。 即使是最终用户 UA，例如 SIP 电话，也有兴趣确定请求发起者的身份。

这种威胁表明需要使 SIP 实体能够对请求的发起者进行身份验证的安全服务。

26.1.2 模拟服务器

请求的目的地域通常在 Request-URI 中指定。 UA 通常直接联系该域中的服务器以传递请求。 但是，攻击者总是有可能冒充远程服务器，并且 UA 的请求可能被其他方拦截。

例如，考虑这样一种情况，一个域 chicago.com 的重定向服务器模拟另一个域 biloxi.com 的重定向服务器。 用户代理向 biloxi.com 发送请求，但位于 chicago.com 的重定向服务器以伪造的响应进行响应，该响应具有适当的 SIP 标头字段，用于来自 biloxi.com 的响应。 重定向响应中的伪造联系人地址可以将原始 UA 引导到不适当或不安全的资源，或者只是阻

止对 biloxi.com 的请求成功。

这一系列威胁拥有庞大的成员，其中许多是至关重要的。 与注册劫持威胁相反，考虑发送到 biloxi.com 的注册被 chicago.com 拦截的情况，该注册以伪造的 301（永久移动）响应回复拦截的注册。此响应似乎来自 biloxi.com，但指定了 chicago.com 作为适当的注册商。 然后，来自始发 UA 的所有未来 REGISTER 请求都将转到 chicago.com。

预防这种威胁需要一种方法，UA 可以通过这种方法对它们发送请求的服务器进行身份验证。

26.1.3 篡改消息体

当然，SIP UA 通过受信任的代理服务器路由请求。 不管这种信任是如何建立的（代理的身份验证在本节的其他地方讨论），UA 可以信任代理服务器来路由请求，但不检查或可能修改包含在该请求中的主体。

考虑一个使用 SIP 消息体来传递媒体会话的会话加密密钥的 UA。 尽管它信任它正在联系的域的代理服务器以正确传递信令，但它可能不希望该域的管理员能够解密任何后续媒体会话。 更糟糕的是，如果代理服务器是主动恶意的，它可能会修改会话密钥，或者充当中间人，或者可能更改原始 UA 请求的安全特性。

这一系列威胁不仅适用于会话密钥，而且适用于 SIP 中端到端承载的大多数可能的内容形式。 这些可能包括应该呈现给用户的 MIME 主体、SDP 或封装的电话信号等。 攻击者可能会尝试修改 SDP 主体，例如，为了将 RTP 媒体流指向窃听设备，以便窃听后续语音通信。

另请注意，SIP 中的某些标头字段是端到端有意义的，例如，主题。 UA 可能会保护这些标头字段以及正文（例如，更改主题标头字段的恶意中介可能会使重要请求看起来像是垃圾邮件）。 但是，由于代理服务器在路由请求时会合法地检查或更改许多标头字段，因此并非所有标头字段都应端到端保护。

由于这些原因，UA 可能希望保护 SIP 消息主体，在某些有限的情况下，端到端的头字段。机构所需的安全服务包括机密性、完整性和身份验证。 这些端到端服务应该独立于用于保护与代理服务器等中介交互的手段。

26.1.4 拆除会话

一旦通过初始消息传递建立了对话，就可以发送修改对话和/或会话状态的后续请求。 至关重要的是，会话中的主体可以确定此类请求不是由攻击者伪造的。

考虑这样一种情况，第三方攻击者在两方共享的对话中捕获一些初始消息，以了解会话的参数（To tag、From tag 等），然后将 BYE 请求插入到会话中 . 攻击者可以选择伪造请求，使其看起来来自任一参与者。 一旦目标接收到 BYE，会话将被提前终止。

类似的会话中威胁包括传输伪造的 re-INVITE 来改变会话（可能降低会话安全性或重定向媒体流作为窃听攻击的一部分）。

对这种威胁最有效的对策是对 BYE 的发送者进行身份验证。 在这种情况下，接收者只需要知道 BYE 来自与之建立相应对话的同一方（而不是确定发送者的绝对身份）。 此外，如果攻击者由于机密性而无法获知会话的参数，就不可能伪造 BYE。 但是，一些中介（如代理服务器）将需要在会话建立时检查这些参数。

26.1.5 拒绝服务和放大

拒绝服务攻击侧重于使特定网络元素不可用，通常通过在其接口处引导过多的网络流量。分布式拒绝服务攻击允许一个网络用户导致多个网络主机用大量网络流量淹没目标主机。

在许多架构中，SIP 代理服务器面向公共 Internet，以便接受来自全球 IP 端点的请求。 SIP 为分布式拒绝服务攻击创造了许多潜在机会，必须由 SIP 系统的实施者和运营商识别和解决。

攻击者可以创建虚假请求，其中包含伪造的源 IP 地址和相应的 Via 头字段，将目标主机标识为请求的发起者，然后将此请求发送到大量 SIP 网络元素，从而使用倒霉的 SIP UA 或代理 生成针对目标的拒绝服务流量。

类似地，攻击者可能会在请求中使用伪造的 Route 标头字段值来识别目标主机，然后将此类消息发送到分叉代理，这些代理将放大发送到目标的消息。

当攻击者确定由请求发起的 SIP 对话将导致大量反向事务时，可以使用 Record-Route 来达到类似的效果。

如果 REGISTER 请求没有得到注册商的正确身份验证和授权，就会出现许多拒绝服务攻击。攻击者可以注销管理域中的部分或全部用户，从而阻止这些用户被邀请到新会话。 攻击者还可以注册大量联系人，为给定的记录地址指定同一主机，以便在拒绝服务攻击中使用注册器和任何关联的代理服务器作为放大器。 攻击者还可能通过注册大量绑定来尝试耗尽注册器的可用内存和磁盘资源。

使用多播传输 SIP 请求会大大增加拒绝服务攻击的可能性。

这些问题表明一般需要定义将拒绝服务风险降至最低的架构，并且需要注意针对此类攻击的安全机制的建议。

26.2 安全机制

从上述威胁中，我们得出 SIP 协议所需的基本安全服务是：保持消息传递的机密性和完整性，防止重放攻击或消息欺骗，提供会话参与者的身份验证和隐私，以及防止 拒绝服务攻击。 SIP 消息中的主体分别需要机密性、完整性和身份验证等安全服务。

SIP 没有定义特定于 SIP 的新安全机制，而是尽可能重用从 HTTP 和 SMTP 空间派生的现有安全模型。

消息的完全加密提供了保护信令机密性的最佳方法——它还可以保证消息不被任何恶意中介修改。 但是，SIP 请求和响应不能完整地端到端加密，因为在大多数网络架构中，诸如 Request-URI、Route 和 Via 等消息字段需要对代理可见，以便正确路由 SIP 请求。 请注意，代理服务器还需要修改消息的某些特性（例如添加 Via 标头字段值）才能使 SIP 正常工作。因此，在某种程度上，SIP UA 必须信任代理服务器。 为此，建议使用 SIP 的低层安全机制，它可以逐跳地加密线路上的整个 SIP 请求或响应，并允许端点验证它们向其发送请求的代理服务器的身份。

SIP 实体还需要以一种安全的方式相互识别。 当 SIP 端点向对等 UA 或代理服务器声明其用户的身份时，该身份应该以某种方式是可验证的。 SIP 中提供了一种加密认证机制来满足这一要求。

用于 SIP 消息体的独立安全机制提供了另一种端到端相互身份验证的方法，并限制了用户代理必须信任中介的程度。

26.2.1 传输和网络层安全

传输层或网络层安全对信令流量进行加密，保证消息的机密性和完整性。

通常，证书用于建立较低层的安全性，并且这些证书还可用于在许多体系结构中提供身份验证手段。

在传输层和网络层提供安全性的两种流行替代方案分别是 TLS [25] 和 IPSec [26]。

IPSec 是一组网络层协议工具，它们可以共同用作传统 IP（互联网协议）的安全替代品。 IPSec 最常用于一组主机或管理域彼此之间存在信任关系的体系结构中。 IPSec 通常在主机的操作系统级别实现，或者在为从特定接口接收的所有流量提供机密性和完整性的安全网关上实现（如在 VPN 体系结构中）。 IPSec 也可以在逐跳的基础上使用。

在许多架构中，IPSec 不需要与 SIP 应用程序集成。 IPSec 可能最适合直接向 SIP 主机添加安全性非常困难的部署。 与其第一跳代理服务器具有预共享密钥关系的 UA 也是使用 IPSec 的良好候选者。 SIP 的任何 IPSec 部署都需要一个 IPSec 配置文件，该配置文件描述了保护 SIP 所需的协议工具。 本文件中没有给出这样的简介。

TLS 通过面向连接的协议（在本文档中为 TCP）提供传输层安全性； “tls”（表示基于 TCP 的 TLS）可以在 Via 头字段值或 SIP-URI 中指定为所需的传输协议。 TLS 最适合在没有预先存在的信任关联的主机之间需要逐跳安全性的架构。 例如，Alice 信任她的本地代理服务器，在证书交换后，该代理服务器决定信任 Bob 信任的 Bob 的本地代理服务器，因此 Bob 和 Alice 可以安全地通信。

TLS 必须与 SIP 应用程序紧密耦合。 请注意，传输机制是在 SIP 中逐跳指定的，因此通过 TLS 向代理服务器发送请求的 UA 无法保证 TLS 将被端到端使用。

在 SIP 应用程序中使用 TLS 时，实施者必须至少支持 TLS_RSA_WITH_AES_128_CBC_SHA 密码套件 [6]。 为了向后兼容，代理服务器、重定向服务器和注册商应该支持 TLS_RSA_WITH_3DES_EDE_CBC_SHA。 实施者也可以支持任何其他密码套件。

26.2.2 SIPS URI 方案

SIPS URI 方案遵循 SIP URI 的语法（在 19 中描述），尽管方案字符串是“sips”而不是“sip”。 然而，SIPS 的语义与 SIP URI 非常不同。 SIPS 允许资源指定应该安全地访问它们。

SIPS URI 可以用作特定用户的记录地址 - 用户通过该 URI 被规范地知道（在他们的名片上，在其请求的 From 标头字段中，在 REGISTER 请求的 To 标头字段中 ）。 当用作请求的 Request-URI 时，SIPS 方案表示转发请求的每一跳，直到请求到达负责 Request-URI 域部分的 SIP 实体，都必须使用 TLS 保护； 一旦它到达有问题的域，它就会根据本地安全和路由策略进行处理，很可能对到 UAS 的任何最后一跳使用 TLS。 当请求的发起者使用时（如果他们使用 SIPS URI 作为目标的记录地址就是这种情况），SIPS 规定到目标域的整个请求路径必须如此安全。

除了 Request-URI 之外，SIPS 方案适用于当今 SIP 中使用 SIP URI 的许多其他方式，包括记录地址、联系地址（联系标头的内容，包括注册方法的内容）) 和路由标头。 在每种情况下，SIPS URI 方案都允许这些现有字段指定安全资源。在任何这些上下文中取消引用 SIPS URI 的方式都有其自己的安全属性，详见 [4]。

特别是 SIPS 的使用需要使用相互 TLS 身份验证，因为应该使用密码套件 TLS_RSA_WITH_AES_128_CBC_SHA。在认证过程中收到的证书应该使用客户端持有的根证书

进行验证； 验证证书失败应该导致请求失败。

特别是 SIPS 的使用需要使用相互 TLS 身份验证，因为应该使用密码套件 TLS_RSA_WITH_AES_128_CBC_SHA。在认证过程中收到的证书应该使用客户端持有的根证书进行验证；验证证书失败应该导致请求失败。请注意，在 SIPS URI 方案中，传输独立于 TLS，因此“sips:alice@atlanta.com;transport=tcp”和“sips:alice@atlanta” .com;transport=sctp" 都是有效的（尽管请注意 UDP 不是 SIPS 的有效传输）。 因此，不推荐使用“transport=tls”，部分原因是它特定于请求的单跳。 这是自 RFC 2543 以来的更改。

将 SIPS URI 作为记录地址分发的用户可以选择操作拒绝通过不安全传输的请求的设备。

26.2.3 HTTP 认证

SIP 提供基于 HTTP 身份验证的质询功能，该功能依赖于 401 和 407 响应代码以及用于携带质询和凭据的标头字段。 无需重大修改，SIP 中 HTTP Digest 身份验证方案的重用允许重放保护和单向身份验证。

第 22 节详细介绍了 SIP 中 Digest 认证的使用。

26.2.4 S/MIME

如上所述，出于机密性目的而端到端加密整个 SIP 消息是不合适的，因为网络中介（如代理服务器）需要查看某些标头字段才能正确路由消息，并且如果这些中介被排除在外 安全关联，那么 SIP 消息基本上是不可路由的。

但是，S/MIME 允许 SIP UA 加密 SIP 中的 MIME 主体，从而在不影响消息头的情况下端到端保护这些主体。 S/MIME 可以为消息正文提供端到端的机密性和完整性，以及相互身份验证。 也可以使用 S/MIME 通过 SIP 消息隧道为 SIP 标头字段提供一种完整性和机密性形式。

第 23 节详细介绍了 SIP 中 S/MIME 的使用。

26.3 实施安全机制

26.3.1 对 SIP 实施者的要求

代理服务器、重定向服务器和注册器必须实现 TLS，并且必须支持双向和单向身份验证。 强烈建议 UA 能够启动 TLS； UA 也可以充当 TLS 服务器。 代理服务器、重定向服务器和注册商应该拥有一个站点证书，其主题对应于它们的规范主机名。 UA 可以拥有自己的证书，用于与 TLS 的相互身份验证，但本文档中没有规定它们的使用。 所有支持 TLS 的 SIP 元素必须具有验证在 TLS 协商期间收到的证书的机制； 这需要拥有一个或多个由证书颁发

机构颁发的根证书（最好是与为 Web 浏览器颁发根证书的网站证书相当的知名分销商）。

所有支持 TLS 的 SIP 元素也必须支持 SIPS URI 方案。

代理服务器、重定向服务器、注册器和 UA 也可以实现 IPSec 或其他较低层的安全协议。

当 UA 尝试联系代理服务器、重定向服务器或注册器时，UAC 应该启动一个 TLS 连接，它将通过该连接发送 SIP 消息。 在某些架构中，UAS 也可以通过此类 TLS 连接接收请求。

代理服务器、重定向服务器、注册器和 UA 必须实现摘要授权，包括 22 中要求的所有方面。代理服务器、重定向服务器和注册器应该配置至少一个摘要领域和至少一个“领域”字符串 给定服务器支持的应该对应于服务器的主机名或域名。

如第 23 节所述，UA 可以支持 MIME 主体的签名和加密，以及使用 S/MIME 传输凭证。如果 UA 持有一个或多个证书颁发机构的根证书以验证 TLS 或 IPSec 的证书，它应该是能够根据需要重新使用这些来验证 S/MIME 证书。 UA 可以持有专门用于验证 S/MIME 证书的根证书。

请注意，随着 S/MIME 实现的出现和对问题空间的更好理解，预计未来的安全扩展可能会提升与 S/MIME 相关的规范强度。

26.3.2 安全解决方案

这些安全机制的协同运行可以在一定程度上遵循现有的网络和电子邮件安全模型。 在高层次上，UA 使用 Digest 用户名和密码向服务器（代理服务器、重定向服务器和注册器）进行身份验证； 服务器使用 TLS 提供的站点证书向一跳外的 UA 或另一台服务器验证自己的身份（反之亦然）。

在点对点级别上，UA 通常信任网络以相互进行身份验证； 但是，当网络不支持或网络本身不受信任时，S/MIME 也可用于提供直接身份验证。

以下是一个说明性示例，其中各种 UA 和服务器使用这些安全机制来防止第 26.1 节中描述的各种威胁。 虽然实施者和网络管理员可以遵循本节其余部分给出的规范性指南，但这些仅作为示例实施提供。

26.3.2.1 注册

当一个 UA 上线并在其本地管理域中注册时，它应该与它的注册器建立一个 TLS 连接（第 10 节描述了 UA 如何到达它的注册器）。 注册商应该向 UA 提供证书，证书标识的站点必须与 UA 打算注册的域相对应； 例如，如果 UA 打算注册地址记录“alice@atlanta.com”，则站点证书必须标识 atlanta.com 域内的主机（例如 sip.atlanta.com）。 当它收到 TLS 证书消息时，UA 应该验证证书并检查证书标识的站点。 如果证书是无效的、被撤销的，或者如果它没有识别适当的一方，UA 不能发送 REGISTER 消息，否则继续注册。

当注册商提供了有效证书时，UA 知道注册商不是可能重定向 UA、窃取密码或尝试任何类似攻击的攻击者。

然后，UA 创建一个 REGISTER 请求，该请求应发送到与从注册器接收的站点证书相对应的 Request-URI。 当 UA 通过现有的 TLS 连接发送 REGISTER 请求时，注册器应该使用 401（需要代理身份验证）响应来质询请求。 响应的 Proxy-Authenticate 头字段中的“realm”参数应该对应于站点证书先前给出的域。 当 UAC 收到质询时，它应该提示用户输入凭据，或者从与质询中的“领域”参数对应的密钥环中获取适当的凭据。 此凭证的用户名应该与 REGISTER 请求的 To 头字段中的 URI 的“userinfo”部分相对应。一旦 Digest 凭证被插入到适当的 Proxy-Authorization 头字段中，REGISTER 应该重新提交给注册商。

由于注册商要求用户代理对其自身进行身份验证，因此攻击者很难伪造对用户记录地址的 REGISTER 请求。 另请注意，由于 REGISTER 是通过机密 TLS 连接发送的，因此攻击者将无法拦截 REGISTER 以记录任何可能的重放攻击的凭据。

一旦注册被注册者接受，UA 应该保持这个 TLS 连接打开，前提是注册者还充当代理服务器，为这个管理域中的用户发送请求。 现有的 TLS 连接将被重用以将传入的请求传递给刚刚完成注册的 UA。

因为 UA 已经在 TLS 连接的另一端对服务器进行了身份验证，所以已知通过此连接的所有请求都已通过代理服务器 - 攻击者无法创建看似已通过该代理服务器发送的欺骗请求。

26.3.2.2 域间请求

现在假设 Alice 的 UA 想与远程管理域中的用户发起会话，即“bob@biloxi.com”。 我们还将说本地管理域 (atlanta.com) 具有本地出站代理。

处理管理域的入站请求的代理服务器也可以充当本地出站代理； 为简单起见，我们假设 atlanta.com 就是这种情况（否则，此时用户代理将启动到单独服务器的新 TLS 连接）。 假设客户端已经完成了上一节中描述的注册过程，当它向另一个用户发送 INVITE 请求时，它应该重用与本地代理服务器的 TLS 连接。 UA 应该重用 INVITE 中缓存的凭证以避免不必要地提示用户。

当本地出站代理服务器验证了 UA 在 INVITE 中提供的凭据时，它应该检查 Request-URI 以确定应该如何路由消息（参见 [4]）。 如果 Request-URI 的“域名”部分对应于本地域 (atlanta.com) 而不是 biloxi.com，则代理服务器将咨询其位置服务以确定如何最好地到达所请求的用户。

如果“alice@atlanta.com”试图联系“alex@atlanta.com”，则本地代理将代理 Alex 在注册时与注册商建立的 TLS 连接的请求。 由于 Alex 会通过他的认证通道接收到这个请求，他可以确信 Alice 的请求已经得到本地管理域的代理服务器的授权。

但是，在这种情况下，Request-URI 指定了一个远程域。 因此 atlanta.com 的本地出站代理服务器应该与 biloxi.com 的远程代理服务器建立 TLS 连接。 由于此 TLS 连接中的两个参与者都是拥有站点证书的服务器，因此应该发生相互 TLS 身份验证。 连接的每一方都应该验证和检查另一方的证书，注意证书中出现的域名，以便与 SIP 消息的标头字段进行比较。例如，atlanta.com 代理服务器应在此阶段验证从远程端接收到的证书是否与 biloxi.com 域相对应。 一旦这样做了，并且 TLS 协商完成，在两个代理之间建立了一个安全通道，atlanta.com 代理可以将 INVITE 请求转发到 biloxi.com。

biloxi.com 的代理服务器应该依次检查 atlanta.com 代理服务器的证书，并将证书声明的域与 INVITE 请求中 From 头字段的“域名”部分进行比较。 biloxi 代理可能有一个严格的安全策略，要求它拒绝与代理它们的管理域不匹配的请求。

可以制定此类安全策略，以防止经常被利用来生成垃圾邮件的 SMTP“开放中继”的 SIP 等价物。

然而，这个策略只保证请求来自它所归属的域； 它不允许 biloxi.com 确定 atlanta.com 如何对 Alice 进行身份验证。 只有当 biloxi.com 有其他方式了解 atlanta.com 的身份验证策略时，它才有可能确定 Alice 是如何证明她的身份的。 然后，biloxi.com 可能会制定更严格的政策，禁止来自管理上未知的域的请求与 biloxi.com 共享通用身份验证策略。

一旦 INVITE 被 biloxi 代理批准，代理服务器应该识别与该请求的目标用户相关联的现有 TLS 通道，如果有的话（在本例中为“bob@biloxi.com”）。 INVITE 应该通过这个通道代理给 Bob。 由于该请求是通过之前已被验证为 biloxi 代理的 TLS 连接接收的，因此 Bob 知道 From 标头字段没有被篡改，并且 atlanta.com 已经验证了 Alice，尽管不一定是否信任 Alice 的身份。

在他们转发请求之前，两个代理服务器都应该在请求中添加一个 Record-Route 头字段，以便此对话框中的所有未来请求都将通过代理服务器。 代理服务器因此可以在此对话的生命周期内继续提供安全服务。 如果代理服务器不将自己添加到 Record-Route，未来的消息将直接在 Alice 和 Bob 之间端到端传递，无需任何安全服务（除非双方同意一些独立的端到端安全性，例如 S/MIME）。 在这方面，SIP 梯形模型可以提供一个很好的结构，其中站点代理之间的协议约定可以在 Alice 和 Bob 之间提供一个相当安全的通道。

例如，利用此架构的攻击者将无法伪造 BYE 请求并将其插入 Bob 和 Alice 之间的信令流中，因为攻击者无法确定会话的参数，而且还因为完整性机制传递保护 Alice 和 Bob 之间的流量。

26.3.2.3 点对点请求

或者，考虑一个声明身份“carol@chicago.com”的 UA，它没有本地出站代理。当 Carol 希望向“bob@biloxi.com”发送邀请时，她的 UA 应该直接与 biloxi 代理发起 TLS 连接（使用 [4] 中描述的机制来确定如何最好地到达给定的 Request-URI） .当她的 UA 收到来自 biloxi 代理的证书时，应该在她通过 TLS 连接传递她的 INVITE 之前对其进行正常验证。然而，Carol

无法向 biloxi 代理证明她的身份，但她在 INVITE 中的“消息/sip”正文上确实有一个 CMS 分离的签名。在这种情况下，Carol 不太可能拥有 biloxi.com 领域的任何凭据，因为她与 biloxi.com 没有正式关联。 biloxi 代理也可能有一个严格的政策，它甚至不会打扰挑战在 From 标头字段的“域名”部分中没有 biloxi.com 的请求 - 它会将这些用户视为未经身份验证的用户。

biloxi 代理对 Bob 有一个策略，即所有未经身份验证的请求都应重定向到针对“bob@biloxi.com”注册的适当联系地址，即 <sip:bob@192.0.2.4>。 Carol 通过她与 biloxi 代理建立的 TLS 连接接收重定向响应，因此她相信联系地址的真实性。

然后，Carol 应该与指定地址建立 TCP 连接，并发送一个新的 INVITE，其中包含收到的联系地址的 Request-URI（在请求准备好时重新计算正文中的签名）。 Bob 在一个不安全的接口上接收到这个 INVITE，但他的 UA 会检查并在这种情况下识别请求的 From 头字段，然后将本地缓存的证书与 INVITE 正文的签名中呈现的证书进行匹配。 他以类似的方式回复，向 Carol 证明了自己的身份，安全对话开始了。

有时，管理域中的防火墙或 NAT 可能会阻止与 UA 建立直接 TCP 连接。 在这些情况下，代理服务器还可以按照本地策略的规定，以一种不涉及信任的方式（例如，放弃现有的 TLS 连接并通过明文 TCP 转发请求）将请求中继到 UA。

26.3.2.4 DoS 保护

为了将拒绝服务攻击对使用这些安全解决方案的架构的风险降至最低，实施者应注意以下准则。

当运行 SIP 代理服务器的主机可从公共 Internet 路由时，它应该部署在具有防御性操作策略的管理域中（阻止源路由流量，最好过滤 ping 流量）。 TLS 和 IPSec 还可以利用参与安全关联的管理域边缘的堡垒主机来聚合安全隧道和套接字。 这些堡垒主机还可以首当其冲受到拒绝服务攻击，确保管理域内的 SIP 主机不会受到多余消息的阻碍。

无论部署何种安全解决方案，指向代理服务器的大量消息都会锁定代理服务器资源并阻止所需流量到达其目的地。 与在代理服务器上处理 SIP 事务相关的计算费用，对于有状态代理服务器来说比对于无状态代理服务器的费用更大。 因此，有状态的代理服务器比无状态的代理服务器更容易受到洪水的影响。

UA 和代理服务器应该只用一个 401（未授权）或 407（需要代理身份验证）来挑战有问题的请求，放弃正常的响应重传算法，从而对未授权的请求表现出无状态的行为。

重新传输 401（未授权）或 407（需要代理身份验证）状态响应会放大攻击者使用伪造的标头字段值（例如 Via）将流量定向到第三方的问题。

总之，通过 TLS 等机制对代理服务器进行相互身份验证显着降低了流氓中介引入可拒绝服务的伪造请求或响应的可能性。 这相应地使攻击者更难将无辜的 SIP 节点变成放大代理。

26.4 限制

尽管这些安全机制在以明智的方式应用时可以阻止许多威胁，但实施者和网络运营商必须理解的机制范围存在限制。

26.4.1 HTTP 摘要

在 SIP 中使用 HTTP Digest 的主要限制之一是 Digest 中的完整性机制不适用于 SIP。具体来说，它们提供对 Request-URI 和消息方法的保护，但不保护 UA 最可能希望保护的任何标头字段。

RFC 2617 中描述的现有重放保护机制对 SIP 也有一些限制。 例如，next-nonce 机制不支持流水线请求。 应使用 nonce-count 机制进行重放保护。

HTTP Digest 的另一个限制是领域的范围。当用户想要对与他们具有预先存在关联的资源进行身份验证时，Digest 很有价值，例如用户是其客户的服务提供商（这是一种非常常见的场景，因此 Digest 提供了非常有用的功能）．相比之下，TLS 的范围是域间或多域，因为证书通常是全局可验证的，因此 UA 可以在没有预先存在的关联的情况下验证服务器。

26.4.2 S/MIME

S/MIME 机制最大的突出缺陷是缺乏面向最终用户的流行公钥基础设施。 如果使用自签名证书（或无法由对话中的参与者之一验证的证书），则第 23.2 节中描述的基于 SIP 的密钥交换机制容易受到中间人攻击， 攻击者可能会检查和修改 S/MIME 正文。 攻击者需要在对话中拦截双方之间的第一次密钥交换，从请求和响应中删除现有的 CMS 分离签名，并插入包含攻击者提供的证书的不同 CMS 分离签名（但这似乎 作为正确记录地址的证书）。每一方都会认为他们已经与另一方交换了密钥，而实际上每一方都拥有攻击者的公钥。

值得注意的是，攻击者只能在两方之间第一次交换密钥时利用此漏洞——在随后的情况下，UA 会注意到密钥的更改。 随着时间的推移（可能会经过几天、几周或几年），攻击者也很难留在双方未来所有对话的路径上。

SSH 在第一次密钥交换时容易受到同样的中间人攻击； 然而，人们普遍承认，虽然 SSH 并不完美，但它确实提高了连接的安全性。 密钥指纹的使用可以为 SIP 提供一些帮助，就像它为 SSH 所做的那样。 例如，如果两方使用 SIP 建立语音通信会话，则每一方都可以读取他们从对方那里收到的密钥的指纹，这可以与原始密钥进行比较。 对于中间人来说，模仿参与者的声音肯定比他们的信号更难（这种做法与基于 Clipper 芯片的安全电话一起使用）。

如果 UA 在其密钥环上拥有记录目标地址的证书，则 S/MIME 机制允许 UA 发送不带前缀的加密请求。 但是，注册地址记录的任何特定设备都可能不会持有该设备当前用户之前使用过的证书，因此它将无法正确处理加密请求，这可能导致 到一些可避免的错误信号。 当

分叉加密请求时，这种情况尤其可能发生。

当与特定用户（记录地址）而不是设备（UA）关联时，与 S/MIME 关联的密钥最有用。 当用户在设备之间移动时，可能难以在 UA 之间安全地传输私钥； 设备如何获取此类密钥超出了本文档的范围。

S/MIME 机制的另一个更普通的困难是它可能导致非常大的消息，尤其是在使用第 23.4 节中描述的 SIP 隧道机制时。 因此，建议在使用 S/MIME 隧道时将 TCP 用作传输协议。

26.4.3 TLS

对 TLS 最常见的担忧是它不能在 UDP 上运行。 TLS 需要一个面向连接的底层传输协议，在本文档中指的是 TCP。

本地出站代理服务器和/或注册商与众多 UA 保持许多同时的长期 TLS 连接也可能很困难。这引入了一些有效的可扩展性问题，特别是对于密集的密码套件。 维护长期 TLS 连接的冗余，特别是当 UA 单独负责它们的建立时，也可能很麻烦。

TLS 只允许 SIP 实体对它们相邻的服务器进行身份验证；TLS 提供严格的逐跳安全性。TLS 和本文档中指定的任何其他机制都不允许客户端对无法与其形成直接 TCP 连接的代理服务器进行身份验证。

26.4.4 SIPS URIs

实际上，在请求路径的每个段上使用 TLS 意味着终止的 UAS 必须可以通过 TLS 访问（可能使用 SIPS URI 作为联系地址进行注册）。 这是 SIPS 的首选用途。 然而，许多有效的架构使用 TLS 来保护请求路径的一部分，但依赖于一些其他机制来实现到 UAS 的最后一跳。因此 SIPS 不能保证 TLS 的使用将是真正的端到端。 请注意，由于许多 UA 不接受传入的 TLS 连接，因此即使是那些支持 TLS 的 UA，也可能需要按照上面的 TLS 限制部分所述保持持久的 TLS 连接，以便通过 TLS 作为 UAS 接收请求。

位置服务不需要为 SIPS 请求 URI 提供 SIPS 绑定。尽管位置服务通常由用户注册填充（如第 10.2.1 节所述），但可以想象各种其他协议和接口可以为 AOR 提供联系地址，并且这些工具可以根据需要自由地将 SIPS URI 映射到 SIP URI。 当查询绑定时，位置服务返回其联系地址，而不考虑它是否收到带有 SIPS 请求 URI 的请求。 如果重定向服务器正在访问位置服务，则由处理重定向的联系人头字段的实体来确定联系人地址的适当性。

确保 TLS 将用于直到目标域的所有请求段有点复杂。沿途经过加密验证的代理服务器不合规或受到损害，可能会选择忽略与 SIPS 相关的转发规则（以及第 16.6 节中的一般转发规则）。 例如，此类恶意中介可以将请求从 SIPS URI 重定向到 SIP URI，以尝试降低安全性。

或者，中介可能合法地将请求从 SIP 重定向到 SIPS URI。 Request-URI 使用 SIPS URI 方案的请求的接收者因此不能仅基于 Request-URI 假设 SIPS 用于整个请求路径（从客户端开

始）。

为了解决这些问题，建议其 Request-URI 包含 SIP 或 SIPS URI 的请求的接收者检查 To 标头字段值以查看它是否包含 SIPS URI（但请注意，如果 此 URI 具有相同的方案，但不等同于 To 标头字段中的 URI）。 尽管客户端可以选择以不同方式填充请求的 Request-URI 和 To 头字段，但当使用 SIPS 时，这种差异可能会被解释为可能的安全违规，因此请求可能会被其接收者拒绝。 接收者也可以检查 Via 头链，以仔细检查整个请求路径是否使用了 TLS，直到到达本地管理域。 S/MIME 也可以由始发 UAC 使用，以帮助确保 To 标头字段的原始形式是端到端传送的。

如果 UAS 有理由相信 Request-URI 的方案在传输过程中被不当修改，UA 应该通知其用户潜在的安全漏洞。

作为防止降级攻击的进一步措施，仅接受 SIPS 请求的实体也可以拒绝不安全端口上的连接。

最终用户无疑会辨别 SIPS 和 SIP URI 之间的区别，并且他们可以手动编辑它们以响应刺激。这可能有利于或降低安全性。 例如，如果攻击者破坏了 DNS 缓存，插入了一个有效删除代理服务器所有 SIPS 记录的虚假记录集，则任何遍历该代理服务器的 SIPS 请求都可能失败。 但是，当用户看到对 SIPS AOR 的重复调用失败时，他们可以在某些设备上手动将方案从 SIPS 转换为 SIP，然后重试。 当然，对此有一些保护措施（如果目标 UA 真的很偏执，它可以拒绝所有非 SIPS 请求），但这是一个值得注意的限制。 从好的方面来说，用户也可能认为“SIPS”是有效的，即使他们只看到一个 SIP URI。

26.5 隐私

SIP 消息经常包含有关其发件人的敏感信息——不仅是他们必须说的话，还包括他们与谁通信、通信时间、通信时长以及他们从何处参与会话。 许多应用程序及其用户要求对不需要知道的任何一方隐藏此类私人信息。

请注意，泄露私人信息的方式也不太直接。 如果用户或服务选择的地址可以从人的姓名和组织隶属关系（描述大多数记录地址）中猜出，那么通过使用未列出的“电话号码”来确保隐私的传统方法就会受到损害。 用户定位服务可以通过向呼叫者泄露他们的具体下落来侵犯会话邀请接收者的隐私； 因此，实现应该能够在每个用户的基础上限制向某些类别的呼叫者提供什么样的位置和可用性信息。 这是一整类问题，预计将在正在进行的 SIP 工作中进一步研究。

在某些情况下，用户可能希望在传达身份的标题字段中隐藏个人信息。 这不仅适用于代表请求发起者的 From 和相关标头，还适用于 To - 向最终目的地传达快速拨号昵称或一组目标的未扩展标识符可能不合适， 当请求被路由时，其中任何一个都将从 Request-URI 中删除，但如果两者最初相同，则在 To 标头字段中不会更改。 因此，出于隐私原因，可能需要创建一个不同于 Request-URI 的 To 头字段。

27 IANA 考虑事项

SIP 应用程序中使用的所有方法名称、标头字段名称、状态代码和选项标签都通过 RFC 中 IANA 考虑部分中的说明向 IANA 注册。

该规范指示 IANA 在 http://www.iana.org/assignments/sip-parameters 下创建四个新的子注册管理机构：选项标签、警告代码 (warn-codes)、方法和响应代码，添加到子注册管理机构已经存在的标题字段的注册表。

27.1 选项标签

本规范在 http://www.iana.org/assignments/sip-parameters 下建立了选项标签子注册表。

选项标签用于标头字段，例如 Require、Supported、Proxy-Require 和 Unsupported，以支持扩展的 SIP 兼容性机制（第 19.2 节）。 选项标签本身是一个与特定 SIP 选项（即扩展）相关联的字符串。 它标识 SIP 端点的选项。

选项标签在标准跟踪 RFC 中发布时由 IANA 注册。 RFC 的 IANA 注意事项部分必须包含以下信息，这些信息与出版物的 RFC 编号一起出现在 IANA 注册表中。

o 选项标签的名称。 名称可以是任意长度，但长度不应超过 20 个字符。 名称必须仅由字母数字（第 25 节）字符组成。

o 描述扩展的描述性文本。

27.2 警告代码

本规范在 http://www.iana.org/assignments/sip-parameters 下建立了警告代码子注册中心，并使用第 20.43 节中列出的警告代码启动其填充。 其他警告代码由 RFC 发布注册。

警告代码表的描述性文本是：

当事务失败由会话描述协议 (SDP) (RFC 2327 [1]) 问题导致时，警告代码提供对 SIP 响应消息中的状态代码的补充信息。

“警告代码”由三位数字组成。 第一个数字“3”表示特定于 SIP 的警告。 在未来的规范描述使用 3xx 以外的警告代码之前，只能注册 3xx 警告代码。

警告 300 到 329 保留用于指示会话描述中关键字的问题，330 到 339 是与会话描述中请求的基本网络服务相关的警告，370 到 379 是与会话描述中请求的定量 QoS 参数相关的警告，390 通过 399 是不属于上述类别之一的杂项警告。

27.3 头域名称

这将废弃有关 http://www.iana.org/assignments/sip-parameters 下标头子注册表的 IANA 说明。

为了注册新的标头字段名称，需要在 RFC 发布中提供以下信息：

o 标头注册的 RFC 编号；

o 正在注册的标头字段的名称；

o 该标题字段的紧凑表单版本（如果已定义）；

一些常见和广泛使用的头域可以分配一个字母的紧凑形式（第 7.3.3 节）。 紧凑型表格只能在 SIP 工作组审核后分配，然后再发布 RFC。

27.4 方法和响应代码

本规范在 http://www.iana.org/assignments/sip-parameters 下建立了方法和响应代码子注册表，并按如下方式启动它们的填充。 初始方法表是：

```
INVITE                          [RFC3261]
ACK                             [RFC3261]
BYE                             [RFC3261]
CANCEL                          [RFC3261]
REGISTER                        [RFC3261]
OPTIONS                         [RFC3261]
INFO                            [RFC2976]
```

响应代码表最初是从第 21 节填充的，这些部分标记为 Informational、Success、Redirection、Client-Error、Server-Error 和 Global-Failure。 该表具有以下格式：

```
Type (e.g., Informational)
      Number    Default Reason Phrase          [RFC3261]
```

为了注册新的响应代码或方法，需要在 RFC 出版物中提供以下信息：

o 注册方法或响应代码的 RFC 编号；

o 响应代码的编号或注册方法的名称；

o 该响应代码的默认原因短语（如果适用）；

27.5 “message/sip”MIME 类型

本文档注册“message/sip”MIME 媒体类型，以允许 SIP 消息作为 SIP 内的主体进行隧道传

输，主要用于端到端安全目的。 此媒体类型由以下信息定义：

```
Media type name: message
Media subtype name: sip
Required parameters: none
```

可选参数： version
version：所附消息的 SIP 版本号（例如，“2.0”）。 如果不存在，则版本默认为“2.0”。
编码方案：SIP 消息由 8 位标头组成
可选地后跟一个二进制 MIME 数据对象。 因此，SIP 消息必须被视为二进制。 在正常情况下，SIP 消息通过支持二进制的传输方式传输，不需要特殊编码。
安全注意事项：见下文
在 23.4 中给出了这种使用作为与 S/MIME 一致的安全机制的动机和示例。

27.6 新的内容配置参数注册

该文档还注册了四个新的 Content-Disposition 标头“处置类型”：警报、图标、会话和渲染。作者要求将这些值记录在 IANA 的 Content-Dispositions 注册表中。

这些“性格类型”的描述，包括动机和例子，在第 20.11 节中给出。

适用于 IANA 注册管理机构的简短描述如下：

alert: body 是一个自定义铃声来提醒用户

icon: 正文显示为用户的图标

render: 正文应该显示给用户

session: 正文描述了一个通信会话，例如，RFC 2327 SDP 正文

28 对 RFC 2543 的更改

本 RFC 对 RFC 2543 进行了修订。它主要向后兼容 RFC 2543。此处描述的更改修复了 RFC 2543 中发现的许多错误，并提供了有关 RFC 2543 中未详细说明的场景的信息。该协议已在此处以更清晰的分层模型呈现。

我们将差异分解为与 RFC 2543 相比有重大变化的功能行为，这在某些情况下会影响互操作性或正确操作，以及与 RFC 2543 不同但不是互操作性问题的潜在来源的功能行为。 也有无数的澄清，这里没有记录。

28.1 主要功能变化

o 当一个 UAC 希望在一个呼叫被应答之前终止它，它会发送 CANCEL。 如果原来的 INVITE 仍然返回 2xx，UAC 然后发送 BYE。 BYE 只能在现有呼叫段（现在在本 RFC 中称为对话）上发送，而在 RFC 2543 中可以随时发送。

o SIP BNF 已转换为符合 RFC 2234。

o SIP URL BNF 变得更通用，在用户部分允许更多的字符集。 此外，比较规则被简化为主要不区分大小写，并描述了在存在参数时比较的详细处理。 最大的变化是带有默认值参数的 URI 与没有该参数的 URI 不匹配。

o 通过隐藏删除。 它有严重的信任问题，因为它依靠下一跳来执行混淆过程。 相反，Via 隐藏可以作为有状态代理中的本地实现选择来完成，因此不再记录。

o 在 RFC 2543 中，CANCEL 和 INVITE 事务混合在一起。 他们现在分开了。 当用户发送 INVITE 然后 CANCEL 时，INVITE 事务仍然正常终止。 UAS 需要以 487 响应来响应原始 INVITE 请求。

o 同样，CANCEL 和 BYE 交易也混杂在一起； RFC 2543 允许 UAS 在收到 BYE 时不向 INVITE 发送响应。 这是不允许的。 原始 INVITE 需要响应。

o 在 RFC 2543 中，UA 只需要支持 UDP。在这个 RFC 中，UA 需要同时支持 UDP 和 TCP。

o 在 RFC 2543 中，分支代理仅在遇到多个挑战时传递来自下游元素的一个挑战。 在这个 RFC 中，代理应该收集所有挑战并将它们放入转发的响应中。

o 在 Digest credentials 中，需要引用 URI； 这在 RFC 2617 和 RFC 2069 中并不清楚，两者都不一致。

o SDP 处理已被拆分为一个单独的规范 [13]，并且更完整地指定为通过 SIP 有效隧道化的正式提议/答案交换过程。 在 INVITE/200 或 200/ACK 中允许 SDP 用于基线 SIP 实现；RFC 2543 提到了在单个事务中在 INVITE、200 和 ACK 中使用它的能力，但这并没有很好地说明。 扩展中允许使用更复杂的 SDP。

o 在 URI 和 Via 标头字段中添加了对 IPv6 的完全支持。 Via 对 IPv6 的支持要求其标头字段参数允许使用方括号和冒号字符。 这些字符以前是不允许的。 理论上，这可能会导致与旧实现的互操作问题。 但是，我们观察到大多数实现都接受这些参数中的任何非控制 ASCII 字符。

o DNS SRV 程序现在记录在单独的规范 [4] 中。 此过程同时使用 SRV 和 NAPTR 资源记录，并且不再像 RFC 2543 中所述组合来自 SRV 记录的数据。

o 循环检测已成为可选的，被强制使用 Max-Forwards 所取代。 RFC 2543 中的循环检测过程有一个严重的错误，如果不是，则会将“螺旋”报告为错误条件。 可选的环路检测程序在这里更完整和正确地指定。

o 标签的使用现在是强制性的（它们在 RFC 2543 中是可选的），因为它们现在是对话识别的基本构建块。

o 添加了 Supported 标头字段，允许客户端指示服务器支持哪些扩展，服务器可以将这些扩展应用于响应，并在响应中使用 Require 指示它们的使用。

o BNF 中缺少几个标头字段的扩展参数，它们已被添加。

o 在 RFC 2543 中对路由和记录路由构造的处理没有明确规定，也不是正确的方法。 它在本规范中进行了大幅修改（并且变得更加简单），这可以说是最大的变化。 仍然为不使用“预加载路由”的部署提供向后兼容性，其中初始请求具有一组 Route 标头字段值，这些值以某种方式在 Record-Route 之外获得。 在这些情况下，新机制是不可互操作的。

o 在 RFC 2543 中，消息中的行可以以 CR、LF 或 CRLF 结尾。 该规范仅允许 CRLF。

o RFC 2543 中没有很好地定义 CANCEL 和 ACK 中路由的使用。现在已经很好地指定了；如果一个请求有一个 Route 头域，它的 CANCEL 或 ACK 对请求的非 2xx 响应需要携带相同的 Route 头域值。 2xx 响应的 ACK 使用从 2xx 响应的 Record-Route 中学习到的 Route 值。

o RFC 2543 允许单个 UDP 数据包中的多个请求。 此用法已被删除。

o 已删除 Expires 标头字段和参数中使用的绝对时间。 它在时间不同步的元素中引起了互操作性问题，这是一种常见的现象。 而是使用相对时间。

o 现在，所有元素都必须使用 Via 标头字段值的分支参数。 它现在扮演唯一交易标识符的角色。 这避免了来自 RFC 2543 的复杂且充满错误的事务识别规则。参数值中使用了一个魔术 cookie，以确定前一跳是否使参数全局唯一，并且当不存在时，比较回退到旧规则 . 因此，确保了互操作性。

o 在 RFC 2543 中，关闭 TCP 连接等同于 CANCEL。 对于代理之间的 TCP 连接，这几乎不可能实现（而且是错误的）。 这已被消除，因此 TCP 连接状态和 SIP 处理之间没有耦合。

o RFC 2543 没有说明 UA 是否可以在另一个事务正在进行时向对等方发起新事务。 现在在这里指定。 对于非 INVITE 请求是允许的，对于 INVITE 是不允许的。

o PGP 已被删除。 它没有被充分指定，并且与更完整的 PGP MIME 不兼容。 它被 S/MIME 取代。

o 为端到端 TLS 添加了“sips”URI 方案。 此方案与 RFC 2543 不向后兼容。现有元素在

Request-URI 中接收具有 SIPS URI 方案的请求可能会拒绝该请求。 这实际上是一个功能；它确保只有在所有路径跃点都可以得到保护的情况下才传递对 SIPS URI 的调用。

o TLS 添加了其他安全功能，这些功能在更大更完整的安全注意事项部分中进行了描述。

o 在 RFC 2543 中，不需要代理将临时响应从上游 101 转发到 199。 这已更改为必须。 这很重要，因为许多后续功能依赖于从 101 到 199 的所有临时响应的传递。

o 关于 RFC 2543 中的 503 响应代码几乎没有提及。此后，它在指示代理中的故障或过载情况方面具有重要用途。 这需要一些特殊的处理。 具体来说，收到 503 应触发尝试联系 DNS SRV 查找结果中的下一个元素。 此外，503 响应仅在某些条件下由代理向上游转发。

o RFC 2543 定义但没有充分指定服务器的 UA 身份验证机制。 那已被删除。 相反，允许 RFC 2617 的相互认证过程。

o 在收到初始 INVITE 的 ACK 之前，UA 不能为呼叫发送 BYE。 这在 RFC 2543 中是允许的，但会导致潜在的竞争条件。

o 在获得请求的临时响应之前，UA 或代理无法为事务发送 CANCEL。 这在 RFC 2543 中是允许的，但会导致潜在的竞争条件。

o 注册中的操作参数已被弃用。 它不足以提供任何有用的服务，并且在代理中应用应用程序处理时会引起冲突。

o RFC 2543 有一些多播的特殊情况。 例如，某些响应被抑制，计时器被调整，等等。 多播现在扮演的角色更加有限，并且协议操作不受使用多播而不是单播的影响。 由此产生的限制被记录在案。

o 基本身份验证已完全删除，禁止使用。

o 代理不再在收到 6xx 后立即转发它。 相反，他们立即取消挂起的分支。 这避免了可能导致 UAC 获得 6xx 后跟 2xx 的潜在竞争条件。 在除此竞争条件之外的所有情况下，结果都是相同的 - 6xx 被转发到上游。

o RFC 2543 没有解决请求合并的问题。 当请求在代理处分叉并随后在元素处重新加入时，就会发生这种情况。 合并的处理仅在 UA 处完成，并且定义了用于拒绝除第一个请求之外的所有请求的过程。

28.2 微小的功能变化

o 添加了 Alert-Info、Error-Info 和 Call-Info 标头字段，用于向用户展示可选的内容。

o 添加了 Content-Language、Content-Disposition 和 MIME-Version 标头字段。

o 增加了“眩光处理”机制来处理双方同时向对方发送 re-INVITE 的情况。 它使用新的 491（请求挂起）错误代码。

o 添加了 In-Reply-To 和 Reply-To 头字段，用于支持稍后返回未接来电或消息。

o 添加了 TLS 和 SCTP 作为有效的 SIP 传输。

o 有多种机制用于在通话期间随时处理故障； 这些现在大体上是统一的。 发送 BYE 以终止。

o RFC 2543 要求通过 TCP 重新传输 INVITE 响应，但指出它实际上只需要 2xx。 这是协议分层不足的产物。 有了这里定义的更一致的事务层，就不再需要了。 只有 2xx 对 INVITE 的响应通过 TCP 重新传输。

o 客户端和服务器事务机器现在基于超时而不是重传计数来驱动。 这允许为 TCP 和 UDP 正确指定状态机。

o 在 REGISTER 响应中使用 Date 标头字段，以提供一种在用户代理中自动配置日期的简单方法。

o 允许注册商拒绝过期时间过短的注册。 为此目的定义了 423 响应代码和 Min-Expires。

29 规范性参考文献

[1] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[4] Rosenberg, J. and H. Schulzrinne, "SIP: Locating SIP Servers", RFC 3263, June 2002.

[5] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[6] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

[7] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[8] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[9] Vaha-Sipila, A., "URLs for Telephone Calls", RFC 2806, April 2000.

[10] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[11] Freed, F. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[12] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[13] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with SDP", RFC 3264, June 2002.

[14] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[15] Postel, J., "DoD Standard Transmission Control Protocol", RFC 761, January 1980.

[16] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[17] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[18] Troost, R., Dorner, S. and K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997.

[19] Zimmerer, E., Peterson, J., Vemuri, A., Ong, L., Audet, F., Watson, M. and M. Zonoun, "MIME media types for ISUP and QSIG Objects", RFC 3204, December 2001.

[20] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[21] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[22] Galvin, J., Murphy, S., Crocker, S. and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[23] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[24] Ramsdell B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[25] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[26] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

30 参考资料

[27] R. Pandya, "Emerging mobile and personal communication systems," IEEE Communications Magazine, Vol. 33, pp. 44--52, June 1995.

[28] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[29] Schulzrinne, H., Rao, R. and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[30] Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. and J. Segers, "Megaco Protocol Version 1.0", RFC 3015, November 2000.

[31] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[32] Hoffman, P., Masinter, L. and J. Zawinski, "The mailto URL scheme", RFC 2368, July 1998.

[33] E. M. Schooler, "A multicast user directory service for synchronous rendezvous," Master's Thesis CS-TR-96-18, Department of Computer Science, California Institute of Technology, Pasadena, California, Aug. 1996.

[34] Donovan, S., "The SIP INFO Method", RFC 2976, October 2000.

[35] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[36] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, September 1998.

[37] Good, G., "The LDAP Data Interchange Format (LDIF) - Technical Specification", RFC 2849, June 2000.

[38] Palme, J., "Common Internet Message Headers", RFC 2076, February 1997.

[39] Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E. and L. Stewart, "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997.

[40] Johnston, A., Donovan, S., Sparks, R., Cunningham, C., Willis, D., Rosenberg, J., Summers, K. and H. Schulzrinne, "SIP Call Flow Examples", Work in Progress.

[41] E. M. Schooler, "Case study: multimedia conference control in a packet-switched teleconferencing system," Journal of Internetworking: Research and Experience, Vol. 4, pp. 99--120, June 1993. ISI reprint series ISI/RS-93-359.

[42] H. Schulzrinne, "Personal mobility for multimedia services in the Internet," in European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS), (Berlin, Germany), Mar. 1996.

[43] Floyd, S., "Congestion Control Principles", RFC 2914, September 2000.

A 定时器值表

表 4 总结了本规范使用的各种定时器的含义和默认值。

Timer	Value	Section	Meaning
T1	500ms default	Section 17.1.1.1	RTT Estimate
T2	4s	Section 17.1.2.2	The maximum retransmit interval for non-INVITE requests and INVITE responses
T4	5s	Section 17.1.2.2	Maximum duration a message will remain in the network
Timer A	initially T1	Section 17.1.1.2	INVITE request retransmit interval, for UDP only
Timer B	64*T1	Section 17.1.1.2	INVITE transaction timeout timer
Timer C	> 3min	Section 16.6 bullet 11	proxy INVITE transaction timeout
Timer D	> 32s for UDP 0s for TCP/SCTP	Section 17.1.1.2	Wait time for response retransmits
Timer E	initially T1	Section 17.1.2.2	non-INVITE request retransmit interval, UDP only
Timer F	64*T1	Section 17.1.2.2	non-INVITE transaction timeout timer
Timer G	initially T1	Section 17.2.1	INVITE response retransmit interval
Timer H	64*T1	Section 17.2.1	Wait time for ACK receipt
Timer I	T4 for UDP 0s for TCP/SCTP	Section 17.2.1	Wait time for ACK retransmits
Timer J	64*T1 for UDP 0s for TCP/SCTP	Section 17.2.2	Wait time for non-INVITE request retransmits
Timer K	T4 for UDP 0s for TCP/SCTP	Section 17.1.2.2	Wait time for response retransmits

Table 4: Summary of timers

致谢

作者地址

完整的版权声明