# 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

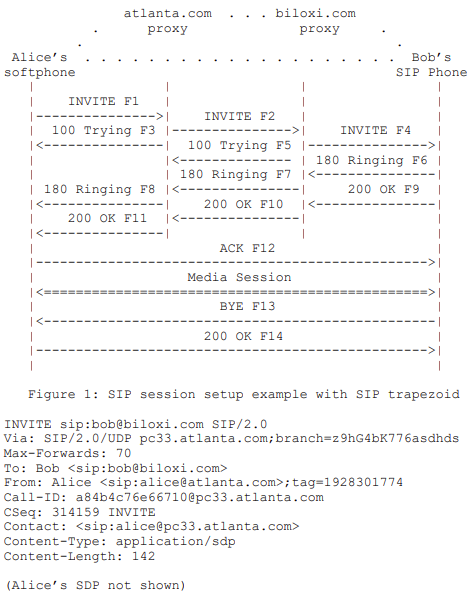
本节通过简单的示例介绍 SIP 的基本操作。 本节本质上是教程，不包含任何规范性陈述。

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：



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。

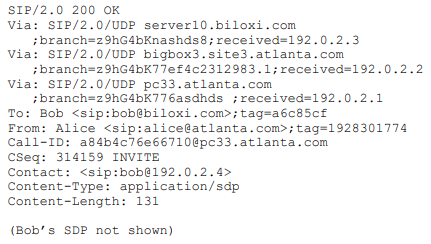
由于软件电话不知道 Bob 的位置或 biloxi.com 域中的 SIP 服务器的位置，因此软件电话将 INVITE 发送到服务于 Alice 的域 atlanta.com 的 SIP 服务器。 例如，atlanta.com SIP 服务器的地址可能已在 Alice 的软电话中配置，或者可能已被 DHCP 发现。

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：



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。

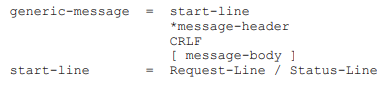
上面定义的代理、位置和注册服务器是逻辑实体； 实现可以将它们组合成一个应用程序。

# 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。



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)。



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。



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：



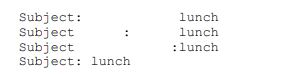
允许将同名的标题字段组合成一个逗号分隔的列表。 Contact 头字段允许使用逗号分隔的列表，除非头字段值为“\*”。

### 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。

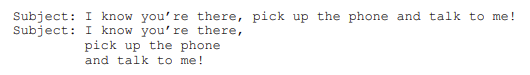


第 25 节中指定的消息头的形式语法允许在冒号的任一侧有任意数量的空格； 但是，实现应避免在字段名称和冒号之间使用空格，并在冒号和字段值之间使用单个空格 (SP)。



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：



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：



以下每个块都是有效的，但不等同于其他块：



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：



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，



相当于



和



相当于



以下两个标头字段不等效：



### 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：



#### 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：



#### 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：



#### 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：



#### 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：



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

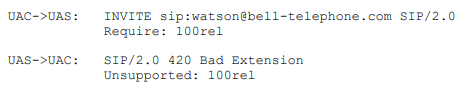
假设 UAS 确定它是处理请求的正确元素，它会检查 Require 头字段（如果存在）。

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：



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。

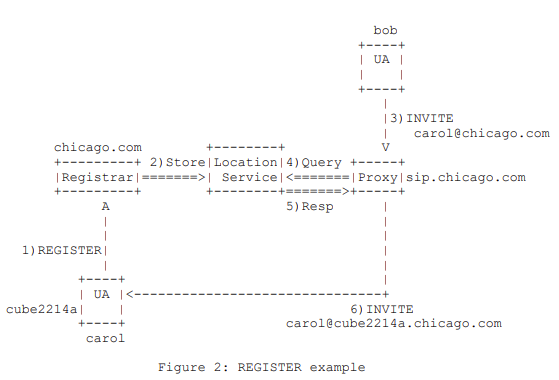
Call-ID：来自 UAC 的所有注册都应该使用相同的 Call-ID 头字段值，用于发送到特定注册商的注册。

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。



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：



## 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。

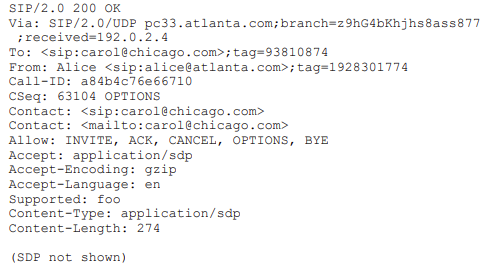
由于 OPTIONS 和 INVITE 请求的代理处理不同，因此使用 OPTIONS 具有局限性。 虽然分叉的 INVITE 可能会导致返回多个 200（OK）响应，但分叉的 OPTIONS 只会导致单个 200（OK）响应，因为它是由使用非 INVITE 处理的代理处理的。 有关规范性细节，请参见第 16.7 节。

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）：



# 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