YD

中华人民共和国通信行业标准

YD/T 1522.1-2006

会话初始协议（SIP）技术要求 第1部分：基本的会话初始协议

Technical Requirments for Session Initiation Protocol

Part 1:Basic Session Initiation Protocol

(IETF RFC 3261,2002,SIP:Session Initiation Protocol,NEQ)

2006-12-11 发布　　2007-01-01 实施

中华人民共和国信息产业部　发布

目　录

前　言

《会话初始协议（SIP）技术要求》预计分为5个部分：

——第1部分：基本的会话初始协议；

——第2部分：基于会话初始协议（SIP）的呼叫控制的应用；

——第3部分：ISDN用户部分（ISUP）和会话初始协议（SIP）的互通；

——第4部分：软交换网络中基于呼叫控制的会话初始协议（SIP）技术要求；

——第5部分：支持移动应用部分。

本部分为《会话初始协议（SIP）技术要求》的第1部分。

本部分对应于IETF RFC3261（2002）《SIP：会话初始协议》，与RFC3261（2002）的一致性程度为非等效。

本部分是会话初始协议（SIP）系列标准之一。该系列标准的预计结构为：

1.《会话初始协议（SIP）技术要求》

——第1部分：基本的会话初始协议；

——第2部分：基于会话初始协议（SIP）的呼叫控制的应用；

——第3部分：ISDN用户部分（ISUP）和会话初始协议（SIP）的互通；

——第4部分：软交换网络中基于呼叫控制的会话初始协议（SIP）技术要求；

——第5部分：支持移动应用部分。

2.《会话初始协议（SIP）测试方法》

——第1部分：基本的会话初始协议；

——第2部分：基于会话初始协议（SIP）的基本呼叫控制。

3.《会话初始协议（SIP）服务器设备技术要求》。

4.《会话初始协议（SIP）服务器设备测试方法》。

《会话初始协议（SIP）技术要求 第1部分：基本的会话初始协议》将与《会话初始协议（SIP）测试方法 第1部分：基本的会话初始协议》配套使用。

随着技术的发展，还将制定后续的相关标准。

本部分的附录A、B、K是资料性附录，附录C、D、E、F、G、H、I、J是规范性附录。

本部分由中国通信标准化协会提出并归口。

本部分起草单位：信息产业部电信研究院

华为技术有限公司

中兴通讯有限公司

北京西门子通信网络股份有限公司

本部分主要起草人：蒋晓琳　魏　亮　李建芳　段世慧　林美玉　张大坤等

会话初始协议（SIP）技术要求
第1部分　基本的会话初始协议

1　范围

本部分规定了会话初始协议的技术要求，包括SIP消息、用户代理基本行为、请求取消、查询能力、对话、会话发起过程、会话更改过程、代理服务器行为、SIP事务层、传输、普通消息成分、头字段、响应代码、HTTP鉴权使用、S/MIME、SIP协议的扩展BNF、安全、IANA考虑等技术要求。

本部分适用于支持SIP协议的终端设备或网络。

2　规范性引用文件

下列文件中的条款通过本部分的引用而成为本部分的条款。凡是注日期的引用文件，其随后所有的修改单（不包括勘误的内容）或修订版均不适用于本部分，然而，鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本部分。

RFC1750（1994）　安全随机建议
RFC1847（1995）　MIME多部分安全格式：签名和加密
RFC2234（1997）　扩展的BNF语法：ABNF
RFC2279（2000）　SIP INFO 方法
RFC2327（1998）　会话描述协议
RFC2369（1998）　通用资源定位符（URI）通用语法
RFC2543（1999）　SIP协议
RFC2612（1999）　CAST-256 加密算法
RFC2616（1999）　超文本传输协议——HTTP/1.1
RFC2617（1999）　HTTP鉴权：基本鉴权和分类接入鉴权
RFC2630（1999）　加密消息语法
RFC2633（1999）　S/MIME 版本3消息规范
RFC2782（2000）　DNS指定服务位置（DNS SRV）
RFC2806（2000）　电话呼叫URL
RFC2809（2000）　通过RADIUS 实现L2TP强制隧道
RFC2822（2001）　Internet消息格式
RFC2976（2000）　SIP INFO 消息
RFC3108（2001）　在ATM承载连接上使用SDP的规程
RFC3261（2002）　SIP：会话初始协议
RFC3262（2002）　SIP协议临时应答的可靠性
RFC3263（2002）　会话初始协议：定位服务器
RFC3311（2002）　UPDATE方法

RFC3428（2002） 即时消息

RFC3515（2003） REFER方法

3 术语、定义和缩略语

3.1 术语和定义

下列术语和定义适用于本部分。

3.1.1 UA（User Agent）

用户代理（UA）是一个逻辑功能实体，当产生请求时作为UAC，接收请求并产生响应时作为UAS。

3.1.2 UAC（User Agent Client）

用户代理客户端（UAC）是一个逻辑功能实体，它产生一个新的请求消息后使用客户事务状态机将该请求发送出去。当终端产生一个请求时，在该请求事务中，它作为UAC。当终端接收到一个请求时，它作为UAS处理该请求并产生响应。

3.1.3 UAS（User Agent Server）

用户代理服务器（UAS）是一个逻辑功能实体，它对SIP请求消息进行处理并产生响应。该响应接受、拒绝或者重定向SIP请求消息。当终端对于接收到的请求消息作出响应时，在该请求事务中，它作为UAS。当终端产生请求时，在该请求事务中，它作为UAC。

3.1.4 B2BUA（Back-to-Back User Agent）

背靠背用户代理（B2BUA）是一个逻辑功能实体，它作为UAS接收请求消息并处理该消息。同时，为了判决该请求消息如何应答，它也作为UAC来发送请求消息。和代理服务器不同的是，B2BUA需要维护一个它所创建的对话状态。

3.1.5 重定向服务器

重定向服务器用响应消息将某一请求的路由信息返回给客户端，从而起到了帮助选路的功能。当请求的发起者收到重定向响应后，它将基于收到的URI发送新的请求。重定向服务器逻辑上由一个服务器端事务层和一个能够访问某种定位服务的事务用户组成。

3.1.6 注册服务器

注册服务器会为特定的区域创建绑定信息，即将一个地址记录URI与一个或多个地址相关联。

3.1.7 代理服务器

为客户机做代理，进行SIP消息的转接和转发。消息机制与UAC和UAS相似。并对收到的请求消息进行翻译和处理后，传递给其他的服务器，同时对SIP请求及响应进行路由。

3.1.8 对话

对话是两个UA之间持续一段时间的点对点的SIP连接，它使UA之间的消息变得有序，同时给出请求消息的正确的路由。

3.1.9 会话

会话是通信参与方及它们之间的媒体流的集合。

3.1.10 内核（Core）

内核指定了某个特定的SIP实体的功能集合。该SIP实体包括有状态代理、无状态代理、UA、注册服务器。除了无状态代理之外，所有的内核都是事务用户。

3.2 缩略语

下列缩略语适用于本部分。

DNS	Domain Name Service	域名服务
DoS	Denial of Service	拒绝服务攻击
HTTP	Hyper Text Transport Protocol	超文本传输协议
ICMP	Internet Control Message Protocol	互联网控制协议
IP	Internet Protocol	互联网协议
IANA	Internet Assigned Numbers Authority	互联网号码分配委员会
IETF	Internet Engineering Task Force	互联网工程任务组
MTU	Maximal Transmission Unit	最大传输单元
PSTN	Public Switched Telephone Network	公共交换电话网
RTP	Real-Time Transport Protocol	实时传输协议
RTCP	Real-Time Transport Control Protocol	实时传输控制协议
SCTP	Stream Control Transmission Protocol	流控传输协议
SDP	Session Description Protocol	会话描述协议
SIP	Session Initiation Protocol	会话初始协议
SSN	Security of Systems and Networks	安全服务网络
S/MIME	Secure / Multipurpose Internet Mail Extensions	安全多用途互联网邮件扩展
TCP	Transmission Control Protocol	传输控制协议
TLS	Transport Layer Security	传输层安全
UA	User Agent	用户代理
UAC	User Agent Client	用户代理客户
UAS	User Agent Server	用户代理服务器
UDP	User Datagram Protocol	用户数据报协议
URI	Universal Resource Identifier	通用资源标识

4 SIP 协议结构

SIP 是一个分层结构的协议，即它的行为根据一组平等独立的处理阶段来描述，每一阶段之间只是松耦合。协议分层描述是为了表达，从而允许功能的描述可在一个部分跨越几个实体。SIP 协议不指定任何方式的实现。当我们说某实体包含某层，我们是指它顺从该层定义的规则集。

SIP 协议框架的最底层是它的语法和编码层。SIP 协议的编码以 BNF 语法来定义。

SIP 协议框架的第二层是传输层。一个客户端事务通过传输层向服务器端事务发送一个请求，服务器端事务对该请求做出响应并发送给客户端，这样就构成了一个事务。事务层负责处理应用层的重传，响应匹配和超时。UAC 的任何任务都是通过一系列的事务而完成的。UA 作为有状态的代理服务器的时候包括事务层；无状态的代理服务器不包括事务层。事务层包括客户端部分和服务器端部分，并都以有限状态机表示。

SIP 协议框架的第三层为事务用户层（TU）。每个 SIP 实体都是一个事务用户。当 TU 要发送一个请求消息时，它首先创建一个客户端事务实例并将请求消息和目的 IP 地址、端口号发送给它。创建客户端事务的 TU 也可以将该事务取消。当一个客户端事务被取消时，它要求服务器端终止对该事务的处理并回

到初始状态，然后产生对该事务产生一个特定的错误响应。

SIP 实体，即用户代理客户端和服务器端、有状态及无状态的代理服务器和注册服务器，都包括一个内核相互区别，除了无状态的代理服务器，其他实体的内核都是事务用户。UAC 和 UAS 的内核的行为都决定于方法（method），对于所有的方法都有一个总的规则。对于 UAC，这些规则决定了请求消息的构成；对于 UAS，这些规则决定了请求的处理和响应的产生。由于注册在 SIP 协议中的重要地位，处理 REGISTER 消息的 UAS 在本协议中就称为注册服务器。

SIP 协议中，某些请求在对话中发送。本部分定义 INVITE 方法是惟一的可以创建对话的方法。

SIP 协议中最重要方法就是 INVITE 方法，它可以用来建立在双方间创建一个会话。

5 SIP 消息

SIP 协议是采用 UTF-8 字符集来进行编码的文本协议。

SIP 协议消息分请求和响应两类，其中请求消息由客户机发往服务器，响应消息由服务器发往客户机。除选用的字符集以及语法定义外，请求和响应消息均采用 RFC2822 定义的基本格式进行编码。请求和响应消息格式由一个起始行、若干个头字段，以及一个可选的消息体组成。其中消息体为可选项，头字段与消息体之间用空行进行分隔。请求和响应消息格式如下：

SIP 消息 = 起始行

*消息头部（1 个或多个头部）

CRLF（回车换行符）

[消息体]

起始行 = 请求行/状态行

如上消息格式定义，“*”表示该消息头部可包含一个或多个，“[]”表示该参数为可选项。本标准规定起始行、每一个消息头部以及空行都必须使用回车换行字符（CRLF）来表示行终结，即使消息中未包含消息体可选项空行也不能省略。

除了以上字符集不同之外，SIP 消息和头字段语法定义与 HTTP 1.1 的语法定义一致。HTTP/1.1 的语法定义见 RFC2612。消息语法定义与 HTTP 类似，SIP 协议并不是 HTTP 的扩展协议。

在SIP应用中使用的所有方法名字、头字段头字段名字、状态编码和选项标记都必须向IANA注册。具体的IANA定义参见附录B。

5.1 请求（Request）

请求消息的起始行为请求行（Request-Line）。请求行的格式由方法名、请求 URL 和协议版本组成，各部分之间均用一个空格字符进行分隔。除此之外，请求行必须用回车换行（CRLF）字符表示行终结。

Request-Line = Method[ ] Request-URI [] SIP-Version CRLF

1）方法（Method）：本部分正文共定义了 6 个方法，INVITE、ACK、CANCEL、OPTIONS、BYE 和 REGISTER。REGISTER 消息用于发送注册请求信息；INVITE、ACK、CANCEL 用于建立会话连接；BYE 用于终结会话连接；OPTIONS 用于查询服务器能力。本协议规定方法名必须使用大写字母。除以上 6 类主要消息外，SIP 协议在其他文档中还规定有若干方法实现协议扩展。本部分在附录 F，G，H.I，J 中定义了其他几个重要的扩展方法，如 NOTIFY/SUBSCRIBE，INFO，REFER 等，具体内容见相关附录。

2）请求 URI（Request-URI）：指示被邀请用户的当前地址。本协议规定 Request-URL 中不允许出现空格或其他控制字符且不能包含在“<>”符号之内。

除使用“sip”和“sips”URI之外，还可以使用tel URI定义机制，有关“tel”的URI定义机制见RFC2806。SIP实体可将非SIP URI翻译成SIP URI、SIPS URI或其他URI定义。

3）版本号（SIP-Version）：用于定义协议的当前版本号。本协议的版本号为SIP/2.0。

5.2 响应（Response）

响应消息的起始行为状态行（Status-Line），状态行由协议版本、状态码和与状态码相关的文本描述组成，各个部分之间用一个空格字符进行分隔。状态行的格式如下所示：

Status-Line = SIP-Version [ ]Status-Code [ ] Reason-Phrase CRLF

除状态行的尾部可使用回车换行CRLF字符之外，状态行内不允许出现CRLF字符。

1）Status-Code（状态码）：该参数为一个3位的十进制整数，用于指示请求消息的执行响应结果。

2）Reason-Phrase（原因）：该参数用于对Status-Code参数进行简单的文本描述。客户机不必检查或显示Reason-Phrase参数。尽管本标准建议使用特定字符表示Reason-Phrase，具体实现过程中Reason-Phrase仍可使用其他的文本字符。

本协议共定义6类状态码，其中状态码的第1位数字用于指示响应类型，后2位数字表示具体响应。本协议规定状态码为“100~199”之间的响应用“1XX”进行标识，“200~299”之间的响应用“2XX”进行标识，依此类推。

1）1XX：临时响应，表示请求消息正在被处理。

2）2XX：成功响应，表示请求已被成功接收，完全理解并被接收。

3）3XX：重定向响应，表示需采取进一步以完成该请求。

4）4XX：客户机错误，表示请求消息中包含语法错误信息或服务器无法完成客户机请求。

5）5XX：服务器错误，表示服务器无法完成合法请求。

6）6XX：全局故障，表示任何服务器无法完成该请求。

5.3 头字段

SIP头字段的语法和语义定义与HTTP头字段定义基本相同。HTTP中定义多行扩展可使用隐含的空格和折叠字符（folding），而本标准定义的多行扩展规则只能使用显式空格和折叠字符（folding），且空格和折叠字符（folding）作为消息的组成部分。

同样，HTTP 中也定义了将具有多个参数值的同一字段扩展为具有相同字段名称的多个字段行的规则。该规则同样适用于本协议，但具体应用时规则会有所不同。SIP协议定义的头字段语法规则如下：

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

如上所示，SIP头字段允许一个头字段可以定义多个参数值，且多个参数值之间用“,”字符进行分隔。当属性值不为“*”时，Contact头字段允许属性值之间用“,”字符进行分隔。

5.3.1 头字段格式

SIP消息的头字段格式遵循RFC2822所定义的通用头字段格式定义规则。头字段格式如下所示，头字段名和字段值之间用字符“:”进行分隔。

field-name：field-value

消息头字段中允许出现多个空格字符。但是在具体实现过程中，本标准建议字段头部右侧，即字段名和“:”字符之间应避免出现空格字符，而字段头部左侧，即字段值和“:”之间用一个空格字符进行分隔，如下所示：

field-name：[]field-value

头字段部分可以通过增加多个空白字符或者 TAB 字符而扩展成多行，本标准规定扩展行首位的多个空白字符以及分行字符都应作为一个空白字符对待。消息头字段内不同头字段名的顺序可任意排列，但是本标准建议与代理服务器处理相关的字段名（如 Via、Route、Record-Route 、Proxy-Require、Max-Forwards、Proxy-Authorization 等）等，应尽可能排在消息头字段的前几位以加快代理服务器对消息的处理速度。相同头字段名的排列顺序也非常重要，本标准规定，当且仅当字段值为多个字段值列表且字段值列表遵循多个字段值之间用“,”分割的规则，则一个消息内允许同时存在多个相同的字段名行。同理，多个相同字段名的字段行可以组成一个单一的字段行，其字段值为一个用“,”字符分隔的值列表。WWW-Authenticate、Authorization、Proxy-Authenticate 和 Proxy-Authorization 字段不遵循以上规则。由于以上字段值不允许出现列表值，因此当消息中出现多个以上字段行时，多个字段行不能组成一个字段行。

具体实现过程中，消息接收实体应按照同样的规则处理字段名相同的多个字段行或一个包含字段列表的组合字段行。

本标准中，头字段值由头字段名进行标识。通常头字段名由 UTF8 字符集所包含的文本字符组成。除此之外，头字段还允许包含空格、破折号、分隔符和引号字符。多数头字段名和头字段属性值之间用“:”进行分隔，且头字段中还可包含参数名和参数值，格式如下所示：

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

本标准对头字段中所包含的参数名数目不作限制，但规定在一个头字段内，同一参数名能且仅能出现一次。

本协议规定，头字段名对大小写字符不敏感，即相同字段名不区分大小写字符。如未特别指出，头字段中所包含的字段参数值，参数名和参数值也对大小写字符不敏感，除此之外，头字段名中所包含的破折号“——”字符对大小写字符不敏感，但是引号字符内的字符内容对大小写字符敏感。

5.3.2 头字段分类

SIP 消息某些头字段仅当存在于请求或响应中的时候才有意义。只能存在于请求消息中的字段称为请求头字段，只能存在于响应消息中的字段称为响应头字段。当消息头字段出现了不匹配的字段类型时，则消息接收实体应忽略该字段。

5.3.3 压缩格式

本协议提供了一种压缩机制用于将头字段名相同的字段进行压缩传送，这种压缩机制可以避免消息传输过程中产生大量消息拆分。采用压缩并不会导致消息语义的改变。本协议规定同一头字段名可出现在压缩头部或非压缩头部中，在具体实现过程中，消息接收实体应能识别并正确处理这两种不同格式的消息。

5.4 消息体

本协议规定 SIP 请求消息可包含消息体部分，消息体部分的解释应与消息请求方法相一致。对于 SIP 响应消息，请求方法和响应状态码可以识别消息体的类型。本协议规定所有 SIP 响应消息应包含一个消息体部分。

5.4.1 消息体类型

消息体的类型必须由“Content-Type”字段进行定义，且如果消息体采用了压缩编码方式，则相应地应在“Content-Encoding”字段中定义其所采用的压缩编码算法。否则，如果消息体未采用了压缩编码方

式，则“Content-Encoding”字段的内容应被忽略。在具体应用过程中，消息接收实体应将消息体的内容作为“Content-Type”头字段值来对待。

本协议规定消息体可以承载使用“multipart”MIME方式进行编码的信息单元。具体实现中，当请求发送方需要发送消息体部分包含MIME信息单元的请求消息时，且消息接收方在“Accept ”头字段指示其不能接受MIME消息体，则消息请求发送方应在消息中附加一个SDP部分作为非MIME消息体进行发送。SIP 消息可以包含二进制编码的消息体。如果未明确指出，本标准中缺省的消息文本编码类型是UTF-8。

5.4.2 消息体长度

消息体的长度由“Content-Length”头字段进行定义，单位为字节。有关“Content-Length”头字段见本部分第18章。

本协议规定SIP消息体不能采用HTTP1.1中所定义的分块（chunked）传送编码机制。在分块传送编码机制中，每一块消息体的长度由相应的标识符进行标识。

5.5 SIP消息帧

SIP协议可以使用UDP或者其他不可靠的报文协议进行承载传送，且每一个报文携带一个请求或响应消息。

具体实现时，如果SIP消息采用面向流的方式进行传送，则SIP消息起始行前的任何CRLF字符应忽略。

“Content-Length ”头字段值用于定义一个SIP消息在流中的结束位置。当SIP消息采用面向流的方式进行传送时，该头字段不能被省略。

6 用户代理（UA）的基本行为

当UAC发出请求，请求消息会通过一些代理服务器被转发到UAS。而UAS产生一个响应后，响应消息会以同样的方式被转发到UAC。

UAC和UAS的处理过程与两个因素有关，即被处理的请求或响应消息是否属于某个对话（dialog）以及请求的方法。对话是一种UA之间的端到端对等关系，它由特定的SIP消息，如INVITE消息建立。关于对话定义见本标准第10章。

对话之外的请求或响应消息的安全性处理参见本部分附录A。另外，UAC和UAS之间存在着相互鉴权机制。消息体可以通过S/MIME进行加密。

6.1 UAC基本行为

本节讨论对话之外的UAC行为。

6.1.1 请求消息的产生

本标准规定，由UAC产生的一个有效的SIP请求消息必须至少包含下列头字段：To、From、CSeq、Call-ID、Max-Forwards和Via头字段，这些头字段在所有的SIP请求消息都是必选的。这6个头字段是构建SIP消息的基本单元，它们共同提供了大部分的关键的消息路由服务，包括消息的寻址、响应的路是由、消息传播距离限制、消息排序，以及事务交互的惟一性标识等。另外，请求行（request line）也是必选的，它包含了请求方法、Request-URI、SIP版本信息。

在对话外发送的请求消息包括：用INVITE消息建立起一个会话，用OPTIONS消息进行能力查询等。

6.1.1.1 Request-URI

除 REGISTER 请求，本协议中所有的请求消息的初始 Request-URI 都应被设为 To 头字段中的 URI 值。另外，出于安全性考虑，UA 有时可能不希望将这两个头字段设为同样的值，尤其是当发端 UA 知道 Request-URI 可能会在传输中改变。

某些情况下，预设路由集的存在能影响到消息的 Request-URI。预设路由集是一个有序的 URI 集合，它标识了一个服务器链，UAC 向这个服务器链发出对话外请求消息。预设路由集通常由用户或服务供应商在 UA 上手工配置，或者通过某些其他的非 SIP 机制配置。本部分建议，在给某个 UA 配置一个外拨代理服务器时服务供应商所提供的预设路由集中只有一个 URI，即外拨代理服务器的 URI。

当存在预设路由集时，必须遵循本标准 12.2 节中描述的过程来设置 Request-URI 和 Route 头字段，其 Request-URI 作为远端目标 URI。

6.1.1.2 To 头字段

To 头字段指定请求消息的逻辑接收者或者是用户或资源的注册地址，该地址同样是作为请求消息的目标地址。To 头字段所指示的不一定为请求消息的最终接收者。它可能包含一个 SIP 或 SIPS URI，或其他的 URI 方案。本标准规定，所有的具体实现过程都必须支持 SIP URI 方案，任何支持 TLS 的实现必须支持 SIPS URI 方案。To 头字段中允许包含一个显示名称。

UAC 可通过多种方式来构造一个请求消息的 To 头字段。通常经过人机接口，由用户输入或者从地址簿中选取。用户通常不会输入一个完整的 URI，而只是提供一个由数字或字母组成的字符串，由 UA 判断并解释该字符串。如果这个字符串被用来构造一个 SIP URI 的用户部分，则用户名称在@符号右侧所示的主域中被解析；如果这个字符串被用来构造一个 SIPS URI 的用户部分，则用户希望进行安全的通信，同时用户名将在@符号右侧所示的主域中被解析。SIP URI 中@符号右侧通常是请求发起方的主域，它使本地主域能够处理外发请求。此外，如果用户输入的是一个电话号码，且 UA 不会指定由某个主域来解释该号码，这时可以使用 tel URL，从而使请求消息所经过的每一个主域都可以处理它。

请求消息的 To 标签标识了一个对话中的对端，如果对话没有建立，标签就不应当出现。对话之外的请求消息中不可以包含 To 标签（tag）。

关于 To 头字段见本标准 18 章。

6.1.1.3 From 头字段

From 头字段是指示请求发起方的逻辑标识，它可能是用户的注册地址。From 头字段包含一个 URI 和一个可选的显示名称。SIP 实体用它来决定如何处理一个请求（如呼叫自动拒绝）。由于不是逻辑名称，因此 From URI 不包含 IP 地址或 UA 所在主机的全称域名（FQDN）。

From 头字段中允许包含一个显示名称。如果一个 UAC 需要隐藏自己的身份，它可以使用“Anonymous”作为显示名称和一个语法正确但没有任何意义的 URI。

通常，某个 UA 产生的请求消息中的 From 头字段值是由用户或用户本地主域的服务器预先设置的。如果 UA 被多个用户所共用，那么它可以有多个可切换的用户配置文件，每一文件中含有某一用户的 URI。请求消息的接收者要对发送者进行鉴权，以确认发送者身份与 From 头字段相一致。

From 头字段中必须包含一个新的由 UAC 选定的 “tag” 参数。

关于 From 头字段见本标准 18 章。

6.1.1.4 Call-ID 头字段

Call-ID 头字段是用来将消息分组的惟一性标识。本协议规定，在一个对话中，UA 发送的所有请求

消息和响应消息都必须有同样的 Call-ID。在注册生存期内，一个 UA 每次注册所用的 Call-ID 也应是一样的。

当 UAC 产生一个新的对话外请求时，除非被某些方法指定，否则它必须为这个请求消息选择一个在空间上和时间上都是全局惟一的 Call-ID 头字段。所有的 SIP UA 都必须保证它所产生的 Call-ID 不会与其他 UA 所产生的相同。当 UA 收到某些失败的响应后，请求会根据响应的内容修改并重发，这些重发的请求不作为新请求处理，因而也就不需要新的 Call-ID 头字段。

本标准建议使用RFC1750中的加密随机标识符生成方法来生成Call-ID。具体实现可采用localid@host的形式。Call-ID 对大小写敏感，需逐字节的进行比较。加密随机标识符方法在一定程度上能防范黑客的会话攻击，并降低了无意中产生 Call-ID 冲突的可能性。Call-ID 的选择不需要通过人机接口和预先设定值来实现。

关于 Call-ID 头字段见本标准 18 章。

6.1.1.5 Cseq 头字段

CSeq 头字段用于标识事务并对事务进行排序。它由一个请求方法和一个序列号组成，请求方法必须与对应的请求消息类型一致。对话外的非 REGISTER 请求，序列号值可以是任意的。但它必须可被表示成一个 32 位的无符号整数，且 $<2^{31}$。只要符合以上规则，客户端可以用任何方式来选择 Cseq 头字段值。

6.1.1.6 Max-Fowords 头字段

Max-Fowords 头字段限定一个请求消息在到达目的地之前允许经过的最大跳数。它包含一个整数值，每经过一跳，这个值就被减一。如果在请求消息到达目的地之前该值变为 0，那么请求将被拒绝并返回一个 483（跳数过多）错误响应消息。

UAC 必须在它发起的每个请求中都插入 Max-Fowords 头字段，值为 70。在任何不出现回路的 SIP 网络中，选择该值为 70 足够大的保证一个请求消息不被丢弃，且在有回路的情况下，这个值也不会太大而过分浪费代理服务器的资源。UA 只有知道网络的拓扑结构时，才可以谨慎地选择更小的跳数值。

6.1.1.7 Via 头字段

Via 头字段定义 SIP 事务的下层（传输层）传输协议，并标识响应消息将要被发送的位置。只有当到达下一跳所用的传输协议被选定后，才能在请求消息中加入 Via 头字段值。

本协议规定，当 UAC 生成请求消息时，它必须在其中插入一个 Via 头字段。Via 头字段的协议名称和协议版本必须分别为“SIP”和“2.0”。Via 头字段中必须包含一个“branch”参数，该参数用来标识由当前请求所建立的事务。该参数既用在客户端也用在服务器端。

对于某个 UA 发出的所有请求，它们的 branch 参数值在空间和时间上必须全局惟一的。但有两种情况例外：一是 CANCEL 请求，以后会说明 CANCEL 请求的 branch 参数与它所要取消的那个请求的 branch 参数是一样的；另一个是对非 2xx 响应的 ACK 请求，这种情况下 ACK 请求与相关的 INVITE 请求有着同样的 branch ID，它所要确认的就是该 INVITE 的响应。

SIP 实体在插入 branch ID 时，必须以"z9hG4bK"开头。这 7 个字符是“magic cookie”，这样 SIP 服务器在收到请求消息时，就能确定现在的 branch ID 是全局惟一的。另外，branch ID 参数的准确格式由具体的实现定义。

Via 头的 maddr、ttl 以及 sent-by 部分在传输层对请求消息进行处理时进行设置，见本标准第 16 章。

关于代理服务器对 Via 头字段的处理见本标准 13 章。

6.1.1.8 Contact 头字段

Contact 头字段指定一个 SIP 或 SIPS URI，后续请求可以用它来联系到当前 UA。任何能够建立对话的请求消息中都必须有 Contact 头字段，并且该头字段中只能含有一个 SIP 或 SIPS URI。在本标准定义的请求方法中，只有 INVITE 能建立对话。对这些能建立对话的请求，Contact 的作用范围是全局的。也就是说，Contact 头字段值中包含的 URI 是 UA 希望用来接收请求的地址，即使用在任何对话外的后续请求消息中，该 URI 也必须有效。

如果请求消息的 Request-URI 或顶端 Route 头字段值中包含了 SIPS URI，那么在 Contact 头字段中也必须包含一个 SIPS URI。

关于 Contact 头字段见本标准 18 章。

6.1.1.9 Supported 和 Require 头字段

如果 UAC 支持某些 SIP 协议的扩展，并且这些扩展可被服务器用来构成请求的响应，那么 UAC 应在请求消息中包含一个 Supported 头字段，列出这些扩展的选项标签。

只有在本标准和后续扩展所对应的选项标签能够在 Supported 头字段中列出。

如果 UAC 要求 UAS 必须理解某项扩展以便处理它所发出的请求，它必须在请求消息中插入一个 Require 头字段，并列出此扩展的选项标签。如果 UAC 希望在请求中使用某项扩展，并要求请求消息经过的所有代理服务器都能理解此扩展，它必须在请求消息中插入一个 Proxy-Require 头字段，并列出此扩展的选项标签。

6.1.1.10 消息的其他部分

在新的请求消息生成且上述头字段被正确构造之后，可加入任何其他的可选头字段和请求方法所要求的特定头字段。

SIP 请求消息中可包含一个按 MIME 格式编码的消息体。无论请求中包含的消息体类型如何，都必须构造某些头字段以表征消息体的内容。这部分头字段的内容见本标准 18 章。

6.1.2 请求消息的发送

首先确定请求消息的发送目的地。除非另有本地策略，目的地址必须按照 DNS 定位过程确定：如果 Route 头字段中路由集的第一个实体是个严格的路由器，DNS 的过程必须被应用于请求消息的 Request-URI；否则，该过程应用于请求消息中的第一个 Route 头字段值，若没有 Route 头字段出现则该 DNS 过程仍然应用于请求消息的 Request-URI。其输出结果是个可以发送的目标地址有序集，其中每一项是由地址、端口号以及传输协议构成的三元组。不论用哪个 URI 作为 DNS 寻址过程的输入，如果 Request-URI 指定一个 SIPS 资源，UAC 就必须把这个 URI 当作 SIPS URI 来解析。具体定位过程见附录 D。

本地策略可指定其他目标地址集。如果 Request-URI 中包含的是 SIPS URI，那么同该地址集中的任何目的地址联系都必须使用 TLS。此外，如果请求消息中不含 Route 头字段，那么对该地址集中的地址没有其他限制。当指定一个外拨代理服务器时，这样做比预设路由集简单。但是，本部分建议用只含单个 URI 的预设路由集来指定外拨代理服务器。如果请求消息中含有 Route 头字段，那么请求消息应当发送到最顶端 Route 头字段值所指示的位置；也可发送到任何其他的服务器。只要 UA 确认这些服务器遵循本部分中对 Route 和 Request-URI 的处理策略。本标准建议，配置了外拨代理服务器的 UAC 应该尝试将请求发往第一个 Route 头字段值所指的位置，而不是将所有消息都发往外拨代理服务器。

UAC 应当尝试连接目标地址集中的每个地址，直到联系上某个服务器。每次尝试都会建立新的事务，

因而每个新请求的最顶端Via头字段值都不同，其中包含着新的branch参数。此外，Via头字段中的transport值也要根据不同的目标服务器来设置。

6.1.3 响应消息处理

响应消息先由传输层处理，然后上传到事务层。经事务层处理后再将其上传给事务用户（TU）。TU对响应消息的处理主要依赖于请求方法，但是一些基本处理方法则同具体的请求方法无关。

6.1.3.1 事务层错误

某些情况下，从事务层返回的并不是SIP消息，而是事务层错误消息。当TU从事务层收到超时错误消息时，对该消息的处理必须同408（请求超时）状态码。如果事务层报告的是严重错误，通常是UDP方式下的严重ICMP错误或TCP连接失败所致，则该消息的处理同503（业务不可用）状态码。

6.1.3.2 无法识别的响应消息

对无法识别的最终响应，UAC必须将之等价于所属响应类别的x00响应码，且UAC必须能够处理所有响应类别的x00响应码。例如，如果UAC收到了一个无法识别的响应码431，那么它能断定自己发出的请求消息出错，因而对该431码的处理同400（错误请求）响应码。但对于任何无法识别的非100临时响应的处理必须同183响应（会话处理中）。UAC必须能够处理100和183响应。

6.1.3.3 多Via头字段值

如果某个响应消息中出现了多个Via头字段值，UAC应当丢弃该响应。

在标识请求发起者的Via头字段值之前出现了其他的Via头字段值，表明消息在传送过程中发生了路由错误，或者已被破坏。

6.1.3.4 3xx响应码处理

客户端在收到重定向响应时，应基于相应的重定向的请求消息的Contact头字段中的URI来构造一个或多个新的请求消息。这与16.5和16.6节中代理服务器递归处理3xx类响应的过程类似。客户端的目标地址集中最初只有一个URI，即初始请求的Request-URI。如果客户端要根据初始请求的一个3xx类响应构造新的请求消息，它应把可能的URI放入目标地址集中。UAC在遵循本部分定义的限制条件下，可以选择Contact头字段中一些URI放入目标地址集。同代理服务器进行递归处理时一样，客户端在处理3xx类响应时不可以将任何URI多次加入到目标地址集中。如果初始请求消息的Request-URI为SIPS URI，客户端可选择到非SIPS URI地址的递归处理，同时应当通知用户其请求被重定向到不安全的URI。

任何新请求的发出都可能收到3xx响应，这些响应的又包含了原始请求的Request-URI。这是因为两个地址有可能被配置成互为重定向目标。所以为了防止出现重定向循环，规定任何URI只能在目标地址集中放置一次。

随着地址集中条目的增长，客户端可按任何次序来产生到这些URI地址的新请求。通常是按Contact头字段值中q值的大小对地址集中的条目进行排序。请求消息可以串行或并行生成。一种做法是按q值的降序分组后串行处理，同组的URI并行处理。另一种方法是只按q值的降序串行处理，q值相等的则次序任选。

如果与目标地址集中的某个地址联系失败，UA应继续试下一条地址，直到所有的地址都用完。如果所有的地址都已用完，即告请求失败。

请求失败应当通过失败响应码来检测，响应码应大于399。对于网络错误，客户端事务将向事务用户（TU）报告传输层失败。有些响应码指示请求可以重发，重发的请求不应视为失败。

当客户端与一个地址的联系失败后，它应继续联系下一个地址。这将创建一个新的事务来发送新的请求。

如果 UAC 基于 3xx 响应码中的某个 Contact 地址来创建新请求，就必须从目标地址集中把相应的 URI 除了 method-param 和 header 外全部拷贝到 Request-URI 中。Header 参数可以用于创建新请求的某些头字段值并重写重定向请求的某些头字段值。

某种情况下，Contact 地址中所携带的头字段可附于最初的重定向请求的现有头字段之后。本协议规定，如果某个头字段能接受逗号分隔的值列表，那么新的头字段值可被附于原始请求的现有值之后；如果某个头字段不接受多值，则原始请求的现有值可被 Contact 地址中携带的头字段值重写。例如：如果 3xx 响应码中返回的 Contact 地址带有下列值：

sip：user@host?Subject=foo&Call-Info=〈http：//www.foo.com〉

那么原始请求的每个 Subject 头字段都将被重写，但那个 HTTP URI 只是被附加到现有的 Call-Info 头字段值之后。

本标准建议，UAC 可以重用原始请求的 To、From 和 Call-ID 头字段。

如果新的请求消息构建完成，它将在一个新的事务中发送。顶端 Via 字段中必须有一个新的 branch ID 参数。

本协议规定，对于所有未述及部分，在收到重定向响应之后发送的新请求应重用原始请求的头字段和消息体。

某种情况下，UAC 会根据收到的响应码以及 Contact 头字段的有效期来临时或永久地缓存某些 Contact 头字段值。具体规则见本标准 19 章。

6.1.3.5 4xx 响应消息处理

某些 4xx 响应码需要特定 UA 的处理，处理方法与请求方法无关。

本协议规定，UAC 在收到 401（未授权）或者 407（代理服务器需要鉴权）响应码时应按依据相应的授权过程来重发携带鉴权证书的请求（见本部分第 20 章）。

如果收到的是 413（请求消息过大）响应，说明请求中包含的消息体超过了 UAS 所能接收的长度。在可能的情况下，UAC 应当去掉消息体或减小它的长度。

如果收到的是 415（不支持的媒体类型）响应，说明请求中包含了 UAS 不能支持的媒体类型。UAC 应重发请求，并只用响应消息的 Accept 头字段中列出的类型、Accept-Encoding 头字段中列出的编码方式和 Accept-Language 头字段中列出的语言种类。

如果收到的是 416（不支持的 URI 方案）响应，说明请求中的 Request-URI 使用了 UAS 不支持的 URI 方案。UAC 应当将其改为 SIP URI 后重发该请求。

如果收到的是 420（错误的扩展）响应，说明请求的 Require 或 Proxy-Require 头字段中列出了某个选项标签，该标签所对应的扩展特性不能被 UAS 或代理服务器所支持。此时，UAC 应当去掉在响应消息的 Unsupported 头字段中列出的所有不支持的扩展，然后重发请求。

所有上述情况下，请求消息重发都是通过对原请求作适当修改之后创建一个新请求来完成的。新请求建立了一个新的事务，并且与原请求有相同的 Call-ID、To 和 From。但 CSeq 应当包含一个新序列号值，它比原请求中 CSeq 的序列号大一。

对其他 4xx 响应，包括那些将要定义的，请求消息重发是否可能要依赖于具体的请求方法和应用场

合。

6.2 UAS 基本行为

UAS 处理对话外请求时，其处理规则与请求方法无关。

如果请求被接受，那么与之相关的所有状态改变都必须被执行；如果请求被拒绝，则所有状态变化全部禁止。

UAS 应当按下述步骤处理请求：即从鉴权开始，然后检查请求方法，检查各个头字段等。

6.2.1 请求方法检查

在请求的鉴权完成后，UAS 必须检查请求的方法。如果 UAS 认识但不支持某个请求方法，它必须产生一个 405（不允许的请求方法）响应。UAS 还必须在此 405 响应中加入一个 Allow 头字段，其中列出所有它所支持的请求方法。

如果 UAS 能够支持请求的类型，则处理继续进行。

6.2.1.1 消息头检查

如果 UAS 不理解请求消息中的某个头字段（即该头字段未在本标准或任何它所支持的扩展中定义），它必须忽略这个头字段并继续进行处理。此外，UAS 应当忽略任何不影响请求处理的格式错误的头字段。

6.2.1.2 To 和 Request-URI 头字段

To 头字段标识了请求消息的原始接收者，它由 From 头字段所标识的用户指定。由于呼叫转发或其他代理服务器操作，请求的原始接收者不一定是最终处理该请求的 UAS。当 To 头字段所标识的接收者不是自己时，UAS 可应用任何策略来决定是否接受该请求。本标准建议，即使 UAS 不能识别请求消息中 To 头字段的 URI 编码方案，或不知道该头字段所要寻址的用户，或该用户并非它的当前用户，UAS 仍然要接收这些请求消息。另外，如果 UAS 决定拒绝某个请求，那么它应当产生一个 403（禁止）响应，并将该响应传递给服务器端的事务发送。

请求消息的 Request-URI 标识了将要处理该请求的 UAS。如果请求消息的 Request-URI 使用了 UAS 所不支持的 URI 方案，UAS 应用一个 416 响应（不支持的 URI 方案）拒绝该请求。如果请求消息的 Request-URI 标识的不是 UAS 想要用来接收请求的地址，UAS 应当用一个 404（未找到）响应拒绝该请求。一般来说，UA 用 REGISTER 请求把自己的注册地址与某个特定的联系地址绑定到一起，其后的请求消息的 Request-URI 等于该联系地址。另外，Request-URI 也源于 UA 建立或更新对话时所发出的请求或响应中的 Contact 头字段。

6.2.1.3 请求消息的合并

如果请求消息的 To 头字段中没有标签，此时，UAS 必须针对该请求检查正在进行的事务。如果请求的 From 标签、Call-ID 和 Cseq 与某个正在进行的事务的有关信息相匹配，但依据本部分 18 章所述的匹配规则，该请求又确实与该事务不匹配，则 UAS 应产生一个 482（检测到路由循环）响应并传递给服务器端的事务进行处理。

6.2.1.4 Require 头字段

当 UAS 认为自己可以处理某一请求时，它将检查请求消息中出现的 Require 头字段。

UAC 用 Require 头字段来通知 UAS 它所希望该 UAS 能够支持的 SIP 扩展，以便 UAS 能正确处理它所发出的请求消息。有关 Require 头字段的格式见本标准 18 章。如果 UAS 不能理解 Require 头字段中所列出的扩展选项标签，它必须回以 420（错误的扩展）响应。同时必须在该响应中加入一个 Unsupported

头字段，以列出请求的 Require 头字段中它不支持的那些扩展选项。

本协议规定，Require 头字段和 Proxy-Require 头字段不能用在 CANCEL 请求中，也不可用在非 2xx 响应的 ACK 请求中。如果这些请求中出现了这两个头字段，该请求必须被忽略。

2xx 响应的 ACK 请求中只能包含对应的原始请求中出现的那些 Require 和 Proxy-Require 值。

UAS 的这种处理方式保证了当会话双方都能理解所有的扩展选项时，客户与服务器间交互可以无延时地顺利进行，而如果扩展选项不被理解则会导致交互过程减慢。对于一个配合良好的客户机—服务器对，理解扩展选项之后省去了能力协商机制，因而交互可以快速进行。此外，当 UAC 对 UAS 提出其不支持的特性时，它使双方的交互和表达能够准确清晰地进行。

6.2.2 消息内容处理

如果 UAS 理解客户端要求的任何扩展，它将检查消息体和描述消息体内容的消息头字段。如果 UAS 不理解消息体中任何由 Content-Type 指示的类型、由 Content-Language 指示的语言或由 Content-Encoding 指示的编码，且对该部分消息体的处理是必选的（如 Content-Disposition 头字段所示），那么 UAS 必须用一个 415（不支持的媒体类型）响应拒绝请求。如果请求中包含了 UAS 不支持的消息体类型，响应消息中必须包含一个 Accept 头字段，其中列出它能理解的所有消息体类型。如果请求中包含了 UAS 不理解的编码方式，响应消息中必须包含一个 Accept-Encoding 头字段，其中列出它能理解的编码方式。如果请求中包含了 UAS 不理解的语言，响应消息中必须包含一个 Accept-Language 头字段，其中列出它能理解的语言。除此之外，消息体的处理和请求的方法以及消息体的类型有关。

6.2.3 扩展的应用

如果 UAS 在响应时要应用某些扩展，那么相应的请求的 Supported 头字段中必须已指明客户端支持这些扩展。如果客户端不支持需要的扩展，UAS 只能依赖基本的 SIP 协议和客户端支持的扩展来达到目的。极个别情况下如果没有所需的扩展支持则请求无法处理，此时 UAS 可发出 421（需要扩展支持）响应。该响应指出如果不支持某一特定扩展就无法产生正确的响应，该响应的 Require 头字段中必须包含需要支持的扩展选项。但是由于这样可能破坏互操作性，因此本部分不建议采用这种方式。

对非 421 响应要用的任何扩展也必须在该 421 响应的 Require 头字段中列出，并且不能使用请求的 Supported 头字段中未列出的扩展。这样，响应消息的 Require 头字段将只能包含标准的后续标准中定义的扩展选项标签。

6.2.4 请求的处理

如果上述检查全部通过，接下来 UAS 的处理就与具体的请求方法有关。

6.2.5 产生响应

当 UAS 要遵循下列章节的流程构造某个请求的响应时，可能还需要某些特定响应码相关的处理，但本节不作描述。

UAS 在完成创建响应的所有流程之后，应把响应消息传递给发起请求的服务器端的事务。

6.2.5.1 发送临时响应

UAS 不应考虑对非 INVITE 请求发送临时响应，而是应尽快对其产生最终响应。

当 100（正在尝试）响应产生后，请求消息中出现的任何 Timestamp 头字段都必须被复制到该响应中去。如果响应产生时出现了延时，UAS 应当在响应的 Timestamp 值上附加一个延时值。这个延时值必须是从收到请求开始到发出响应为止的以秒计算的时间间隔。

6.2.5.2 头字段和标签

响应消息的 From 头字段、Call-ID 头字段和 Cseq 头字段必须分别与被响应的请求的 From 头字段、Call-ID 头字段和 Cseq 头字段相等。本标准要求，Via 头字段不但值必须要相等，各个值之间的顺序也必须保持一致。

如果请求消息中包含 To 标签，那么响应消息的 To 头字段必须与请求消息的 To 头字段相等。然而，如果请求消息的 To 头字段不包含标签，那么响应消息中 To 头字段的 URI 必须与请求消息中 To 头字段的 URI 相等；此外，UAS 必须在响应消息的 To 头字段中添加一个标签（100 响应例外），这是为了标识正在应答的 UAS，也许会成为一个对话 ID 的组成部分。同一个请求的所有响应必须使用同一个标签，包括临时响应和最终响应（100 响应例外）。

6.3 重定向服务器

重定向服务器自己不发送任何 SIP 请求。在收到除 CANCEL 以外的请求时，重定向服务器可以拒绝它，或者通过定位服务获得一个可选地址列表并返回一个 3xx 最终响应。对格式正确的 CANCEL 请求，重定向服务器应返回 2xx 响应。该响应将结束被取消请求的 SIP 事务。重定向服务器维护整个 SIP 事务的状态。客户端检测重定向服务器之间发生的转发循环。

当重定向服务器返回某个请求的 3xx 响应时，它在 Contact 头字段中装入一个地址列表（含一个或多个可选地址）。Contact 头字段值中还可能提供“expires”参数，以指明 Contact 数据的有效期。

Contact 头字段包含了可供发送的 URI，这些 URI 给出了新的目标位置或新的用户名，或者只简单地指定了一些其他的传输参数。301（永久移动）或 302（临时移动）响应可能会给出与初始请求一样的目标位置和用户名，同时又指定另外的传输参数，比如一个不同的服务器地址或多播地址，或者将 SIP 消息的传输方式从 UDP 改为 TCP 或从 TCP 改为 UDP，等等。

然而，重定向服务器决不能将请求重定向到一个与请求的 Request-URI 相同的 URI 地址。如果重定向的目的地 URI 所指并非当前重定向服务器，它可能将请求向重定向目的地转发，或用一个 404（未找到）响应来拒绝请求。

Contact 头字段值也可能指示与原始被叫不同的资源位置。例如，一个连接到 PSTN 网关的 SIP 呼叫可能需要发送一个特定的录音通知。

响应消息的 Contact 头字段可能包含任何指示被叫位置的 URI 类型，而并不仅限于 SIP URI。

Contact 头字段值的 expires 参数指出了该值中包含的 URI 地址的有效期。这个参数的值是以秒为单位计算的。如果没有提供该参数，那么 URI 地址的有效期由 Expires 头字段值来确定。格式错误的有效期值应被视为 3600 来处理。RFC2543 中允许 Expires 头字段包含绝对时间。如果收到了用绝对时间表示的有效期值，它将被认为格式错误并缺省地用 3600 代替。

如果请求中包含不能理解的内容（包括不可识别的头字段，Require 中任何未知的选项标签，甚至请求方法名等），重定向服务器必须忽略它们并继续进行重定向处理。

7 请求的取消

CANCEL 用于取消客户端发送的前一个请求。特别是当要求 UAS 终止对被取消请求的处理，并对那个请求产生一个错误响应。对于 UAS 已经给出了最终响应的请求，CANCEL 是无效的。因此，CANCEL 主要用于取消服务器端需要长时间才能给出响应的请求。CANCEL 请求最适合于 INVITE 请求，因为它会需要很长的时间来产生响应。在这种情况下，如果 UAS 收到了针对 INVITE 的 CANCEL 请求，但还没有为之发出最终响应，则 UAS 将"停止振铃"，然后用一个特定的错误响应（487）来应答 INVITE。

代理服务器和 UAC 都可以产生并发送 CANCEL 请求。

代理服务器应当应答收到的 CANCEL，而不是简单地转发从下游实体收到的响应。因此，由于 CANCEL "跳" 到每一个的代理服务器时都会被应答，所以它是个 "逐跳"（hop-by-hop）的请求。

7.1 客户端行为

CANCEL 请求不应当用于取消非 INVITE 请求。

下列过程用于构造一个 CANCEL 请求消息：CANCEL 请求的 Request-URI、Call-ID、To、CSeq 的数字部分，以及 From 头字段，包括各个标签，都必须与被取消请求的对应部分相同。客户端产生的 CANCEL 必须只能有一个 Via 头字段值，并与被取消请求的顶端 Via 值相匹配。这些头字段值的一致性使得 CANCEL 请求能与被取消的请求相匹配。但是，Cseq 头字段中"method"部分的值必须是"CANCEL"，以保证 CANCEL 请求能够被识别并在一个它自己的事务中被处理。

如果被取消请求包含一个 Route 头字段，CANCEL 请求中必须包含此 Route 头字段的值。这个要求是为了让无状态代理服务器能够正确地路由 CANCEL 请求。

CANCEL 请求不能包含任何 Require 或 Proxy-Require 头字段。

一旦 CANCEL 请求构造完毕，客户端应当检查是否已收到了被取消请求（下文将统称为原请求）的任何响应消息。

如果还没有收到临时响应消息，则 CANCEL 请求不能发送，而是一直等待一个临时响应到来。如果原请求已经产生了一个最终响应，由于 CANCEL 对已经产生了最终响应的请求是无效的，那么 CANCEL 请求也不应被发送。当客户端决定要发送 CANCEL 时，它为该请求创建一个客户端事务，并将 CANCEL 消息和目的地址、端口、以及传输方式一起传递给新创建的事务。CANCEL 请求的目的地址、端口和传输方式必须与原请求的相同。

如果允许在收到响应消息之前发送 CANCEL 请求，那么服务器端有可能在收到原请求之前先收到 CANCEL 请求。

对应于原请求的事务和 CANCEL 请求的事务都将独立完成。但是，一个执行请求取消的 UA 不能依赖于收到原请求的 487 响应消息来进行后续处理，UAS 不会产生 487 响应。如果在 64×T1 秒内没有收到原请求的最终响应，那么 UAC 应认为原请求已经被取消，并终止原请求的事务处理。

7.2 服务器端行为

CANCEL 方法请求服务器端的事务用户（TU）取消一个未决的事务。事务用户假定原请求的方法是除了 CANCEL 或 ACK 之外的任何一个，并进行事务匹配，匹配到的事务将被取消。

服务器如何处理 CANCEL 请求依赖于服务器的类型。无状态代理服务器将简单地转发请求；代理服务器会对其做出应答并产生自己的 CANCEL 请求；UAS 将对 CANCEL 请求做出应答。

UAS 首先按 13 章的基本规则对 CANCEL 进行处理。由于 CANCEL 请求是逐跳处理的并且不能重传，

所以 UAS 不能对它们发出质询（Challenge）来得到在 Authorization 头字段中携带的证书（Credentials）。另外，CANCEL 请求不包含 Require 头字段。

然后，UAS 对 CANCEL 请求进行事务匹配。如果依据上述的过程没有找到匹配的事务，那么 UAS 应当用一个 481 响应（事务或呼叫方不存在）来应答 CANCEL 请求。如果匹配到了原请求的事务，那么 UAS 的处理行为将取决于是否已经发送了对原请求的最终响应。如果响应已经发送，那么 CANCEL 请求不影响对原请求的处理，也不影响任何会话状态和原请求的响应。如果最终响应没有发送，那么 UAS 的处理方式将取决于原请求的类型。如果原请求是一个 INVITE，UAS 应立即对它发送 487（请求终止）响应。而对本标准中定义的任何其他的请求方法，CANCEL 请求对它们的事务处理都不起作用。

不管原请求的类型是什么，只要 CANCEL 匹配到了一个现存的事务，UAS 都要用一个 200（OK）响应来应答 CANCEL 请求。CANCEL 响应的 To 标签应当与原请求响应的 To 标签相同。CANCEL 的响应消息被传递给一个服务器端事务以便发送。

8 注册

8.1 概述

SIP 协议中应用发现机制，如果一个用户想和另一个用户建立一个会话，必须首先发现用户当前使用的终端设备，这种发现过程通常是由 SIP 的实体设备实现的，比如代理服务器、重定向服务器，它们负责接收请求，然后根据所学习到的用户地址信息，将请求转发到目的用户。定位服务器提供地址和特定域的绑定，这些地址绑定映射到一个 SIP 或是 SIP URI 的人呼叫，或者是到一个或多个与通信用户距离较近的 URI。最终，代理服务器借助定位服务将地址映射到一个接收地址域的用户代理上。

当区域内的代理服务器收到一个请求，并且它的 Request-URI 与地址记录（address-of-record）匹配，那么代理就将请求转发到注册的那个地址记录。通常，在一个区域内注册一个地址记录对于定位服务是有意义的，那么到该地址的请求将被路由到指定区域中。在多数情况下，这就意味着注册区域需要匹配 address-of-record 的 URI 字段。

可以使用多种方法来建立定位服务的内容。然而，SIP 为 UA 提供了一种显式的创建绑定的机制，即注册。

UA 要求发送 Register 请求给注册服务器。注册服务器作为区域定位服务的前端设备，它根据注册请求的内容来读写映射表。因为代理服务器负责路由请求到某个区域，所以定位服务需要借助代理服务器。

SIP 并不要求特殊的机制来实现定位服务，惟一的要求是区域的注册服务器要能够读写定位服务的数据。同样的，代理或重定向服务器也必须能够读相同的数据。注册服务器可以和特定的 SIP 代理服务器存在于相同的地址域。

SIP 协议的注册过程见图 1。

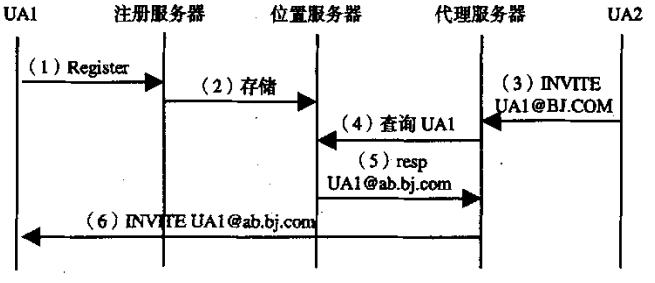


图 1 Register 示例

8.2 构造注册请求

注册服务器要求增加，删除和查询绑定信息。一个注册请求可以在 address-of-record 和一个或多个 contact 地址之间增加一个新的绑定。一个特殊的地址注册可以由一个适当的通过授权的第三方来执行。客户端可以删除先前的绑定，或是根据查询结果来决定替代当前的地址绑定信息。

除了上面提到的，Register 请求消息的构造和客户端发送 Register 请求的行为和一般的用户代理客户端的行为完全一样。

REGISTER 请求不会建立一个对话。UAC 可以在 Register 请求的路由头字段中包含预先定义的路由信息。在 REGISTER 请求和响应报文中 Record-Route 字段没有意义，如果存在必须忽略。不管对 REGISTER 请求的响应中是否包含 Record-Route 字段域，UAC 都不可以创建一个新的路由。

除了 contact，在 register 请求中必须包含以下的头字段，contact 头字段可能包含在：

request-URI：定义了对于注册过程有意义的定位服务所在的区域，在 SIP URI 中一定不能出现"userinfo"和"@"部分。

To：To 报头字段包含被创建，查询和修改的记录地址。To 报头字段和 Request-URI 域完全不同，因为在 To 报头字段包含用户名信息，address-of-record 一定是 SIP URI 或者是 SIPS URI。

From：From 域包含发出注册请求的用户地址信息，这个值和 To 域的相同，除非是第三方的注册。

Call-ID：来自 UAC 的注册应当使用相同的 Call-ID 值，发送到特定的注册服务器。如果同一个客户使用不同的 Call-ID 值，注册服务器不能判断是否是一个延时的 REGISTER 请求失序到达。

Cseq：CSeq 值确保 Regiater 请求的正常顺序。对于与 Call-ID 相同的每个 Register 请求，UA 必须递增 Cseq 值。

Contact：Register 请求可以包含一个 Contact 字段，值为 0 或者多个地址绑定信息。

在没有收到来自注册服务器对先前注册请求的响应或者是请求超时之前，UAS（用户代理）不能发送一个新的注册（也就是说，包含新的 Contact 域值，这不同于重传）。

以下包含在注册请求报文 Contact 域的值含有特殊的含义：

Expires："expires" 参数说明 UA 将认为绑定信息的有效的时间间隔。这个值以秒为单位。如果这个值没有提供，那么采用 Expires 头字段的值。

8.2.1 增加绑定

Register 请求应当根据报文 To 字段中 address-of-record 地址发送到包含 contact 地址的注册服务器。

Contact 域值通常是由 SIP 或者是 SIP URI 组成，它们用来标识特定的 SIP 端点，但是可以使用任意的 URI 配置。UA 可以选择注册电话号码或者是电子邮件地址作为 address-of-record 的联系地址。

例如，Carol，地址标识是 "sip：carol@chicago.com"，将用 chicago.com 的域名向 SIP 注册服务器进行注册。在 chicago.com 域内的代理服务器就可以使用该注册信息路由目的地是 carol 的请求到她的 SIP 终端。

一旦客户端在注册服务器上建立了绑定，那么它可以根据需要发送新的绑定或者是更新已有的注册信息。在注册请求 2xx 响应报文的 contact 字段中，将包含一个已经在注册服务器上注册的完整的地址（address-of-record）绑定列表。

如果注册请求 To 字段域 address-of-record 是 SIPS URI，那么请求报文 Contact 字段值都应该是 SIPS URIs。当由 contact 地址确定的资源安全能够通过其他的方法得到保证时，客户端只能注册非 SIPS URIs。

注册并不需要更新所有的绑定，一个用户代理只需要更新它自己的 contact 地址。

8.2.2 设置 Contact 地址的过期间隔

当一个客户发送注册请求，它可以建议一个过期的间隔，指示客户端期望注册多长时间有效。（正如在 10.3 中描述的，注册服务器基于本地策略选择一个实际的间隔）。

对于一个绑定，客户端可以有两种方式建议一个过期的间隔：通过 Expires 字段或者是“expires”Contact 参数。当一个注册请求中包含多个绑定时，后者允许根据每个绑定建议一个过期间隔。然而，前面的方式对所有 contact 地址建议一个过期间隔，不包含“expires”参数。

如果在注册中上述两种方式都不存在，那么客户端希望服务器端做出选择。

8.2.3 Contact 地址的优先级

如果在注册请求中有多个 Contact 地址被发送，用户代理试图将所有 Contact 字段的 URI 值与 To 字段的 address-of-record 值相关联。这个列表可以通过用 Contact 字段的“q”参数来区分优先级，“q”参数指示了相对优先级。

8.2.4 删除绑定

注册必须要定期的刷新，否则注册信息就会过期，当然也可以删除该注册信息。客户端可以改变注册服务器的过期间隔。用户代理可以通过指定注册请求中的过期间隔为 0 来立即删除绑定。用户代理应当能够支持这种机制以便在过期间隔到达之前就可以删除绑定。

Contact 字段的特殊值“*”可以应用于所有的注册消息，但是只有当 Expires 字段的值为 0 时才可以使用。

Contact 字段使用特殊值“*”时，允许一个注册用户代理删除所有不知道确切值的与 address-of-record 地址相关的绑定。

8.2.5 获取绑定

对于任何注册请求的一个成功响应，应当包含现有绑定的完整列表，而无论请求是否会包含 Contact 字段。如果在注册请求中没有 Contact 字段，将不包含没有更改的绑定。

8.2.6 刷新绑定

每个用户代理负责刷新由它先前已经建立的绑定。用户代理不应当刷新别的用户代理建立的绑定。

注册服务器 200 响应包含一个 Contact 字段列表，它包含了当前所有的绑定。用户代理比较每一个 contact 地址。如果是它自己创建的，它将根据 expires 参数，或者是 Expires 字段值，更新过期间隔。用户代理在每个绑定过期之前发送注册请求，而且它可以在一个注册请求中包含多个绑定更新。

在一个初始化周期内，对于所有的注册用户代理都使用相同的 call-ID。除非使用重定向服务器，注册刷新应当被发送到与原始注册相同的网络上。

8.2.7 设置内部时钟

如果注册请求的响应包含一个日期字段，客户端可以使用这个字段来获取当前的时间已用于设置内部的时钟。

8.2.8 发现注册设备

用户代理可以使用三种方式来决定发送注册到哪个地址：通过配置、使用地址记录、组播。一个用户代理能够用注册服务器地址配置，如果没有配置注册服务器地址，用户代理应当使用地址记录，主机部分地址作为 Request-URI，并且发送到那个地址，使用通常的 SIP 服务器定位机制。

最后，用户代理可以配置使用组播。组播注册被发送到一个众所周知的 SIP 组播服务器，地址（224.0.1.75，适用于 IPv4）。目前还没有分配一个 IPv6 的组播地址，如果需要将在一个单独文档中阐述。SIP 用户代理可以在那个地址上监听，来了解本地址域其他的用户，然而它并不响应请求。

组播注册在某些环境下是不适宜的，例如，如果多个公司共享同一个本地网的情况。

8.2.9 传送请求

一旦注册的方法已经构造，并且消息的目的地址明确，那么 UAC 将会将 Register 请求递交给事务层。如果 Register 没有产生响应，事务层返回一个超时错误，那么 UAC 不应当立即重新向同一个注册服务器注册。

立即重新尝试可能也会超时，因此应当根据引起超时的条件等一个合理的时间间隔来减少网络中不必要的负载。没有一个指定的间隔时间。

8.2.10 错误响应

如果 UA 收到一个 423 响应，在确定注册请求中所有合同地址的超时间隔等于或是大于在 423 响应的 Min-Expires 报头字段的超时间隔以后，它可以尝试重新注册。

8.3 处理注册请求

注册服务器就是用户代理服务器，负责响应注册请求和维护管理域中用于访问代理服务器和重定向服务器的绑定信息。一个注册服务器处理注册请求见本部分第 8 章和 17 章，但是它只接收注册请求。注册服务器不会产生响应码为 6xx 的响应。

注册服务器可以按照适当的方式重定向注册请求。一般地，注册服务器可以在一个组播的接口上侦听然后重定向组播注册请求到一个自己的单播接口，并且以 302 响应。

如果在注册请求报头中包含 Record-Route 字段，注册服务器必须忽略该字段，而且注册服务器在它的注册请求响应报文中也不能包含 Record-Route 字段。

注册服务器可以接收由代理转发的请求信息，对于代理来讲它无法识别注册请求，因此会添加 Record-Route 字段信息。

注册服务器必须要知道它所维护的绑定信息归属区域设置，比如可以通过配置的方式。注册服务器必须按照接收请求信息的顺序处理请求。注册请求必须被处理，也就是说要么完全处理，要么根本不处理。每个注册消息必须独立于其他的注册请求或者绑定变化来处理。

注册服务器收到一个注册请求按照如下的步骤进行处理：

1）注册服务器检查 Request-URI 字段，决定是否需要访问该字段标识域的绑定信息。如果不需要，它也要充当代理服务器的角色。服务器应当转发请求到地址域。

2）为了确保能够支持任意必要的扩展，注册服务器必须能够处理在本标准所要求的用户代理服务器必须处理的字段值。

3）注册服务器应当能够鉴别用户代理客户端。SIP 协议的用户鉴别机制见本标准 20 章。

4）注册服务器应当能够决定是否允许被授权的用户更改地址记录的注册信息。例如，注册服务器应该可以参考鉴权数据库的数据，映射用户名到一个地址列表，用户被授权能够更改的绑定信息。如果授权用户没有被授权更改绑定，注册服务器必须返回 403（禁止），跳过余下的部分处理。

在支持第三方的注册体系结构中，通信方可以负责更新与多个地址记录相关的注册信息。

5）注册服务器从请求报文的 To 字段摘取地址记录中的地址信息。如果该地址在 request-URI 字段限

定的区域中无效，那么注册服务器必须发送 404 的响应（未找到）。URI 的参数必须删除（包括用户参数部分），任意转义字符必须更改成它们非转义形式。更改后的结果作为绑定列表的索引。

6）注册服务器检查是否包含 contact 字段，如果没有，将直接跳到最后一步的处理，如果存在，注册服务器检查 contact 域是否包含特殊值“*”和终止域。如果请求包含附加的 contact 域或者是不为 0 的终止时间，请求是无效的，服务器必须返回 400（无效的请求），跳过余下的处理部分。否则，注册服务器必须检查 call-ID 的值是否与保存在每个绑定的值一致。如果不一致，必须删除绑定，如果一致，只有在请求报文中的 Cseq 值大于绑定中保存的值时，才从绑定中删除。否则必须放弃更新，请求失败。

7）注册服务器处理 contact 字段中的每一个地址。对于每一个地址，过期间隔按照如下的方式确定：

——如果字段中包含有“expires”参数值，那么这个值必须作为请求的过期间隔。

——如果没有这样的参数，但是请求含有 expires 字段，那么这个值必须作为请求的间隔。

如果上述两种情况都不存在，本地配置的缺省值必须作为请求过期间隔。

注册服务器可以选择一个小于请求的过期间隔。只有当请求的间隔同时大于 0 小于 1 个小时，或者小于注册服务器配置的最小间隔时，它可以以 423 响应拒绝注册（间隔太短）。这个响应包含一个 Min-Expires 字段，它描述了注册服务器愿意接受的最小的过期间隔，然后跳过余下的处理部分。

允许注册服务器设置注册间隔来保护遭受过多的注册请求，同时降低它需要维护的状态，并降低注册过期的可能性。注册的过期间隔通常在服务的建立过程中使用。因此，注册服务器应当接受短时注册。如果间隔太短以至于频繁的注册刷新而降低了服务器的性能，则请求应被拒绝。

对于每个地址，注册服务器使用 URI 的比较规则查找当前的绑定列表。如果绑定不存在，URI 被暂时添加到地址中，如果绑定存在，注册服务器检查 call-ID 的值，如果在绑定表中保存的 call-ID 值与请求消息中的 call-ID 值不同，并且过期计时间隔是 0，那么绑定必须删除，否则更新绑定。如果它们的值相同，注册服务器比较 Cseq 的值，如果请求消息中的值大于绑定表中的值，那么它必须更新或者从表中删除。如果不是，必须终止更新，请求失败。

这种算法确保了来自相同用户代理的失序请求被忽略。

每个绑定项记录了请求消息中的 call-ID 和 CSeq 的值。

只有在所有的绑定更新和附加的操作都成功执行，才能够提交（也就是说使得代理服务器和重定向服务器能够看到上述信息）。如果任何操作失败，必须以 5XX 响应请求失败，并且所有的暂时的绑定更新必须删除。

8）注册服务器返回200响应。响应必须包含Contact字段，它的值列举了所有当前的绑定。每个Contact值必须用一个过期参数来标识，以指示注册服务器所接受的过期间隔。响应还应当包含一个Date头字段。

9 能力查询

OPTIONS 用于一个 UA 向另外一个 UA 或者代理服务器查询对方的能力。包括下列信息：包括支持的方法、内容类型、扩展名、以及编解码方法等等。例如，客户机要在 INVITE 中插入 Require 头字段之前，并不确定目的地 UAS 是否支持 Require 中所列的选项，则客户机就可以使用 OPTIONS 方法看目的地 UAS 回应的 Supported 字段中是否包括这个选项以确定对方是否支持它。所有的用户代理都应支持 OPTIONS 方法。

OPTIONS 请求的目标由 Request-URI 来指定，它可以指定另外一个 UA 或者 SIP 服务器。

如果 OPTIONS 发送到一个代理服务器，那么 Request-URI 的设置方法就如同 REGISTER 请求，其中没有用户部分。

服务器收到的 OPTIONS 请求中的 Max-Forwards 头字段的值是 0，那么它可以响应这个请求，而不必考虑 Request-URI 的值。这种处理如同 HTTP/1.1，可以用作“路径”功能，通过发送一系列包含递增的 Max-Forwards 值的 OPTIONS 请求来来查询路径中每个服务器的能力。

如果 OPTIONS 没有响应，事务层就返回一个超时错误来表明目标不可达。

发送 OPTIONS 请求也可以作为已建立的对话的一部分，可以询问对等体的能力以备将来的对话所用。

9.1 OPTIONS 请求的构造

OPTIONS 请求中可以包括一个 Contact 头字段。

OPTIONS 请求中应包括一个 Accept 头字段，该字段指明在所收到的响应中，UAC 支持的消息体的类型。一般情况下，该字段被设置成用来描述 UA 媒体能力的格式，例如 SDP（application/sdp）。

OPTIONS 的响应的范围是初始请求的 Request-URI。然而，只有 OPTIONS 请求作为已建立的对话的一部分发送时，后续的请求才能够被产生 OPTIONS 响应的服务器收到。

9.2 OPTIONS 请求的处理

如何构造 OPTIONS 响应码的选择同 INVITE 请求。即，当 UAS 准备接受呼叫时返回响应 200（OK），UAS 忙时返回响应 486（正忙）。这样，用 OPTIONS 来确定 UAS 的基本状态并指出该 UAS 是否接受 INVITE 请求。

在对话中收到 OPTIONS 请求时发送响应 200（OK）响应，这等同于在对话之外构建响应，并对该对话没有任何影响。

代理服务器处理 OPTIONS 和 INVITE 的不同导致 OPTIONS 的使用限制。由于代理服务器使用 non-INVITE 的方法处理 OPTION 请求，一个分叉代理的 INVITE 请求可以有多个 200（OK）响应，而分叉代理的 OPTIONS 请求只能有一个 200（OK）响应。

如果 OPTIONS 请求消息的响应是由代理服务器发出的，并列出该服务器的能力，例如响应 200（OK）。该响应不包括消息体。

响应码 200（OK）中应该包括以下头字段：Allow，Accept，Accept-Encoding，Accept-Language 和 Supported。如果该响应是由代理服务器发出，由于代理服务器不能识别方法，因此 Allow 头字段就应被忽略。响应 200（OK）中可包括 Contact 头字段，并且与 3xx 响应中的语义完全相同。也就是说它们可以列出一组可选择的名字和方法。该响应中也可以包括 Warning 头字段。

可被发送的消息体的类型由 OPTIONS 的 Accept 字段指定（缺省为 application/sdp）。如果该类型能

够描述媒体能力，UAS 就应该在响应中构建一个消息体。

10 对话（dialog）

对话包括一个解释 SIP 消息的上下文。有关对话外独立 UA 处理请求和响应的方法见本部分第 5 章。本章将规定如何使用那些请求和响应去建立一个对话，以及在对话过程中如何发送后续的请求和响应。

任何 UA 上的对话都是由对话 ID 来标识的，这个对话 ID 包含一个 Call-ID，一个本地标签和一个远端标签。对话中的每个 UA 的对话 ID 是不同的。另外，一个 UA 的本地标识符与对端 UA 的远端标识符相等。标签（tags）在惟一的对话 ID 的生成过程中是不透明的。

对话 ID 与其 To 头字段中包含一个标签（tag）的所有响应和请求有关。某个消息中的对话 ID 的计算规则取决于 SIP 实体是 UAC 还是 UAS。对于 UAC，对话 ID 中的 Call-ID 由消息的 Call-ID 头字段设置，远端标签（tag）由 To 头字段的 tag 设置，本地标签由于 From 头字段的 tag 设置。对于 UAS，对话 ID 中的 Call-ID 由消息中的 Call-ID 头字段设置，远端标签由消息 From 头字段的 tag 设置，本地标识符于消息 To 头字段的 tag 设置；

对话中包括一些对话中的后续消息所需的状态，包括：对话 ID、本地序列号、远端序列号、本地 URI、远端 URI、远端目的、布尔型标记“secure”和路由集，其中路由集是一个顺序的 URI 集，指定发送一个请求到对端所需遍历的服务器地址。

临时的响应被创建时，对话处于“初始状态”，当一个 2xx 的最终响应到达时转为“确认状态”，如果是其他响应或无响应到达，“初始状态”终止。

10.1 对话的创建

对话在请求消息得到了明确的非失败响应之后才被创建。

依此规则，当 INVITE 请求消息收到包含 To 标签的 2xx 响应和 101~199 响应时，对话被创建。当一次对话随着一个非终止的响应被创建的时候，它处于“初始”状态，也可称其为初始对话。扩展中可以定义其他的对话创建方法。有关 INVITE 方法的定义见本标准第 5 章，本章所及的对话状态创建的过程是不依赖于此方法的。

用户代理必须指定对话 ID 各部分的值。

10.1.1 UAS 行为

UAS 对一个请求消息作出响应从而建立对话时，它必须将请求消息中的 Record-Route 头字段拷贝到响应中去，包括 URI，URI 参数和任何 Record-Route 头字段参数，并必须保持所有值的顺序。另外，UAS 必须在响应中添加一个 Contact 头字段。该 Contact 头字段包括了一个地址，在该地址上，UAS 可以接收该对话中的后续请求消息。通常，URI 的主机部分是 IP 地址或者主机的全资格域名（FQDN）。在 Contact 头字段中提供的 URI 必须是 SIP 或者 SIPS URI。如果一个请求消息开始一个对话，该消息的 Request-URI 或 Record-Route 头字段值中存在一个 SIPS URI，或者如果该请求没有 Record-Route 头字段而在 Contact 头字段中包括 SIPS URI，则其响应中的 Contact 头字段必须是 SIPS URI。而且该 URI 必须是全局的。同样 INVITE 方法中 Contact 头字段里 URI 的作用范围也不仅仅局限于该对话中，它也可以在该对话外的发往 UAC 的消息中使用。

然后 UAS 构建对话的状态，在整个对话持续过程中，状态必须保持。

如果请求基于 TLS（传输层安全协议）传送，同时 Request-URI 中包含一个 SIPS URI，“secure”标签置为“TRUE”。

路由集必须设置为请求消息的 Record-Route 头字段中的 URI 列表，并保留顺序和 URI 参数。 如果在请求消息中无 Record-Route 头字段，路由集必须为空。路由集即使为空也必须覆盖该对话的后续请求消息的现有的路由集。远端目的必须设置为来请求消息的 Contact 头字段的 URI。

远端序列号必须设置为请求消息的 CSeq 头字段中的序列号值，本地序列号为空；

对话 ID 中的呼叫标志符必须设置为请求消息的 Call-ID 值；

本地标签必须设置为该请求的响应消息的 To 头字段的 tag 值；

远端标签必须设置为请求消息的 From 头字段的 tag 值；

另外，UAS 必须能够接收一个在 From 头字段中没有标签的请求消息，这种情况下标签值认为为空。

远端 URI 必须设置为 From 头字段的 URI；

本地 URI 必须设置为 To 头字段的 URI。

10.1.2 UAC 行为

当 UAC 发送一个建立对话的请求（如 INVITE 请求）来建立对话时，该请求消息的 Contact 头字段必须有一个全局的 SIP/SIPS URI。如果请求消息中存在一个 Request-URI 或者在 Record-Route 头字段的顶部包含一个 SIPS URI，那么 Contact 头字段必须包含一个 SIPS URI。

UAC 在接收到一个建立对话的响应消息后构建对话的状态。这个状态在整个对话过程中必须保持不变。

如果请求基于 TLS 传送，同时 Request-URI 中包含一个 SIPS URI，“secure” 标签应设置为 “TRUE”；

路由集必须设置为响应消息的 Record-Route 头字段的 URI 列表，保持相反的顺序并保留所有的 URI 参数，如果在响应消息中无 Record-Route 头字段，路由集必须设置为空，路由集即使为空也必须覆盖该对话的后续请求消息的现有的路由集；

远端目的必须设置为响应消息 Contact 头字段的 URI；

本地序列号必须设置为请求消息的 CSeq 头字段的序列号值；

远端序列号必须为空，当远端 UA 发送一个本次对话中的请求后该值才可确定；

呼叫标识符必须设置为请求消息的 Call-ID 值；

本地标签必须设置为请求 From 头字段的标签值；

远端标签必须设置为响应 To 头字段的标签值；

另外，UAC 必须可以接收一个 To 头字段中不含标签值的响应，这种情况下标签值为空；

远端 URI 必须设置为 To 头字段的 URI 值；

本地 URI 必须设置为 From 头字段的 URI 值。

10.2 对话中的请求

对话在两个 UA 之间建立之后，任一个 UA 都可根据需要在对话中发起一个新的事务。在该事务中，发送请求消息的 UA 作为 UAC，接收请求消息的 UA 作为 UAS。这种情况，UA 的角色可能和建立对话是的事务中的有所不同。

对话中的请求消息可包括 Record-Route 和 Contact 头字段。但是即使修改了远端目的 URI，这些请求仍不会改变对话的路由集。另外，非目标刷新的请求消息不会修改对话的远端目的 URI，而目标刷新的请求消息则会修改它们。由 INVITE 请求建立的对话，惟一的目标刷新请求定义为 re-INVITE。针对由其他方式建立的对话，有相应的目标刷新请求的扩展。

ACK 不是一个目标刷新请求消息。

目标刷新请求消息仅是更新对话的远端目的 URI，而不是更新 Record-Route 头字段的路由集。

10.2.1 UAC 行为

10.2.1.1 发起请求

一个对话中的请求消息由对话所保存的状态信息来构建。

请求消息的 To 头字段的 URI 必须设置成对话状态中的的远端 URI；

请求消息的 To 头字段的标签值设置为对话 ID 的远端标签值；

请求消息的 From 头字段的 URI 必须设置为对话状态中的本地 URI；

请求消息的 From 头字段的标签值设置为对话 ID 的本地标签；

如果本地或远端标签值为空，相应的 To 或 From 头字段中的标签参数必须头字段省略。

本协议规定，请求消息中的 Call-ID 必须设置为对话的 Call-ID。

对话内的中间请求必须包含 CSeq 序列号，CSeq 序列号是一个随着 CSeq 值依照各自方向严格单向按 1 递增的值，因此，如果本地序列号非空，Cseq 序列号值必须按 1 递增，同时必须放在 CSeq 头字段内。如果本地序列号为空，必须按照本标准定义方法为其设置初始值。CSeq 头字段中的请求方法必须与请求消息中的方法一致。

UAC 采用远端目的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 Request-URI 置为远端目的 URI 值，同时不必在请求消息中添加 Route 头字段。

如果路由集不为空，同时路由集的第一个 URI 中包含一个 lr 参数，UAC 也必须将 Request-URI 置为远端目的 URI 值，同时必须包含一个 Route 头字段，且按顺序包含路由集的值及所有参数。

如果路由集不为空，同时第一个 URI 不包含 lr 参数，UAC 必须置 Request-URI 为路由集的第一个 URI，且不允许去掉任何参数。同时 UAC 必须添加一个 Route 头字段，按顺序包含路由集剩余部分值及所有参数。UAC 必须将远端目的 URI 置为 Route 头字段的最后一个值。

如果路由设置的第一个 URI 不包含 lr 参数，代理表示不理解本标准所规定的路由机制，它将 Request-URI 用它收到的前向消息中的 Route 头字段的第一个值来代替。

对话过程中的任一个更新目的的请求消息应该包含一个 Contact 头字段，如果无特殊情况需要更改，Contact 的 URI 应该与该对话前面的请求消息中的 URI 相同。如果“secure”标签为“TRUE”，URI 就必须是一个 SIPS URI。目的更新请求消息的 Contact 头字段会更新远端目的 URI。

然而，非目的地更新请求消息不会影响此次对话的远端目的 URI。

一旦请求消息被构建，服务器地址被确定，同时请求消息被发送，其处理同对话外的请求消息的处理。请求将会被发送到 Route 头字段的最顶端所标明的地址，如果无 Route 头字段的，请求消息将被发送至 Request-URI 标明的地址上。特殊情况下，也允许将请求发送至一个可替换的地址上。

10.2.1.2 响应处理

UAC 从事务层收到对请求的响应消息。如果客户端事务层返回一个超时信息，对其的处理与 408 响应相同，对话过程中，UAC 收到 3xx 响应的处理与对话外响应的处理相同。

但当 UAC 进行位置选择时，它还会使用对话的路由集去构建请求消息的 Route 头字段。

当收到对目的刷新请求消息的 2xx 响应时，UAC 必须将对话的远端目的 URI 设置为响应消息中存在

的 Contact 头字段的 URI 值。如果响应为 481（呼叫/事务不存在）或者 408（请求超时），UAC 应该终止对话；在对方无响应的时候 UAC 也应终止对话，客户端事务处理将会通知 TU 超时。

以 INVITE 请求消息发起的对话，UAC 通过发送 BYE 来终止对话。

10.2.2 UAS 行为

如果 UAS 接收一个特定的请求，所有相关的状态将会被迁移；如果请求被拒绝，将不会有状态迁移发生。

仅有一些特殊的请求会影响状态迁移，比如 INVITE 请求。

UAS 会收到来自于事务层的请求消息。如果该请求 To 头字段中存在标签，UAS 内核（UAS core）将会计算与此请求相关的对话标识符，同时将其与已有的对话进行比较。如果相匹配的话，该请求即为一个对话内的请求。在这种情况下，UAS 采用与对话外请求消息处理规则相同的处理流程对其作出响应。

如果请求消息的 To 头字段包含标签，但对话标识符与与已有的对话不匹配，则表示 UAS 可能曾经重新启动过，或者收到了发给其他的 UAS 请求消息，或者接收到一个被发错的请求。基于 To 头字段标签值，UAS 可以接收或拒绝这个请求。接收请求，导致拆掉的对话得以延续。支持这种能力的 UA 必须考虑到如选择单向递增的 CSeq 序列号、路由集的重建、以及接收越界的 RTP 时间信息与序列号等问题。

如果 UAS 不愿重建此次对话而拒绝该请求，那么它必须回一个 481（呼叫/事务不存在）的状态码给服务器端的事务。

UAS 可以接收不改变对话状态的请求消息，如 OPTIONS 请求，对它的处理与在对话外的请求消息的处理相同。

如果对话的远端序列号为空，那它必须被设置为请求 CSeq 头字段的序列号值；

如果远端序列号非空，但是请求中序列号值比远端序列号低，则认为请求次序颠倒，必须回 500（服务器内部出错）响应拒绝请求；

如果远端序列号非空，同时请求消息中序列号值比远端序列号高，请求次序正常，且 CSeq 序列号与远端序列号相比差值大于 1，此时不认为出错，同时 UAS 应该准备接收此请求并对其进行处理，UAS 必须设置远端序列号为请求消息的 CSeq 头字段的序列号值。

当 UAS 接收到一个目的更新请求，它必须用请求消息已有的 Contact 头字段的 URI 值替换对话的远端目的 URI 头字段。

10.3 终止对话

初始对话的终止不依赖于发起请求的具体方法，只要收到一个非 2xx 的终止响应即可将其终止。证实/确认对话（confirmed dialogs）的终止机制与确切方法相关。本协议规定，BYE 方法终止一次会话（session），同时终止一次与其相关的对话，具体内容见本标准第 13 章。

11 会话发起过程

11.1 概述

UA 通过向服务器发送 INVITE 消息开始会话发起过程，该请求可能通过网络中间的服务器设备的转发，最终到达 UAS。如果 UAS 同意建立本次会话，则返回 2XX 响应，否则返回 3XX、4XX、5XX、6XX 响应。在收到最终响应之前，UAS 也可以发送临时响应（1XX）来通知 UAC 当前的处理进展情况。

由于该 INVITE 请求可能被分叉代理（Forking），UAC 可能会收到一个或多个 2XX 响应或者一个非 2XX 响应。UAC 收到最终响应消息之后需要向 UAS 侧发送 ACK 消息。

每一个 2XX 响应都会创建一个对话，因此如果 UAC 收到多个 2XX 响应，那么每个 2XX 都创建一个对话，这些对话都属于同一个呼叫。

11.1.1 创建初始 INVITE 消息

由于初始的 INVITE 消息是对话之外的请求，因此它的构造过程需要遵循通用的对话外请求消息构造过程。下面针对 INVITE 消息的特殊情况进行说明：

必须包含 Allow 头字段，用于表示本次会话过程中 UAC 支持的请求方法。

必须包含 Supported 头字段，用于表示 UAC 支持的扩展功能。

INVITE 消息中可以包含 SDP 信息，这时 Content-Type 头字段应该是 application/sdp。

本部分中关于媒体协商的过程是通过在消息中携带 SDP 来完成的。关于 SDP 协商的过程需要遵循如下规定：

INVITE 消息中携带 SDP 请求，200 消息中返回 SDP 响应。

200 消息中携带 SDP 请求，ACK 消息中返回 SDP 响应。

SDP 请求和响应这个协商过程，不能并行，只能当一次交互完成之后才能发起新的协商过程。

会话中的两个实体可以使用提供/应答模式进行 SDP 的交互来使得它们之间的多媒体会话达成一致。在该模式中，一个参加者向另一个参加者提供它期望建立的会话的描述，另一个参加者应答它期望建立的对话。SIP 协议中采用提供/应答模式来进行媒体协商的过程见附录 E。

11.1.2 处理对 INVITE 消息的响应

1XX 响应：在收到最终响应之前，UAC 可能收到一个或者多个临时响应，或者不收到任何的临时响应。

3XX 响应：一个 3XX 响应中可以包含一个或者多个 Contact 头字段，这些头字段代表被呼叫方的新的地址。UAC 可以根据具体的响应类型以及本地的业务策略来选择是否向这些地址发起新的会话请求。

4XX、5XX 和 6XX 响应：UAC 只能收到一个非 2XX 响应。其中可以包含一个 Contact 头字段用于提供关于会话建立失败信息的查询地址。UAC 收到非 2XX 响应，即认为 INVITE 事务已经完成，向 UAS 发送 ACK。

2XX 响应：由于 INVITE 请求可能被分叉代理（Forking），因此 UAC 可能收到多个 2XX 响应。这些响应通过 To 头字段中的 tag 参数相区别，每一个响应都代表一个独立的对话。UAC 必须对每个 2XX 都返回 ACK。

11.2 UAS 的处理过程

11.2.1 UAS 处理 INVITE 消息包括下述几种状况：

正在处理：如果 UAS 不能马上作出应答，它可以发送相应的 1XX 响应来通知 UAC 当前的处理进展（例如振铃、被前转等）。

重定向：如果 UAS 希望将该请求重定向，它可以发送 3XX 响应，其中携带新的被叫地址，要求 UAS 向新的地址发起呼叫。

拒绝：如果 UAS 由于某种原因不能接收这个呼叫请求，它可以根据具体的原因，发送相应的 4XX、5XX、6XX 响应。

接受：UAS 返回 2XX 响应，表示接受本次呼叫请求。这个响应消息同时也建立了一个对话。

2XX 响应中必须包含 Allow 头字段，用于表示 UAS 支持的请求方法。2XX 中必须包含 Supported 头

字段，用于表示 UAS 支持的扩展能力。

如果 INVITE 请求中，包含 SDP 提供，那么 UAS 必须在 2XX 返回一个 SDP 应答。

UAS 必须重发 2XX 直到接收到 ACK 确认消息。重发间隔为 T1，每次重发间隔加倍，直到达到 T2。如果这时仍未收到 ACK，UAS 必须结束本次会话，向 UAC 发送 BYE 请求。

12 会话更改过程

12.1 概述

会话建立之后，任何一方都可以发起会话更改请求，修改会话的某些属性，例如，增加删除媒体流、改变媒体发送接收地址等。这是通过在已经建立的对话中发送一个新的 INVITE 请求来完成。我们称之为 reINVITE。

12.2 UAC 的处理过程

reINVITE 过程所采用的 SDP 协商过程与建立会话的 INVITE 过程相同。会话中的任意一方可以通过在 reINVITE 中携带一个新的 SDP 请求来更新会话内容。

reINVITE 消息中的 To、From、Call-ID、Cseq 和 Request-URI 头字段的生成方法采用通用的对话中请求消息的生成规则相同。

reINVITE 不会被分叉代理，因此只可能收到一个最终响应。

reINVITE 过程不能重叠，如果已经有一个 reINVITE 事务正在执行，就不能发起新的 reINVITE 事务。

对于 reINVITE 事务的 ACK 和 2XX 响应的生成，与初始 INVITE 过程相同。

12.3 UAS 的处理过程

UAS 收到 reINVITE 消息后，必须检查其中的 SDP 是否更改，并对相应的会话参数做出调整。如果新的媒体描述不可接受，UAS 可以返回 488 拒绝响应。

如果 UAS 返回了 2XX 响应，但是没有收到 ACK，它必须发送 BYE 来结束本次会话。

13 会话结束过程

13.1 概述

会话的中止可以通过对 INVITE 请求返回拒绝相应，对已建立的会话发送 BYE 请求等方式来完成。

13.2 通过 BYE 请求结束会话

13.2.1 UAC 的处理过程

会话中的任意一方可以通过发送 BYE 请求来结束已经建立的会话。BYE 请求的生成与通用的对话中请求消息的生成规则相同。

BYE 请求对应一个新的事务。UAC 发送 BYE 请求之后即认为本次会话已经结束了。

13.2.2 UAS 的处理过程

UAS 收到 BYE 请求之后，需要查询匹配的会话。如果找不到则返回 481 响应；如果找到对应的会话，UAS 必须结束该会话，然后对 BYE 返回 2XX 响应。对于正在处理的请求消息，UAS 返回 487 响应。

14 代理服务器的行为

14.1 概述

代理服务器是将请求消息路由到UAS以及将响应消息路由到UAC的实体。一个请求消息在到达UAS之前可能要经过若干个代理服务器的转发，每个代理服务器都要进行路由决策，并在将请求消息转发到下一个实体之前对其进行修改。响应消息将遍历请求消息所经的那些服务器，但顺序却完全相反。

代理服务器是一个逻辑 SIP 实体。当一个请求消息到来时，一个能作为代理服务器的 SIP 实体首先决定是否需要由自己来应答这个请求，例如请求消息中可能有格式错误，或者在执行代理功能之前需要先获得客户端的鉴权证书等，而该实体亦可用任何适当的错误码来响应。SIP 实体直接应答一个请求时，它承担的角色就是 UAS。

对每个新请求，代理服务器既可以在有状态模式下工作，也可以在无状态模式下工作。当以无状态模式工作时，代理服务器只是作为一个简单的消息转发实体，它根据请求消息来做转发目的地和路由决策，然后把请求转发到下游的某个实体；对于响应消息，则只简单地将其往上游方向转发。一旦消息转发完毕，无状态代理服务器将丢弃所有与此消息相关的信息。有状态代理服务器会记住它所收到的每个请求的信息，如事务状态，以及作为某一请求的处理结果而发送的任何请求的信息。这些信息将影响它对后续的、与先前接收的某一请求相关的消息的处理。有状态代理服务器可能选择“分叉代理”（fork）转发一个请求，即将一个请求向多个目的地路由。任何被转发到多个地点的请求都必须在有状态模式下处理。

有些情况下，代理服务器可能采用有状态的传输方式（比如 TCP）来转发请求，这时不必保留事务状态。例如，代理服务器可能在不保留事务状态的情况下将一个请求从一个 TCP 连接转发到另一个 TCP 连接，只要它的请求消息有足够的信息，使它能把对应的响应消息沿着接收请求所用的 TCP 连接转发。当请求消息在不同类型的传输方式之间转发时，代理服务器必须采用有状态模式，代理服务器的事务用户必须保证消息的可靠传输。

一个有状态的代理服务器可能在请求处理过程中的任何时侯转到无状态工作模式下，在做这样的转换时，所有的事务状态信息都将被丢掉。代理服务器不应主动发起 CANCEL 请求。

14.2 有状态代理服务器

一个工作在有状态模式下的代理服务器就是一个 SIP 事务处理引擎。有状态代理服务器的行为模型可根据客户端事务和服务器端事务的有关定义而建立。一个有状态代理服务器包含一个服务器端事务，一个或多个客户端事务，以及将它们关联起来的一个被称为代理服务器内核的（见图 2）高层处理模块。从外部收到的请求由服务器端事务来处理。处理之后的请求送给代理服务器内核层，内核层决定将请求路由到何处，选择一个或多个下一跳位置。发往每个下一跳位置的请求消息都由一个它自己的客户端事务进行处理。内核层从客户端事务收集响应，并依据这些响应来向服务器端事务发送响应。

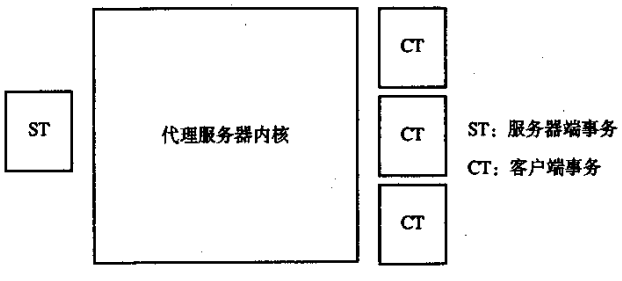


图 2 有状态代理服务器模型

有状态代理服务器为每个收到的请求创建一个新的服务器端事务。任何重传的请求由服务器端事务的处理过程见本章第 14.1 节。当立即回送临时性响应消息（例如 100 Trying）到服务器端事务时，代理服务器内核必须作为 UAS 进行处理。有状态代理服务器不应当对非 INVITE 请求产生 100（Trying）响应。

对所有新请求，包括任何方法未知的请求，一个执行代理功能的实体必须：

1） 确认请求的有效性；

2） 预处理路由信息；

3） 确定请求发送的目的地；

4） 将请求向每个目的地转发；

5） 处理所有的响应消息。

14.3 确认请求的有效性

一个实体在进行代理服务之前必须确认请求的有效性。一个有效的请求消息必须通过如下的检查：

1） 合理的语法；

2） URI 方案；

3） 最大转发数；

4） 路由循环检测（可选）；

5） Proxy-Require；

6） Proxy-Authorization。

如果任何一项检查失败，代理服务器必须像 UAS 那样处理并用一个错误码来应答。

本标准规定，代理服务器不要求进行请求合并的检测，并且不应把合并的请求当作错误处理。接收请求的端点将会解决请求合并问题。

14.3.1 消息的语法检查

请求消息的语法构造必须正确，这样服务器端事务才可以处理该请求。任何与请求的有效性相关的消息内容，或者与请求的转发相关的消息内容，其构造必须正确。对任何其他的消息内容，在消息转发时应忽略其语法构造是否正确并保持原样。

本部分是可扩展的。将来的扩展可能定义新的请求方法和新的消息头字段。一个实体不可以因为某个请求包含了它不认识的请求方法或者消息头字段而拒绝代理该请求。

14.3.2 URI 编码格式检查

如果一个请求的 Request-URI 使用了代理服务器不能理解的 URI 方案，代理服务器应当用一个 416（不支持的 URI 方案）响应拒绝该请求。

14.3.3 最大转发数检查

Max-Forwards 头字段用来限制一个 SIP 请求消息所能经过的实体的最大数目：

如果请求消息中不包含 Max-Forwards 头字段，本项检查通过；

如果请求消息中包含 Max-Forwards 头字段，并且其值大于 0，本项检查通过；

如果请求消息中包含 Max-Forwards 头字段，并且其值等于 0，那么 SIP 实体不可以转发此请求。如果请求方法是 OPTIONS，那么 SIP 实体可作为消息的最终接收者并进行响应；否则，SIP 实体必须返回一个 483（跳数过多）响应。

14.3.4 路由循环检测（可选）

SIP 实体在转发一个请求前可能进行转发循环的检测。如果 SIP 实体收到的某个请求中包含了一个 Via 头字段，其 sent-by 参数值与该实体作为代理服务器时在以前的请求消息中加入的值相同，则检测到该请求已经被该实体转发过。这种情况可能是请求消息被循环路由，或者是合法地"螺旋路由"地经过这个 SIP 实体。循环路由检测规则见本标准 14 章，SIP 实体可以根据该规则的第 8 步来计算该消息的 branch 参数，并将结果与上述 Via 头字段中的 branch 参数进行比较。如果两个参数相匹配，那么请求消息被循环路由；反之，则请求消息被"螺旋路由"，且处理继续。如果检测到路由循环，SIP 实体可能返回一个 482（检测到路由循环）响应。

14.3.5 Proxy-Require 检查

本协议的扩展以后可能会引入一些要求代理服务器所具有的特殊的处理特性。终端在使用这些新特性的请求消息中加入一个 Proxy-Require 头字段，代理服务器如果不理解指定特性将不能进行请求消息的处理。

如果请求消息中包含了一个 Proxy-Require 头字段，其中列出的某项或某几项扩展能力选项标签不能为 SIP 实体所理解，SIP 实体必须返回一个 420（错误的扩展）响应。该响应中必须包含一个 Unsupported 头字段，其中列出那些不能被理解的选项标签。

14.3.6 Proxy-Authorization 检查

如果 SIP 实体在转发一个请求之前需要鉴权证书，则需要先进行消息检查。关于消息检查及检查失败后 SIP 实体如何处理见本标准 19 章。

14.4 路由信息预处理

代理服务器必须检查请求消息的 Request-URI。如果请求消息的 Request-URI 是它以前插在 Record-Route 头字段中的值，代理服务器则必须用 Route 头字段的最后一个值替换 Request-URI，并且从 Route 头字段中删除该值。在这之后，代理服务器必须按修改过的请求来进行处理。

上述情况只有当发送请求到代理服务器的 SIP 实体为严格路由器（代理服务器）时才会发生。

本部分中并不要求代理服务器为了检测它以前插入到 Record-Route 头字段中的那些 URI 而保留状态信息。代理服务器只需要在那些 URI 中加入足够的信息，使其在以后出现时能够被识别。

如果 Request-URI 中包含 maddr 参数，代理服务器必须检查其值是否在自己所负责的地址集或主域集中。如果在其中，且对应的请求消息是通过 Request-URI 中指定或缺省的端口和传输方式接收的，代理服务器必须去掉 maddr 参数和所有非缺省的端口及传输方式参数，并按照没有这些参数的请求消息来进行处理。

代理服务器收到的某个请求消息中可能有与该服务器相匹配的 maddr 参数，但接收该消息时所用的端口和传输方式却与 Request-URI 中指定的不同。这样的请求需要被转发到使用 Request-URI 中指定的端口和传输方式的代理服务器。

如果请求消息中 Route 头字段的第一个值指示的就是本代理服务器，那么它必须将该值从请求消息中删掉。

14.5 确定请求的发送目标

代理服务器计算请求消息的发送目标。发送的目标地址集可以由请求消息的内容预先决定，也可以是从某个抽象的定位服务获得。地址集中的每个目的地都被表示成一个 URI。

如果请求的 Request-URI 中包含一个 maddr 参数，Request-URI 必须被作为惟一的目标 URI 放在目标

地址集中，代理服务器对其处理的过程见本部分 14 章。

如果 Request-URI 所指示的域不是代理服务器所负责的域，Request-URI 必须被作为惟一的发送目的地放进目标地址集中，然后该实体必须对该消息进行转发。

如果目标地址集不能通过上述的方法确定，当前 SIP 实体则负责 Request-URI 中的主域，并且该实体可以用任何机制来决定向何处转发请求。不论是哪种机制，其模型都可以化为访问一个抽象定位服务的过程。转发机制有：从 SIP 注册服务器创建的某个定位服务获得信息；读取数据库；查询一个呈现（Presence）服务器；使用其他协议；或者执行一个简单的算法对 Request-URI 进行替换等等。

如果 Request-URI 没有提供足够的信息使得代理服务器能够决定目标地址集，代理服务器应当返回一个 485（不明确）响应。响应中应当包含一个 Contact 头字段，其中列出了可供尝试的新地址的 URI。例如，如果某代理服务器的定位服务中有多个 John Smith，则一个构造为 sip：John.Smith@company.com 的 INVITE 请求对该代理服务器是不明确的。

任何在请求消息中，或者与请求消息相关的，或者实体的当前环境相关的信息都可能用于目标地址集的构建。例如，根据某些头字段和消息体是否出现及其内容如何，或请求消息到达的时间、或接收请求所用的接口、或以前请求消息的失败情况，甚至实体当前的可用性级别等等，可能会构造出不同的目标地址集。

通过上述定位服务确定了可能的目标地址之后，它们的 URI 就被加入到目标地址集中。目标地址只能被放置到目标地址集一次。根据 URI 相等的定义，一个目标 URI 已经在目标地址集中出现，那么它就不能被再次加入该地址集。

如果原请求的 Request-URI 所指示的资源位置不在本代理服务器负责的范围之内，代理服务器不得在目标地址集中加入其他地址。

如果代理服务器负责原始请求的 Request-URI 所指定的资源，则在请求转发开始之后它可能继续向目标地址集加入地址。它可以使用处理过程中获得的任何信息来决定新的目标地址。比如，代理服务器可以将一个重定向响应（3xx 类响应）中获得的 contact URI 并入目标地址集中。如果代理服务器在构造目标地址集时使用了动态信息源，它应当在请求处理过程中监视所用信息源的变化。如果有新的可用地址，那就应当将其加入目标地址集中。同样的地址决不能被多次加入。

例如，“无操作”也是一个简单的定位服务，此时目标 URI 就等于接收请求的 Request-URI。请求消息将被发送到特定的下一跳代理服务器以进一步处理。见 13.6 节第 6 条的请求转发处理，用 SIP 或 SIPS URI 表示的下一跳地址标识被作为 Route 头字段最顶端的值插入到请求消息中。

如果在 Request-URI 中指定的某个资源不存在，代理服务器必须返回一个 404（找不到）响应。

按上述所有的方法处理之后，目标地址集仍然为空，代理服务器就必须返回一个错误响应，并且应当是 480（临时不可用）响应。

14.6 请求的转发

只要目标地址集非空，代理服务器就可以开始转发请求。有状态代理服务器可以按任何次序来处理目标地址集。代理服务器可以对多个目标地址采用串行处理的办法，使得一个客户端事务完成之后才开始下一个；它也可以同时开始所有的目标地址处理，即并行处理相应的所有客户端事务；它还可以将目标地址集任意分组，组与组之间串行处理，组内的目标地址并行处理。

一种常用的排序机制是利用目标地址的 qvalue 参数，这些参数来自 Contact 头字段。目标地址按

qvalue 值的大小从最高往最低处理，qvalue 值相同的可能会同时处理。

有状态代理服务器必须有目标地址集的维护机制，这样当收到各个转发请求的响应后就能把它们与原始请求联系起来。本模型中，这是“响应上下文”的机制（response context），它由代理服务器层在转发第一个请求之前创建。

对每个目标地址，代理服务器按如下步骤转发请求：

1）复制收到的原始请求；

2）更新 Request-URI；

3）更新 Max-Forwards 头字段；

4）添加一个 Record-Route 头字段值（可选）；

5）添加其他头字段（可选）；

6）路由信息后处理；

7）确定下一跳的地址、端口和传输方式；

8）添加一个 Via 头字段值；

9）必要时加入一个 Content-Length 头字段；

10）转发新请求；

11）设置定时器 C。

14.6.1 请求消息复制

代理服务器首先对收到的请求做一份拷贝。拷贝最初必须包含所收到的请求的所有头字段。在下面描述的处理过程中未提及的字段也不能删除。拷贝中各个头字段的顺序应当与所收到的请求保持一致。代理服务器决不能将头字段名相同的头字段值重新排序。代理服务器不能添加、修改或删除消息体。

具体实现不需要进行真正的消息复制。其基本要求是对每个下一跳的处理都要从同一个请求消息开始。

14.6.2 Request-URI

拷贝请求消息的起始行的 Request-URI 必须被替换为当前目标地址的 URI。如果该 URI 中包含任何在 Request-URI 中不允许出现的参数，这些参数必须被删掉。

代理服务器通过这种机制把一个请求消息路由到它的目的地。

有些情况下，所收到的请求消息的 Request-URI 没有经过任何修改就被放到了目标地址集中。对这样的目标地址，上述的替换实际上是一种”无操作”行为。

14.6.3 Max-Forwards 字段

如果消息拷贝中包含一个 Max-Forwards 头字段，代理服务器必须将其值减一。

如果消息拷贝中不含 Max-Forwards 头字段，代理服务器必须加入该字段，其值应当为 70。

现有的某些 UA 在请求消息中不提供 Max-Forwards 头字段。

14.6.4 Record-Route

如果代理服务器希望自己留在后续请求消息的传输路径上，而这些后续的请求消息属于由当前请求创建的一个对话，那么即使请求中已经出现了 Route 头字段，代理服务器也必须在请求消息的拷贝中任何已有的 Record-Route 头字段值之前插入一个 Record-Route 头字段值。

建立对话的请求消息中可能包含预置的 Route 头字段。

如果当前请求已经属于某一个对话，那么代理服务器要想留在该对话中后续请求消息的传输路径上，它应当在请求中插入一个 Record-Route 头字段。正常情况下，这些 Record-Route 头字段将不会对终端所用的路由集产生任何影响。

对于已经属于某个对话的请求消息，即使代理服务器不在其中插入 Record-Route 头字段值，它也能留在后续请求的传输路径上。然而，在端点发生了错误后重建对话时，这个代理服务器将被从以前的传输路径上移除。

代理服务器可能在任何请求中插入 Record-Route 头字段值。如果请求不是用来发起对话，那么终端将会忽略这个 Record-Route 头字段值。

请求传输路径上的每个代理服务器都独立决定是否在请求消息中添加一个 Record-Route 头字段值。请求消息中已经出现的 Record-Route 头字段，并不要求代理服务器一定得添加新值。

在 Record-Route 头字段中放置的 URI 必须是一个 SIP URI 或 SIPS URI。这个 URI 必须包含一个 lr 参数。对请求转发的每个目的地，这个 URI 都可能不同。该 URI 不应包含传输方式参数，除非代理服务器明确相邻的下游实体支持该参数所示的传输方式，同时该下游实体将出现在后续请求的传输路径上。

当前代理服务器提供的 URI 将被其他 SIP 实体用于路由决策。通常来说，代理服务器无法知道其他实体的能力，所以它必须支持 SIP URI 以及 TCP 或 UDP 传输方式。定位服务器对 Record-Route 头字段中加入的 URI 进行解析，其结果必须是插入该 URI 的 SIP 实体，这样使得后续的请求能够到达同一个 SIP 实体。如果请求消息的 Request-URI 包含了一个 SIPS URI，或者最顶端的 Route 头字段值在路由信息后处理之后包含了一个 SIPS URI，那么代理服务器在 Record-Route 头字段中加入的 URI 也必须是 SIPS URI。另外，如果请求消息不是通过 TLS 方式接收的，那么代理服务器必须插入一个 Record-Route 头字段。类似地，如果请求消息是通过 TLS 接收的，但转发的请求消息中 Request-URI 或最顶端 Route 头字段值在路由信息后处理之后不是 SIPS URI，那么代理服务器必须在转发的请求消息中插入一个非 SIPS URI 的 Record-Route 头字段。

一个处于某一安全范围内的代理服务器，在对话当中必须保留在该安全范围内。

当插在 Record-Route 头字段中的 URI 通过响应消息又被传回来时，如果代理服务器需要重写该 URI，则该 URI 必须清晰明确以便被准确查找（请求消息可能通过螺旋路由多次经过一个代理服务器，导致代理服务器加入多个 Record-Route 头字段值）。

代理服务器可能在 Record-Route 头字段值中包含一些参数。这些参数会在请求的某些响应消息中被送回，如 INVITE 请求的 200（OK）响应。如果想在消息中而不是代理服务器中保存一些状态信息，便可以使用这些参数。

如果不论对话是什么类型，代理服务器都需要留在它的消息传输路径上，代理服务器则应当在每个请求消息中都加入一个 Record-Route 头字段值，即使它不认识请求的方法。不认识的请求也可能会建立或属于某个对话。

代理服务器加到某一请求的 Record-Route 头字段中的 URI 只在相应的对话生命期中有效，这个对话由该请求所在的事务创建。例如，一个保持对话状态的代理服务器在对话终止之后，可能会拒绝接收 Request-URI 的某个 URI 的后续请求消息，因为该 URI 的有效期已过。当然，一个没有对话状态的代理服务器无从得知对话是否终止，但它可以在插入的 URI 中添加足够的信息，再将其同后续请求的对话标识相比较，其对话标识与信息不相匹配的请求将有可能被拒绝。终端不可以在某个对话外使用在该对话

中得到的 Record-Route 头字段的 URI。

Record-Route 操作可能是某些特定业务所要求的，这些业务需要代理服务器察看一个对话中的所有消息。但是这种操作降低了系统的处理速度和系统的可扩展性，所以只有在特定的业务要求时，代理服务器才应进行 Record-Route 操作。

Record-Route 的操作是处理能够发起一个对话的 SIP 请求消息。本部分所涉及这类请求只有 INVITE 请求。

14.6.5 其他消息头字段的加入

代理服务器可以在请求消息的拷贝中加入其他合适的头字段。

14.6.6 路由信息后处理

代理服务器可以有本地策略，强制某个请求消息在被发到目的地之前必须经过某些特定的代理服务器。但它必须保证这些特定的代理服务器都是松散路由器。通常来说，只有同一个管理域中的代理服务器才能确切知道彼此是否为松散路由器。这些特定的代理服务器由一系列的 URI 表示，每个 URI 都含有 lr 参数。这些 URI 必须被加到请求消息拷贝中的 Route 头字段的任何已有的值之前。如果请求拷贝中没有出现 Route 头字段，那么需要加入一个新的 Route 头字段，该字段中包含那些必经的代理服务器的 URI。

如果代理服务器有一个本地策略，要求请求消息必须经过某一个特定的代理服务器，可以简单的直接按该代理服务器的地址、所用的端口以及传输方式把请求发送给它。但如果请求消息中有 Route 头字段，那么代理服务器就不能采用这种方法，除非它能确定那个下一跳必经代理服务器是个松散路由器。采用上述在请求消息中插入 Route 头字段值的办法，有着更好的鲁棒性、灵活性、通用性和操作上一致性，因此本标准建议应优先考虑这种方式实现。此外，如果请求的 Request-URI 中包含的是个 SIPS URI，那么在与那个必经代理服务器通信时必须使用 TLS。

如果请求消息的拷贝中包含 Route 头字段，代理服务器必须检查其第一个值所含的 URI。如果那个 URI 没有包含 lr 参数，那么代理服务器必须按如下方式修改请求消息的拷贝：

——首先，代理服务器必须将 Request-URI 移至 Route 头字段中，并将其作为 Route 头字段的最后一个值；

——然后，代理服务器必须把 Route 头字段中的第一个值作为 Request-URI，并将其从 Route 头字段中删除。

在请求消息在通过严格路由实体时，需要一种机制来保证 Request-URI 所携带的信息不被丢失，将 Request-URI 附在 Route 头字段之后则是这种机制的一部分。将 Route 头字段的第一个值"弹出"到 Request-URI 中使得一个严格路由实体在接收请求时可以得到它所希望的消息格式（请求消息的 Request-URI 即为自己的 URI，请求消息需要访问的下一个位置由 Route 头字段的第一个值给出）。

14.6.7 确定下一跳的地址、端口、传输方式

代理服务器可以有一个本地策略，即将请求消息发往一个特定的 IP 地址，使用特定的端口和传输方式，而与请求消息的 Route 头字段和 Request-URI 值无关。如果代理服务器不能确定对应于特定地址、端口、以及传输方式的服务器是一个松散路由器，那么它就不能使用这种策略。本部分不建议采用这种机制发送请求消息到特定下一跳位置。

如果没有这种本地策略，代理服务器按如下方式决定向何处转发请求：如果象 12.6.6 中描述的那样，代理服务器为将请求消息发送到一个严格路由实体而对其作了修改，那么它必须将相应的解析过程应用

于解析请求消息的 Request-URI。否则，代理服务器必须将其用于解析 Route 头字段的第一个值，如果无 Route 头字段，则仍然用于解析 Request-URI。解析过程将产生一个有序的结果集，其中的每个元素是一个由 IP 地址、端口和传输方式构成的三元组。不管使用哪个 URI 解析过程的输入，如果请求消息的 Request-URI 是一个 SIPS URI，代理服务器都要把输入 URI 当作 SIPS URI 来处理。

代理服务器必须尝试按解析结果集的第一个三元组发送请求消息，然后依序继续进行，直到发送成功为止。

在尝试发送的每个三元组，代理服务器必须对消息作适当的修改，并用一个新的客户端事务来发送请求。

既然每次发送尝试都用一个新的客户端事务，即代表一个新的对话分支（branch）。因此，12.6.8 中插入的 Via 头字段中的 branch 参数对每次发送尝试都必须不同。

如果客户端事务报告发送失败，或其状态机超时，代理服务器将按次序尝试解析结果集中的下一条地址。如果该集合中的地址全部试完，则说明请求消息无法转发到目标地址集中当前 URI 所指的实体。此时代理服务器不需要在响应上下文中放置任何信息，就如同该实体返回了一个 408（请求超时）响应。

14.6.8 添加一个 Via 头字段值

代理服务器必须在请求消息的拷贝中已有 Via 头字段值之前插入一个新值。这意味着代理服务器将计算它自己的 branch 参数，该参数为全局惟一的，来标识相应的对话分支，并包含必需的 magic cookie。这也意味着对于循环路由或螺旋路由多次经过某个代理服务器的请求消息而言，每次经过该代理服务器时它得到的 branch 参数是不同的。

如果代理服务器选择进行路由循环检测，它们用于构造 branch 参数的值将有更多的限制。这些代理服务器创建的 branch 参数应当可以分为两部分：第一部分必须满足上述 8.1.1.7 节的要求；第二部分用于执行路由循环检测以及区分循环路由和螺旋路由。

路由循环检测的方法是：当一个请求消息返回到代理服务器时，如果那些对请求消息处理有影响的头字段都没有改变，则认为出现了路由循环。branch 参数中用于路由检测部分的值应当体现出所有这些头字段的信息（包括任何 Route，Proxy-Require 以及 Proxy-Authorization 头字段）。这就保证当请求消息被再次路由到代理服务器时，如果这些头字段中有一个发生改变，就可认为请求消息发生了螺旋路由而非循环路由。创建路由检测部分的值有一个常用方法，即计算一个包括所有相关头字段相关信息的加密哈希表（hash），包括 To tag，From tag，Call-ID 头字段，转换前的原始接收请求的 Request-URI，最顶端 Via 头字段，Cseq 头字段的序号值，以及任何可能出现的 Proxy-Require 头字段和 Proxy-Authorization 头字段。哈希表的产生算法依赖于具体的实现，比较常用的是用十六进制表示的 MD5 算法。

如果代理服务器要检测循环路由，那么它提供的 branch 参数必须依赖于所有对请求处理有影响的信息，包括原始接收请求的 Request-URI 以及任何影响请求消息的接受或路由的头字段。这样作对区分发生路由循环的请求和返回到当前服务器之前路由参数发生改变的请求时是必需的。

计算 branch 参数时不能包括请求的方法。对于非 2xx 响应的 CANCEL 和 ACK 请求与对应的被取消或被确认的请求必须有相同的 branch 参数值。在处理这些请求的服务器上，branch 参数用于将这些请求相互关联起来。

14.6.9 加入 Content-Length 头字段

如果要采用流传输协议将请求消息发往下一跳，请求消息中没有 Content-Length 头字段，那么代理服

务器必须在其中插入一个 Content-Length 头字段，该头字段值等于消息体的长度。

14.6.10 请求消息转发

有状态代理服务器必须为转发请求创建一个新的客户端事务，并指示这个事务使用由本部分中所指定的方法确定请求发送的地址、端口和传输方式。

14.6.11 设置定时器 C

事务用户使用了一个定时器 C，用来处理无法得到最终响应的 INVITE 请求。当 INVITE 请求被代理转发时，每个客户端事务都必须设置这个定时器。定时器 C 的值必须大于 3min。

14.7 响应的处理

当一个 SIP 实体收到响应时，首先它会查找与响应消息相匹配的客户端事务。如果事务相匹配，那么这个响应由所匹配的事务来处理。

客户端事务把响应消息传给代理服务层之后，必须执行下述处理：

1）查找正确的响应上下文；

2）对临时响应更新定时器 C；

3）删除最顶端 Via 头字段值；

4）把响应消息加到响应上下文中；

5）检查响应是否需要立即转发；

6）需要时，从响应上下文中选取最优的最终响应。

如果同响应上下文相关的所有客户端事务都已终止，但还没有送出最终响应，代理服务器必须从它已有的响应中选取一个"最优的"转发出去。

对每个要转发的响应消息都必须作下述处理。每个请求都会至少有一个临时响应和一个最终响应将要转发。

1）必要时汇聚所有的 authorization 头字段值；

2）重写 Record-Route 头字段值（可选）；

3）转发响应消息；

4）生成任何必要的 CANCEL 请求。

消息的处理包括下列步骤：

1. 查找响应上下文

代理服务器使用关键字查找与响应匹配的响应上下文，该上下文是它在转发原始请求之前创建的。接下来的处理都在这个响应上下文中进行。

2. 对临时响应更新定时器 C

对于 INVITE 事务，如果响应消息是一个响应码在 101 到 199 之间的临时响应，代理服务器必须重置相应客户端事务的定时器 C。定时器 C 可以被设为不同的值，但必须大于 3min。

3. Via 头字段

代理服务器必须从响应消息中将最顶端的 Via 头字段值删除

删除之后，响应消息中无 Via 头字段值，则该响应消息就是发给当前实体的，不能再被转发。然后实体按照 UAC 的方式进行处理。

4. 将响应消息加入响应上下文

收到的最终响应被存在响应上下文中，直到与这个上下文相关的服务器端事务生成了一个最终响应。当前加入的响应可能是“最优”最终响应之一，这个“最优”最终响应将由相关的服务器端事务返回。即使当前加入的响应不是最优的，它所携带的信息也可能用于构造“最优”响应。

如果代理服务器要对 3xx 响应中的任何 contact 地址进行递归处理，从而需要将这些地址加入到目标地址集中。在响应加入到响应上下文之前代理服务器必须把这些地址从响应消息中删除。如果原始请求的 Request-URI 是一个 SIPS URI，那么代理服务器不应对非 SIPS URI 进行递归处理。如果代理服务器对 3xx 响应的所有 contact 地址都进行了递归处理，那么它就就不应当把由此产生的无 contact 地址的响应加入到响应上下文中去。

在将 3xx 响应加入到响应上下文之前把被递归处理的 contact 地址从其中删除，可以防止上游下一个实体重复尝试这个地址。

3xx 响应消息中可能同时包含 SIP、SIPS 以及非 SIP URI。代理服务器可以对 SIP 和 SIPS URI 进行递归处理，把其他类型的 URI 放在响应上下文中，它们以后可能会在最终响应中返回。

如果代理服务器收到了一个 416（不支持的 URI 方案）响应，该响应对应的请求的 Request-URI 方案为非 SIP 类型，而所收到的原始请求消息的 Request-URI 却是 SIP 或 SIPS 方案，即代理服务器在代理转发一个请求消息时将其 Request-URI 从 SIP 或 SIPS 改成了其他类型，那么代理服务器应在目标地址集中加入一个新的 URI，这个 URI 应是该非 SIP URI 的 SIP URI 版本。如果那个非 SIP URI 是 tel URI，那么 URI 转换规则如下：

把 tel URI 的 telephone-subscriber 部分作为 SIP URI 的 user 部分；把前一请求发送的主域作为 SIP URI 的 host 部分。

代理服务器通过 SIP 或 SIPS URI 来“递归处理”416 响应，同 3xx 响应，这个 416 响应也不应当被加到响应上下文中。

5. 检查响应是否需要立即转发

以下响应都必须立即转发，直到服务器端事务发送最终响应：

——除 100（Trying）之外的临时响应；

——2xx 响应。

有状态代理服务器收到的 6xx 响应不必立即转发。但是它应取消所有进行中的客户端事务，并且不可以在响应上下文中再创建任何新的对话分支。

INVITE 事务中，在其他对话分支上可能会收到 2xx 响应，此时代理服务器必须转发这个 2xx 响应。如果要求立即转发 6xx 响应，会导致某个 UAC 收到了 6xx 响应之后又接着收到 2xx 响应。本部分要求，代理服务器在收到 6xx 响应后将发出 CANCEL 请求，通常这会使所有未决的客户端事务都收到 487 响应，这样，代理服务器才能往上游方向转发 6xx 响应。

在服务器端事务发出最终响应之后，代理服务器必须立即转发 INVITE 请求的 2xx 响应。

除此之外，有状态代理服务器决不可以立即转发其他任何响应，特别 100（trying）响应。那些以后可能作为“最优”的响应被转发的响应应放在响应上下文中。

有状态代理服务器对非 INVITE 请求恰好只转发一个最终响应；对 INVITE 请求转发一个非 2xx 响应或者一个及多个 2xx 响应。

6. 选择最优响应

经过上述步骤，没有最终响应消息被立即转发，并且响应上下文中的所有客户端事务都已终止，那么有状态代理服务器必须给响应上下文的服务器端事务发送一个最终响应。

有状态代理服务器必须从已收到并存放于响应上下文中的响应中选出一个"最优"的最终响应。

如果响应上下文中没有最终响应，代理服务器必须发送一个 408（请求超时）响应给服务器端事务。

否则，代理服务器必须转发一个存在于响应上下文中的响应。服务器首先选择响应上下文中的 6xx 响应。如果上下文中没有 6xx 响应，服务器就应当选择响应上下文中保存的最低类别的响应，并可以在该类别中选择任意一个响应。代理服务器应优先考虑那些对重发请求提供有用信息的响应。如选择的是 4xx 类响应，就应该优先考虑 401、407、415、420 以及 484 响应。

除非代理服务器能确定所有被转发的后续请求都会产生 503 响应，否则它就不可以向上游转发所收到的 503（业务不可使用）响应。如果代理服务器转发 503 响应则表示自己不能再处理之后的任何请求消息。如果所收到的惟一响应是个 503 响应，代理服务器则应产生一个 500 响应并向上游转发。

例如，如果代理服务器向 4 个地点转发了一个请求，并且分别收到了 503、407、501 和 404 响应，那么它会转发 407（代理服务器需要鉴权信息）响应。

对话建立过程中可能产生 1xx 响应和 2xx 响应。当请求中不包含 To 标签时，UAC 就用响应中的 To 标签来区分创建对话的请求消息的多个响应。如果请求中不含 To 标签，代理服务器就不能在 1xx 或 2xx 响应的 To 头字段中插入标签，同时，代理服务器也不能修改 1xx 或 2xx 响应中的 To 标签。

由于代理服务器不能在没有 To 标签的请求消息的 1xx 响应中插入 To 标签，那么它自己也不能发送非 100 临时响应。然而，它可以把该请求转给一个与自己共享同一实体的 UAS，该 UAS 可以返回自己的临时响应，从而与请求的发送者进入对话的初始状态。该 UAS 的处理不需要与代理服务器分开，它可以是一个虚拟 UAS 并与代理服务器在同一代码空间中实现。

3xx 到 6xx 的响应是逐跳发送的。当发送这些响应消息时，转发实体作为 UAS，基于从下游实体收到的那些响应发出自己的响应。如果一个请求消息中不含 To 标签，那么当实体简单地转发该请求的 3xx 到 6xx 响应时，应当保存响应消息的 To 标签。

请求中如果包含 To 标签的，代理服务器在转发其任何响应是不能修改它们的 To 标签。

代理服务器是否替换被转发的 3xx 到 6xx 响应的 To 标签对上游实体来说没什么不同，但保留原始标签有助于调试错误。

当代理服务器收集了几个响应的信息时，选择其中哪个响应消息的 To 标签是任意的，也可产生一个新的 To 标签，这样可能使错误调试更容易。例如：合并 401（请求未经授权）和 407（代理服务器需要鉴权）响应的认证质询以及合并未加密的和未鉴权的 3xx 响应的 Contact 值。

7. 汇聚 Authorization 头字段值

如果所选的响应是一个 401（请求未经授权）或 407（代理服务器需要鉴权）响应，那么代理服务器必须收集响应上下文中收到的所有的 401 和 407 响应中的任何 WWW-Authenticate 和 Proxy-Authenticate 头字段值，并在转发前将它们原样地加入到所选的响应中。最终发送的 401 或 407 响应中可能会有几个 WWW-Authenticate 和 Proxy-Authenticate 头字段值。

请求消息转发的任意一个或所有的目的地都可能需要鉴权证书（credential），客户端需要接收所有的认证质询（chanllenge）并在请求重发时为每一个质询提供鉴权证书。因此上述做法是必须的。具体内容参见本标准附录 A。

8. Record-Route 头字段

如果所选响应包含一个由当前代理服务器提供的原始 Record-Route 头字段值，那么代理服务器可以在转发响应前重写这个头字段值。这样，代理服务器能对相邻的上游及下游实体提供不同的 URI。对于多宿主主机就可以使用这种机制。

如果代理服务器是通过 TLS 接收请求，然后用非 TLS 连接将其转发出去，那么代理服务器必须把响应消息中 Record-Route 头字段的 URI 重写为 SIPS URI。如果代理服务器是通过非 TLS 连接收到的请求，然后用 TLS 连接将其转发出去，那么代理服务器必须把响应消息中 Record-Route 头字段的 URI 重写为 SIP URI。

代理服务器加入的新的 URI 与插入到请求消息中 Record-Route 头字段的 URI 除了必须满足同样的要求外，还要作如下修改：新的 URI 不应当包含传输参数，除非代理服务器明确相邻的上游实体支持该参数所表示的传输方式，并且该实体又将出现在后续请求的传输路径上。

当代理服务器决定修改响应消息的 Record-Route 头字段时，它需要找到它所插入的 Record-Route 头字段值。如果请求消息发生了螺旋路由，每次重复经过代理服务器时，都会被插入一个 Record-Route 头字段值。本标准建议，当代理服务器要重写 Record-Route 头字段值时，它在请求消息中插入的 URI 应当十分明确，这样重写时才能便于查找。另外一种方法是，代理服务器在 URI 的 user 部分附加一个惟一性标识符，以标记此该代理服务器。

代理服务器在响应消息到达时修改标识符与实例相匹配的第一个 Record-Route 头字段值。修改后的 URI 中 user 部分不再附加实例标识符数据。下一次重复经过该服务器时，使用相同的算法，查找最上面的与实例标识符参数相匹配的 Record-Route 头字段值，就可以准确地找出上一次所插入的 Record-Route 头字段值。

对于有些请求，虽然代理服务器向其加入了 Record-Route 头字段值，但其对应的响应中可能并不包含 Record-Route 头字段。如果响应中确实包含了 Record-Route 头字段，该字段包含代理服务器以前所加入的值。

9. 转发响应

经过上述的处理，代理服务器可以对选定响应进行任何特定操作。代理服务器不可以添加、修改、或者删除消息体。除非另有说明，否则代理服务器决不能删除 Via 头字段值以外的任何其他头字段值。代理服务器不可以删除顶端 Via 头字段值（即原始响应消息的第二个 Via 头字段值）中可能出现的任何 received 参数，该参数是在处理该响应的请求时加入的。代理服务器必须把响应消息发送给与响应上下文相关的服务器端事务进行处理。这将使响应消息被发送到最顶端 Via 头字段值所指的位置。如果服务器端事务不再可用，代理服务器必须用无状态方式转发响应，直接把消息发送到传输层。服务器端事务可能会指示响应消息发送失败，或者发出状态机超时信令。这些出错信息可能会被记入日志以用于错误诊断。

即使最终响应已经发出，只要相关事务没有全部终止，代理服务器也必须一直维护响应上下文的内容。

10. 生成 CANCEL 请求

当代理服务器转发一个最终响应，或者当它收到 6xx 响应时，它都必须对响应上下文的所有未决客户端事务发出一个 CANCEL 请求。本标准中，未决客户端事务是指已经收到了临时响应，但没有收到最

终响应，并且没有收到相关的 CANCEL 请求的事务。

转发最终响应之后取消所有未决客户端事务并不能保证终端不会收到多个 INVITE 请求的 200（OK）响应。在 CANCEL 请求发出并得到处理之前，可能在多个对话分支上就已经产生了 200 响应。

14.8 处理定时器 C

如果定时器 C 超时，代理服务器必须以新值来重置定时器，或者终止相应的客户端事务。如果客户端事务已收到临时响应，代理服务器必须为之生成一个匹配的 CANCEL 请求；如果客户端事务没有收到临时响应，代理服务器必须按其收到了一个 408（请求超时）响应来处理。

代理服务器通过重置超时的定时器 C 来根据当前情况动态地延长事务的生存期。

14.9 处理传输错误

如果传输层通知代理服务器在请求消息转发时发生了错误，代理服务器必须按收到了一个 503（业务不可用）响应的情况来处理。

如果代理服务器在转发响应消息时被通知有错误发生，它必须放弃该响应。代理服务器不应因此取消任何与响应上下文相关的未决客户端事务。

14.10 CANCEL 请求的处理

有状态代理服务器在收到了请求消息的临时响应后可以对它产生的任何其他请求发送一个 CANCEL 请求。当收到 CANCEL 请求时，代理服务器必须取消与之匹配的响应上下文的所有未决客户端事务。

由于终端会通知事务终止的事件，所以有状态代理服务器一般不需要基于 INVITE 请求中 Expires 头字段指示的有效期来来对 INVITE 请求的未决客户端事务产生 CANCEL 请求。

当 CANCEL 请求在由它自己的服务器端事务处理时，有状态代理服务器不会为它创建新的响应上下文。反之，代理服务器从现有的响应上下文中查找与 CANCEL 请求相关的那个请求所在的服务器端事务。如果找到了匹配的响应上下文，那么代理服务器必须立即对 CANCEL 请求返回一个 200（OK）响应。这种情况下当前实体就作为一个 UAS。另外，代理服务器必须为响应上下文中的所有未决客户端事务产生 CANCEL 请求。

如果没有找到相匹配的响应上下文，当前实体没有任何与 CANCEL 请求对应的请求的信息，它必须以无状态方式转发这个 CANCEL 请求。

15 SIP 事务层

15.1 概述

SIP 是事务型协议，一个 SIP 事务是由请求和其响应，包括临时响应和最终响应组成的。INVITE 消息的响应为非 2xx 响应时，对应的事务也包括 ACK 消息。如果其响应是 2xx，则相应的事务中不包括 ACK 消息。

事务可分为客户端事务和服务器端事务。客户端事务发送请求，而服务器端事务发送响应。客户端和服务器端事务都是一种逻辑功能，包含在任何实现此功能的实体中。

一个无状态的代理服务器没有客户端或服务器端事务功能。客户端事务负责从上层实体（称为事务用户 TU，可以是 UA 或者有状态的代理服务器）接收请求，并将请求可靠的发送到服务器端事务；它也负责接收响应并将响应传送给 TU，同时过滤重复的响应和一些非法的响应，如对 ACK 的响应。另外，对于 INVITE 请求的非 2xx 的最终响应，客户端事务还负责产生相应的 ACK 响应。

服务器端事务负责从传输层接收请求并发送给上层 TU，同时滤掉从网络上收到的重复的请求。它也

负责从 TU 接收响应并将其发送给传输层。在 INVITE 请求事务中，服务器端事务负责吸收对非 2xx 的最终响应的 ACK。

2xx 的响应的重传和和对应的 ACK 请求的产生是由 UA 内核完成的，由 UAS 重传 2xx 响应，UAC 负责产生对应的 ACK。这种端到端的处理使呼叫者能够了解接受呼叫的所有用户的完整列表。所以在路径上的每个代理服务器都仅仅是转发 INVITE 的 2xx 响应以及对应的 ACK。

15.2 客户端事务

15.2.1 简述

客户端事务通过维护一个状态机来实现相应的功能。上层 TU 与客户端事务通过一个简单的接口进行通信。当 TU 初始化一个新的事务时，它产生一个新的客户端事务实例，要发送的 SIP 请求传送给实例，并带上要发往目的地的 IP 地址、端口号和传输协议等参数。客户端事务实例开始执行状态机，同时将有效的响应上传给 TU。

根据 TU 发送的请求的方法类型，有两种客户端事务实例的状态机。一种是处理 INVITE 请求的客户端事务实例，称为 INVITE 客户端事务；另一种是处理除 INVITE 和 ACK 之外的所有请求的客户端事务实例，称为非 INVITE 客户端事务。没有对应于 ACK 的客户端事务存在。如果 TU 要发送一个 ACK，它将直接将对应的请求发送给传输层。

15.2.2 INVITE 客户端事务

15.2.2.1 INVITE 事务简述

INVITE 事务由三次握手组成：客户端事务发送 INVITE 请求，服务器端事务发送响应，然后客户端事务发送 ACK 消息。对于 UDP 等不可靠传输，客户端事务开始以 T1 的间隔重传请求，并在每次重传后将定时间隔加倍。T1 是往返时间（RTT）的估计值，缺省为 500ms。本部分所涉及的大部分事务定时器都是以 T1 为度量标准的，可以通过更改 T1 的值来更改其他定时器的值。对于可靠传输，请求不进行重传。在收到 1xx 的临时响应后，停止重传，客户端事务等待进一步的响应。对于不可靠传输服务端事务可以发送另外的 1xx 响应，最后发送最终响应。对于不可靠传输，响应周期性的进行重传，而对于可靠传输，响应只传输一次。对于每个收到的最终响应，客户端事务都发送 ACK 消息，以便使对端停止重传响应。

15.2.2.2 状态机模型

INVITE 客户端事务的状态机如图 3 所示。在 TU 用 INVITE 请求来生成新事务时，客户端事务首先进入 Calling 状态，同时客户端事务将请求传送给传输层进行网络传输。如果使用不可靠传输协议，客户端事务必须用 T1 启动定时器 A；如果使用可靠传输协议，客户端事务不必启动定时器 A。无论何种情况，事务都必须用 64 × T1 启动定时器 B（定时器 B 控制整个事务的超时）。

如果定时器 A 超时，则客户端事务必须将请求进行重传，然后必须将定时器的间隔设为 2 × T1 进行更新并重新启动。若 A 在 2 × T1 秒后超时，则继续重传请求，同时 A 的间隔时间继续加倍，如此反复。上述重传只在“Calling”状态下进行。T1 的值可根据具体的网络性质进行一定的调整，但上述的重传过程必须遵循。

如果在事务处于“Calling”状态的情况下，定时器 B 超时，则事务应该通知 TU 超时的发生。事务层不能产生 ACK。定时器 B 的值在不可靠传输的情况下刚好等于重传 7 次请求的时间。

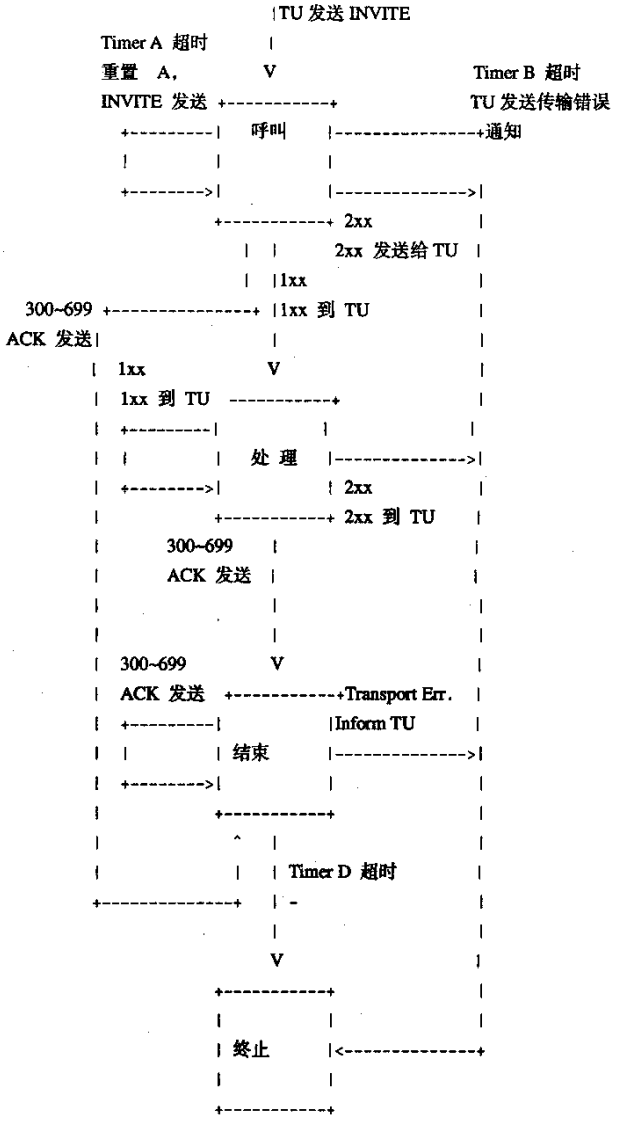


图 3 INVITE 客户端事务

如果事务在“呼叫”状态收到了临时响应，则状态迁移到“处理”状态。在此状态，事务不应该重传请求。同时临时响应必须上传给 TU。在此状态，每个收到的临时响应都要上传给 TU。

无论在“呼叫”还是“处理”状态，在收到状态码为 300~699 的响应时，事务的状态都迁移移到“完成”状态，并且将响应上传到 TU，同时必须生成相应的 ACK 并发送到传输层进行传输。ACK 发往的地址必须和请求发往的地址具有同样的 IP 地址，端口号和传输协议。在进入“完成”状态后，客户端事务应该启动定时器 D，对于不可靠传输定时时间至少为 32 秒，而对可靠传输时间为 0 秒。

在“完成”状态，任何收到的重传的最终响应都必须使本端事务重传 ACK，但这些收到的重传响应不再上传给 TU。如果在此状态定时器 D 超时，则事务必须立刻迁移到“终止”状态，并将超时事件上报给 TU。

在“呼叫”或“处理”状态，如果收到 2xx 的响应，则事务立刻迁移到“终止”状态，同时将响应上传给 TU。这个响应的处理方法取决于 TU 是代理内核还是 UAC 内核，对于 UAC 内核将产生相应的

ACK，而 Proxy 内核则仅仅前转 200（OK）即可。这种处理的不同就是 200（OK）的处理不放在事务层的原因。

事务一旦进入“终止”状态，就必须将实例清除。如果传输层收到一个找不到对应事务的响应，则直接将响应发送给 TU。

15.2.2.3 ACK 请求的生成

事务层生成的 ACK 请求必须与原发送的请求具有同样的 Calll-ID，From，和 Request-URI 头字段。ACK 的 To 头字段必须与对应的响应的 To 头字段具有同样的值，一般情况下，因为多了 Tag 参数而与原请求的 To 头部不同。ACK 请求必须具有单个的 Via 头字段，而且是与原请求的 Via 头字段顶部值具有同样的值。Cseq 头字段必须具有与原请求同样的序列号，且方法必须是“ACK”。

如果原 INVITE 请求具有 Route 头字段，则这些字段也必须出现在 ACK 请求中，这是为了保证 ACK 能被接下来的无状态的代理服务器正确的路由。

本部分建议，在对应非 2xx 最终响应的 ACK 请求中不应包含消息体。如果有消息体的话则分两种情况：对应响应不是 415，ACK 中的消息体类型要严格限制在 INVITE 中包含的消息体类型范围内；如果响应是 415，则 ACK 中包含的消息体的类型可以是在响应的 Accept 头字段中所列的消息体类型。

15.2.3 非 INVITE 客户端事务

15.2.3.1 非 INVITE 事务的简述

非 INVITE 事务不使用 ACK，是简单的请求-响应的模式。对于不可靠数据传输，请求将首先以 T1 的时间间隔进行重传，并且每次超时将 T1 加倍，直到与 T2 相等。如果收到了临时响应，请求的重传将继续，但重传的时间间隔为 T2。服务器端事务只有在收到重传请求的情况下才将最后发送的临时或最终的响应进行重传。因此客户端事务在收到临时响应的情况下仍然重传请求是为了能使最终响应得到可靠的传输。

非 INVITE 事务对 2xx 响应无特殊处理。

15.2.3.2 状态机模型

非 INVITE 客户端事务的状态机与 INVITE 客户端事务的状态机非常相似。在 TU 生成新事务时事务实例进入“进行中”状态，此时应该设置定时器 F（值为 64×T1 秒），同时将请求发送到传输层进行传输。如果使用不可靠传输协议，则必须设置定时器 E，时间为 T1，如果仍在该状态且 E 已到期，则复位定时器 E，但时间设置为 min（2×T1，T2）。如果定时器再次到期，则复位 min（4×T1，T2），依此类推，直到达到 T2，然后将定时器时间设为 T2。T2 的缺省取值为 4s。所以对于 T1 和 T2 的取值，这个间隔为 500ms、1s、2s、4s、4s、4s、等等。

如果事务在“进行中”状态定时器 F 到期，则事务实例应该通知 TU 该超时事件，同时状态迁移到“终止”状态。若在此状态期间收到临时响应，则事务实例将临时响应发送给上层 TU，然后将实例状态迁移到“处理”状态；若在此状态期间收到最终响应（状态码为 200~699），则事务实例必须将响应发送给上层 TU，然后将实例状态迁移到“结束”状态。

如果事务在“处理”状态定时器 E 到期，则必须将请求再次发送给传输层进行重传，同时定时器 E 复位，值为 T2 秒。若在此状态定时器 F 到期，则事务实例必须通知 TU 超时事件，并将状态迁移到“终止”状态；若在此状态收到最终响应（状态码为 200~699），则事务实例必须将响应发送给上层 TU，然后将实例状态迁移到“结束”状态。

一旦客户端事务实例进入“结束”状态，事务必须启动定时器 K，对于不可靠传输定时时间为 T4 秒，而对可靠传输时间为 0 秒。“结束”状态的存在是为了在不可靠传输时吸收重传的响应。而 T4 则代表了等待的时间，T4 的缺省值为 5 秒。若在此状态定时器 K 到期，则事务实例必须立刻迁移到“终止”状态。

一旦客户端事务实例进入“终止”状态，实例必须立刻清除。

具体过程见图 4。

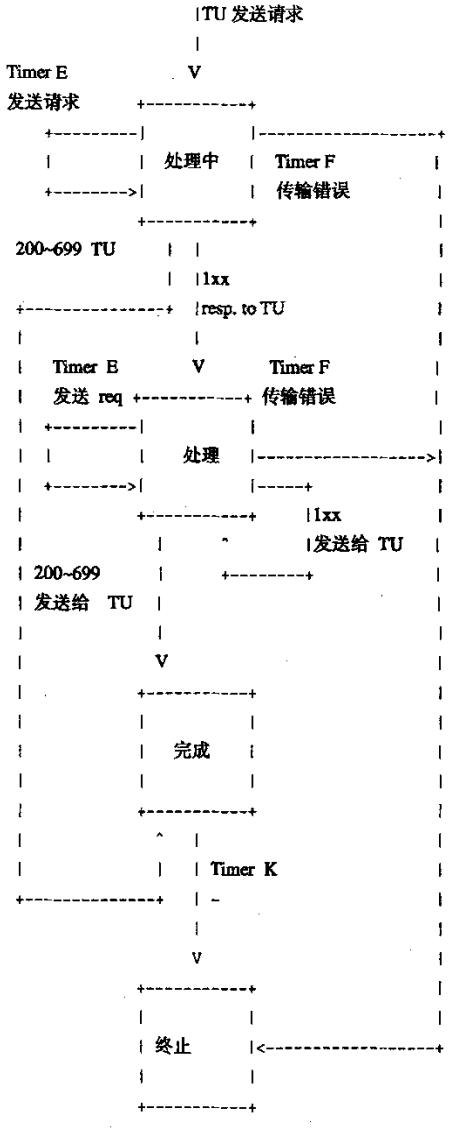


图 4 非 INVITE 客户端事务

15.2.4 响应与客户端事务的匹配

当客户端传输层收到一个响应的时候，它必须决定由某个事务来遵循上述过程处理这个响应。在以下两种情况同时满足时，响应和一个客户端事务相匹配：

1）响应中 Via 头字段顶部中的 branch 参数与产生此事务的原请求中 Via 顶端头字段中 branch 参数是相同的；

2）响应中 Cseq 头字段中的方法参数与产生此事务的原请求中 Cseq 头字段中的方法参数是相同的。

此处需要方法匹配是因为 CANCEL 请求与原请求虽属不同的事务，却具有相同的 branch 参数值。

先后两个匹配同一个事务的的响应是重传的响应。

如果请求是经多播发送的，那么它将从不同的服务端返回多个响应。所有的这些响应在 Via 顶端头字段中都具有相同的 brance 参数，但具有不同的 To 标签值。根据上述的规则，则第一个响应会被正常处理，而其他的响应则会视为重传。这并非错误，本标准规定，在多播发送的情况下只处理一个响应消息。

15.2.5 传输错误处理

在客户端的事务实例将相应的请求发到传输层后，在传输层传输失败的情况下事务层实例应该将传输失败通知给上层 TU，并将实例的状态迁移为“终止”。

15.3 服务器端事务

服务器端事务负责将请求发送给上层 TU，并将相应的响应发送给传输层进行网络传输。服务器端事务同样通过状态机来实现相关的功能。在收到一个新的请求时，生成一个新的服务器端事务。和客户端一样，状态机也根据请求是否为 INVITE 而分为两种状态机模型，流程见图 5。

15.3.1 INVITE 服务器端事务

一个新的 INVITE 请求到来时，服务器端生成新的 INVITE 事务实例，并进入“处理”状态。如果不清楚 TU 是否会在 200ms 内产生临时响应或最终响应，事务实例必须生成 100（Trying）临时响应。这个 100 响应可以用来快速避免因请求的重传而造成的网络拥塞。请求消息必须上传给 TU。

TU 可以发送多个临时响应给服务器端事务实例。在“处理”状态下，事务实例将这些响应发送给传输层进行网络传输，但不影响实例的状态。若在此状态收到重传的请求，则事务实例将最近一次的响应进行重传。若在此状态收到 TU 的 2xx 响应，则事务实例将 2xx 响应发送到传输层进行传输，但并不负责此响应的重传，2xx 响应的重传由 TU 负责。然后事务实例将状态迁移到“终止”状态。若在此状态收到 TU 的 300~699 的响应，则事务实例将响应发送到传输层进行传输，并将状态迁移到“完成”状态。对于不可靠的传输，启动定时器 G，值为 T1。

一旦进入“完成”状态，事务启动定时器 H，值为 64×T1。定时器 H 决定了何时放弃重传响应，它的取值与客户端事务定时器 B 一致，而 B 决定了客户端重传请求的时长。若定时器 G 到期，则重传响应，同时将重新启动 G，其取值为 2×T1 和 T2 的最小值，此后每次 G 到期都重传响应，并将定时时长加倍，直到其值等于 T2 后，将一直取 T2 为时长。若在此状态收到重传的请求，则应该重传响应。

若在“完成”状态收到 ACK 请求，则事务必须迁移到“确认”状态。在此状态，忽略定时器 G，所以重传将停止。

若在“结束”状态定时器 H 到期，表示事务没收到 ACK，则事务将状态迁移到“终止”状态，并通知上层 TU 事务失败。

在“确认”状态，事务实例可以吸收因重传响应而引起的重传的 ACK。进入“确认”状态后，事务启动定时器 I，对于不可靠传输定时时间为 T4 秒，而对可靠传输时间为 0 秒。若在此状态定时器 I 到期，则事务实例必须立刻迁移到“终止”状态。

一旦事务实例进入“终止”状态，实例必须立刻清除。具体流程如图 5 所示。

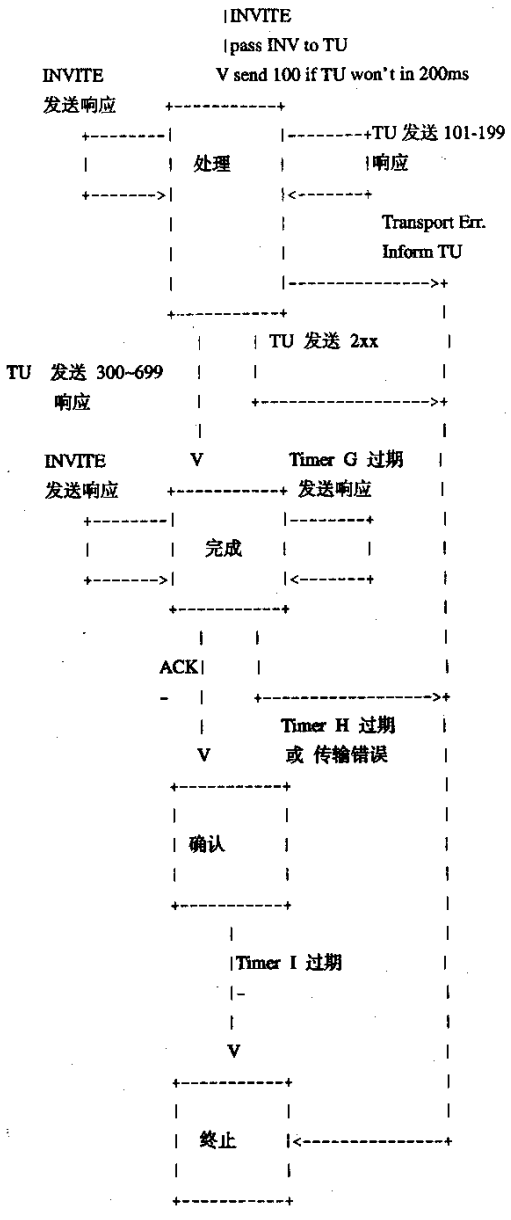


图5 INVITE 服务器端事务

15.3.2 非 INVITE 服务器端事务

对于非 INVITE 和 ACK 的请求，生成非 INVITE 服务器端事务实例，初始状态机为“处理中”状态，并将该请求发送给上层 TU。一旦进入“处理中”状态，任何收到的重传请求都将被丢弃。

在“处理中”状态，若收到上层 TU 的临时响应，服务器端事务实例进入“处理”状态，并将临时响应发送到传输层。在此状态从 TU 接收到的任何临时响应都将发送到传输层进行网络传输。若在此状态收到重传的请求，则事务将最近的响应进行重传。若在此状态收到 TU 的最终响应（状态码 200~699），则事务迁移到“完成”状态，同时将响应发送到传输层进行发送。

当事务进入到“完成”状态，必须启动定时器 J，对于不可靠传输值为 64×T1 秒，对于可靠传输则为 0 秒。在“完成”状态，收到重传请求的情况下事务将最后的响应进行重传；并且在此状态，事务收

到的任何 TU 的最终响应都将丢弃。当定时器 J 超时后，事务迁移到“终止”状态。

一旦事务实例进入“终止”状态，实例必须立刻清除。具体流程如图 6 所示。

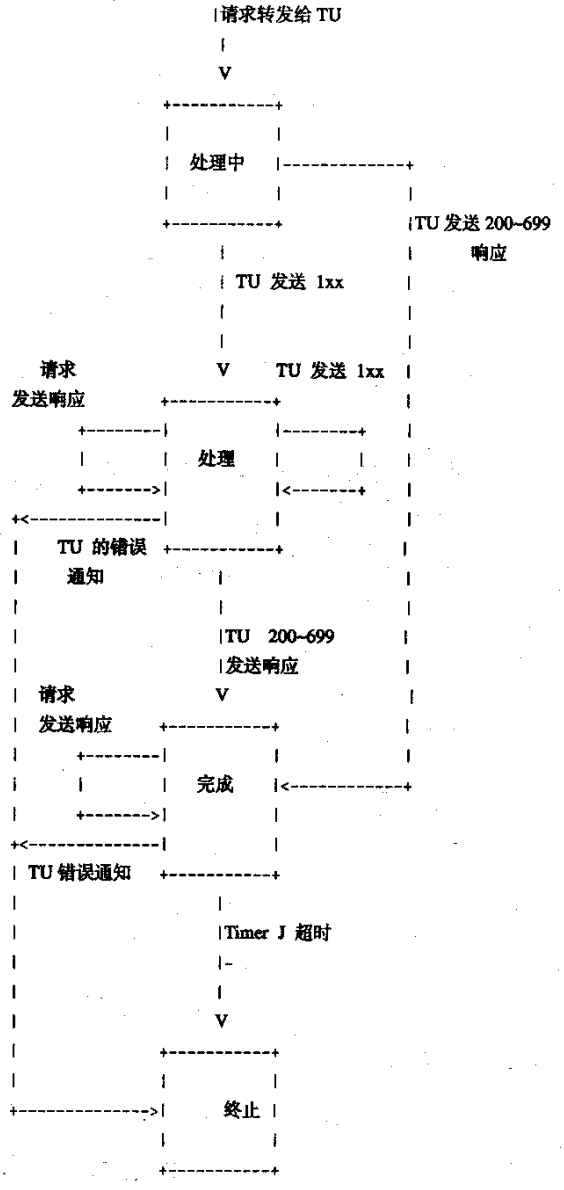


图 6　非 INVITE 服务器端事务

15.3.3　请求与服务器端事务的匹配

在服务器收到网络上的一个请求后，它必须将请求与已有的事务进行匹配，匹配规则如下：

首先检查请求的最顶端的 Via 字段中的 branch 参数。如果 branch 参数存在并且以“z9hG4bK”开始，说明此参数是由与兼容的客户端事务产生的。所以此 branch 参数在此客户端产生的事务中应该是惟一的。需要满足下列条件，请求与事务相匹配：

1）请求中的 branch 参数与产生事务的请求最顶端 Via 中的 branch 参数相同；

2）请求中的 send-by 值与产生事务的请求相同；

3）除了 ACK 请求，请求中的 Method 要与产生事务的请求相同。

匹配规则对于 INVITE 以及非 INVITE 事务都适用。

其中 send-by 参数用于匹配规则是因为有可能不同客户端事务所产生的 branch 参数有可能相同。

如果请求中不存在 branch 参数或 branch 参数不以“z9hG4bK”开始，按照下面的规则进行匹配：

如果 INVITE 请求的 Request-URI，To 标签，From 标签，Call-ID，Cseq 和顶端 Via 头字段与产生事务的 INVITE 请求相匹配，则该请求与事务相匹配。该情况下，INVITE 请求为原请求的重传请求。如果 ACK 请求的 Request-URI，From 标签，Call-ID，Cseq 中的数字（不包括方法）和顶端 Via 头字段与产生事务的 INVITE 请求相匹配，并且该请求的 To 标签与服务器端事务产生的 To 标签相匹配，则该 ACK 请求与事务相匹配。头字段对于所有其他的请求方法，如果 Request-URI，To 标签，From 标签，Call-ID，Cseq 和顶端 Via 所有头字段与产生事务的请求相匹配，则匹配成功。如果一个非 INVITE 请求与一个存在的事务相匹配的话，则认为它是产生这个事务的请求的重传。

由于以上匹配规则与 Request-URI 有关，所以服务器端事务无法将一个响应与存在的事务匹配起来。当 TU 传送一个响应给服务器端事务时，它必须被传送给响应所标明的指定的服务器端事务。

15.3.4 传输错误处理

在服务器端的事务实例将相应的响应发到传输层后，在传输层传输失败的情况下，可进行如下处理：

首先，可以选择将响应发往另一个作为备份的实体。

如果上述尝试都传输失败，事务层应该将失败通知给 TU，并将相应的事务实例的状态迁移为“终止”。

16 传输

传输层负责通过网络传送请求和响应。传输分为面向连接传输和非面向连接传输。当传输为面向连接时，还要确定请求或者响应可用的连接。

传输层可以管理永久连接，这些连接使用的传输协议有 TCP 和 SCTP，或者基于 TCP 和 SCTP 的 TLS，包括那些开放的传输层的协议。这些连接还包括客户机或者服务器传输输层的开放连接，客户机和服务器传输层可以共同使用该连接。可以通过该连接的远端地址、端口、传输协议所组成的三元组来检索该连接。如果一个连接开放，该索引设成目的地 IP 地址、端口和传输协议的三元组。如果一个连接被传输层所接受，该索引设成源 IP 地址、端口和传输协议的三元组。由于源端口通常是暂时的，所以传输层所接受的连接一般不会重新使用。所以通常两个对等代理服务器在使用面向连接传输协议时，会使用两个连接，用于各自方向发起的事务。

本标准建议，在最后一个消息通过之后，该连接仍需要开放一段时间。该时间段至少等于实体将一个事务从实例进行到终止状态所用时间的最大值。这样，事务可以在其发起的连接完成（例如请求、响应、INVITE、用于非 2xx 响应的 ACK）。该时间段的值至少为 64×T1。然而如果一个实体使用一个更大的定时器 C，这个持续时间将更长。

16.1 客户端

16.1.1 发送请求

传输层的客户端负责发送请求并接受响应。传输层的用户将请求、IP 地址、端口、传输协议以及用于组播目的地的 TTL 传递到客户机传输层。

如果一个请求在小于 200 字节或者大于 1300 字节的路径 MTU 中，而该 MTU 值未知，那么该请求必须使用拥塞控制协议。如果这与顶端 Via 中所指定的传输协议不同，那么顶端 Via 的值需要修改。这有效的防止了使用 UDP 会造成的消息碎片，并且为大块的消息提供了拥塞控制。对于 UDP 来说能够处理

的最大数据包就是65，535 字节（包括IP和UDP字头）。

消息尺寸和MTU之间的200字节的“缓冲区”适应了SIP中响应消息一般比请求大的情况。例如，可以在INVITE响应中附加了Record-Route头字段值。有了这个额外的缓冲区，响应可以比请求大170字节而在Ipv4中不必被分块（假设没有IPSec时，IP/UDP消耗大约30字节）。当路径MTU未知时，基于以太网的MTU是1500字节来将MTU设为1300字节。

如果一个实体由于消息大小的缘故才需要通过TCP发送某请求，否则该请求可以通过UDP传输，如果建立该连接时引起下面两种情况：ICMP协议不支持，和重新建立TCP连接，此时该实体就应该使用UDP重新发送请求。

向组播地址发送请求的客户机必须在Via头字段值中附加“maddr”参数指明目的地组播地址，对于Ipv4来说应该增加一个值为1的“ttl”参数。IPv6的组播的内容不在本部分范围之内。这个规定对SIP组播产生了限制。它的基本功能是“单跳发现（single-hop-discovery-like）”业务，也就是向一组同类的服务器发送请求，但是只需要其中一个服务器处理该请求即可。注册时可以使用该功能。见本部分15.1.3中事务处理的规定，客户机事务层应该接收第一个请求，其余的因为包含同样的Via分支标识符而被视为重发。

发送请求之前，客户端事务层必须在Via头字段中插入一个“sent-by”字段。该字段包括一个IP地址或者是主机名和端口。本部分建议使用FQDN（全资格域名）方式。该字段用于以下情况下发送请求。如果没有规定端口，根据传输协议不同而取不同的端口缺省值，UDP取5060而TCP和SCTP取5061。

在进行可靠传输时，应在收到请求的连接上发送响应。因此，客户端传输层必须准备在同一个连接上接收响应。发生错误时，服务器可以建立一个新连接发送响应。在这种情况下，传输层也必须在源IP地址上引入一个连接，从源IP地址和“sent-by”字段中所指的端口发送请求。它还必须准备在服务器端选择的地址和端口上引入一个连接。

在不可靠的单播传输时，客户端传输层必须准备在源IP地址上接收响应，源IP地址就是发送请求的地址，端口号为“sent-by”字段中的值。另外，与可靠传输一样，某些特定情况下，响应也将被传输到其他地方。

在组播的情况下，客户端传输层必须作为发送请求的组播群中的一员在与发送请求相同的组播群和端口上接收响应。

如果一个原有的连接对于某请求的目的IP地址、端口和传输协议开放，则本部分建议这个连接用于发送请求而另外一个连接也可以开放并使用。

如果某请求使用组播发送，它就会被送往一组用户所提供的地址、端口和TTL。如果某请求使用不可靠的单播传输方式发送，它就会被送往传输用户所提供的IP地址和端口。

16.1.2 接收响应

客户机传输层收到响应后需要检查顶端Via的值。如果“sent-by”参数中的值与客户端传输层插入到请求中的值不一样，该响应就必须被丢弃。

如果存在一些客户端事务，客户机传输层将在原有的事务中查找与该响应所匹配的事务。如果找到匹配事务，就必须将该响应送到对应的事务中处理。反之该响应就必须被送到内核服务器做进一步处理。根据内核服务器的不同，对这些响应有不同的处理。

16.2 服务器

16.2.1 接收请求

服务器可以通过 DNS 来搜索对方服务器，搜索结果以 SIP 或者 SIPS URI 形式给出，服务器应该通过搜索结果中的任何 IP 地址、端口和传输协议来接收请求。另外，必须将一个 URI 放到 REGISTER 请求或者重定向响应中的 Contact 字段，或者放在请求或者响应中的 Record-Route 头字段中。URI 也可以用放到网页或名片上的形式给出。本部分建议，服务器在公共接口上接受请求是应使用默认的端口值（TCP 和 UDP 是 5060，而 TCP 上的 TLS 是 5061），私有网络或者同一个主机上运行多个服务器的情况除外。因为如果消息太大则必须使用 TCP 而不能用 UDP 传输，所以任何使用 UDP 端口和接口的服务器必须能够在 TCP 上使用同样的端口和接口。反之，服务器不必因为使用了 TCP 的一个特定的端口和接口就要求在 UDP 情况下使用同一特定端口和接口。服务器传输层通过任何传输协议收到请求后，必须检查 Via 头字段顶端的“sent-by”参数的值。如果“sent-by”参数的主机位置包含一个域名或者包含一个不同于数据包源地址的 IP 地址，服务器必须在 Via 头字段中添加一个“received”参数。该参数必须包括所收数据包的源地址，这样，服务器传输层就可以将响应发送到发出请求的源 IP 地址。

然后，服务器传输层将在服务器端事务中寻找与请求相匹配的事务。如果找到了对应的服务器事务，就将请求送到这个事务来处理；如果没有找到相匹配的事务，就将请求送到内核服务器，并建立一个新的服务器端事务来处理它。如果 UAS 内核服务器对 INVITE 请求发出 2xx 响应，服务器事务即终结，这就意味着如果 ACK 到达，将没有相匹配的服务器事务。基于上述原则，ACK 就被送往 UAS 内核进行处理。

16.2.2 发送响应

服务器传输层使用 Via 字段的顶端值来决定将响应发送目的。具体过程如下：

- 如果“sent-protocol”为可靠的传输协议，如 TCP 或者 SCTP，或者基于 TCP 或 SCTP 的 TLS，且现有的连接仍然可用，则必须使用该连接传输响应到建立该事务的初始请求的源地址。这就要求服务器传输层保持服务器端事务和传输连接的关联。如果该连接不可用，服务器应与“received”参数中的 IP 地址之间开放一个新连接，并使用“sent-by”中的端口值，如果没有指定端口就使用该传输协议的缺省端口值。如果该连接失败，服务器应该确定所要开放的连接的 IP 地址和端口并向其发送响应消息；
- 否则，如果 Via 字段包含“maddr”参数，就必须将响应转发到参数中所列的地址，所用端口为“sent-by”中指定端口值，如果没有指定，端口值为 5060。如果地址为组播地址，就应该使用“ttl”参数中指定的 TTL 值来发送响应，如果该参数不存在，TTL 的值为 1；
- 在不可靠的单播传输的情况下，如果 Via 含有“received”参数，则响应就被送往“received”中指定的地址，所用端口为“sent-by”中指定端口值，如果没有指定，端口值就为 5060；
- 否则，如果没有接收者标签，就应该将响应送到“sent-by”中指定的地址。

16.3 数据帧

非面向消息传输的情况下，如果消息包含 Content-Length 头字段，则消息体则被应为该字段所指示的长度。超长的数据包必须被丢弃。如果传输的数据包在消息体结束之前结束，则产生错误。如果消息是一个响应，则它必须被丢弃。如果消息为一个请求，实体就应该产生一个 400（Bad Request）响应。如果该消息没有 Content-Length 头字段，那么消息体的长度就认为是所传输的数据包的长度。

面向流传输的情况下，Content-Length 头字段指示消息体的大小。面向流传输的情况下必须使用 Content-Length 头字段。

16.4 错误处理

如果传输的用户通过不可靠传输协议传输消息，并产生 ICMP 错误，则按照 ICMP 错误的类型确定处理方法。如果为主机、网络、端口或者协议不可用的错误或者参数错误，应由传输层通知用户。源终止和 TTL 超时的 ICMP 错误都将忽略。

如果传输的用户要求通过可靠传输协议传输消息且连接失败，那么传输层应该通知用户。

17 普通的消息组成

17.1 SIP 和 SIPS URI

SIP 和 SIPS URI 用于指示一个通信资源。像所有的 URI 一样，SIP 或 SIPS URI 也可以用于网页、电子邮件消息和出版物。其中包含了与该通信资源建立并维持会话所需要的信息。

通信资源的例子如下：

- 在线业务的用户；
- 多方电话的情况；
- 消息系统的语音信箱；
- 网关业务的 PSTN 号码；
- 某组织里的一个组。

SIPS URI 所指示的为安全的资源，即，UAC 与该 URI 所属的域之间使用 TLS 传输。该资源与某个用户间通信是安全的，且所使用安全机制由该域的本地策略所决定。如果希望得到安全的通信，任何由 SIP URI 所指示的资源在改变其 URI 方案之后可以升级至 SIPS URI。

17.1.1 SIP URI 的构成

“SIP URI”和“SIPS URI”的方案见 RFC 2396。格式类似于邮件 URL，分别规定 SIP 头字段和 SIP 消息体。这就可以在网页上或者电子邮件消息里使用 URI 规定会话的主题、媒体类型和紧急程度等。SIP URI 的通用格式为：

sip：user：password@host：port;uri-parameters?headers

SIPS URI 的格式也是一样的。

以上符号定义如下：

user：指定被寻址的主机资源的标识符。“host”通常用来指示一个域。URI 中“userinfo”由 user 和 password 字段以及 @ 符号组成。如果目的主机没有用户或者主机本身就是被指定的资源，则 userinfo 部分为可选。如果有 @ 符号，那么 user 字段就不可为空。

Password：password 与 user 相关。SIP 和 SIPS URI 的语法中允许该字段存在。但是本部分中不建议使用该字段。因为鉴权信息以明码文本的形式通过，几乎在所有的情况下都不安全。password 只是 user 部分的延伸，本部分不对其进行特殊的定义，可以把“user：password”部分看作一个字符串。

Host：指定 SIP 资源。Host 部分包含一个 FQDN 或者是一个数值表示的 IPv4 或者 IPv6 地址。本部分建议该字段使用 FQDN 方式。

Port：指定请求要被发送的端口。

URI parameter：包括 transport，maddr，ttl，user，method 以及 lr 参数。这些参数用于以 URI 构建一个请求，它们附加在 hostport 之后，以分号隔开，其格式为：parameter-name“=”parameter-value。虽然一个 URI 可以有若干个参数，但是一个参数名最多只能出现一次。

Transport：用来确定发送 SIP 消息的传送机制。SIP 可以使用任何网络传送协议。各种协议的参数名的定义见 UDP、 TCP 和 SCTP 的相关规范。

Maddr：用于指定用户所要联系的服务器地址，它优先于 host 字段的地址。Maddr 存在时，URI 中的 port 和 transport 字段都用于 maddr 所指定的地址。为了发送请求，需要获得目的地的地址、端口以及传输协议。Maddr 字段是松散源路由的一个简单的形式。它允许 URI 指定一个到达目的地时必须通过的代理服务器。本部分不建议在该路径上继续使用 maddr 参数，而是应该使用文件中讲的另外一种路由机制来预先建立一个路径组，它含有一个描述所要通过的节点的完整 URI。

Ttl：UDP 组播数据包的生存时间值，只能用于 maddr 为组播地址并且传送协议是 UDP 的情况下。例如确定对于 alice@atlanta.com 的呼叫，使用 ttl 值为 15，组播地址为 239.255.255.1，URI 形式如下：

sip：alice@atlanta.com;maddr=239.255.255.1;ttl=15

这组有效的 telephone-subscriber 字符串是有效的 user 字符串的一个子集。URI 中的 user 参数用来区分那些电话号码和 user 名不同的 URI。如果 user 字符串包含一个格式为 telephone-subscriber 的电话号码，则应该存在“phone”这个值。即使没有这个参数，如果本地用户名的规定允许，SIP URI 也可以把@之前的部分看做一个电话号码。

Method：规定由 URI 构建 SIP 请求的方法，由方法参数指定。

lr：该参数存在的情况下，用来指示负责该资源的实体实施路由机制。该参数可以存在于 URI 代理服务器中 Record-Route 头字段，也可以存在于预先建立的路由组 URI 中。一个基于 URI 发送的请求如果没有 lr 参数，则接收实体就被认为执行的严格路由机制并重定消息格式来保存 Request-URI 中原有信息。既然 URI 参数机制是可扩展的，SIP 实体就必须忽略其不理解的 URI 参数。

Headers：该字段位于由 URI 构建的请求消息中。SIP 请求的 Headers 字段可以在 URI 中用“?”来指定。Header 的名字和值是由&分开的 hname = hvalue 对。当 hname 为“body”时表明相关的 hvalue 是 SIP 请求的消息体。

表1 SIP URI 中各成分的使用及其缺省值

	缺省	请求 URI	To	From	Contact	R-R/Route	外部
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	UDP	o	-	-	o	o	o
lr-param	--	o	-	-	-	o	o
other-param	--	o	o	o	o	o	o
headers	--	-	-	-	o	-	o

(1)：缺省的 port 值与传送协议有关。Sip：使用 UDP，TCP，或者 SCTP 时为 5060。Sip：使用基于 TCP 的 TLS 是 5061。

表一总结了在URI出现在不同的上下文中时SIP和SIPS URI各个组成部分的用法。“外部”这一列讲述了URI位于SIP消息体之外的情况，例如网页上或者商业名片上。标注“m”表示强制使用，标注“o”表示可选，标注“-”表示禁止。如果URI中有禁止项存在，SIP实体处理到它的时候应该忽略这些项。第二列指出如果一个可选项没有出现时的缺省值。“--”指出这项或者不可选或者没有缺省值。

Contact头字段中的URI根据字段出现的上下文的不同而有不同的限制。一组适用于建立和维持对话的消息（INVITE和200响应），另外一组适用于注册和重定向消息（REGISTER，对它的200（OK）响应以及对于任何方法的3xx响应）。

17.1.2 字符转义的要求

在SIP URI中，定义一组需要转义的字符时，就要遵循RFC 2396中的要求使用“%”“HEX HEX”机制进行转义。即任何给定的URI组成部分中所保留的字符集实际是由该组成部分所定义的。一般情况下，如果一个URI的字符使用转义的US-ASCII码所代替后语义被改变，这个字符就应该保留。US-ASCII字符之外的字符，例如空格、控制符还有URI中的分隔符都必须被转义。URI不可以包含未被转义的空格和控制符。

对于每个部分来说，有效的BNF的扩展定义了可以不用转义的字符。除此以外的字符都要被转义。例如"@"不是user成分的字符，因此user为j@s0n时，至少@符需要编码为“j%40s0n”。扩展的hname和hvalue符号表明所有的URI头字段名和值中保留的字符必须被转义。

User部分的telephone-subscriber子集的转义有特殊的要求。telephone-subscriber未保留的字符集中包含许多不同语法形式的字符，当使用SIP URI时，这些字符需要被转义。任何telephone-subscriber中出现的字符如果没有出现在BNF的扩展定义中则必须被转义。

SIP和SIPS URI中的host部分不允许字符转义（%字符在扩展中无效）。将来国际化域名定案之后这个规定可能会发生改变。

17.1.3 SIP URI举例

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

sip：alice@192.0.2.4

sip：atlanta.com;method=REGISTER?to=alice%40atlanta.com

sip：alice;day=tuesday@atlanta.com

最后一个例子的user字段值为“alice;day=Tuesday”。转义规则允许该字段中分号保留未转义。本协议的目的就是要求该字段不透明，所以这个值如何构成也仅仅在负责该资源的SIP实体中有意义。

17.1.4 URI比较

本标准中定义的某些操作要求确定两个SIP或SIPS URI是否等价。本标准规定，注册服务器需要比较REGISTER请求中的Contact URI的绑定，并根据下列规则判定SIP或SIPS URI是否等价：

- SIP和SIPS URI永远不等价；
- 比较SIP和SIPS URI的userinfo（含密码的userinfo或者telephone-subscriber格式）时需要区分大小写，除非特殊要求，比较URI的其余成分都不区分大小写；

- 比较 SIP 和 SIPS URI 时，参数和头字段的顺序与等价性无关；
- 除了“保留”组以外，所有的字符都等价于它们对应的“%”“ HEX HEX”编码；
- DNS 搜索主机名得到的 IP 地址与该主机名不一致；
- 两个 URI 等价，要求用户、密码、主机、端口都必须一致。

省略用户部分的 URI 与包含用户部分的 URI 不一致。省略密码的 URI 与包含密码的 URI 也不一致。

省略任何含有缺省值部分的 URI 与包含该部分并且指定值等于缺省值的 URI 不一致。例如，省略掉可选的端口的 URI 与指定端口为 5060 的 URI 不一致，transport-parameter，ttl-parameter，user-parameter，以及 method 都是同样道理。从相同的 URI 中得到的地址等价。sip：user@host：5060 的端口总是 5060 而 sip：user@host 可以通过 DNS SRV 机制解析为另外的端口。

- URI 的参数比较规则如下：

—两个 URI 中出现的参数必须一致；

—user，ttl，或 method 参数只在其中一个 URI 中出现，即使它们包含有缺省值，也一定不能与另外的 URI 一致；

—包含 maddr 参数的 URI 与不包含 maddr 的 URI 不一致；

—其余的参数如果只在其中一个 URI 中出现时，比较的时候，忽略不计；

—URI 的 header 成分一定不能忽略。两个一致的 URI 中必须同时包含成分一致的 header。具体的每个字段的规定见本标准第 18 章。

17.1.5 由 URI 构成请求

直接由 URI 构成请求的时，商业名片、网页以至协议内部资源（如已注册地址）的 URI 的头字段或者消息体可能包含一些不合适部分。

构成请求中必须包括 Request-URI 中的 transport，maddr，ttl，user 参数。如果 URI 包含一个 method 参数，它的值必须作为请求的方法。Request-URI 中不可以有 method 参数。未知的 URI 参数必须放在消息的 Request-URI 中。

出现在 URI 中的头字段或者消息体都应当包含在消息里，基于请求中每个组成部分明显标记该请求。

具体实现中，不应明显标记那些容易受到威胁的头字段 From、Call-ID、Cseq、Via 和 Record-Route。

具体实现中，为了在恶意攻击的情况下不被作为不知情的代理利用，不可以明显标记那些请求的 Route 字段的值。

有些头字段可能会引起位置或能力的错误判断，所以不应该明显标记它们。包括 Accept，Accept-Encoding，Accept-Language，Allow，Contact，Organization，Supported，User-Agent。

具体实现中，应当校验描述性头字段的准确性，包括：Content-Disposition，Content-Encoding，Content-Language，Content-Length，Content-Type，Date，Mime-Version 和 Timestamp。

如果从给定的 URI 构建的请求不是有效的 SIP 请求，那么该 URI 也是无效的。该请求不应被传输，并应寻找出现无效 URI 的原因。构建请求无效的原因有很多，包括头字段语法错误，URI 参数组合无效，消息体描述错误等。

发送一个由给定的 URI 构成的请求中所要求的能力可能难以达到。例如，URI 可能指定一种不可实现的传送协议或者扩展。这种情况下应该拒绝发送这个请求而不是修改它来适应它的能力。不可以发送需要某个它不支持的扩展协议的请求。

例如，这种请求可以通过现有的 Require 头参数来构造，也可以通过或方法 URI 参数来构造，该参数带有未知的值或明确不支持的值。

17.1.6 SIP URI 和 tel URL 的关联

如果一个 tel URL 要被转换为 SIP URI 时，tel URL 中的整个 telephone-subscriber 部分和一些参数都被放在 SIP URI 的 userinfo 中。通常，以这种方式，等价的“tel” URL 转变成的 SIP 或 SIPS URI 可能不等价。SIP URI 中的 userinfo 比较的时候区分大小写。tel URL 中大小写不区分的部分和 tel URL 参数重新排序虽然不影响 tel URL 的等价性，但是转变成 SIP 或 SIPS URI 就不等价了。

为解决这个问题，将放到 userinfo 部分的 telephone-subscriber 中大小写不区分的部分统一改为小写（tel URL 所有的成分除将来扩展的参数以外都是大小写不区分的），telephone-subscriber 参数按照名字排序（isdn-subaddress 和 post-dial 这种参数除外，它们总是排在前面）。

17.2 选项标签

选项标签是用来指定 SIP 中扩展选项的识别符且为惟一的。这些标签用于 Require、Proxy-Require、Supported 以及 Unsupported 这些头字段中。这些选项以“选项参数=符号”的形式作为参数出现在这些头字段中。选项标签在 RFC 的相关标准中定义。为了保证厂商互通性，可参考 IANA 注册的选项标签。

17.3 标签

“tag”参数用于 SIP 消息中的 To 和 From 头字段。它通常用来识别一个对话，由 Call-ID 和对话的双方各一个的标签组成。当 UA 在对话之外发送请求的时候，它只包含一个 From 标签，提供“一半”对话 ID。对话由收到响应结束，双方在 To 标签里提供另一半 ID。分叉代理的 SIP 请求即可以由一个请求建立多个对话。因此，对话双方都必须有标识符：没有接收者确认，发件人不能确认这个对话是由一个请求建立的多个对话的哪一个。

UA 生成一个标签插入到请求或者响应中，这个标签必须是全局惟一的并且至少是 32bit 的随机编码。当 UA 要将自己加入一个会话中去，它需要在 INVITE 请求的 From 头字段中插入一个标签，在其响应的 To 头字段中插入一个不同的标签。同样，两个对于不同呼叫的 INVITE 有不同的 From 标签，对于两个不同的呼叫的响应也有不同的 To 标签。

除了要求全局惟一，根据具体实现不同对建立标签的规则也有不同的要求。系统在允许的范围之内出错，标签有助于将对话倒换到备用的服务器上。在出错的服务器上通过备份可以识别出这个对话的部分请求，UAS 可以选择标签来恢复该对话以及其他的相关状态。

18 头字段

头字段与各种方法和代理服务器的的关系总结于表 2。

表2 头字段摘要

头字段	位置	代理	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

表2（续）

头字段	位置	代理	ACK	BYE	CAN	INV	OPT	REG
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
Priority	R	ar	-	-	-	o	-	-
Proxy-Authenticate	407	ar	-	m	-	m	m	m
Proxy-Authenticate	401	ar	-	o	o	o	o	o

表2（续）

头字段	位置	代理	ACK	BYE	CAN	INV	OPT	REG
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

“位置”列是可以使用该头字段的请求和响应的类型。每个值的含义如下：

R：该头字段只能用于请求；

r：该头字段只能用于响应；

2xx、4xx 等等：该头字段可以用于的响应代码；

c：该头字段从请求中复制到响应中。

“代理”列讲述的是代理服务器可以在该头字段上进行的操作：

a：如果没有该字段，代理服务器可以增加或者连接该字段；

m：可以修改头字段原有的值；

d：可以删除一个头字端的值；

r：代理服务器可以读该字段，因此该字段一定不能加密。

剩下的 6 列涉及的是某方法中可以存在的头字段：

c：Conditional，即该头字端的要求依赖于消息的上下文；

m：mandatory，即该头字段是强制使用的；

m*：该头字段应该被发送，但是客户机/服务器可以接收没有该字端的消息；

o：optional，即该头字段是可选的；

t：该头字段应该被发送，但是客户机/服务器可以接收没有该字端的消息。如果使用基于流的传送协议（例如 TCP），必须含有该头字段；

*：消息体不为空时，该头字段是必需的；

-：不可使用该字段。

“可选的”意味着在请求中或者响应中可以包含该头字段，但是 UA 也可以忽略它，Require 字段例外。“强制的”要求请求中或者响应中必须含有该头字段，收到该请求的 UAS 或者 UAC 必须能够理解该字段。“不可使用”要求请求消息不可以含有该头字段，UAS 必须忽略含有这些字段的请求消息；同样，对于响应，标注了“不可使用”的头字段意味着 UAS 不可以将该字段放于响应中，并且 UAC 必须忽略响应中的该字段。

UA 应该忽略它所不理解的扩展的头字段参数。当整个消息太大时，某些普通的头字段可以使用缩略形式。Contact、From 以及 To 字段都包含一个 URI。如果该 URI 包含逗号、问号或者分号，该 URI 必须用三角括号括起来。任何 URI 参数也都包括在这些三角括号内。如果某 URI 没有在三角括号内并且参数使用分号分隔，那么这些参数就认为是头字段参数，而不是 URI 参数。

18.1 Accept 头字段

Accept 头字段参照 HTTP 的语法规定，语义也基本相同，惟一不同的是如果 Accept 头字段不存在，服务器应该默认其值为 application/sdp。如果 Accept 字段为空，意味着没有可接受的媒体类型。举例如下：

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

18.2 Accept-Encoding 头字段

Accept-Encoding 字段类似于 Accept，限制响应中可接受的内容编码

Accept-Encoding 字段允许为空。它等价于 Accept-Encoding：identity，也就是说只有 identity 时允许没有编码。如果没有 Accept-Encoding 字段，服务器应该假设缺省值是 identity。

18.3 Accept-Language 头字段

Accept-Language 头字段用来在请求中指定响应中首选使用的语言，用于响应中作为消息体的原因描述、对话描述或者状态响应。若无该字段服务器应该假定客户机接受任何一种语言。

Accept-Language 的语法遵照 HTTP 协议，基于“q”参数排序的规定也适用于 SIP。

18.4 Alert-Info 头字段

Alert-Info 位于 INVITE 请求时，该字段对 UAS 定义了另外一个可用的铃音。当 Alert-Info 头字段位于 180（Ringing）响应中时，该字段对 UAC 定义了另外一个可以使用的回铃音。代理服务器插入该字段是可以提供一种独特的铃声。

另外，用户应该也可以修改铃声能够使铃声可用。这有助于防止由于不信任的实体使用了该字段而导致通信中断。

18.5 Allow 头字段

该字段列出请求发起 UA 所支持的方法。UA 能理解的所有方法就必须列于该头字段中。消息中若无该头字段，则意味 UA 未提供任何关于它所支持的方法的信息，并不意味着 UA 不支持任何方法。

响应消息中的 Allow 头字段包含 OPTIONS 头字段以外其他的方法的，因此可以减少所需消息数量。

18.6 Authentication-Info 头字段

Authentication-Info 头字段为通信双方提供 HTTP 分类鉴权信息。如果某请求已经基于 Authorization 头字段完成了鉴权，那么 UAS 可以在该请求的 2XX 的响应中包含一个 Authentication-Info 字段。

18.7 Authorization 头字段

Authorization 头字段中含有一个 UA 的鉴权证书。

Authorization 和 Proxy-Authorization 头字段不遵循多头字段值的一般规则。虽然多个数值之间没有逗号分隔，该头字段名仍可以出现多次。

18.8 Call-ID 头字段

Call-ID 头字段惟一的标识某个客户端的某个特定的邀请或所有的注册请求。一个多媒体会议可以发起几个 Call-ID 不同的呼叫，例如，某用户可以多次邀请某人参与同一个会议。Call-ID 区分大小写并逐字节比较。缩写形式为 i。

18.9 Call-Info 头字段

Call-Info 头字段若在请求中则提供主叫的附加信息，若在响应中来则提供被叫附加信息。“purpose”参数描述了 URI 的用途；“icon”参数指定主叫或者是被叫的表示图像；“info”参数则为主叫或者被叫的一般描述；“card”参数用于提供一个商务名片；其余的参数可以通过 IANA 注册。

但是，使用 Call-Info 字段可能引起安全隐患。如果某被叫收到一个恶意主叫提供的 URI，那么该被叫可能会显现一些不恰当或者攻击性的内容，或者一些危险的或者非法的内容。因此本部分建议，仅当 UA 能够证明发出 Call-Info 字段的实体是真实并可信的，该 UA 才能使用 Call-Info 的信息。这并一定要求是对等的 UA，代理服务器也可以在请求中插入该字段。

18.10 Contact 头字段

Contact 字段的值含有一个 URI，其含义取决于该字段所在的请求或者响应的类型。

Contact 头字段值中还可以包含一个显示名称、含有 URI 参数的 URI 和头字段参数。本部分定义了 Contact 参数“q”和“expires”。这些参数只用于 REGISTER 请求及其响应以及 3xx 响应。

当 Contact 头字段包含一个显示名称的时候，带有所有的 URI 参数的 URI 应放于三角括号<>中，否则，URI 后面的参数都认为是头字段参数而不是 URI 参数。

即使“display-name”为空，只要“addr-spec”包含逗号、分号或者问号，也必须使用“name-addr”中的格式。名字和“<”之间 LWS 可有可无。

解析显示名称、URI、URI 参数以及头字段参数的规则同样适用于 To 和 From 头字段。

Contact 头字段的作用类似于 HTTP 中的 Location 头字段。但是 HTTP 的 Location 头字段只允许一个不用引号标注的地址。由于 URI 中可以包括逗号和分号，它们可能被误认为头字段或者参数的分隔符。

Contact 头字段的缩写是 m（“moved”）。

18.11 Content-Disposition 头字段

Content-Disposition 头字段描述了 UAC 或者 UAS 如何翻译消息体或者多部分消息体中的每部分消息体。该头字段扩展了 MIME 内容类型。

本协议在 Content-Disposition 头字段中定义了几个新的类型。“session”表示该消息体部分描述了一个呼叫时或者呼叫前的媒体会话。“render”值表示该消息体部分应被显示，否则回传给用户。在这里使用“render”而不是“inline”参数值是为了避免 MIME 消息体的隐藏部分作为整个消息的一部分而被显现出来。为了后向兼容，如果 Content-Disposition 头字段丢失，服务器应该假设 Content-Type 为

application/sdp 的消息体是所要描述的“session”，其他消息体的值为“render”。

“icon”表示该消息体包含一个图像作为表示主叫或者被叫的图标，当某个消息被接收时由 UA 递交或者在某个对话期间存留。“alert”指明该消息体部分包含一些信息应该由 UA 通知用户接收请求的时候回传给用户，通常一个请求发起一个对话。例如，180 临时性响应发出之后，告警该消息体应该作为一个铃声信号回传给电话机。

任何含有“disposition-type”的 MIME 消息体只有在该消息被鉴权之后才能被处理。

参数 handling-param 指定了如果 UA 不理解所收到的某消息体的内容类型或者处理类型，UAS 应该如何处理。该参数定义了“optional”和“required”两个值，如果没有该参数，默认值为“required”。

18.12 Content-Encoding 头字段

Content-Encoding 头字段的值指定了适用于该实体的编码以及为了获得 Content-Type 指定的媒体类型所需要使用的解码机制。Content-Encoding 头字段主要用于消息体压缩而且不丢失底层媒体类型标识。如果某个实体可以使用多个编码方式，则编码方式按其使用的顺序排列。

content-coding 头字段的所有值都不区分大小写，其符号值必须在 IANA 上注册。

客户端可以在请求中的消息体中使用 Content-conding 头字段指定的编码方式。服务器端则可以在响应中的消息体中使用这些编码方式。服务器端在请求中 Accept-Encoding 字段中列举的编码方式。

Content-Encoding 的缩写形式为 e。

18.13 Content-Language 头字段

Content-Language 头字段用来描述所附的消息体的接受者的原始语言。但是该语言于消息体内中所使用的所有语言并不等值。该头字段的格式如下：

Content-Language = “Content-Language” “:” 1#language-tag

Content-Language 主要的作用是允许用户根据自己喜好的语言来指定并区分消息体。具体内容见 RFC2616。

18.14 Content-Length 头字段

Content-Length 头字段用十进制数指定发送的消息体的大小（字节数）。具体实现时，应该使用该字段指示所传递消息体的大小而不考虑该实体的媒体类型。本协议规定，使用基于流的传输协议时必须使用该字段。

本字段所指示的消息体的大小不包括分隔头字段和消息体的 CRLF 符。该字段的有效值为大于等于零的数。如果消息中无消息体，那么该字段值必须设为零。省略 Content-Length 字段简化了动态的产生响应的类 CGI 脚本程序。

该字段的缩写形式为 l。

18.15 Content-Type 头字段

Content-Type 头字段指定消息体的媒体类型。本协议规定，如果消息体不为空，Content-Type 头字段必须存在。如果消息体为空且消息中包含 Content-Type 头字段，表明所指定的媒体类型长度为零（例如一个空的音频文件）。

该字段缩写形式为 c。

18.16 Cseq 头字段

Cseq 头字段位于请求消息中，包含一个十进制数字序列和一个请求方法。该数字序列必须一个 32

比特的无符号整数来表示。Cseq 头字段的方法部分区分大小写。该头字段用于把某对话中的事务进行排序且提供了一种惟一标识某事务的方法，并能够区分某请求是新的请求还是重发的请求。如果两个 Cseq 的数字序列以及方法都相等那么这两个 Cseq 就是等价的。

18.17 Date 头字段

Date 头字段包含日期和时间。[H3.3]中 SIP 的时区限制为“GMT”。

Date 头字段反映了请求或者响应第一次发送的时间。没有后备电池的终端系统可以利用 Date 字段获得当前时间。然而，使用 GMT 格式的时候，要求客户机知道本地与 GMT 时间的偏移。

18.18 Error-Info 头字段

Error-Info 头字段用来提供出错状态码的附加信息。UAC 具备用户接口功能，包括 PC 软客户的弹出窗口接口、音频接口、通过网关连接的音频电话或端点的接口。当服务器产生错误信号的时候，它可以发送一个带有详细原因描述的出错状态码，或者可发送一个音频记录，但是 Error-Info 头字段可以同时发送这两种信号。UAC 可以选择任何一种方式通知主叫。

UAC 可以把 Error-Info 字段里的 SIP 或 SIPS URI 当作重定向消息的 Contact 头字段，并发起一个新的 INVITE 请求并建立一个已记录的的会话。非 SIP URI 可以回送给用户。

18.19 Expires 头字段

Expires 头字段给出消息（内容）的有效期时间，其含义因请求方法不同而不同。INVITE 的有效时间并不影响它所引起的会话的实际持续时间。会话持续时间的表达方法由会话描述协议 SDP 规定。

该字段的值是一个十进制整数，单位为秒，，其值介于 0~（$2^{32}-1$）之间。从收到请求的那一刻开始计时。

18.20 From 头字段

From 头字段用于指示请求的发起者。这可能与对话的发起者并不同。被叫发送给主叫的请求在 From 字段中使用被叫的地址。

“display-name”参数可选，主要用于人机接口。如果某用户需要隐藏其标识，则本字段应为“Anonymous”。即使本字段为空，如果“addr-spec”包含一个逗号问号或分号，那么必须使用“name-addr”的格式。

如果两个 From 字段的 URI 一致并且参数一致，则这两个字段是等价的。如果其中一个字段中有扩展的参数而另外一个没有，那么比较的时候应该忽略这个参数。这意味着名字和三角括号的并不影响其等价性。From 的缩写形式是 f。

18.21 In-Reply-To 头字段

In-Reply-To 头字段列举了本次呼叫涉及或者返回的所有的 Call-ID。客户端可以缓存这些 Call-ID，并放在所返回的呼叫的 in-Reply-To 字段中。这允许自动呼叫分配系统（ACD）将返回的呼叫路由给到第一个呼叫发起者。这也允许被叫过滤这些呼叫，仅返回那些被接受的呼叫。这个头字段并不可替代请求鉴权。

18.22 Max-Forwards 头字段

本部分中，Max-Forwards 头字段必须和任和方法一起规定向下游转发消息的代理服务器和网关的个数。当某客户端沿着某条链路发送请求消息的时候，使用该字段可以有效的防止链路中出错或者发生回环。

Max-Forwards 头字段的值是一个 0~255 的整数，指示了该请求还允许被转发的次数。转发该请求时每经过一个服务器该值就减一。本部分建议起始值为 70。

实体如果无法保证环路检测则应该在消息中插入该头字段。

18.23 Min-Expires 头字段

Min-Expires 头字段定义了由服务器控制的软状态实体更新间隔的最小值。这也包括注册服务器所存储的 Contact 字段。该头字段的值是一个 0~（$2^{32}-1$）的十进制整数，单位为秒。

18.24 MIME-Version 头字段

见 HTTP/1.1。

MIME-Version 头字段用于指示构造消息得 MIME 协议的版本号。使用该头字段可以指示消息于 MIME 协议完全兼容。

MIME-Version 头字段的格式如下：

MIME-Version = “MIME-Version” “:” 1*DIGIT “.” 1*DIGIT

18.25 Organization 头字段

Organization 头字段给出发出请求或者响应消息的实体所属的组织的名称。该字段可以被客户端软件用来过滤呼叫。

18.26 Priority 头字段

Priority 头字段定义客户端请求的紧急程度。该头字段描述了对于接收用户或其代理，请求消息所具有优先级。例如，它可以影响呼叫路由的选择以及是否决定接受。请求中若无该头字段则应认为优先权为 “normal”。Priority 头字段并不影响通信资源的使用，比如其不会影响路由器中数据包转发优先级或者接入到 PSTN 网关中的情况。该字段的值可以为 “non-urgent”，“normal”，“urgent” 和 “emergency”。

18.27 Proxy-Authenticate 头字段

Proxy-Authenticate 头字段值包含一个鉴权质询口令。关于本字段见 RFC2616。

18.28 Proxy-Authorization 头字段

Proxy-Authorization 头字段用于用户对要求鉴权的代理服务器标识自己。Proxy-Authorization 头字段的值中包含一个证书，其中包含 UA 对于代理服务器的鉴权信息和所要请求的资源的域。具体内容见 RFC2616。

关于多字段名的一般规定并不适于 Proxy-Authorization 和 Authorization 头字段。虽然多个数值之间没有逗号分隔，该头字段名仍可以出现多次，但是不可以将它们结合成一个单独的头字段行。

18.29 Proxy-Require 头字段

Proxy-Require 头字段指定代理服务器所必须支持的某些特征。

18.30 Record-Route 头字段

Record-Route 头字段由代理服务器插入请求消息中，这样可以使该对话中将来的请求仍能经过该代理服务器。

18.31 Reply-To 头字段

Reply-To 头字段中包含一个逻辑返回 URI，该 URI 与 From 头字段可能不同。如果用户需要保持匿名，则该字段必须从请求中省略掉，或者使用一种不显现任何个人信息的方式。

即使“display-name”为空，但如果“addr-spec”包含一个逗号、问号或者分号，那么必须使用“name-addr”

的格式。

18.32 Require 头字段

UAC 通过 Require 头字段告知 UAS 处理请求时需要支持的选项。该字段为可选，但不可以被忽略。

该字段包含一系列选项标签。选项标签指定了一系列的扩展头字段，这些字段在处理请求时必须被理解。

18.33 Retry-After 头字段

Retry-After 头字段用在 500（服务器内部错误）或 503（服务不可用）响应中，可以对请求发起客户端指示多久以后服务不可用，用于 404（未找到）、413（请求过大）、480（暂时不可用）、486（正忙）、600（忙）或者 603（下降）响应中表明被叫何时可用。该值用一个十进制的正整数表示，单位为秒，从响应发出开始计时。

本字段中可以使用一个注释项标明于回呼时间的附加信息，该注释项为可选。可用“duration”参数指明被叫从空闲的开始保持空闲的时长，该参数为可选。

18.34 Route 头字段

Route 头字段中有一个代理服务器列表，用来指定请求消息的路由。代理服务器对用于 route 头字段的处理参见附录 K。

18.35 Server 头字段

Server 头字段包含 UAS 处理请求所使用的软件信息。如果显示服务器特定的软件版本信息，一旦该软件包含安全缺口的话，就容易使该服务器受到攻击。具体实现中，应把 Server 字段作为可配置的选项。

18.36 Subject 头字段

Subject 头字段包括呼叫的概述或者呼叫的性质，过滤呼叫时，可以不必解析会话描述。会话描述不必与邀请使用同一个对象。该字段的缩写形式为 s。

18.37 Supported 头字段

Supported 头字段列举了 UAS 或者 UAC 支持的所有的扩展。该头字段中包含一个 UAS 或者 UAC 所支持的选项标签的列表。如果此头字段为空表示则没有可以支持的扩展定义。该字段的缩写形式为 k。

18.38 Timestamp 头字段

Timestam 头字段给出 UAC 向 UAS 发出请求的时间。本部分中没有定义该头字段的用法，但可以使用某些扩展 来获得 RTT（TCP 往返传输时间）估计值。

18.39 To 头字段

该头字段指定请求的逻辑接收者。“display-name”参数用于人机接口，为可选。“tag”参数一般用于标识对话。

To 头字段等价性的判定同 From 头字段。该字段的缩写形式为 t。

18.40 Unsupported 头字段

Unsupported 头字段列出 UAS 不支持的特征。

18.41 User-Agent 头字段

User-Agent 头字段含有与请求发起 UAC 有关的信息。该字段的语义见 RFC 2616。

如果显示 UA 的特定的软件版本信息，一旦软件存在安全缺口的话，该 UA 就容易易受到攻击。应把 User-Agent 字段作为可配置的选项。

18.42 Via 头字段

Via 头字段指定目前请求消息经过的路径，同时指定响应也要按该路径返回。该字段值中的 branch ID 参数是一个事务标识符，代理服务器用它来检测环路。

Via 头字段包含一个用来发送消息的传送协议和客户端的主机名或者网络地址，该头字段还可能包含一个接受响应的端口号。本字段还可以包含下列参数："maddr"、"ttl"、"received" 和 "branch"。具体实现时，branch 参数的值必须从 "z9hG4bK" 这个字符串开始。

本字段定义的传送协议有 "UDP"，"TCP"，"TLS" 和 "SCTP"。举例如下：

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 字段的缩写形式为 v。

上面例子中，一个拥有两个地址（192.0.2.1 和 192.0.2.207）的多穴主机发出消息，发送方可能用错了网络接口。Erlang.bell-telephone.com 发现不匹配，就在前一跳的 Via 字段中插入一个包含实际发送该数据包的地址的参数。

主机或者网络地址以及端口号不需要遵循 SIP URI 的语法。":" 或者 "/" 的任何一边的 LWS 都是允许的。

虽然本标准要求所有的请求中都包含 branch 参数，该字段的 BNF 仍然规定该参数为可选的。

如果两个 Via 字段的 sent-protocol 和 sent-by 参数相等（有同样的参数组并且相同的参数值）则两个头字段等价。

18.43 Warning 头字段

Warning 头字段中是与响应状态有关的附加信息。该字段值放于响应中发送，其中包含一个三位告警码、主机名以及告警文本信息。

"warn-text" 应为可被接收用户所理解的自然语言。可以根据以下几种情况选择：用户的位置、请求中的 Accept-Language 字段、响应中的 Content-Language 字段。默认的语言是 i-default 。

以下列举了已定义的 "warn-code"，和英文的 warn-text。这些告警是由会话描述所引入的失败。告警码的第一位是 3 则表明为 SIP 告警。300~329 表明是因为会话描述中的关键词引起的错误；330~339 表示错误与会话描述要求的基本网络业务有关；370~379 表明错误与会话描述要求的 QoS 参数有关；390~399 是除上述情况之外的错误。

300 网络协议不兼容：会话描述中的一个或多个网络协议不可用。

301 网络地址格式不兼容：会话描述中的一个或多个地址格式不可用。

302 传送协议不兼容：会话描述中的一个或多个传送协议不可用。

303 带宽单位不兼容：会话描述中的一个或多个带宽度量单位不被理解。

304 媒体类型不可用：对话描述中的一个或多个媒体类型不可用。

305 媒体格式不兼容：对话描述中的一个或多个媒体格式不可用。

306 媒体特征不被理解：对话描述中的一个或多个媒体特征不被支持。

307 对话描述参数不被理解：除上述几种参数之外的参数不被理解。

330 组播不可用：用户站点不支持组播。

331 单播不可用：用户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：对话描述中定义的或者媒体定义的带宽超出可用带宽。

399 混合告警：该告警表示用户存在的任意一种错误，收到该告警的系统不可以采取任何自动的动作。

其余的“warn-code”可以由 IANA 定义。

18.44 WWW-Authenticate 头字段

WWW-Authenticate 包含一个鉴权质询。

19 响应代码

SIP 的响应代码在 HTTP/1.1 的基础上有所扩展。本标准只涉及到 SIP 响应代码，并补充了 6xx 响应代码。

19.1 临时响应 1xx

临时性响应即报告性的响应，用来指明所联系的服务器还没有确定性的响应。如果服务器需要 200ms 以上的时间才能发出最终响应，则它就需要首先发送一个 1xx 响应。1xx 响应不能进行可靠传输。它也不能让客户端发送一个 ACK 请求。临时响应（1xx）可以包括一些消息体，其中包含会话描述 SDP。

在 SIP 中，响应分为两类，即临时响应和最终响应。最终响应传递呼叫请求处理的结果，并且保证可靠传送；临时响应提供呼叫请求处理过程中的信息，通常并不保证可靠传送。但在某些特定的情况下可靠地传递临时响应非常重要，因此需要任选能力集来支持临时响应的可靠传递。关于临时响应的可靠传递参见附录 C。

19.1.1 100（处理中）

尝试响应（100）表明下一跳服务器已经收到该请求，但是对这次呼叫的请求并未进行具体的处理。和其他临时响应一样，该响应使 UAC 停止重发 INVITE 请求。与其他的临时性响应不同，该响应不能使用有状态服务器前转。

19.1.2 180（振铃中）

UA 收到 INVITE 请求之后用该响应通知用户，该响应可以发起一个本地回铃音。

19.1.3 181（呼叫正在前转）

服务器可以使用该状态码表示该呼叫正被前转到另外一组终点。

19.1.4 182（排队）

如果被叫方正忙，服务器可以将本次呼叫放于队列中等待而非拒绝它。当被叫空闲时，将返回适当的最终响应。该响应中可包含一个关于呼叫状态的原因短语。服务器可以向主叫发出多个 182 响应来更新呼叫等待的状态。

19.1.5 183（会话进行）

该响应用来传递关于呼叫进程的信息。其中包括原因短语、头字段、消息体来描述呼叫进程更详细的信息。

19.2 2xx（请求成功）

该响应表明请求成功。

19.2.1 200（成功）

该响应表示请求成功。与响应一起返回的信息取决于请求中使用的方法。

19.3 3xx（重定向）

该响应指定用户新位置信息，或者指定可以满足本次呼叫所需要的的其他服务。

19.3.1 300（多选择）

请求中的地址可以解析为几个选项，每个选项都有自己特定的位置。用户或者 UA 可以确定一个首选的通信端点并且将该请求重定向到该位置。

请求中的消息体可能包含一个资源特征和位置的列表，用户或者 UA 可以从中选择一个 Accept 头字段中所允许的最合适的资源特征和位置。该消息体不定义 MIME 类型。

这些选项可作为 Contact 字段列出。SIP 响应可包含几个 Contact 字段或者在一个 Contact 字段中有多个地址。UA 可以使用 Contact 字段中的值自动重定向，也可以要求用户对选项进行确认。自动选择的标准定义不再本部分范围之内。如果被叫可以通过多个不同位置到达，且服务器不能代理该请求，则可以使用该状态码。

19.3.2 301（永久重定向）

用当户已经不在 Request-URI 头字段的地址中，请求发起用户就应该重发请求至 Contact 头字段中的新地址。请求方应该更新本地姓名地址簿和用户位置并将将来的请求重定向到新地址。

19.3.3 302（暂时重定向）

请求发起用户应向 Contact 头字段中的新地址重发请求。新请求中的 Request-URI 是响应中 Contact 头字段里的值。Contact 头字段中 URI 的有效期可以由 Expires 头字段或者 Contact 头字段中的 expires 参数来定义。代理服务器和 UA 可以在有效期内使用 URI。如果没有明确的定义有效期，则该地址仅可有效递归一次，而以后的事务就不可以再使用该 URI。

如果用户向 Contact 头字段中的 URI 发送请求失败，则应向重定向请求中的 URI 尝试发送请求。有效期过后之后该 URI 就不再使用，这时可以使会使用另一个新的暂时性的 URI。

19.3.4 305（使用代理）

用户必须通过 Contact 头字段中指定的代理服务器接入到请求的资源。Contact 头字段指定代理服务器的 URI。接收者将通过代理服务器中继该请求。该响应只可由 UAS 发出。

19.3.5 380（可选择服务）

呼叫不成功但是有另外可供选择的服务。该响应中的消息体中描述了这种另外可以选择的服务。本部分不涉及消息体的格式定义。

19.4 4xx（请求失败）

该响应由服务器发出表明请求失败。客户机不应（例如增加合适的授权）将原请求不加修改并重新发送。但将原请求发向不同的服务器也可能成功。

19.4.1 400（错误请求）

该响应表示请求由于语法错误该而不能被理解。响应的原因短语中应详细指出语法错误.

19.4.2 401（未鉴权）

该响应表示请求消息需要用户鉴权。该响应由 UAS 和注册服务器发起。407（代理服务器要求鉴权）由代理服务器发起。

19.4.3 402（需要付费）

保留。

19.4.4 403（禁止）

该响应表示服务器能理解但是拒绝执行请求消息，即使该请求已经鉴权也不能进行中继。

19.4.5 404（未找到）

该响应表示服务器可以确定用户不在 Request-URI 头字段指定的域中。如果 Request-URI 头字段所指定的域与请求的接收方所能处理的域不一致时，也应该发送该响应。

19.4.6 405（方法不允许）

该响应表示在 Request-URI 头字段指定的地址上，请求中的方法能够被理解但并不允许使用。该响应中必须包括一个 Allow 头字段来列举指定地址上所允许的方法。

19.4.7 406（不接受）

该响应表示，根据请求的 Accept 头字段，该请求所指定的资源生成响应的消息体中包含的某些内容特性是不被接受的。

19.4.8 407（代理服务器要求鉴权）

该响应类似于 401（未鉴权），不同的它指定客户机必须首先向代理服务器鉴权自己。该状态码可用于接入到通信信道中。

19.4.9 408（请求超时）

该响应表示服务器不能在适当的时长内产生响应。例如当它不能及时确定用户的位置时，客户端收到该响应后，可以不加修改便重发原请求。

19.4.10 410（Gone）

该响应表示服务器中被请求的资源不可用且服务器不知道转发地址，并且这种情况是永久性的。如果服务器不知道这种情况是否为永久性的，此时则应该使用 404（未找到）状态码。

19.4.11 413（请求消息体过大）

该响应表示，如果请求的消息体超出服务器能够处理范围，服务器将拒绝处理该请求。服务器可以关闭此连接以防客户端不断发送同一个请求。

如果这种情况是暂时的，服务器应在响应中加入一个 Retry-After 头字段用来指定多久以后客户机可以重发该请求。

19.4.12 414（Request-URI 过长）

该响应表示，如果 Request-URI 超出服务器能够处理的范围，服务器将拒绝处理该请求。

19.4.13 415（不支持媒体类型）

该响应表示服务器不支持某请求方法的消息体格式而拒绝处理该请求。根据具体内容的不同，服务器必须用响应 Accept、Accept-Encoding 或 Accept-Language 头字段返回服务器可以接收的格式列表。

19.4.14 416（不支持的 URI 方案）

该响应表示，由于服务器不理解 URI 的方案而不能处理该请求。

19.4.15 420（错误扩展）

该响应表示，服务器不理解 Proxy-Require 或 Require 头字段中协议的扩展规定。服务器必须在响应中的 Unsupported 字段中包含一个它不支持的扩展的列表。

19.4.16 421（扩展要求）

该响应表示，UAS 需要某个特定的扩展才能处理该请求，但是这种扩展没有列在请求中的 Supported

头字段中。该响应必须包含一个 Require 头字段列举所需要的扩展。

除非 UAS 不能向客户提供任何其他所需的业务否则不应该使用该响应。如果 Supported 字段中没有所需的扩展，服务器只能用客户端所支持的扩展规定对该请求进行 SIP 的基本处理。

19.4.17 423（间隔太短）

该响应表示，由于请求更新资源的间隔时间太短，服务器拒绝该请求。注册请求的 Contact 字段中定义的有效期太短，注册服务器可以使用该响应拒绝请求。

19.4.18 480（暂时不可用）

该响应表示，与被叫方成功的联系上，但是被叫目前不可用者。在响应的 Retry-After 字段中可以指定一个合适的呼叫时间，在原因短语应该给出一个详细的原因指明为什么被叫方不可用。这个值可以由 UA 来设置。

如果重定向服务器或者代理服务器知道 Request-URI 中指定的用户但是目前并没有其有效位置，就可以返回该状态码。

19.4.19 481（呼叫/事务不存在）

该响应表示 UAS 收到的请求与现有的对话或者事务没有相对应的。

19.4.20 482（环路检测）

该响应表示服务器检测到有环路。

19.4.21 483（跳数太多）

该响应表示，服务器收到请求中的 Max-Forwards 值为零。

19.4.22 484（地址不完整）

该响应表示，服务器收到的 Request-URI 不完整并应在原因短语中提供附加信息。该状态码允许异步拨号（overlapped dialing）。当用户使用异步拨号的方式时，客户端并不知道拨号字串的长度，因此它发送的字符串比实际的长，并提示用户输入更多的数字，直到不再收到 484 状态码为止。

19.4.23 485（不明确）

该响应表示请求中的 Request-URI 不明确。该响应中的 Contact 字段中可以包含另外一个明确的地址，由于显示替代的 Request-URI 可能会破坏用户或者组织的保密性，这种情况下，服务器必须可以作出 404（未找到）响应，或者服务器能够禁止列举可能的 URI 选项。

有些电子信箱以及语音信箱系统可以提供这种功能。该状态码与 3xx 状态码语义不同：对于 300 响应，假设可以通过所提供的选项到达同一个人或者服务。自动化的选择或者连续的查找只对 3xx 有意义，而使用 485（Ambiguous）响应时则需要用户干涉。

19.4.24 486（忙）

该响应表示，已经成功的和被叫终端连接，但是被叫目前不能在该终端系统执行呼叫，响应中的 Retry-After 字段可以指定一个合适的呼叫时间。该用户在其他地方可用。如果客户机知道没有别的终端系统可以接收本次呼叫那么应该采用 600（忙）状态码。

19.4.25 487（请求被终止）

该响应表示，请求被 BYE 或者 CANCEL 请求终止。CANCEL 请求不可以返回该响应。

19.4.26 488（此处不接受）

该响应与 606 响应的含义相同，但是仅指 Request-URI 中指定的资源，如果在别处，该请求可能成功。

该响应中可能存在包含媒体能力描述的消息体，该消息体格式根据 INVITE 请求中的 Accept 字段（如果不存在就是 application/sdp）规定。21.4.27 491 请求挂起响应。

该响应表示，UAS 收到请求但是在同一个对话中该 UAS 还有一个等待处理的请求。

19.4.27 493（无法解密）

该响应表示，UAS 收到的请求包含一个加密的 MIME 消息体而接收方没有合适的解码密钥。该响应可以只包含一个消息体，该消息体包含一个公共密钥用来加密发送给 UA 的 MIME 消息体。

19.5 5xx（服务器错误）

该响应表示服务器内部出错导致失败。

19.5.1 500（服务器内部错误）

该响应表示，服务器遇到意外的情况使它不能执行该请求。客户端可以显示这种特定的出错情况，并且可以在几秒钟内重发该请求。

如果情况是暂时的，服务器可以在 Retry-After 字段中指定多久之后客户机可以重发该请求。

19.5.2 501（未实现）

该响应表示服务器不支持实现该请求所需要的功能。如果 UAS 无法识别该请求的方法并且不支持该方法，就可以发送该响应。如果为代理服务器，它转发请求时都不考虑请求的方法。

如果服务器识别了请求中的方法但是并不支持该方法应该发送 405（方法不允许）响应。

19.5.3 502（错误网关）

当服务器作为网关或者代理服务器时，需要接入到某下行服务器来完成请求，该下行服务器发出该响应表示其为无效网关。

19.5.4 503（服务不可用）

该响应表示由于服务器过载或者正在维护而导致服务器暂时不能处理该请求。服务器可以在 Retry-After 中指明何时可重发该请求。如果没有 Retry-After 则客户端必须按照收到的是 500（服务器内部错误）响应处理。

代理服务器或 UAC 收到一个该响应之后应该将该原请求转发到替代的服务器上。如果响应中存在 Retry-After 字段，则在该字段定义的时间之内该客户机不能再向原来的服务器发送任何请求。

服务器也可以不必发送该响应，而直接拒绝连接或者丢弃原请求。

19.5.5 504（服务器超时）

该响应表示，服务器接入到一个外部服务器来处理请求，但是没有及时收到该外部服务器的响应。如果在 Expires 字段中规定的时间之内没有收到上行服务器发来的响应，就应该采用 408（Request Timeout）响应。

19.5.6 505（不支持版本）

该响应表示服务器不支持请求中的协议的版本。

19.5.7 513（消息过大）

该响应表示，由于消息体的长度超过服务器的处理能力限制，服务器不能处理该请求。

19.6 6xx（全局失败）

该响应表示服务器对于某一特定用户的确定的信息。

19.6.1 600（忙）

该响应表示与被叫终端连接成功，但是被叫因为忙而不能接收呼叫。该响应中的 Retry-After 字段指定过多久之后可以重新呼叫。如果被叫不希望给出拒绝本次呼叫的原因则应该使用 603（拒绝）。该响应只用于客户端知道没有别的终端可以接收该请求的情况，否则应该返回 486 响应。

19.6.2 603（拒绝）

该响应表示与被叫已经成功连接，被叫用户明确表示不能参与此呼叫。该响应中的 Retry-After 字段可以指定过多久之后可以重新呼叫。只有客户端知道没有别的终端可以接收该请求才可以使用该响应。

19.6.3 604（用户不存在）

该响应通知服务器 Request-URI 中的用户根本不存在。

19.6.4 606（无法接受）

该响应表示与用户代理已经成功连接，但是会话描述中如请求的媒体、带宽或者地址形式等都不可接受。

该响应可以包含一个 Warning 头字段来详细说明不支持该会话描述的原因。

该响应中可以有一个包含媒体能力描述的消息体，它的格式依据 INVITE 请求中的 Accept 头字段，如果没有该字段，则依据 application/sdp。同 OPTIONS 请求消息的 200（OK）响应的消息体。

通信中不希望有频繁的协商，但如果一个新用户被邀请参加原有的一个会议根据邀请发起者是否发送 606 响应来决定是否需要协商。

该响应只用于客户机知道没有别的终端可以响应该请求。

20 SIP 协议的鉴权机制

SIP 提供一种无状态的基于质询的鉴权机制，并基于 HTTP 的鉴权机制。代理服务器或者 UA 在任何时候收到用户发来的请求时都可以向请求者质询鉴权。一旦请求方身份证实，接收方就能够确信该用户是否被授权发出请求。本部分不涉及任何授权系统的内容。

分类（digest）鉴权机制只提供了消息的鉴权和重发保护，不涉及到保证消息的完整性和机密性的内容。上文中的保护措施不是分类鉴权所提供的，用来阻止攻击者修改 SIP 请求和响应。

由于基本鉴权（Basic）机制因其安全薄弱性而不再使用。服务器不可以接受使用“Basic”机制的证书，而且服务器也不能使用“Basic”机制发出质询。

20.1 结构

SIP 鉴权的结构类似 HTTP。尤其是 auth-scheme，auth-param，challenge，realm，realm-value，以及 credential 这些的 BNF 语法都是一样的。本协议中，UAS 使用 401（未鉴权）响应询问 UAC 的身份。注册服务器和重定向服务器也使用该响应要求鉴权。只有代理服务器使用的是 407（代理服务器要求鉴权）响应。在不同的消息中可能分别要求包含 Proxy-Authenticate，Proxy-Authorization，WWW-Authenticate 和 Authorization 头字段。

由于 SIP 协议中没有标准 ROOT URI 的概念，SIP 中保护空间的概念就不同。本协议中，realm 字符串定义了保护空间。由于 UAC 发送的 Request-URI 与发出质询的服务器收到的 Request-URI 可能不同，而且实际上 UAC 可能并不知道该 URI 的最终格式，而且 realm 字符串与 Request-URI 共同定义的保护域的方式中，依赖于 Request-URI 中的 SIP URI，这样不能使用别的 URI 方案。

UA 或者代理服务器要求对所收到的请求进行鉴权，服务器创建一个 realm 字符串的规则如下：

- realm 字符串必须是惟一的。本部分建议，一个 realm 字符串包含一个主机名或一个域名，这部分

内容见 RFC 2617。

• realm 字符串应该包含一个可读的标识符并可以提交给用户。

举例如下：

INVITE sip：bob@biloxi.com SIP/2.0

Authorization：Digest realm= “biloxi.com”，<...>

一般，SIP 鉴权只对某个特定的保护域有意义。对于分类鉴权，每个这样的保护域都有自己的一组用户名和密码。如果服务器对某请求不要求鉴权，它可以接受一个默认的用户名 “anonymous” 并且密码为空。同样，UAC 代表许多用户，例如 PSTN 网关，可能有自己的特定设备的用户名和密码，而不是本域内特定的用户账号。

服务器可以合法的质询大多数 SIP 请求，但 ACK 和 CANCEL 需要特殊鉴权。

目前的鉴权机制如分类鉴权，要求响应应带有用于计算 nonce 值的数值，这样没有得到响应的请求，包括没有得到 ACK 响应的请求会产生某些问题。因此，收到 ACK 响应的服务器一定要接受 INVITE 请求的证书。发出 ACK 响应的 UAC 将 INVITE 请求中的 Authorization 和 Proxy-Authorization 头字段的值复制到 ACK 响应的对应的字段中。服务器不可以向 ACK 响应发出质询。

尽管 CANCEL 会得到响应 2xx，但是由于该请求不能重新提交，因此服务器也不能向 CANCEL 提出质询。一般情况下，如果某一个代理服务器发出的请求被取消，那么服务器应该接受该代理服务器发出的 CANCEL 请求。

如果 UAC 收到一个质询，而 UAC 的设备还不知道被质疑的 realm 中的证书，那么该 UAC 应该将询问（WWW-Authenticate 头字段或者 Proxy-Authenticate 字段中）中的 “realm” 参数的内容提交给用户。当询问一个预配置的设备时，预先给 UA 的 realm 配置证书的业务提供者应该知道该用户是否给该 realm 提供自己的证书。

即使一个 UAC 可以找到与适当的 realm 有关的证书，也可能出现以下情况：该证书无效，或者发出质询的服务器由于某种原因不再接受这些证书。这种情况下服务器可能会重复质询，或者发出 403 禁止响应。UAC 不可以使用被拒绝过的证书重新发出请求。

20.2 用户间鉴权

当 UAS 处理所收到的 UAC 的请求之前，UAS 可以对请求者进行鉴权。如果请求中没有在 Authorization 头字段中提供证书，UAS 则可以使用 401（未鉴权）状态码拒绝请求，并向请求者发起质询要求其提供证书。

401（未鉴权）响应消息中必须有 WWW-Authenticate 头字段。该字段的至少包含一个质询来指定适用于该 realm 的鉴权机制和参数。

当请求发起 UAC 收到 401（未鉴权）响应后，它应该使用合适的证书重新发起请求。UAC 也可以在处理请求之前要求用户直接输入证书。如果用户直接提供证书或者 UA 可以从内部关键词字符串找到证书，UA 就应该把证书以一个给定的值缓存到 To 头字段和 “realm” 字符串中，并且在下一个请求中重用该值。

如果找不到某个域的证书，UAC 可以使用 “anonymous” 的用户名和空密码重试该请求。

找到证书之后，UA 如果可以在请求中包含一个 Authorization 头字段完成对 UAS 和注册服务器的鉴权。在 Authorization 头字段可以带有一个证书，该证书包含所请求的资源的域的鉴权信息以及鉴权和重

发保护所需的参数。

UAC 收到 401（未鉴权）或者 407（代理服务器要求鉴权）响应之后重新提交含有证书的请求，并且必须将请求中的 Cseq 字段的值加 1。

20.3 代理服务器到用户的鉴权

UAC 向代理服务器发送请求的时候，代理服务器在处理请求之前可能要求 UAC 鉴权。如果请求中的 Proxy-Authorization 头字段中没有证书，代理服务器可以使用 407（代理服务器要求鉴权）状态码拒绝请求并要求鉴别用户权限。该 407 响应必须包含一个 Proxy-Authenticate 头字段。

代理服务器不可以向 Proxy-Authorization 头字段中增加值。所有的 407 响应必须按上行方向转发到 UAC。UAC 应该向 Proxy-Authorization 头字段中增加相应的证书，如果代理服务器要重新提交该请求，它需要在新请求中将 Cseq 的值加 1。因为 Cseq 的值的改变，原始请求发起的 UAC 将丢弃 UAS 的响应。

发起 UAC 收到 407 响应以后，它应该重新发起包含证书的请求。“realm”参数的确定同 401 响应的处理。

如果找不到某个域的证书，UAC 可以使用“anonymous”的用户名和空密码重试该请求。

UAC 应该缓存用于重新发起请求的证书。

本部分建议代理服务器使用下面规则缓存证书：

如果 UA 收到含有某个特定 Call-ID 的请求的 401/407 响应中的 Proxy-Authenticate 头字段值，它应该合并随后所有含有相同 Call-ID 的请求的证书。这些证书不能越过对话进行缓存；如果某 UA 被配置了本地对外代理服务器的域，如果有一个域存在，那么该 UA 可以越过对话缓存该域证书。这并不代表对话中以后的请求通过 Route 路径上的每个代理服务器都不需要证书。

任何向代理服务器鉴权自身的UA通常收到407响应以后，可以在请求中包含一个Proxy-Authorization 头字段值来完成鉴权。Proxy-Authorization 头字段允许客户机向要求鉴权的代理服务器证明自己或者其用户的身份。Proxy-Authorization 头字段值是由证书组成，该证书中包含 UA 对于代理服务器或请求的资源域的鉴权信息。

Proxy-Authorization 头字段值只用于某些代理服务器，其标识放于“realm”参数中，这些代理服务器以前可能曾利用 Proxy-Authenticate 头字段要求鉴权。路径中使用多个代理服务器时，Proxy-Authorization 头字段的值不可以用于与“realm”参数匹配的代理服务器。

如果某鉴权方案不支持 Proxy-Authorization 头字段中使用的域，那么代理服务器必须试着解析所有的 Proxy-Authorization 头字段的值来确定是否存在有效证书。但在大型的网络中，为了节省时间，代理服务器应该在 Proxy-Authorization 头字段中使用支持该域的鉴权机制。

如果请求被分叉代理，可能会有不同的代理服务器或 UA 向该 UAC 发出质询。这种情况下，分叉的代理服务器负责将这些质询合并在一个请求中。分叉代理的请求的响应中每一个 WWW-Authenticate 和 Proxy-Authenticate 头字段值必须放在分叉代理服务器发给 UA 的响应中，其顺序任意。如果某代理服务器在响应中发出质询，它将停止代理该请求，直到 UAC 重新发出含有有效证书的请求。一个分叉代理服务器可能会同时向多个需要鉴权的代理服务器转发某个请求，这些代理服务器都将停止转发该请求，直到 UAC 在各个域中分别鉴权本身。如果 UAC 没有向每个质询提供证书，发出质询的代理服务器将不再向 UA 转发该请求。

UAC在收到包含多个质询的401或者407响应之后该重新提交请求。该请求中的在WWW-Authenticat

值中应包含一个 Authorization 值，Proxy-Authenticate 值中应包含一个 Proxy-Authorization 值。一个请求中的多个证书应该通过它们的“realm”参数来区分。

多个质询可能只与 401 或者 407 响应中的同一个域相关。当多个代理服务器使用同一个域，当 UAC 向这些代理服务器发出多个分叉代理的请求，该请求中的可 Authorization 或 Proxy-Authorization 头字段中可能有多个含有相同的“realm”参数的证书。相同的证书应用于同一个域。

20.4 分类鉴权

SIP 分类鉴权方案与 HTTP 分类鉴权类似，以下的内容是 HTTP 分类鉴权方案用于 SIP 鉴权时的所作的修改。

SIP 服务器不可以接受或者发出 Basic 鉴权的请求。

SIP 分类鉴权机制遵从 RFC2617 定义的规则，但需把其中的“HTTP/1.1”变为“SIP/2.0”。具体变化如下：

1）质询中 URI 的 BNF：

URI = SIP-URI /SIPS-URI

2）HTTP Digest 鉴权的 Authorization 字段的'uri'参数没有使用引号而 SIP 的'uri'必须使用引号。

3）digest-uri-value 的 BNF 语法为：

digest-uri-value = Request-URI。

4）基于 Etag 选择 nonce 并不适用于 SIP。

5）RFC 2617 中关于缓存的操作不适用于 SIP。

6）RFC 2617 要求服务器检查请求中的 URI 和 Authorization 头字段中的 URI 是否指向相同的资源。在 SIP 中，由于请求要在一些代理服务器进行转发，因此这两个 URI 可以指不同的用户。SIP 服务器可以检查 Authorization 头字段中的 Request-URI，如果跟用户匹配，那么服务器就愿意接受转发的或者直接的请求，但是如果这两个域不同也不一定会导致失败。

7）分类鉴权方案中消息完整性保护中，作为 A2 值的计算的净化，具体实现时，应该假设消息体为空，实体的无用信息部分就被解析为 MD5 空字符串，或者是：

H（entity-body）= MD5（“”）=“d41d8cd98f00b204e9800998ecf8427e”

8）“qop”参数对于客户端和服务器也是可选的。但是在 WWW-Authenticate 和 Proxy-Authenticate 头字段中必须发送“qop”参数。如果客户机在质询中发现了“qop”参数，在随后的鉴权字段中都必须包含“qop”参数。本部分允许使用 Authentication-Info 头字段，该头字段可以提供消息体完整性检查和相互鉴权。

21 安全/多用途因特网邮件扩展（S/MIME）（可选）

SIP 协议消息可以携带使用 MIME 进行编码的消息体，以实现对消息体进行完整性和安全性加密保护。MIME 类型可分为“multipart/signed”和“application/pkcs7-mime”四类，具体内容分别见 RFC1847、RFC2630 和 RFC2633。网络中存在一些可以查看或者修改 SIP 消息（通常是 SDP 部分）的网络实体，因此，当 SIP 消息体使用 MIME 进行安全编码时，可以保证消息体的安全性。这种应用尤其适用于防火墙中。

21.1 S/MIME 证书（Certificates）

S/MIME 证书用于标识一个使用 S/MIME 的终端用户，这些 S/MIME 证书用于确认消息体由终端用户地址进行标识。其中，终端用户的 S/MIME 证书由“用户信息@域名”标识，通常这些证书与终端用

户的注册地址相同。

通常，S/MIME 证书与用于加密 SIP 消息体的密钥有关。虽然消息体被消息发送方分配一个私有密钥，但是消息体采用与消息接收方相关的公共密钥进行加密。因此，消息发送方必须事先获悉消息接收方的密钥以实现对消息体的加密。通常，公共密钥存储于 UA 中的虚密钥环（keyring）中。

本协议规定，任何支持 S/MIME 的用户代理必须有一个密钥环作为终端用户的 S/MIME 证书，该密钥环应实现终端用户注册地址和相应的 S/MIME 证书之间的映射。本协议规定终端用户应使用相同的注册地址和证书来标识 SIP 消息的发送方。

任何依赖终端用户证书的加密机制是有局限性的，因为目前尚不存在统一的为终端用户授权证书的认证机构。然而，终端用户必须从公开的证书认证机构获取一个证书。除此之外，终端用户也可以自定义一个证书。具体应用时，终端用户可使用一个预先配置的证书，这些预先配置的证书确保与所有 SIP 实体保持一定的信任关系。

尽管目前尚没有公开的集中目录用于记录终端用户的 S/MIME 证书，但是 S/MIME 证书的持有者应在任何公开目录上发布其证书。同样，用户代理客户（UAC）应支持使用手动或自动的方式来引入其在公开目录上的找到的与 SIP 请求目标 URI 相关的证书。

21.2 S/MIME 密钥交换

SIP 协议可使用以下方式来分发公共密钥：

1）当 SIP 消息中的 S/MIME 部使用 CMS SignedData 消息时，则该消息必须包含一个用于承载公共密钥的 S/MIME 证书来核查签名。

2）当 UAC 发送一个包含 S/MIME 的 SIP 请求时，或 UAC 发送一个非 INVITE 请求时，UAC 应将请求消息体部分构造成‘multipart/signed’CMS SignedData 消息体进行发送。如果 CMS 使用封装数据（Enveloped Data）且目标用户的公共密钥已知，则 UAC 应在 SignedData 消息中封装 EnvelopedData 消息。

3）当 UAS 接收到一个包含 S/MIME 证书的 S/MIME CMS 消息体的请求消息，则 UAS 应首先核实证书的合法性。UAS 应确认证书的 Subject，通常在 S/MIME 中 SubjectAltName 中会包含一个相应的身份标识，并将 SubjectAltName 的内容与 SIP 请求消息 From 头字段进行比较。如果消息使用无法被核实的自定义的证书或由非公开机构分配的证书，或者消息使用的证书可以被核实，但其 subject 与 From 请求头字段一致，则 UAS 必须向 UAC 通报证书的状态（状态包括证书 subject、证书分配者和密钥 fingerprint 信息）且必须向 UAC 请求证书使用资格。如果证书已被正确核实，且证书 subject 与 From 头字段一致，或如果 UAC 已授权 UAS 证书使用资格，则 UAS 应把该证书加入到自身的密钥环中，且密钥环中由证书拥有者的注册地址来标识不同的证书。

4）当 UAS 发送携带 S/MIME 部分的 SIP 响应消息来对第一个请求消息或非 INVITE 请求消息进行应答，UAS 应将请求消息体构造成‘multipart/signed’CMS SignedData 消息体进行发送。如果 CMS 使用封装数据（Enveloped Data），则 UAS 应在 SignedData 消息中封装 EnvelopedData 消息。

5）当 UAC 接收到携带包含 S/MIME 证书的 S/MIME CMS 的 SIP 响应时，UAC 应核实其证书的合法性。UAC 应确认证书的 Subject 并将 Subject 的内容与 SIP 响应消息 To 头字段进行比较，通常情况下这两部分内容是不相同的。如果消息使用无法被核实的自定义的证书或由非公开机构分配的证书，则 UAC 必须向 UAS 通报证书的状态（状态包括证书 subject、证书分配者和密钥 fingerprint 信息）且必须向 UAS 请求证书使用资格。如果证书已被正确核实，且证书 subject 与 To 头字段一致，或者如果 UAS 已授权 UAC

证书使用资格，则 UAC 应把该证书加入到自身的密钥环中，且密钥环中由证书拥有者的注册地址来标识不同的证书。如果 UAC 未预先将其自身的证书传送给 UAS，则 UAC 的下一个请求或响应消息必须使用 CMS SignedData 消息体。

6）当 UA 接收到的请求或响应消息时，如果消息中 From 头字段的值与 UA 中密钥环的某个值相等，则 UA 应将接收到消息中所包含的证书与密钥环中的证书进行比较。如果证书不同，则 UA 必须向其用户通知证书发生改变，并向用户请求证书使用资格。如果用户授权证书使用，则 UA 必须将此被授权的证书添加到密钥环中，且证书与用户的注册地址相对应。

当采用自定义证书或由非权威机构分配的证书时，以上密钥交换机制无法保证密钥的安全交换，密钥交换过程仍容易受到非法攻击。然而，以上密钥交换机制要优于目前被广泛应用的 SSH 机制。

如果 UA 接收到一个包含 S/MIME 消息体的 SIP 消息，且 S/MIME 消息体部分被一个由消息方无法识别的公共密钥进行加密，则 UA 必须向对方发送 493 响应消息拒绝该请求消息，且响应消息中应在参数类型为“certs-only”和“smime-type”的 MIME 消息体部分包含一个与消息接收方相关的合法证书。

如果 493 响应消息中未包含任何证书信息，则指示尽管消息响应方支持 S/MIME 签名，消息响应方无法识别 S/MIME 加密消息，。

当 UA 接收到一个包含 S/MIME 消息体的请求消息且 S/MIME 为非可选项，如果 UA 无法识别 MIME 的类型，则 UA 必须向消息发送方回送 415 响应消息（不支持的媒体类型）。收到 415 响应消息的 UA 应通知其用户远端实体不支持 S/MIME，或者 UA 也可以重新发送不携带 S/MIME 消息体的请求消息。然而，在某些情况下，415 响应可能会构成等级降低（downgrade）攻击。

当 UA 发送一个包含 S/MIME 消息体的请求消息时，如果 UA 接收到的响应消息中包含未加密的 MIME 消息体，则 UAC 应通知其用户本次呼叫流程不安全。然而，当支持 S/MIME 的 UA 接收到一个包含未加密消息体的请求消息时，UA 不应向对方回送包含加密消息体的响应消息。如果 UAS 期望从消息发送方接收到 S/MIME 消息体，例如，消息发送方的 From 头字段与密钥环中的标识符相同，则 UAS 应向其用户通知本次呼叫流程不能被加密。

证书管理出现异常时，会导致大量的通知用户情况发生，此时，用户会询问如何处理这类事件。一般来说，此类事件发生通常是由于证书发生异常改变或在需要安全的情况下无法保证安全而引起的，而并不一定是受到攻击所致。此时，用户可能会终止当前的任何网络连接或者拒绝所接收到的请求消息。在电话网络中，用户可能会挂机和回拨。并且用户可能希望寻找另外一种有效的方法来与第三方联系并确认用户与第三方之间的密钥是合法改变的。但是，有时用户会被迫改变他们的证书，例如当他们意识到私有密钥的安全性受到威胁时，用户必须采用合法手段生成一个新的密钥并与其他拥有其旧密钥的用户重新建立新的信任关系。

如果 UA 在对话过程中接收到一个 CMS SignedData 消息，且其中包含的证书与先前对话交换过程中使用的证书不一致，则 UA 必须向其用户通知证书发生改变，并暗示有潜在的安全隐患。

21.3 MIME 消息体加密

本协议目前只考虑使用两种类型的 MIME 消息体，这两类消息体的使用规则与 S/MIME 技术规范的定义大致相同，但存在少量的差异。

1）当且仅当与 CMS 分离签名一起使用时，必须使用类型为“multipart/signed”的 S/MIME 消息体。这样可实现与非 S/MIME 接收方的后向兼容。

2）包含 S/MIME 消息体的消息应携带“Content-Disposition”头字段，且“handling”参数的值应为“required”。

3）如果 UAC 希望向对方发送消息时，且其密钥环中不包含与对方注册地址相关的证书，则 UAC 不能向对方发送类型为“application/pkcs7-mime”的加密 MIME 消息体。UAC 也可向对方发送一个 OPTIONS 请求消息，且消息中包含一个 CMS 分离签名以获得对方的证书，其中 CMS 分离签名应包含在类型为“message/sip”的消息体中，有关类型为“message/sip”的消息体见本部分第 20.4 节。

S/MIME 消息体的发送方应使用“SMIMECapabilities”属性参数来描述发送方的能力和其通话过程中的使用规则。尤其是使用“preferSignedData”能力集的消息发送方，期望从接收方接收到类型为 CMS SignedData 的 MIME 消息体。

4）S/MIME 消息体发送方和接收方至少应支持 SHA1 和 3DES 算法，其中 SHA1 作为数字签名算法，3DES 作为加密算法。除此之外，还可以支持其他的数字签名和加密算法。发送方和接收方应在“SMIMECapabilities”参数中来协商双方所使用的数字签名和加密算法。

5）本协议规定 SIP 消息中的每一个 S/MIME 消息体仅能分配一个与之关联的证书。如果 UA 接收到一个消息中包含多个数字签名，则最后的那个签名应被视为该消息体的惟一的签名。本协议不允许消息中出现多个平行的数字签名。

21.4 SIP 隧道

S/MIME 可以将整个 SIP 消息封装到类型为“message/sip”的 MIME 消息体中以保证端到端鉴权完整性或 SIP 消息头字段的保密性，且按照与传统 SIP 消息体相同的 MIME 安全机制来处理这些被封装的 SIP 消息体。

如果 UAS 接收到一个包含类型为“message/sip”的 S/MIME 消息体，则 UAS 应回送一个包含相同 S/MIME 消息体类型的响应消息。

任何传统的 MIME 消息体（如 SDP）应封装到消息体内以获得相应的安全保障。且如果任何非加密的 MIME 类型需要在请求消息中被发送，则类型为“message/sip”的消息体可以承载于类型为“multipart/mixed”的消息体中进行发送。

21.4.1 SIP 头字段的完整性和保密性

当采用 S/MIME 完整性和保密性机制时，被封装的消息和封装后的消息可能会有所不同，因此，本节定义了头字段不同时的处理规则。

为了实现不同时戳，本协议规定所有被封装成类型为“message/sip”的 MIME 消息体的 SIP 消息应在封装前和封装后消息中包含一个 Date 头字段。

21.4.2 完整性

当执行头字段完整性检查时，应参照本标准第 20 章所定义的比较规则来检查被封装前与封装后消息的头字段值。

其中，可以被代理服务器合法修改的头字段有 Request-URI、Via、Record-Route、Route、Max-Forwards 和 Proxy-Authorization。因此，这些字段无法实现端到端的完整性不应被认为是缺乏安全保障。然而，除此之外的其他任何头字段发生改变时，则可造成完整性威胁，UA 应向其用户通知头字段之间存在差异。

21.4.3 保密性

当 SIP 消息被加密时，其头字段应包含于封装前消息体内而不应出现在封装后消息的头字段中。其

中，某些头字段必须以明文方式出现在请求和响应消息中，这些头字段包括：To、From、Call-ID、Cseq 和 Contact。通常，Call-ID、Cseq 和 Contact 头字段没有必要进行加密，而 To 和 From 头字段可以在封装后的消息中以另外一种方式出现。经过加密后的消息体不能用于标识事务交换或对话。如果经过加密后的消息体中所包含的 From 头字段与封装后消息中 From 头字段的值不同，则用户应显示被加密的消息体内 From 头字段的值，但该值不能用于以后的封装后的任何消息加密后的头字段中。

通常，用户代理需要对一些具有端到端含义的头字段进行加密，这些头字段包括 Subject，Reply-To、Organization、Accept、Accept-Encoding、Accept-Language、Alert-Info、Error-Info、Authentication-Info、Expires、In-Reply-To、Require、Supported、Unsupported、Retry-After、User-Agent、Server 和 Warning。如果上述头字段出现在加密消息体内，则封装后的消息中该头字段的值也应用加密形式代替，且这些头字段不能用于其他消息中的头字段中。本部分规定加密后，封装前的消息头字段和封装后头字段的值应保持不变。

由于 MIME 实体是承载于封装前消息体中，则 UA 还应对与 MIME 相关的头字段进行加密，这些头字段包括 MIME-Version、Content-Type、Content-Length，、Content-Language、Content-Encoding 和 Content-Disposition。封装后消息应包含与其承载的 S/MIME 消息体一致的 MIMIE 头字段，当接收到包含这些头字段的消息时，消息接收方应将这些 MIME 头字段看作普通 MIME 头字段和消息体来处理。

通常，以下头字段没有必要进行加密传输，如 Min-Expires、Timestamp、Authorization、Priority 和 WWW-Authenticate，以及如前所述那些可以被代理服务器改变的头字段。本标准规定如果以上头字段没有包含于封装前的消息中，则用户代理（UA）不能将这些头字段包含于加密后的消息体内，且 UA 接收到加密消息体内包含以上头字段的消息，应忽略这些头字段。

SIP 协议进行扩展后可能会定义一些新的头字段，因此，进行扩展同时应描述这些新增加的头字段的完整性和保密性。如果用户代理（UA）接收到一个包含未知头字段的 SIP 消息，且该头字段无法保障完整性，则 UA 应忽略这些未知的头字段值。

21.4.4 隧道完整和鉴权

如果消息发送方将需要加密传送的头字段复制在类型为“message/sip”的 MIME 消息体内，且 MIME 消息体由 CMS 分离数字签名进行标识，则承载于 S/MIME 消息体内的 SIP 隧道消息可以为 SIP 头字段提供完整性保护。

如果类型为“message/sip”的 MIME 消息体内包含基本的对话标识头字段（如 To、From，、Call-ID 和 CSeq），则该由 CMS 分离数字签名进行标识的 MIME 消息体可以提供一些基本的鉴权机制。如果消息接收方无法识别用于标识消息体的证书，或者该证书无法被核实，则用于标识消息体的数字签名可确保对话中其后的请求可被同一个证书拥有者传送。且如果消息接收方必须确保证书的可靠性，则数字签名可用于标识证书主体的身份。

为了避免由于头字段的增减所造成的歧义，消息发送方应将整个头字段值复制于被加密的消息体内，且任何需要完整性保护的消息体应附加于加密的消息体内。

如果加密的消息体内包含 Data 头字段，则消息接收方应将该头字段值与其自身的时钟进行比较。如果时间存在显著偏差，则用户代理（UA）需要向其用户通报异常状态及安全隐患。

如果消息接收方检查到请求消息的完整性被破坏，则消息接收方应回送 403（Forbidden）响应消息，拒绝对该请求消息进行处理，且 UA 应向其用户通报状态并请求解决问题处理方法。

21.4.5 隧道加密

隧道加密机制用于将 CMS EnvelopedData 消息内的类型为“message/sip”的 MIME 消息体进行加密。具体实现时，网络会利用多数消息头字段来获取相关信息，因此，对 S/MIME 消息进行加密通常是对消息体部分（如 SDP）进行加密，而不是对消息头字段进行加密。诸如 Subject 和 Organization 等可以保证端到端完整性的头字段。除此之外，另外一个可以被加密传送的头字段的具体应用是采用匿名发送的方式，例如，可以将一个请求消息中的 From 头字段值设置为匿名（如：sip：anonymous@anonymizer.invalid）。因此，下一个消息中包含消息发送方真实注册地址的 From 头字段应被加密成类型为“message/sip”的 MIME 消息体内，且仅对话接收方可获取该 From 头字段的值。

如果采用这种匿名传送机制，SIP 消息中的 From 头字段不能被消息接收方用作索引值来获取密钥环中的证书，从而无法获得消息发送方的 S/MIME 密钥。因此，消息接收方首先应对加密后的消息进行解密操作，解密后的 From 头字段值才可作为索引值来获取密钥环中的证书。

为了保证端到端消息的完整性，被加密的类型为“message/sip”的 MIME 消息体应由消息接收方分配一个数字签名进行标识，标识后生成类型为“multipart/signed”的 MIME 消息体，该 MIME 消息体包含一个被加密的实体和一个数字签名，被加密的实体和数字签名的类型都为“application/pkcs7-mime”。

以下示例为包含一个加密消息体和一个数字签名的 SIP 消息，消息中由“*”符号组成的文本框内的内容为加密的消息体。

22 SIP 协议的扩展 BNF 语法

本标准中的所有机制都在以文本形式和RFC 2234中定义的扩展Backus-Naur形式（BNF）语法进行描述。

22.1 基本规则

下列规则在整个标准中采用以描述基本的语义分解结构。US-ASCII编码的字符集由ANSI X3.4-1986定义。

```
alphanum  =  ALPHA / DIGIT
```

在RFC 2396中体现的几个规则已经更新以使这些规则与RFC 2234相兼容，这些规则包括：

```
reserved    =  ";" / "/" / "?" / ":" / "@" / "&" / "=" / "+" / "$" / ","
unreserved  =  alphanum / mark
mark        =  "-" / "_" / "." / "!" / "~" / "*" / "'" / "(" / ")"
escaped     =  "%" HEXDIG HEXDIG
```

如果连线以空格或者水平tab符开始，则SIP的头头字段字段值可以用多个线括起来。所有的LWS，包括，都具有与SP相同的语义。在解释域值或者向下游转发消息之前，接收者可能以单个SP代替任何LWS。如RFC 2616所描述的，这样做是按照HTTP/1.1来正确地运作。当LWS作为任选时，在符号和分隔符之间应使用SWS构造法。

```
LWS  =  [*WSP CRLF] 1*WSP; linear whitespace
SWS  =  [LWS]; sep whitespace
```

为了将报头名字与其他值分开，可以使用冒号。根据以上规则，冒号前允许有空格但没有行结束符，冒号后允许有空格，可以包括空格。HCOLON定义了这种构造法。

```
HCOLON  =  *( SP / HTAB ) ":" SWS
```

TEXT-UTF8规则仅仅用于描述性的域内容和值，这些是将不会被解析。TEXT-UTF8的字符包括来自UTF-8字符集的字符。TEXT-UTF8-TRIM规则用于描述性的域内容，这些内容不是被引号括起的字符串，且最前面和最后面的LWS也是没有意义的。在这方面，SIP不同于HTTP，HTTP使用ISO 8859-1字符集。

```
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
```

在TEXT-UTF8-TRIM定义中允许出现CRLF，但仅仅做为头字段延续的一部分。在解释在TEXT-UTF8-TRIM值之前，可以用单个SP替代t括起的 LWS是。

在某些协议单元中使用了16进制的数字字符。某些单元如鉴权单元，强制16进制字母是小写的。

```
LHEX  =  DIGIT / %x61-66；小写字母a~f
```

许多SIP头字段值由被LWS分隔的词或特殊字符组成。除非其他地方有所陈述，符号是对大小写不敏感的。这些特殊的字符必须位于被引号引用的字符串中以用于参数值中。在Call-ID中使用的构词法允许采用大多数分隔符。

```
Token       =  1* ( alphanum / "-" / "." / "!" / "%" / "*"
                / "_" / "+" / "`" / "'" / "~" )
separators  =  " ( " / " ) " / "<" / ">" / "@" / ", " / ";" / ": " / "\" / DQUOTE /
                "/" / "[" / "]" / "?" / "=" / "{" / "}" / SP / HTAB
word        =  1* ( alphanum / "-" / "." / "!" / "%" / "*" /
                "_" / "+" / "`" / "'" / "~" / " ( " / " ) " / "<" / ">" /
                ": " / "\" / DQUOTE / "/" / "[" / "]" / "?" / "{" / "}" )
```

当使用符号或者在单元之间使用分隔符时，空格经常被允许使用在这些字符之前或者之后：

```
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

在SIP头字段字段中包括注释，用圆括号括起。注释仅仅允许在包含了“comment”的头字段中出现，做为其头字段值定义的一部分。在所有的其他头字段中，圆括号被认为是字段值的一部分。

comment = LPAREN *（ ctext / quoted-pair / comment ）RPAREN

ctext = %x21-27 / %x2A-5B / %x5D-7E / UTF8-NONASCII / LWS

ctext包括了所有的字符，除了左右括号和反斜线符号。如果一串文字被双引号所包围，则该文字串按照单个词来分析。在引号引用的字符串中，需要避免引号（ “ ）和反斜线符（\）。

quoted-string = SWS DQUOTE *（ qdtext / quoted-pair ）DQUOTE

qdtext = LWS / %x21 / %x23-5B / %x5D-7E / UTF8-NONASCII

仅仅在引用的字符串和注释的构造中，反斜线符（ “\” ）可以当做单个字符引用机制来使用。不象HTTP/1.1，字符CR和LF因此转义来避免与line folding和头分隔冲突，

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见RFC 2809。但是那里所允许的任何字符在SIP URI的用户部分都是不允许的，都必须要避免。

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-toke1 =  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
```

附 录 A
（资料性附录）
安全：威胁模式和安全建议

SIP协议不易保证安全。协议中介的使用、多方信任关系、网络实体之间无信任的应用及用户间的操作使得安全很难保证；现有的解决方案在各种环境和用法中没有得到广泛的配合。因此，需要一些独特的实现机制。

SIP信令安全本身对RTP等配合SIP使用的协议的安全或者是涉及到SIP可能携带的某些特定部分的安全是没有关系的，虽然MIME在SIP安全上有着重要的作用。任何与对话有关的媒体都可以不依赖于任何相关的SIP信令来进行端到端的加密。本部分不涉及媒体加密的内容。

A.1 攻击和威胁模型

下面所介绍的威胁对大多数的SIP配置都很常见。这些威胁用来说明SIP要求的每种安全服务。

以下的示例并不是一个针对SIP威胁的详细列表，确切地说，这些“经典”的威胁证明了所需的特殊的安全服务有能力防止所有的威胁。

下列攻击假设攻击者可以读到任何网络上的分组环境，即知道SIP将在公网上经常使用。网络上的攻击者能够在某些中介上修改分组。攻击者也可能窃取服务，窃听或者中断。

注册攻击：

SIP的注册机制允许一个UA标识自己是一个用户所在的设备的注册服务器，该用户有一个指定的记录地址（address-of-record）。注册服务器可以从REGISTER消息的From 头字段中的ID来判定这个请求是否可以修改To头字段中和记录地址有关的联系地址，虽然这两个字段常常是相同的，仍然有许多有效的配置可以使第三方注册为代表用户的联系地址。

SIP请求消息的From 头字段可以被UA的所有者任意修改，因此就容易受到恶意的注册攻击。攻击者可以成功地假扮授权方来改写记录地址的联系地址，例如它可以取消所有URI现存的联系地址并把攻击者的设备注册为正确的联系地址，从而使所有发向被攻击用户的请求指向攻击者的设备。

这是一种针对没有保护的请求发起者的威胁。任何作为一个有值服务的SIP UAS都希望能够通过对所收到的请求鉴权来控制接入自己的资源，甚至每个终端用户UA也希望确认一个请求发起者的身份。SIP实体对请求发起者进行鉴权对于安全服务是很必要的。

A.2 伪装服务器

请求消息的目的域通常都在request-URI中指定。UA为了发送请求消息通常联系该域中的服务器。但是这样攻击者就可以伪装远端服务器来截获UA的请求消息。

例如，一个位于chicago.com域中的重定向服务器伪装一个位于biloxi.com中的服务器。用户代理向biloxi.com发送请求消息，chicago.com的重定向服务器以伪造的响应回答，但响应带有来自于biloxi.com的相应的SIP头。重定向响应中伪造的关联地址使发起UA指向非安全的资源或者使得发向biloxi.com的请求无法继续。

这种威胁有很多种方式，其中一些是很严重的。例如，一个发往bolixi.com的注册消息被chicago.com

截获，并回以一个伪造的301响应（Moved Permanently）。这个响应看来像是从boloxi.com发送的但是却将chicago.com指定为正确的注册服务器，以后所有UA发起者的注册请求都将送到chicago.com。

为了防止此类威胁，UA必须可以鉴权其请求消息发向的服务器。

A.2.1 篡改消息体

UA请求消息都是通过可信任的代理服务器路由的。无论该信任关系是如何建立的，UA必须信任一个代理服务器来路由请求消息，并且不应检查或是修改该请求所包含的消息体。

如果UA可以用一个SIP消息体来传送媒体会话密钥，虽然它信任所联系的域代理服务器可以正确地传送信令，但是该域管理员不应解密随后的任何媒体会话，而且如果该代理服务器是恶意的，它就可以修改会话密钥，也可以作为一个中间方，或者改写发起UA所要求的安全特性。

这类威胁不仅针对会话密钥，而且也针对大多数SIP消息中端到端所携带的内容格式。这包括发给用户的MIME消息体、SDP或者封装的电话信令等等。攻击者可以修改SDP消息体，例如，为了窃听话音通信而将一个RTP媒体流指向一个窃听装置。

某些SIP的头字段为端到端的，如Subject头字段。UA也可以保护这些头字段和消息体的安全。但是，由于代理服务器在路由一个请求时，该请求的许多头字段可以被服务器合理地检查或更改，所以不是所有的头字段都可以保证端到端的安全。

UA希望能够保证SIP消息体和有限的头字段的端到端的安全。此类安全服务要求包括机密性、完整性和鉴权。这些端到端的安全服务应该独立于用于保护代理服务器之类的媒介的交互手段的安全服务。

A.2.2 断开会话

一旦对话经过初始化消息而建立，其随后发送的请求消息可以修改对话或者会话的状态。该会话的参与者必须确认这些请求消息是否经攻击者伪造。

例如，某第三方攻击者为了获得对话的参数如to 标签和from标签等，而截获某些对话的初始话消息，该消息是通信双方共享的，然后在该对话中插入一个来自对话中任意方的BYE请求消息。目标地址收到BYE消息，这个对话就会过早地结束。

类似的会话中（mid－session）威胁还包括发送伪造的re－INVITE消息来修改对话，这样会降低会话的安全性或者通过重定向媒体流来窃听攻击。

针对于该威胁的最有效的方法就是对BYE发送者进行鉴权。接收者不需要与对发送者的ID进行绝对的确认，只需要知道BYE来自于该对话中的同一方（由于消息的机密性，攻击者在不知道该会话参数的情况下就不可能伪造BYE消息）。但是，一些媒介（如代理服务器）在会话建立时需要检查这些参数。

A.2.3 服务拒绝和放大

服务拒绝攻击（Denial-of-service）通常是通过让过多的网络流量指向某个特定网络实体的接口而使该实体不可用。分布式的服务拒绝攻击是某个网络用户让多个网络主机用极大的流量注入目标主机中。

在许多网络结构中，SIP代理服务器为了接收全球的IP终端发来的请求消息面向公共INTERNET。而某些分布式服务拒绝攻击者必须被SIP系统的实施者和运营者所验证和寻址，这样，该SIP代理服务器会受到该类型的攻击。

攻击者能够产生带有伪造的源IP地址和相应的Via头字段的假请求消息，从而把一个目标主机标识为请求的发起者，然后将这个请求发给很多的SIP网络实体，从而用 UA或者代理来对这个目标发起拒绝服务的流量。

类似的的攻击还有：攻击者也可以在一个请求中使用伪造的Route头字段值来标识目标主机，并发送这种消息给分叉代理服务器（forking proxy）来放大送往目标的消息。

在攻击者确信由请求发起的对话将向后产生许多事务的时候，用记录路由（record－route）也可以有类似的作用。

如果REGISTER请求消息没有被注册服务器正确地鉴权，那么很多服务拒绝攻击就会开始。攻击者可以使某个管理域中的一些或者所有的用户被取消注册，从而使这些用户不能被加入一个新的会话中。为了在服务拒绝攻击中将注册服务器和任何有关的代理服务器作为放大器，攻击者可以以一个给定的记录地址注册许多个指向同一个主机的联系地址。攻击者也可通过将大量的绑定放入寄存器来删除注册服务器的可用的内存和磁盘资源。

使用多播来传送请求消息能够在很大程度上增加服务拒绝攻击的可能性。

A.3 安全机制

SIP协议所需的基本安全服务包括消息的机密性及完整性保留、重发攻击或消息欺骗的预防、对会话参与者提供保密及鉴权服务以及预防服务拒绝攻击。SIP消息中每个部分各自需要其机密性、完整性和鉴权的安全服务。

SIP协议除了采用新的专用的安全机制之外，还重用了那些来自于HTTP和SMTP的已有的可用的安全模式。

消息的全加密是为信令传送提供机密性最好的方法——它可以保证消息不被恶意中介修改。但由于在大多数的网络结构中为了正确路由SIP请求，消息中如Request－URI、Route和Via这样的字段对代理应该是可见的，因此SIP请求和响应消息不能被简单地进行全部的端到端的加密。代理服务器也需要修改消息的某些特性如增加via 头字段的值等来完成某种功能。因此，UA必须信任代理服务器。本部分建议使用低层的安全机制，即在传输线上对整个SIP请求或响应进行逐跳的加密，这样就可以让端点校验其请求发向代理服务器的身份了。

SIP实体也需要在一种安全的方式下标识自己。当一个端点对一个对等UA或者一个代理服务器声明它的用户的身份时，该实体应该是可证实的。此时，可以采用密码鉴权机制。

SIP消息体的独立的安全机制对端到端的相互鉴权提供了一个可供选择的方案，同时也在某种程度上规定了用户代理必须要相信中介。

A.3.1 传输层和网络层安全

传输或网络层的安全服务包括加密信令流并保证消息的机密性和完整性。

通常，证书被用于建立低层的安全。在许多网络结构中，这些证书也可用来作为一种鉴权的方式。

TLS和IPSec分别用于传输层和网络层提供安全服务。

IPSec是一套网络层协议工具，它可以全部替换传统IP的安全策略。它常用在主机或者管理域已经有着相互信任关系的网络结构中。IPSec通常在主机的操作系统或者在为从一个特定接口收到的所有流量提供机密性和完整性的安全网关上执行。IPSec也可以基于逐跳的应用。

在许多网络结构中，IPSec不需要集成在SIP应用中；IPSec适用于难以直接在主机上加入安全配置的网络结构中。与它们的第一跳代理服务器有着共享密钥环的UA很适合用IPSec保证安全。任何用于SIP的IPSec配置都需要一个描述SIP安全所需的协议工具的轮廓。

TLS提供了基于有连接的协议的传输层的安全；“tls”（TLS over TCP）可以在一个Via 头字段值或SIP-URI中被指定为所需的传输协议。TLS适用于在主机间无现存信任关系并需要逐跳安全的网络结构中。例如，Alice信任她的本地代理服务器，在证书交换之后，该服务器决定信任Bob所信任的本地代理服务器，这样Bob和Alice可以安全地通信。

TLS必须和SIP应用一起实现。SIP中的传输机制是逐跳指定的，这样在TLS上向代理服务器发送请求的UA就不能保证TLS是可以用于端到端安全的。

在具体实现TLS时必须至少支持TLS_RSA_WITH_AES_128_CBC_SHA 密码适配集。为了后向兼容，代理服务器、重定向服务器和注册服务器应该支持TLS_RSA_WITH_3DES_EDE_CBC_SHA。具体实现时，也可以支持任何其他的密码适配集。

在本标准中，TLS为可选。

A.3.2 SIPS URI 方案（可选）

虽然SIPS URI方案附于SIP URI的语法上（19所述），但其语义和SIP URI有很多不同的。SIPS URI中，资源可以指定自己是安全的。

SIPS URI可以作为一个特定用户的记录地址，通过该URI可以得知这个用户。SIPS 方案当在请求消息中作为Request - URI时，即表示在该请求到达负责Request-URI中的那部分域的SIP 实体之前的前向上的每一跳，且必须以TLS保证安全；如果它到达一个可疑的域时，将和本地的安全与路由政策相协调，可在任何最后一跳上使用TLS。SIPS在用于请求发起者侧时，如发起者用SIPS URI作为目标记录地址，它将指定到达目标域的整个请求路径的安全。

SIPS 方案可以用于记录地址、联系地址（包括那些REGISTER方法，Contact头的内容）和Router头字段中。在每种情况中，SIPS URI 方案都允许这些现有的字段来指定安全资源。上述任何情况下，任何解除SIPS URI的引用方式都有着自己的安全特性。

本协议规定，SIPS的用法应该使用相互的TLS鉴权方式，如密码适配组。

TLS_RSA_WITH_AES_128_CBC_SHA所要求，在鉴权过程中收到的证书应该与客户所持的根证书一起被证实；证书证实失败应该导致请求失败。

注意到在SIPS URI 方案中，传输是独立于TLS的，虽然UDP不是有效的SIPS传输方式。这样“sips：alice@atlanta.com;transport=tcp”和“sips：alice@atlanta.com;transport=sctp”就都是有效的。本标准不建议使用“transport=tls”，部分原因是它是特定地针对请求的每一跳的。

把SIPS URI作为记录地址分发出去的用户可以选择那些会拒绝非安全传输的请求的设备。

A.3.3 HTTP鉴权

SIP 中可以使用基于HTTP鉴权的质询，该鉴权是依靠401和407响应代码和携带口令及证书的头字段来进行的。HTTP 分类鉴权方案在SIP中无需作大的改变就可以重用，并可完成重发保护和单向鉴权。

关于分类鉴权见本标准第20章。

A.3.4 S/MIME（可选）

如上所述，由于网络中介（如代理服务器）为了正确路由消息而必须看到消息中某个头字段，所以不适于对整个SIP消息进行端到端的加密。如果该中介被安全联盟所排斥，则SIP消息不可路由。

S/MIME允许SIP UA加密SIP中的MIME部分，在不影响消息头的情况下保证这些部分的端到端安全。S/MIME可以保证消息体端到端的机密性、完整性和相互鉴权。通过SIP消息隧道，可以使用S/MIME来保

证SIP头字段的完整性及机密性。

关于S/MIME见本标准第21章。

A.4 安全机制的实现

A.4.1 SIP实现要求

代理服务器，重定向服务器和注册服务器必须使用TLS，并且必须支持交互和单向认证。本部分建议，UA可以初始化TLS；并可以作为TLS服务器。代理服务器，重定向服务器和注册服务器应该拥有一个站点证书，其题目与其规范的主机名一致。UA可以有自己的证书用来与TLS进行相互认证，对于它们的用法不再本部分的范围之内。所有支持TLS的SIP 实体必须有对在TLS协商中收到证书的证实机制；这样就必须拥有由证书授权者所发的一个或多个根证书。

所有支持TLS 的SIP 实体必须也支持SIPS URI 方案。

代理服务器、重定向服务器、注册服务器和UA也可以实现IPSec或者其他的低层安全协议。

当UA试图连接一个代理服务器、重定向服务器或者注册服务器时，UAC应该初始化一个TLS连接并在其上发送SIP消息。在一些网络结构中，UAS也可以接收TLS连接的请求消息。

代理服务器，重定向服务器和UA必须实现分类鉴权，具体要求见本标准第19章。代理服务器、重定向服务器和注册服务器配置的Digest域应该至少有一个，并且至少有一个给定的服务器所支持的“realm”字符串与该服务器的主机名和域名一致。

UA可以支持MIME消息体的签名和加密，关于S/MIME的证书的传输详见本标准第20章。如果UA有一个或多个授权的根证书来验证TLS或IPSec的证书，它也应该重用该根证书来校验S/MIME证书。UA可以有专为验证S/MIME证书的根证书。

A.4.2 安全解决方案

与这些安全机制相应的操作在某种程度上可以遵循现有的 Web 或 E-mail 的安全模式。在更高的水平上，UA 对服务器包括代理服务器、重定向服务器和注册服务器通过一个分类鉴权的用户名和验证 TLS 或 IPSec 的证书的密码来为自己鉴权；服务器也对 UA 或者对另一个服务器在每一跳上通过一个由 TLS 发送的站点证书为自己鉴权。

在一个对等的水平上，UA 相信网络上一般的相互鉴权；但是，S/MIME 也可以在网络不信任或者网络不被信任的情况下用来提供直接鉴权。

以下的例子说明这些安全机制被不同的 UA 和服务器用来防止 26.1 中所介绍的攻击类型。当实施者和网络管理者可以遵循本章中余下部分的标准化指导，这些就仅作为实现的范例了。

A.4.2.1 注册

当 UA 在线并向它的本地管理域注册，它应该建立一个与其注册服务器间的 TLS 连接。关于 UA 如何连接到它的注册服务器见本标准 7 章。注册服务器应该为 UA 提供证书，并且证书的站点标识必须与 UA 要注册的域相一致。例如，如果 UA 要注册记录地址 “alice@altanta.com”，站点证书就必须标识一个在 atlanta.com 域中的主机（如 sip.atlanta.com）。当 UA 收到 TLS Certificate 消息，它就应该校验该证书并检查站点标识。如果证书是无效的或废止的，或者它不能标识正确方，则 UA 就不可以发送 REGISTER 消息或者继续注册。

当注册服务器已经提供了一个有效的证书，UA 就知道该注册服务器不是可能重定向该 UA，窃取密

码或者进行其他类似的攻击。

那么 UA 创建了一个 REGISTER 请求，该请求应该定位到一个与从注册服务器那里收到的证书相一致的 Request - URI。当该 UA 在现有的 TLS 连接上发送 REGISTER 请求，注册服务器应该用 401 响应验证这些请求。响应的 Proxy - Authenticate 头字段中的“realm”参数应该与之前在站点证书中给出的域相一致。当 UAC 收到口令，它应该命令用户提供证书或者从口令中“realm”参数对应的密钥环（keyring）中得到一个正确的证书。证书中的用户名应该与 REGISTER 的 To 头字段中 URI 的“userinfo”部分一致。一旦 Digest 证书被插入到一个正确的 Proxy-Authorization 头字段中，REGISTER 就应该被重新提交给注册服务器。

由于注册服务器要求用户代理验证自己，所以攻击者就很难对一个用户记录地址伪造 REGISTER 请求。另外，由于 REGISTER 在一个保密的 TLS 连接上发送，所以攻击者在任何的重发攻击中也不能截取 REGISTER 消息来复制证书。

当注册被注册服务器所接受，UA 就应该让该 TLS 连接始终开放，以便注册服务器也可以作为该管理域中的用户请求发向的代理服务器。现存的 TLS 连接将被重用并把引入的请求传送到刚刚完成注册的 UA。

由于 UA 已经验证了 TLS 另一端的服务器，因此该连接上的所有请求在通过代理服务器的时候都是可见的，因此攻击者不能制造经这台代理服务器发送的欺骗请求。

A.4.2.2 域内请求

假设 Alice 的 UA 要初始化与一个在远端管理域的用户（bob@biloxi.com）之间的会话，也假设本地管理域（atlanta.com）有一个本地的出代理。

一个管理域的处理入请求的代理服务器也可以作为一个本地的出代理。为了简单起见，我们假设这是一个 atlantic.com 的例子（否则用户代理要初始化一个新的 TLS 连接隔开该地的服务器）。假设客户已经完成了上述的注册过程，当其发送 INVITE 到另一个用户时，它就应该重用与本地代理服务器之间的 TLS 连接。UA 应该重用 INVITE 消息中缓存的证书以避免对用户不必要的提示。

当本地的出代理服务器验证用户在 INVITE 中的证书时，它应该检查 Request-URI 来决定此消息该如何被路由。如果 Request-URI 中的“domainname” 部分符合本地域名（atlanta.com）而不是 biloxi.com，则代理服务器就会咨询它的本地服务以决定如何更好地到达请求的用户。

假设如果 alice@atlanta.com 试图连接“alex@altanta.com”，本地代理服务器就会将该请求转到由 Alex 注册时于注册服务器间已经建立的 TLS 连接上。由于 Alex 收到从它信任的信道上送来的请求，它就确信这个 Alice 请求已经被本地管理域的代理服务器所授权。

但是，在这个由 Request-URI 指定的远端域的例子中，atlanta.com 的本地出代理服务器应该因此同远端 biloxi.com 域中的服务器建立一个 TLS 连接。由于本 TLS 连接上的双方都是进行站点认证的服务器，所以应该有相互的 TLS 鉴权。连接的双方应该校验并检查对方的证书，记录出现在证书中的域名以便于和 SIP 消息的头字段相比较。比如，atlanta.com 代理服务器应该在这个阶段校验这个从远端收到的证书是否与 biloxi.com 域一致。当它完成这些工作并且 TLS 协商完成，就在两个代理间生成一个安全信道，这样，atlanta.com 代理就可以将 INVITE 消息转给 biloxi.com。

Biloxi.com 的代理服务器应该依次检查 atlanta.com 的代理服务器的证书并将该证书中声明的域与 INVITE 的 From 头字段中的“domainname”部分相比较。Biloxi 代理可以有一个严格的安全政策要求拒

绝与代理它们的管理域不匹配的请求。

制定这样的安全政策可以防止经常产生垃圾邮件的 SIP 等价物 SMTP “开放转发”（open relay）。

但是，这个政策只能保证从其归属域中发来的请求的安全，它不能让 biloxi.com 得知 atlanta.com 是如何鉴权 Alice 的。只有当 biloxi.com 从其他方面得知 atlanta.com 的鉴权政策，它才有可能确认 Alice 是如何证实自己的身份的。Biloxi.com 然后可能制定一个更加严格的政策禁止来自于不知道是否与 biloxi.com 共享同一个鉴权政策的管理域的请求。

一旦 INVITE 消息被 biloxi 代理所证实，则该代理服务器就应该标识现存的与该消息的目标用户（本例中 “bob@biloxi.com”）有关的 TLS 信道。INVITE 消息应该在这个信道上转给 Bob。由于该请求是从之前被证实为 biloxi 代理的 TLS 连接上收到的，虽然不需要相信 Alice 的身份，Bob 也该知道 From 头字段没有被篡改并且 Alice 已经被 atlanta.com 所验证。

每个代理服务器转发请求之前都应该在该请求中加入一个 Record-Route 头字段以使以后此对话中的所有请求都会经此代理服务器，从而代理服务器可以继续在对话的整个生存期内提供安全服务。如果代理服务器没有将自己加入到 Record-Route 中去，以后的消息就会在没有任何安全服务的情况下直接在 Alice 和 Bob 间进行端到端的传送（除非双方都同意某个独立的端到端的安全，如 S/MIME）。在这点，SIP 梯形模型就可以提供一个更好的网络结构，即站点代理间的协定可以在 Alice 和 Bob 间提供一个合适的安全信道。

假设攻击者想破坏这种结构，由于它无法确定会话的参数并且完整性机制也保护着 Alice 和 Bob 间的业务，所以它就不能伪造 BYE 请求并将其插入到 Bob 和 Alice 间的信令流中去。

A.4.2.3 对等端请求

另外，UA 所声明的标识 carol@chicago.com 没有本地的出代理。当 Carol 想发送 INVITE 到 bob@biloxi.com，她的 UA 应该初始化一个直接到 biloxi 的 TLS 连接。当它的 UA 收到一个从 biloxi 代理发来的证书，在 Carol 于该 TLS 连接上传送 INVITE 消息之前该证书就应正常被验证。然而 Carol 没有办法向 biloxi 代理证明其身份，但在 INVITE 的 “message/sip” 部分上有一个 Carol 与 CMS 分开的签名。在这种情况下，由于 Carol 和 biloxi.com 没有正式的关联，所以她就不太可能有任何 biloxi.com 域的证书。Biloxi 代理也可能有严格的政策对于在 From 头字段的 “domainname” 部分中无 biloxi.com 的请求甚至不用验证即将其排除，因为它认为这些用户未经授权。

Biloxi 代理对 bob 的政策是所有未授权的请求应该被重定向到正确的并不是注册为 bob@Biloxi.com 的联系地址上，即<sip：bob@192.0.2.4>。Carol 收到了它到 biloxi 代理间建立的 TLS 连接上的重定向响应，则它就会相信这个联系地址是准确的。

然后，Carol 应该建立一个与指定的地址间的 TCP 连接并发送一个新的其 Request-URI 中包括接收联系地址（当请求被读时重计算签名）的 INVITE 消息。Bob 在不安全的接口上收到这个 INVITE 消息，但其 UA 在这种情况下检查并承认该请求 From 头字段，并随后用一个本地缓存的证书来匹配 INVITE 消息体的签名中的证书。它以同样的方式回答，对 Carol 验证自己并开始一个安全的对话。

有时管理域中的防火墙或者 NAT 可以把与 UA 直接建立的 TCP 连接排除在外。在这种情况下，代理服务器也可能以一种无本地服务器指定的信任关系（例如放弃一个现存的 TLS 连接并在明文 cleartext 的 TCP 上转发请求的方式把请求中继到 UA。

A.4.2.4 DoS保护

为了将使用这些安全方案的网络结构遭受服务拒绝攻击的风险降到最小，实施者应该遵循以下的处理程序。

当一个 SIP 代理服务器主机在公共 Internet 上是可路由的，它应该放置在一个有防御性政策（阻塞源端路由的业务，最好过滤 PING 业务）的管理域中。TLS 和 IPSec 也都能利用参与集合了安全隧道和 Socket 的安全联盟位于管理域边缘的有防护的主机。这些有防护的主机也可以承受拒绝服务的攻击，保证管理域内的 SIP 主机不会被多余的消息传送所消耗资源。

无论使用什么安全方案，指向代理服务器的消息的洪泛都会锁住代理服务器的资源并且阻止有用的业务到达目的地。代理服务器上有与处理一个 SIP 会话有关的可计算的损失，并且这种损失对于一个有状态（stateful）的代理服务器比对一个无状态（无状态）的服务器要大，因此有状态的代理要比无状态的代理更易受到洪泛的攻击。

UA 和代理服务器应该只用单个的 401（未鉴权）或 407（代理鉴权要求）验证可疑的请求，放弃常规的响应重传算法，从而对于未鉴权的请求就成为无状态请求。

401（未鉴权）或 407（代理鉴权要求）状态响应的重传使攻击者用伪造的头字段值（如 Via）来使业务指向第三方的问题变得更加严重。

总之，代理服务器通过如 TLS 这样的机制进行相互鉴权在很大程度上减少了欺诈中介引入伪造的请求或响应拒绝服务的可能性，这使得攻击者更难使无辜的 SIP 节点变成放大的代理。

A.5 局限性

虽然用这些安全体制作为判断方法可以制止很多威胁的发生，但实施者和网络运营者必须认识到在这些机制的范围内仍然有着局限性。

A.5.1 HTTP 分类鉴权

SIP 采用 HTTP 分类鉴权是有着一定的局限性的，其中最主要的就是分类鉴权的完整性机制在 SIP 中用处不大。特别是它们提出了对 Request-URI 的保护和消息的方法对于 UA 最希望保证安全的头字段却没有任何作用。

现有的重发保护机制在 SIP 中也有局限性。如“next - nonce”机制，不支持管道传送的请求。“nonce-count”机制应该被用于重发保护。

HTTP 分类鉴权的另一个局限性就是域的范围。当用户想对一个与之已有关联的资源（如某个视此用户为客户的服务提供者）鉴权自己的时候，就可以采用分类鉴权，并且分类鉴权提供了很有用的功能。TLS 使用范围是域内的或者是多域的，由于证书是普遍可验证的，所以 UA 可以验证之前有关联的服务器。

A.5.2 S/MIME

S/MIME 最突出的缺点就是缺少对终端用户很流行的公有密钥结构。如果使用自签名的证书（或者不能被对话中的某个参与方所验证的证书），那么基于 SIP 的密钥交换机制就容易受到中间者攻击，这时，攻击者可能检查并修改 S/MIME。攻击者需要中断对话双方的第一次密钥交换，从请求及响应中去掉已有的与 CMS 分开的签名，并插入一个不同于自己的包含证书的并且与 CMS 分离的签名（但是看来是一个正确的记录地址的证书）。对话双方都认为相互已经交换了密钥，但其实它们只是有了攻击者的公有密钥。

攻击者可以在双方第一次交换密钥时只针对以上弱点进行攻击，随后，密钥的改变对 UA 是显而易

见的。以后攻击者也很难留在这个进行以后所有对话的通道里。

SSH 在初次交换密钥时很容易受到同样的中间方的攻击，但是 SSH 能很好地改善连接的安全，并可以像 SSH 那样使用指纹密码。例如，通信双方用 SIP 建立一个语音通信的会话，它们应该读出所收对方的密钥中的指纹并可以与最初的相比较。当然，同信令相比，中间方更难模仿参与者的话音（基于 CLIPPER 芯片的安全电话）。

采用 S/MIME 机制，UA 可以发送一个加密的但是无同步码的请求，如果该请求在密钥环中含有一个目标记录地址的证书。但是，任何注册为某个目标地址的设备不能持有任何曾经被该设备的当前用户所使用的证书。因此它就不能正确地处理一个加密的请求，这样就不可避免地产生错误的信令。当一个加密的请求被分叉代理的时候这种情况就最可能发生。

S/MIME 有关的密钥适用于与一个特定的用户（记录地址）而不是一个设备（UA）相关联的时候。当用户在设备间移动时，它也许很难在 UA 间安全地传输私有密钥。关于设备如何得到这种密钥不在本标准的范围之内。

S/MIME 中，很难解决的问题是它能产生很大的消息，特别是当采用 SIP 隧道机制时。因此，本标准建议在使用 S/MIME 隧道技术时，应当使用 TCP 作为传输协议。

A.5.3 TLS

TLS 需要一个有连接的下层传输协议，即 TCP 协议。TLS 不能用于 UDP 之上。

本地出代理服务器和/或注册服务器很难同时维持和很多的 UA 进行长时间的 TLS 连接。尤其对于 UA 单独负责建立连接并维护冗余的长时间生存的 TLS 连接的时候。

TLS 只允许 SIP 实体来鉴权其相邻的服务器，它提供严格的逐跳的安全。TLS 和本文所述的任何其他的机制都不可以允许客户授权它们无法建立直接的 TCP 连接的代理服务器。

A.5.4 SIPS URIs

在请求路径的任何部分使用 TLS 都必须使终结的 UAS 在 TLS 上是可达的（可以和 SIPS URI 注册一个关联地址）。例如，许多有效的网络结构采用 TLS 来保证部分请求路径的安全，而用其他的一些机制保证到 UAS 的最后一跳的安全。这样就不能保证 TLS 是真正的端到端实现的。注意到由于许多 UA 不接受引入的 TLS 连接，甚至那些支持 TLS 的 UA 可能也被要求维持永久的 TLS 连接，以便于 UAS 能在 TLS 上接收请求消息。

定位服务不要求为 SIPS Request-URI 提供 SIPS 绑定。虽然定位服务通常由用户注册服务器提供，见本标准 7.2.1。其他的各种协议和接口也是可为 AOR 提供可信的联系地址，并且这些工具也可自由地将 SIPS URI 映射成正确的 SIP URI 。定位服务在询问绑定时返回它的联系地址而不管是否收到了一个带有 SIPS Request-URI 的请求。如果重定向服务器访问位置服务，将由该实体处理重定向的 Contact 头字段以决定该联系地址是否合适。

由目标域来确保 TLS 可以被用于所有的请求部分的方法实现起来很麻烦。因此在转发路径上，以密码鉴权的不兼容的代理服务器可能不考虑与 SIPS 有关的转发规则。总的转发规则见本标准 13.6。例如，这些恶意的中介可能将一个请求从 SIPS URI 重指向到一个 SIP URI 来试图破坏安全。

另外，某中介可能将请求由 SIP 重指向到 SIPS URI。该请求的接收者的 Request-URI 采用 SIPS URI 方案，这样就不能基于 Request-URI 来保证 SIPS 被用于整个的请求路径上。

本标准建议，如果请求的 Request-URI 包括一个 SIP 或者 SIPS URI，请求接收者应检查 To 头字段值

是否包括 SIPS URI，如果该 URI 有着相同方案但并不等于 To 头字段中的 URI 不会破坏其安全。虽然客户可以选择不同的方式组装请求的 Request-URI 和 To 头字段，但当使用 SIPS 时，这种不同就被认为是一个可能的安全隐患，因而该请求就随即被接收者所拒绝。接收者也可以检查 Via 头字段链而确信是否 TLS 被用在直到本地管理域整个的请求路径上。S/MIME 也可以被用在发起 UAC 上来帮助确保端到端携带的 To 头字段的最初格式的安全。

如果 UAS 相信 Request-URI 的方案在传输中被不正当修改了，UA 就应该通知其用户潜在的安全破坏。

以后为了预防下降攻击，只接受 SIPS 请求的实体也可以拒绝在不安全端口上的连接。

终端用户可以判断 SIPS 和 SIP URI 之间的区别，并且它们可以手动编辑自己的响应，这对于安全既有利又有弊。例如，如果攻击者破坏 DNS 的缓存，为了有效去掉所有的服务器的 SIPS 记录而插入假的记录集，然后任何穿过该服务器的 SIPS 请求都会失败。然而，当一个用户看到一个到 SIPS AOR 的重复呼叫失败，他们就会在某些设备上手动将 SIPS 方案变成 SIP 方案并重试。当然，也有些对此的防护措施（如果目标 UA 很多疑，他就会拒绝所有的非 SIPS 的请求），但是其局限性使之没有什么价值。好的一面是，用户也可以知道即便当它们只带有 SIP URI 出现时，“SIPS” 也可以是有效的。

A.6 保密性

SIP 消息经常带有发送者的敏感信息，包括通话内容、通信的对方、通信时长、双方在会话中从何地加入。许多应用和他们的用户要求的个人信息对任何不需要知道的参与者都应该隐藏。

但是，很少有直接泄漏个人信息的方式。如果用户或服务选择在由姓名和组织从属关系（描述大多数的记录地址）可推测出的地址上是可达的，传统的保密性措施（即有一个未列出的“电话号码”）就被破坏了。用户定位服务可以由泄漏呼叫者详细位置来破坏会话邀请的接收方的保密性。具体实现时，应该能够基于每个用户来限制何种位置和何种可用的信息是可被某类呼叫者所知的。在某些情况下，用户可能希望在传送标识的头字段中隐藏个人信息。这不仅可以用在代表请求发起方的 From 及相关的头字段，还有 To 头字段——发送给最终目的一个快速拨号的别名或者一个目标组的未展开的标识可能不太合适，它们在路由请求时都可能从 Request-URI 中被去掉，但如果它们最初是相同的，那么在 To 中就不会改变。因此，为了保密，需要创建一个与 Request-URI 不同的 To 头字段。

附 录 B
（资料性附录）
IANA 定义

在SIP应用中使用的所有方法名字、头字段头字段名字、状态编码和选项标记都必须向IANA注册。

规范指示IANA在http：//www.iana.org/assignments/sip-parameters下创建4个子注册项目：选项标记、告警编码（warn-codes）、方法和响应编码，这些被增加到头字段的子注册项中，头字段已经在该注册项中存在了。

B.1 选项标记

本标准在http：//www.iana.org/assignments/sip-parameters下建立了选项标记子注册项。

本标准定义可以在头字段中使用选项标签来用于SIP协议的兼容和扩展，诸如Require、Supported、Proxy-Require和Unsupported。选项标签自身是一个与特定SIP选项相关的字符串（即扩展），它定义了SIP终结点的可选项。

选项标签公布时由IANA注册。IANA相关章节必须包括下列信息，这些信息和出版的RFC号码一样在IANA注册登记中出现。

1）选项标记的名字。名字可以是任意长度，但应该不超过20个字符长度。名字必须仅由字母和数字组成。

2）描述扩展内容的描述性文字。

B.2 告警编码

本标准在http：//www.iana.org/assignments/sip-parameters下建立了告警编码子注册项，并以本标准中列出的告警编码作为其初始的告警编码。另外的告警编码由RFC出版物注册。

告警编码表的描述性文字要求：

1）当会话描述协议（SDP）导致事务处理故障时，告警编码为SIP响应消息中的状态编码提供补充信息。

2）“warn-code”由3个数字组成。第一个数字“3”指出告警是针对SIP协议的。在后续的规范描述告警编码的用法不是3XX以前，可能仅有3XX的告警编码被注册。

3）300~329的告警被预留用于与指示会话描述中关键字相关的问题；告警330~339是与会话描述中所请求的基本网络业务相关的告警；告警370~379是与会话描述中所请求的QOS量化参数相关的告警；告警390~399是不属于上述告警类型的其他各种告警。

B.3 头字段名字

为了注册新的头字段名字，需要在RFC的出版物中提供下列信息：

1）报头被注册的RFC号码；

2）所注册的头字段名字；

3）该头字段的紧凑形式版本，如果定义了版本号字段。

B.4 方法和响应编码

本标准在http：//www.iana.org/assignments/sip-parameters下建立了方法和响应编码子注册项，将下列编码作为其初始的编码。初始的方法编码表是：

表B.1 方法编码表

INVITE	[RFC3261]
ACK	[RFC3261]
BYE	[RFC3261]
CANCEL	[RFC3261]
REGISTER	[RFC3261]
OPTIONS	[RFC3261]
INFO	[RFC2976]

响应编码表见本部分19章。编码的类型部分标记为Informational、Success、Redirection、Client-Error、Server-Error和Global-Failure。编码采取下列格式：

Type（例如，Informational）

Number Default Reason Phrase [RFC3261]

为了注册一个新的响应编码或方法，需要在RFC出版物中提供下列信息：

1）方法或响应编码所注册的RFC号码；

2）所注册的响应代码或方法名字；

3）某个响应编码的缺省原因短语。

B.5 “message/sip” MIME类型

为了允许SIP消息通过SIP实体中的隧道，本文档注册了“message/sip” MIME媒体类型，主要是用于实现端到端的安全。该媒体类型由下列信息定义：

Media type name：message

Media subtype name：sip

Required parameters：none

Optional parameters：version

Version：所附加的消息的SIP版本号码（如“2.0”）。如果该参数不存在，则缺省值为“2.0”。

Encoding scheme：SIP消息由8比特组成，其后可选地跟随着一个二进制的MIME数据对象。同样地，SIP消息必须按照二进制编码来处理。在正常情况下，SIP消息在支持二进制的传输媒体上传输，不需要特殊的编码。

Security considerations：作为与S/MIME一致的安全机制，该用法的目的和例子见本部分20.4节。

B.6 新的Content-Disposition参数注册

RFC3261注册了4个新的Content-Disposition报头“disposition-types”：alert、icon、session和render。这些值要求记录在IANA注册项中以用于Content-Disposition。

对这些“disposition-types”的描述，包括目的和例子，见本标准第20章。

与IANA注册内容相配的简短描述是：

alert	该实体是提醒用户的振铃音调
icon	该实体作为一个图标向用户显示
render	该实体应该向用户显示
session	该实体描述了通信会话

附 录 C
（规范性附录）
SIP 协议中临时响应的可靠性

C.1 简介

SIP 是一个基于请求响应的协议，用于发起和管理会话。在 SIP 中，响应分为两类，即临时响应和最终响应。最终响应传递呼叫请求处理的结果，并且保证可靠传送；临时响应提供呼叫请求处理过程中的信息，并且不保证可靠传送。

但某些情况下，可靠地传递临时响应非常重要，其中包括和 PSTN 互通这种情况，因此需要任选能力集来支持临时响应的可靠传递。

本标准建议，可以通过借鉴最终响应 2xx 的可靠传递机制来实现临时响应的可靠传递。事务处理用户部分（TU）将周期性地发送最终响应 2xx，直到收到 ACK 为止。2xx 沿着 INVITE 消息的发送路径逐段回送，ACK 消息则端到端传递。为了可靠传递临时响应，需要采用相同的机制。事务处理用户部分 TU 采用按照指数递增的定时器控制临时响应的重传，当收到 PRACK 消息时停止临时响应的重传。但是 PRACK 和 ACK 不同，PRACK 是一个常规消息，和 BYE 类似，PRACK 消息通过有状态的代理服务器逐跳传送，并且有对应的响应消息。

每个临时响应都有一个序列号，包含在 RSeq 头部字段中。PRACK 消息中应包含 RAck 头部字段，指示被证实的临时响应的序列号。为了防止拥塞，一次只应发送一个临时响应并且 PRACK 消息不进行累积。

C.2 UAS 的行为

只要初始的 INVITE 请求（该请求的临时响应进行可靠传递）中包含 Supported 头部字段，且该头部字段中的任选标记为 100rel，UAS 就可以可靠地传递任何非 100 临时响应。虽然本标准不允许对 INVITE 之外的其他方法可靠地传递临时响应，但是在定义用于建立对话的新方法的扩展协议中，可以采用该机制。

如果初始请求中包含 Require 头部字段，且该头部字段中的任选标记为 100rel，UAS 就必须可靠地传递任何非 100 临时响应。如果 UAS 不能满足此要求，则 UAS 必须用 420 响应来拒绝初始请求，并且在响应中包含 Unsupported 头部字段，该头部字段中的任选标记为 100rel。

UAS 不能可靠地传递 100 响应，只有编号为 101~199 的临时响应进行可靠传递。如果请求方法中未包含指示该特性的 Supported 头部字段或 Require 头部字段，则 UAS 就不需要可靠地传递临时响应。

100 响应只能进行逐跳传递，本标准中描述的可靠机制要端到端进行传递。

能够充当代理的单元也可以可靠地传递临时响应，此时该单元可以看作是 UAS。但是，如果请求方法的 To 字段中包含标记，则该单元不应进行可靠传递临时响应，即代理不应在对话上下文中发送的请求产生可靠的临时响应。和 UAS 不同，当代理单元接收到 PRACK 方法，该方法和任何未确认的可靠的临时响应都不匹配时，那么将作为代理消息继续传递。

有几种原因可能导致 UAS 发送临时响应。一种原因是 INVITE 事务处理可能需要一段时间才能产生最终响应，此时 UAS 需要定期发送临时响应，请求延长事务处理时间。本标准规定，代理能够每隔 3min 收到一个临时响应，但由于传递的过程中可能有包丢失，所以 UAS 应该以更短的周期发送临时响应（建

议发送周期为 1min)。如果 UAS 能够可靠地传递临时响应，可以每隔 2.5min 发送一次。本标准建议扩展的事务处理中使用可靠的临时响应。

假设初始请求中包含 Supported 或 Require 头部字段且字段中包含标记 100rel，此时需要可靠地传递临时响应。

需要可靠传递临时响应由 UAS 内核构建。临时响应中应包含 Require 头部字段，其中的任选标记为 100rel，并且应包含 RSeq 头部字段。在事务处理中，第一个可靠的临时响应中的 RSeq 头部字段的值应在 1 和（2^{31}–1）之间，推荐在该范围内统一进行选择。RSeq 的编号空间仅在一个事务处理内有效，即不同请求的临时响应可以使用相同的 RSeq 编号。

可靠的临时响应也可以包含消息体。

可靠的临时响应周期地传递到事务处理层，初始间隔为 T1 秒，然后每次重传的间隔都扩大两倍。一旦向事务处理传递了可靠的临时响应，UAS 内核应将该响应增加到内部未被证实的可靠的临时响应列表中。事务处理层应前传 UAS 内核重发可靠的临时响应。

和可靠的临时响应的重发不同，2xx 响应的重发间隔为 T2 秒，这是因为当收到 2xx 响应时触发 ACK 的重传，而 PRACK 的重传和 1xx 响应的接收相互独立。

当 UA 内核收到匹配的 PRACK，将停止可靠的临时响应的重传。PRACK 类似于对话中的其他请求。匹配的含义是 PRACK 和对应的响应属于同一个对话，并且 RAck 头部字段中的方法、Cseq-num 和 response-num 分别和可靠的临时响应中 Cseq 字段的方法、Cseq 字段的序列号和 RSeq 字段的序列号相同。

如果 UA 收到的 PRACK 请求和任何未被证实的可靠的临时响应都不匹配，UAS 应该发送 481 响应。如果 PRACK 和某个未被证实的可靠的临时响应相匹配，则 UAS 回应 2xx，此时 UAS 可以确定临时响应已经被按序接收，应该停止对该响应的重发，同时应从未被证实的临时响应列表中删除该响应。

如果以 64 × T1 秒的间隔重发可靠的临时响应，但当定时器超时时仍没有收到相应的 PRACK，UAS 应拒绝初始的请求并发送 5xx 响应。

如果 PRACK 包含会话描述，则按照本标准中描述的方式进行处理。如果 PRACK 包含的是其他类型的消息体，那么该消息的处理方式和 ACK 中消息体的处理方式相同。

第一个可靠的临时响应被证实之后，UAS 可以发送其他可靠的临时响应。在第一个可靠的临时响应被证实之前，UAS 不应发送第二个可靠的临时响应。第一个可靠的临时响应除外，建议在先前发送的可靠的临时响应被证实之前，UAS 不应发送其他可靠的临时响应。第一个可靠的临时响应需要特殊处理，因为该响应传递初始序列号。如果在第一个可靠的临时响应被证实之前发送其他可靠的临时响应，UAS 将不能确定这些响应是否被按序接收。

对于同一个请求，每个后续可靠的临时响应中 RSeq 的值应 > 1。RSeq 编号不应循环。RSeq 的值应 $\leq 2^{31} - 1$，所以每个请求可以有 2^{31} 个可靠的临时响应。

对于所有未被证实的可靠的临时响应，在收到相应的 PRACK 之前，UAS 可以对初始请求发送一个最终响应，除非最终响应是 2xx 且某个未被证实的可靠的临时响应中包含会话描述，此时在所有临时响应被证实之前，UAS 不能发送最终响应。如果仍有可靠的临时响应未被证实时 UAS 发送了最终响应，那么 UAS 不应继续重发未被证实的可靠的临时响应，但应该准备处理针对这些响应的 PRACK 请求。对初始请求发送了最终响应之后，UAS 不应发送新的可靠的临时响应。

C.3 UAC的行为

当 UAC 创建新的请求时，可以要求该请求的临时响应必须可靠传递，为此 UAC 可以在请求中插入任选标记为 100rel 的 Require 头部字段。包含任选标记 100rel 的 Require 头部字段不能在 INVITE 之外的请求中出现。

如果 UAC 不要求可靠的临时响应，但可以在请求中包含 Supported 头部字段，其中任选标记为 100rel，这表明如果 UAS 需要传递可靠的临时响应它也能够支持。UAC 应在所有的 INVITE 请求中包含该字段。

如果收到初始请求的临时响应，并且该响应中包含 Require 头部字段，其中的任选标记为 100rel，则应该可靠地传递该响应。如果该响应是 100（相对于 101~999），必须忽略该任选标记，并且不执行下面描述的程序。

如果还没有创建对话，那么收到临时响应时必须创建一个对话。

假设响应被可靠地传递，UAC 必须创建一个新的 PRACK 请求，该请求在和临时响应关联的对话中传递。PRACK 请求可以包含消息体，可以根据消息体的类型和处理方式进行解释。

PRACK 和对话中的其他非 INVITE 请求类似。当收到重发的临时响应但该响应已经被证实过，此时 UAC 不应重发 PRACK 请求。

收到可靠的临时响应之后，应该舍弃重发的响应。当响应的对话 ID、Cseq、RSeq 和曾经收到过的响应相关参数匹配时，那么该响应就是一个重发响应。UAC 应对序列号进行维护，序列号指示了近来依次收到的初始请求的可靠临时响应，应一直维护此序列号直到收到初始请求的最终响应。序列号的初始值等于初始请求的第一个可靠临时响应中 RSeq 头部字段的值。

除了要保证可靠的临时响应的次序之外，按照如上所述的同样规则处理同一个初始请求后续可靠的临时响应。因此，如果 UAC 收到了同一请求的另一个可靠的临时响应，但响应中 RSeq 的值没有维护的序列号的值大，那么 UAC 就不应用 PRACK 证实该响应，并且不应进一步进行处理。具体实现时可以丢弃该响应，或者如果希望收到丢失的响应则将该响应缓存。

UAC 可以对在最终响应之后收到的可靠的临时响应进行证实，也可以丢弃该响应。

C.4 Offer/Answer模式和PRACK

本部分基于对 Offer 和 Answer 出现的消息集总的描述，从而规定了需要 Offer/Answer 的其他情况。如果 INVITE 消息中包含 Offer 字段，且 UAC 支持 Offer/Answer 字段，则 UAS 可以在可靠的临时响应中产生 Answer 字段，这样会话可以在呼叫完成之前建立。如果一个可靠的临时响应是送回到 UAC 的第一个可靠的临时响应，并且在 UAC 发送的 INVITE 中没有包含 Offer 字段，则在该临时响应中应该包含 Offer 字段。

如果 UAC 收到包含 Offer 字段的可靠的临时响应，则 UAC 应在 PRACK 中产生 Answer 字段；如果 UAC 收到包含 Answer 字段的可靠的临时响应，那么 UAC 可以在 PRACK 中产生另外的 Offer 字段；如果 UAS 收到包含 Offer 字段的 PRACK，那么它必须响应 2xx。

即使 UA 还没有对初始的 INVITE 消息产生响应，但只要发送或收到 Answer 字段，UA 就应基于 Offer 和 Answer 字段中的参数建立会话，

如果 UAS 在某个可靠的临时响应中包含了会话描述，那么 UAS 必须延迟发送 2xx 直到该临时响应被证实。但是需要通过正确地进行 Offer/Answer 字段的交互来保证 1xx 消息的可靠性。

所有支持本扩展规范的用户代理必须支持所有的 Offer/Answer 交互。

C.5 PRACK方法的定义

本部分定义了新的 SIP 方法——PRACK，该方法的语义如上所述。表 C.1 和 C.2 是针对该方法对 RFC3261 中表 2 和表 3 的扩展。

表C.1 头部字段总结

头部字段	位 置	PRACK
Accept	R	O
Accept	2xx	—
Accept	415	C
Accept-Encoding	R	O
Accept-Encoding	2xx	—
Accept-Encoding	415	C
Accept-Language	R	O
Accept-Language	2xx	—
Accept-Language	415	C
Alert-Info	R	—
Alert-Info	180	—
Allow	R	O
Allow	2xx	O
Allow	r	O
Allow	405	M
Authentication-Info	2xx	O
Authorization	R	O
Call-ID	c	M
Call-Info		—
Contact	R	—
Contact	1xx	—
Contact	2xx	—
Contact	3xx	O
Contact	485	O
Content-Disposition		O
Content-Encoding		O
Content-Language		O
Content-Length		T
Content-Type		*
CSeq	c	M
Date		O
Error-Info	300~699	O
Expires		—
From	c	M

表 C.1（续）

头部字段	位　置	PRACK
In-Reply-To	R	—
Max-Forwards	R	M
Min-Expires	423	—
MIME-Version		O
Organization		—
Priority	R	—
Proxy-Authenticate	407	M
Proxy-Authenticate	401	O
Proxy-Authorization	R	O
Proxy-Require	R	O
Record-Route	R	O
Record-Route	2xx，18x	O
Reply-To		—
Require		C
Retry-After	404，413，480，486 500，503 600，603	O
Route	R	C
Server	R	O
Subject	R	—
Supported	R	O
Supported	2xx	O
Timestamp		O
To	C	M
Unsupported	420	M
User-Agent		O
Via	C	M
Warning	r	O
WWW-Authenticate	401	M

C.6　头字段的定义

本部分定义了两个新的头字段，RAck 和 RSeq。表 C.2 是针对这些头字段对 RFC3261 的扩展。

表C.2　RAck和RSeq头字段

头部字段	位置	代理	ACK	BYE	CAN	INV	OPT	REG	PRA
RAck	R		—	—	—	—	—	—	m
RSeq	1xx		—	—	—	o	—	—	—

C.6.1　RSeq

RSeq 头字段用在临时响应中用来保证响应的可靠传送。该字段只包含的数值范围从 1~（$2^{31}-1$）。

C.6.2 RAck

RAck 头部字段用在 PRAck 请求中用来保证临时响应的可靠性。该字段包含两个数值和一个方法标记。第一个数值等于要被证实的临时响应中 RSeq 头部字段的值，第二个数值和方法从要被证实的响应中 Cseq 字段复制得到。RAck 头部字段中方法名称区分大小写。

C.7 安全

攻击者可能插入 PRACK 请求，强迫可靠的临时响应的重发停止。由于这些响应可能传递重要信息，所以 PRAck 消息也应被鉴权。

C.8 BNF集

本节定义 RAck 和 RSeq 头部字段以及 PRACK 方法的 BNF。

PRACKm	=	%x50.52.41.43.4B ; PRACK in caps
Method	=	INVITEm / ACKm / OPTIONSm / BYEm / CANCELm / REGISTERm / PRACKm / extension-method
Rack	=	"RAck" HCOLON response-num LWS CSeq-num LWS Method
response-num	=	1*DIGIT
CSeq-num	=	1*DIGIT
Rseq	=	"RSeq" HCOLON response-num

附 录 D
（规范性附录）
SIP 的定位过程

SIP 协议通过 DNS 程序，客户端可以将 SIP URI 解析成下一跳的 IP 地址、端口号和传输协议。同时利用 DNS 程序，当主客户端故障时允许服务器向备用的客户端发送响应。

D.1 简介

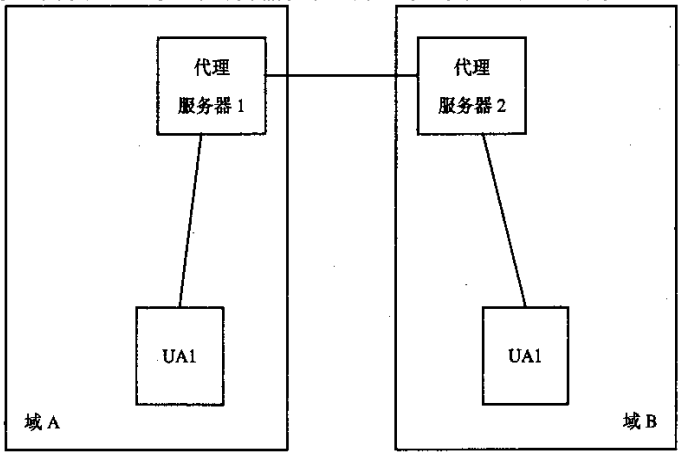
SIP 协议是一个基于客户机服务器协议，用于发起和管理用户间的会话。SIP 终端系统称为用户代理，中间实体称为代理服务器。典型的 SIP 配置如图 D.1 所示。在图 D.1 中，域 A 中的主叫用户（UA1）想呼叫域 B 中的用户 UA2。首先 UA1 需要和所属域（域 A）中的代理服务器 1 通信，代理服务器 1 向被叫用户所属域（域 B）内的代理服务器——代理服务器 2 前传呼叫请求，然后代理服务器 2 向被叫用户——UA2 前传呼叫。

作为呼叫流程的一部分，代理服务器 1 需要为域 B 确定 SIP 服务器。为此代理服务器 1 使用 DNS 程序，同时使用 SRV 和 NAPTR 记录。

D.2 DNS需要解决的问题

利用 DNS 系统帮助解决上述整个呼叫流程存在两个问题。一是代理服务器 1 如何找到域 B 中的 SIP 服务器，以便把呼叫前传给 UA2；二是当代理服务器 1 在前传呼叫请求之后发生故障时，代理服务器 2 如何识别代理服务器 1 的备份服务器。

首先，代理服务器 1 需要明确确定域 B 中 SIP 服务器的 IP 地址、端口号和传输协议，尤其是传输协议的选择。SIP 协议可以承载在各种传输协议之上，包括 TCP、UDP 和 SCTP，因此客户端应能自动确定哪一种传输协议可用。发送请求的代理服务器有一个它所支持的特定的传输协议集，并且在传输协议集中有一个优选协议。代理服务器 2 有它自己支持的传输协议集和相应的优选传输协议。所有的代理服务器必须同时支持 UDP 和 TCP，因此总是存在能力的交集。代理服务器 1 可能通过某种类型的 DNS 程序来寻找域 B 中 SIP 业务可用的传输协议，以及这些传输协议中优选的传输协议。代理服务器 1 找到它所支持的传输协议列表和代理服务器 2 所支持的传输协议列表的交集，从中选出代理服务器 2 优选的协议。



图D.1 SIP协议的典型配置

本标准规定，在处理呼叫的过程中可以多次执行 DNS 查询。通常，要发送请求的单元（称为客户端）可能需要执行 DNS 程序以便确定下一跳单元的 IP 地址、端口号和传输协议。在两个单元之间的任何一跳上都可能执行 DNS 查询程序。

SIP 协议基于事务交互式通信，主被叫用户之间事务处理的完成时间非常重要。通常从主叫用户发起呼叫到被叫用户被提醒之间的时间应不大于几秒。假定呼叫过程中要经过多跳，并且假设除了可能的近似于实时的操作之外每一跳都进行执行 DNS 查询，这样对于每一跳 DNS 查询的时间将受到限制。

SIP 协议中可扩展性和可用性也很重要。在实现 SIP 业务的过程中可能采用多种技术。如前述示例中，代理服务器 2 可能是一组同样配置的代理服务器，此时需要 DNS 提供为域 B 配置一组服务器的能力，包括优先级和权重，以便为基于能力的负荷分担提供一个粗略的等级区分。

SIP 利用上行网络实体检测故障的机制来确保其可用性。当代理服务器从某个特定的域收到第一个呼叫时，代理服务器应该依次联系该域内的服务器，直到找到一个可用的服务器为止。理想情况下这个可用服务器的标识应该缓存一段时间，以便减少后续呼叫建立的时延。客户端不应连续地查询故障服务器来确定该服务器是否又重新可用。而且，最终要刷新可用性状态，以便向已经恢复的单元重新分配负荷。

网络实体在事务处理过程中可能发生故障。例如，代理服务器 2 向 UA 2 前传了请求之后，代理服务器 1 故障，UA 2 向代理服务器 2 发送响应，代理服务器 2 试图向代理服务器 1 前传，而此时代理服务器 1 已经不再可用。此时，就需要 DNS 协助解决 D.1 中描述的呼叫流程中的第二个问题，代理服务器 2 要标识代理服务器 1 的备份服务器，以便向备份的服务器发送该响应。这个问题在 SIP 协议比较突出，因为有些 SIP 响应需要很长的时间才能产生。因此，在呼叫请求和呼叫应答之间常需要几十秒的时间。

D.3 客户端DNS用法

假设客户端是有状态的服务器。当客户端需要向 SIP URI 或 SIPS URI 标识的资源发送请求时，将调用本节中的程序。URI 可以标识请求要发往的 URI，或者标识这中间的某一跳。此时 DNS 查询的结果不会改写此 URI，只是产生请求可发往的 IP 地址、端口号和传输协议。在有些情况下，请求可以发送到不是用 SIP URI 标识，而是用主机名或数字型 IP 地址标识的特定的中间代理服务器，这时就会使用一个临时 URI。该 URI 的形式为 sip：<代理服务器>，此处的代理服务器可以是下一跳代理服务器的 FQDN 或数字型 IP 地址。该过程即在 DNS 中解析 SIP URI 或 SIPS URI，以便确定请求要发往的主机的 IP 地址、端口号和传输协议。

每个事务处理必须正确地执行一次该程序。也就是说，一旦和 SIP 服务器成功建立了连接，所有重传的 SIP 请求和 ACK 消息（即为证实响应 INVITE 消息而发送的 non-2xx 响应）都必须发往同一个主机。而且对于特定 SIP 请求的 CANCEL 消息必须发送到 SIP 请求发送到的同一个 SIP 服务器。

如果在 URI 中包含 maddr 参数，那么将目标作为该参数的值，否则将目标作为 URI 中主机端口部分的主机值。目标标识了要建立连接的域。

D.3.1 选择传输协议

首先客户端要选择传输协议。

如果 URI 在 transport 参数中指定了传输协议，则应使用该传输协议。

否则，如果没有指定传输协议，但目标是一个数字型 IP 地址，那么对于 SIP URI 客户端应该采用 UDP，对于 SIPS URI 客户端应该采用 TCP。同样，如果没有指定传输协议，并且目标不是数字型的 IP 地址，

但是提供了具体的端口号，那么对于 SIP URI 客户端应该采用 UDP，对于 SIPS URI 客户端应该采用 TCP。

否则，如果没有指定传输协议或端口号，并且目标不是数字型 IP 地址，那么客户端应该对 URI 中的域名执行 NAPTR 查询。和传输协议选择相关的业务是包含 NAPTR 业务字段且该字段的值为“SIP+D2X”和“SIPS+D2X”的业务，字母 X 对应该域所支持的传输协议。本部分为 UDP 定义了 D2U，为 TCP 定义了 D2T。在 NAPTR 业务字段中 NAPTR 记录提供域名到 SRV 记录的映射，以便使用特定的传输协议和服务器建立连接。资源记录将包含一个空的规则表达式和一个替换值，该实体记录是针对特定传输协议的 SRV 记录。如果服务器支持多个传输协议，将有多个 NAPTR 记录，每个有不同的业务值。客户端丢弃业务字段不可用的记录。规则如下：

首先，解析 SIPS URI 的客户端应丢弃业务字段中协议不为 SIPS 的所有业务，相反却不正确。解析 SIP URI 的客户端应保留记录。其次，客户端应丢弃业务字段（标识解析的业务）的值不为"D2X"的所有业务，X 的值指示客户端所支持的传输协议。和 SRV 记录相同，针对服务器，处理 NAPTR 能得出客户端所支持的、并且服务器最优选的是传输协议。

例如，客户端要解析 sip：user@example.com。针对域名客户端执行 NAPTR 查询，将返回表 D.1 的 NAPTR 记录。

表D.1 NAPTR记录

	次序	优先级	标志	业务	规则表达式	替换值
IN NAPTR	50	50	"s"	"SIPS+D2T"	""	_sips._tcp.example.com.
IN NAPTR	90	50	"s"	"SIP+D2T"	""	_sip._tcp.example.com
IN NAPTR	100	50	"s"	"SIP+D2U"	""	_sip._udp.example.com.

既然客户端支持 TCP 和 UDP，那么可以使用 TCP 和通过 SRV 查询“_sip._tcp.example.com”确定的主机建立联系。

表D.2 SRV记录

	优先级	权重	端口号	目标
IN SRV	0	1	5060	server1.example.com
IN SRV	0	2	5060	server2.example.com

如果通过查询 NAPTR 记录可以和 SIP 代理服务器、重定向服务器或注册服务器建立连接，那么至少有三个记录：一个记录包含“SIP+D2T”业务字段，一个包含“SIP+D2U”业务字段，一个包含“SIPS+D2T”业务字段。业务字段中协议为 SIPS 的记录应比业务字段中协议为 SIP 的记录优先级高（即次序字段的值较小）。业务字段为“SIPS+D2U”的记录不应出现在 DNS 中。

在 NAPTR 替换值字段中域名的后缀没必要和初始请求中的域名（即 example.com）相匹配。然而，域必须为初始请求中的域名维护 SRV 记录，即使 NAPTR 记录在另一个域中。例如，即使对 TCP 已经有 SRV 记录“ _sip._tcp.school.edu”，还必须同时有 SRV 记录“_sip._tcp.example.com”。

如果服务器使用站点证明，则基于 TLS 交互中服务器提供的站点证明、包含 SIPS 协议字段的 NAPTR 记录、NAPTR 查询中的域名和 NAPTR 记录中替换值字段中的域名必须有效。基于相同的站点证明，SRV 查询中的域名和 SRV 记录中目标字段中的域名必须有效。另外，攻击者可能修改 DNS 记录使替换值为一个不同的域名，但是客户端不能确认该操作是一个合理的行为还是攻击的结果。

如果没有返回 NAPTR 记录，客户端将对它所支持的传输协议创建 SRV 查询，并且执行每个查询。对 SIP URI 使用业务标识“_sip”，对 SIPS URI 使用业务标识“_sips”执行 SRV 查询。如果查询成功，

那么将支持特定的传输协议。客户端可以使用任何它想使用的并且服务器支持的传输协议。

如果没有返回 SRV 记录，对于 SIPS URI，客户端应使用 TCP，对于 SIP URI 应使用 UDP。

D.3.2 确定端口号和 IP 地址

确定了传输协议之后就要确定 IP 地址和端口号。

如果目标字段是数字型 IP 地址，客户端将使用该地址。如果 URI 中包含端口号，客户端也将使用该端口号。如果没有指定端口号，将使用传输协议对应的默认端口号。

如果目标字段不是数字型 IP 地址，但是在 URI 中包含端口号，那么客户端将执行域名的记录查询。查询结果为一列 IP 地址，每个 IP 地址可以使用 URI 中的端口号和之前确定的传输协议进行连接。客户端应尝试第一个记录。如果尝试失败，将尝试下一个记录，依此类推。

如果目标不是数字型 IP 地址，并且在 URI 中未包含端口号，那么如果执行了 NAPTR 处理，客户端将根据 NAPTR 处理返回的记录执行 SRV 查询。如果没有执行 NAPTR 处理，因为已经指定传输协议，客户端可以针对该传输协议执行 SRV 查询，并且对于 SIPS URI 使用业务标识 "_sips"，对于 SIP URI 使用业务标识 "_sip"。如果因为没有返回 NAPTR 记录而未执行 NAPTR 处理，但是针对所支持的传输协议 SRV 查询成功，那么将选择返回的 SRV 记录。

如果没有返回 SRV 记录，那么客户端将执行域名查询。查询结果将为一列 IP 地址，每个地址可以使用之前确定的传输协议和所对应的默认端口号进行连接。一旦查询到记录，将对具体的端口号执行上述程序。

D.3.3 RFC 2782 中的处理方法

RFC 2782 描述了一组 SRV 记录如何分类和尝试。然而，它仅仅说明了客户端应"尝试建立连接到（协议、地址、业务）"，而没有详细描述故障时发生的情况。此处将描述这些细节问题。

对于 SIP 请求，如果事务处理层发送 503 或是某一类的传输失败响应。如果事务处理层定时器（即定时器 B 或定时器 F 超时）超时时没有收到过任何响应，包括临时响应或最终响应，那么此时也发生了故障。如果故障发生，客户端应该创建一个新的请求，该请求和先前的请求相同，除了 Via 字段中的 branch ID 和先前的请求具有不同的值，以便创建新的事务 ID。该请求发往 RFC 2782 中所指定列表中的下一个单元。

D.3.4 无状态代理服务器

前面章节中描述的处理程序都是针对有状态代理服务器的。当一个服务器被成功连接，对于该事务处理所有重发请求、证实非最终响应 2xx 的 ACK 以及 CANCEL 请求，都必须发送到相同的服务器。

事务处理状态中包含成功连接的服务器的标识。但是无状态代理服务器也需要在事务处理中将所有请求都发送到同一个服务器。

cookie 会话中的 HTTP 事务处理基于 DNS 随机选路到不同服务器。通常工作的服务器共享保存会话数据的后台数据存储区。当工作的服务器收到 HTTP 请求，且请求中包含会话标识，服务器将根据会话标识获取处理请求所需的状态。收到未包含会话标识的请求时将创建一个新的状态。但是，经过无状态代理服务器传递的请求在事务处理中被看作重传的请求。既然这些重传的请求中都不含会话标识（即完整的对话标识），那么应在每个服务器中创建一个完全相同的新的对话。这可能导致到同一个电话创建了多个呼叫。

如果与服务器建立连接时没有发生故障，则比较容易防止这种情况的发生。当无状态代理服务器收

到请求时，它将执行上述DNS查询。对于具有相同优先权的记录，无状态代理服务器必须使用配置的算法确定记录的次序。一种建议是将这些记录按照字母顺序排列，另一种更常用的方式是按照标准的ASCII编码对记录进行排列。要使无状态代理服务器的处理简单，建议域名管理者使具有相同优先级的SRV记录具有不同的权重（例如：如果两个服务器的优先级相同，分配不同权重1000和1001比分配相同权重1000好），对NAPTR记录也类似处理。如果成功地和第一个服务器建立连接，代理服务器将仍作为无状态服务器。但是如果没有成功地和第一个服务器建立连接，而是和后续的服务器建立了连接，那么对于该事务处理代理服务器不能继续作为无状态服务器。如果服务器是无状态服务器，那么如果在重传过程中故障服务器恢复将很有可能使重传请求发送到不同的服务器。因此，当代理服务器不能和第一个服务器成功建立连接时，该服务器应作为有状态的代理服务器。

但是无状态代理服务器仍然有可能向另外的服务器转发重传请求，即使该无状态服务器遵循上述描述。在事务处理过程中如果DNS TTL超时并且列项已经改变，这种情况不可避免。网络设计者应该意识到这种限制，如果这种错误比较严重应不使用无状态代理服务器接入DNS。

D.4 服务器用法

通常，对于单播UDP请求，响应应该回送到发送请求的源IP地址，使用Via头部中的端口号。对于可靠的传输协议，应在收到请求的连接上发送响应。但是当在发送请求和接收响应的过程中客户单元故障时，对故障克服机制的支持就非常重要。

对于可靠的传输协议，服务器将在收到请求的连接上发送响应，对于不可靠的传输协议，响应将发送到请求中的源地址和Via头部字段中的端口号。当服务器尝试向相应的地址发送响应但发生故障时调用此处描述的程序。“故障”指的是在发送响应之前，收到请求的传输连接关闭或者传输层指示通信中发生重大错误。

在这些情况下，服务器检查最顶部Via头部字段中Sent-by参数的值。如果该值是数字型IP地址，那么服务器尝试向该地址发送响应，并且使用Via头部字段中的传输协议；如果Sent-by参数中包含端口号则使用该端口号，否则使用该传输协议对应的缺省端口号。

如果Sent-by参数字段包含域名和端口号，那么服务器将使用该域名查询记录。服务器尝试向查询结果IP地址列表中的每个地址发送响应，使用Via字段中的端口号和传输协议。和客户端的处理相同，如果失败则尝试列表中的下一个IP地址。

如果sent-by字段包含域名但不包含端口号，那么服务器将使用业务标识“_sip”查询该域名的SRV记录。对结果列表按照RFC 3262中的描述进行分类，并且将响应发送到位于新列表中最顶部的单元。如果失败，则尝试下一个列项。

D.5 构建SIP URI

在许多情况下，单元需要构建SIP URI，以便包含在REGISTER的Contact头部字段中，或者包含在INVITE的Record-Route头部字段中。这些URI必须解析成插入它们的单元。但是，如果这些URI只用IP地址构成，例如：

sip：1.2.3.4

那么如果单元故障，则没有办法将请求或者响应选路到备用单元。

SRV提供了一种解决这个问题的办法。代替使用IP地址，解析成SRV记录的域名可以使用：

sip：server23.provider.com

对于特定的目标可以构建 SRV 记录，使其中一个记录的优先级较小，并且该记录指向构建该 URI 的特定单元。同时也应该有其他记录，这些记录的优先级较大，并且用来指向故障时的备用单元。

D.6 安全

攻击者可以通过缓存来分流消息。经常和相同的域建立连接的客户端应缓存 NAPTR 记录，而无论该 NAPTR 记录的业务字段是否包含 SIPS。如果出现这样的记录，而在以后的查询中不再出现，那么这可能是隐性攻击。在这种情况下，客户端应产生提示或告警信息并且可以拒绝请求。

另外一种情况是处于系统端用户之间的代理服务器经常要发起 NAPTR 查询，因此代理服务器有可能忽略业务字段为 SIPS 的记录，即使有这些记录，这将降低安全性。很难采取措施来防止这样的攻击。客户端简单地依赖代理服务器完成呼叫，就必须相信代理服务器能够正确地实现提供一定安全性的协议。通过篡改业务（在缺少 DNSSEC 的情况下）可以伪造 DNS 记录，妥协和强迫代理服务器请求中断。

D.7 传输协议应用

本节描述 NAPTR 的用法，使用动态删除发现系统（DDDS）框架。DDDS 代表 NAPTR 资源记录的演进。DDDS 定义应用，该应用针对特定的解析业务使用 NAPTR 记录。该应用称为传输协议确定应用，目的是将输入的 SIP URI 或 SIPS URI 映射到一组 SRV 记录，这些记录对应处理该 URI 的各种服务器。

下面是 DDDS 请求应用提供的信息：

应用惟一字符串：应用惟一字符串（AUS）是解析业务的输入。此处是要解析 URI。

规则一：从 AUS 中提取的关键字。此处是 SIP URI 或 SIPS URI 的主机部分。

有效的数据库：在单一的数据库中查寻规则一中的关键字。此处有效的数据库是 DNS。

期望的输出：应用的结果是对应要连接的服务器的 SRV 记录。

附 录 E
（规范性附录）
SDP 的提供/应答（Offer/Answer）模式

利用该机制两个实体可以采用 SDP 对它们之间的多媒体会话达成一致。在该模式中，一个参加者向另一个参加者提供它期望建立的会话的描述，另一个参加者应答它期望建立的对话。提供/应答模式在单播会话中最有用，此时需要两个参加者的信息来完成会话的协商。SIP 协议可以采用提供/应答模式。

E.1 简介

SDP 最早用作描述多媒体网络中的组播会话。SAP 被设计成一种组播机制来传送 SDP 消息。虽然 SDP 规范中允许用于单播操作，但是该规范不完善。组播会话需要所有参加者对所使用的对话进行整体协商，但是单播会话不同，单播会话只包含两个参加者，会话的协商仅需要两个参加者的信息，且仅需要两个参加者对各种参数达成一致。

例如，组播会话需要对特定的媒体流通告一个组播地址，而对于单播会话，需要两个地址——每个参加者一个；又例如，组播会话要指出会话中采用的编解码，而对于单播会话，需要确定一组两个参加者都支持的编解码。

因此，虽然 SDP 能够描述单播对话，但是它缺少具体的语义和操作描述。本文通过基于 SDP 定义的简单提供/应答模式对此进行补充。在该模式中会话的一个参加者产生 SDP 消息，该 SDP 消息构成提供——提供者希望使用的一组媒体流和编解码，以及提供者想用来接收媒体的 IP 地址和端口号。提供传送给另一个接收者，称为应答者。应答者产生应答，应答是 SDP 消息用来响应提供者的提供。应答中包含与提供中相同的媒体流，指示该媒体流是否被接受，以及所使用的编解码和应答者希望用来接收媒体所使用的 IP 地址和端口号。

组播会话也有可能像单播会话一样工作。两个用户之间协商会话参数，类似于单播会话，但是和单播会话不同，两个用户都向同一个组播地址发送分组。本文也讨论了提供/应答模式对组播媒体流的应用。同时本文也描述了在初始提供/应答交互之后如何采用提供/应答模式来更新会话。

本文不规定提供和应答传递的方式。这里定义的提供/应答模式是 SIP 必须采用的基本机制。

E.2 定义

代理：代理是在提供/应答交互中涉及的协议实现，共涉及两种代理。

应答：应答者发送的 SDP 消息，用来相应从提供者收到的提供。

应答者：一个代理，从其他代理接收描述期望进行的媒体通信的会话描述，然后用自己的会话描述进行响应。

媒体流：单个媒体实例，例如：音频流或视频流，也可以是单个白板或共享应用组。在 SDP 中，媒体流用“m=”行和相关的属性描述。

提供：提供者发送的 SDP 消息。

提供者：一个代理，产生会话描述用来创建或修改会话。

E.3 协议操作

提供/应答交互假设存在高层协议（例如 SIP），为了会话建立该协议能够在代理之间交换 SDP 信息。

当一个代理向另一个代理发送初始提供时协议操作开始。如果提供位于高层协议建立的所有上下文之外，那么该提供即为初始提供。假设高程协议提供对某类上下文的维护，该上下文允许同时进行各种相关 SDP 消息的交互。

收到提供的代理可以产生应答，也可以拒绝该提供。拒绝提供的方式依赖于高层协议。提供/应答的交互是自动的；如果应答被拒绝，会话恢复到提供之前的状态（可能没有对话）。

任何时候，任何代理都可以产生新的提供来更新对话。但是，如果该代理已经收到一个提供但还没有进行应答或还没有拒绝，那么该代理就不能产生新的提供。另外如果它已经产生了一个提供但还没有收到应答或拒绝消息，那么它也不能产生新的提供。如果代理在发送提供之后又收到一个提供，但是对发送的提供还没有收到响应，这时将被认为发生了“同抢”。

同抢最初用在电路交换网中，用来描述两个交换机同时想占用同一条中继上的同一个空闲电路。这里，同抢指的是两个代理想同时发送更新的提供。

高层协议需要提供一种解决这种情况的方式。高层协议需要在每个方向上对消息进行排列。SIP 协议能够满足这种要求。

E.4 产生初始提供

提供和应答应是有效的 SDP 消息，如 RFC 2327 中的定义，但是有一个例外。RFC2327 要求在 SDP 消息中必须同时包含 e 和 p 行。本部分放松了该限制；提供/应答应用描述的 SDP 可以同时省略 e 和 p 行。o 行中会话 ID 的值和版本号的值应能够用 64 比特符号整数表示。版本号的初始值应小于 2^{62}-1 以避免环回。虽然 SDP 规范允许将多个会话一起放入一个大的 SDP 消息中，但是提供/应答模式中使用的 SDP 消息应只包含一个会话描述。

SDP 中的“s=”行传送会话主题，主要用于组播而非单播。对于单播对话，建议该行采用一个空格（0x20）或破折号（——）表示。

但是，SDP 不允许“s=”行为空。

SDP 的“t=”行传送会话时间。通常，单播会话的媒体流使用外部信令的方式建立和终止，如 SIP。此时，“t=”行应填充之“0 0”。

提供将包含零个或多个媒体流（每个媒体流用“m=”行和相关的属性描述）。零个媒体流隐含地指出提供者希望进行通信，但是会话的媒体流将在之后通过修改提供的方式增加。媒体流可以同时用于单播和多播；在用于多播的情况下，在相关的“c=”行中指示多播地址。

每个提供的媒体流的创建依赖于该媒体流是单播还是组播。

E.4.1 单播媒体流

如果提供者希望在一个媒体流上仅向它的对等发送媒体，那么它应通过“a=sendonly”属性来标记该流为只发不收。如果在媒体流属性或会话属性中有方向属性，那么该流就被标记了一定的方向。如果提供者希望从它的对等只接收媒体，它应标记该流为只收不发。如果提供者希望进行通信，但是此时既不希望发送也不希望接收媒体，那么它应用属性“a=inactive”标记该流。此时该流未激活，在 RFC3108 中定义了未激活的方向属性。注意，对于实时传输协议（RTP），对于只发、只收和未激活的流仍然要发送

和接收 RTCP。也就是说，媒体流的方向性不影响 RTCP 的使用。如果提供者希望和它的对等之间同时进行媒体的接收和发送，那么它可以包含属性“a=sendrecv”，也可以省略该属性，因为收发是默认的。

对于只收和收发媒体流，提供中的端口号和地址指示提供者希望从哪里接收媒体流。对于只发 RTP 流，地址和端口号隐含地指示提供者希望从哪里接收 RTCP 报告。除非明确指明，否则 RTCP 报告将发送到比指定值大一的端口号。提供中的 IP 地址和端口号没有对提供者发送 RTP 分组和 RTCP 分组的源 IP 地址和源端口号进行指定。提供中的端口号为 0 指示提供媒体流但不能使用该媒体流。这在初始提供中没有太大意义，但是为了完成，应答可以包含 0 端口号来指示拒绝媒体流。而且已经存在的媒体流可以通过将端口号设置为 0 来终止。通常，0 端口号指示不期望该媒体流。

每个媒体流的媒体格式列表传达了两个信息，即提供者能够发送和/或接收（依赖于方向属性）媒体的格式集（在使用 RTP 的情况下，主要是编解码以及和编解码相关参数），以及在使用 RTP 的情况下，用于标识这些格式的 RTP 载荷的类型值。如果列出了多个格式，那么意味着提供者能够在会话过程中使用其中的任何一个。也就是说，在会话过程中不用发送新的提供，应答可以使用列出的这些格式中的任何一个改变媒体格式。对于只发媒体流，提供指示对于该媒体流提供者期望发送的媒体格式。对于只收媒体流，提供指示对于该媒体流提供者期望接收的媒体格式。对于收发媒体流，提供指示提供者期望发送和接收的媒体的编解码。

对于只收 RTP 媒体流，载荷类型值指示对于该编解码提供者期望接收的 RTP 分组中载荷类型字段的值。对于只发 RTP 媒体流，载荷类型值指示对于该编解码提供者期望发送的 RTP 分组中载荷类型字段的值。对于收发 RTP 媒体流，载荷类型值指示提供者期望接收和发送的 RTP 分组中载荷类型字段的值。但是，对于只发和收发媒体流，对于相同的编解码，应答可能指示不同的载荷类型值，在这种情况下，提供者必须使用应答中的载荷类型值发送 RTP 分组。

为了和 H.323 互通，在每个方向上可能需要不同的载荷类型值。

可以包含 fmtp 参数来提供媒体格式的其他参数。

在使用 RTP 媒体流的情况下，所有的媒体描述应包含“a=rtpmap”属性，指示从 RTP 载荷类型到编码的映射。如果没有“a=rtpmap”，将使用默认的载荷类型映射。

在所有情况下，“m=”行中格式必须按照优先级排列，第一个格式是优选格式。优选意味着应使用能够接受的具有最高优先级的格式接收提供。

如果对于媒体流包含属性 ptime，那么它指示提供者希望接收期望的分组的间隔。Ptime 属性的值必须大于 0。

如果对于媒体流包含属性 bandwidth，那么它指示提供者期望使用的接收带宽。该属性的值可以为零，但不建议。0 值指示不应发送任何媒体。在使用 RTP 的情况下，也不允许发送 RTCP 分组。

如果包含多个不同类型的媒体流，那么它意味着提供者期望同时使用这些媒体流。一个典型的情况就是在视频会议中同时使用音频和视频媒体流。

如果在提供中包含多个相同类型的媒体流，那么它意味着提供者期望同时发送（或/和接收）多个该类型的媒体流。当发送多个同一类型的媒体流时，如何将该类型的每个媒体源（例如：视频画面和视频中的 VCR）映射到每个媒体流上取决于本地策略。当对于一个特定的媒体类型，用户端只有一个源时，只使用一个策略是合理的：源被发送到相同类型的每一个流上。每个流可以使用不同的编码。但接收多个同一类型的媒体流时，如何将每个媒体流映射到该类型的各种媒体接收器（例如：音频中的扬声器或

录音装置）上取决于本地策略。但是对策略还是有些限制。首先，当接收多个相同类型的媒体流时，每个流必须至少映射到一个接收器，以便显示给用户。也就是说，接收多个相同类型的媒体流意味着这些媒体应该同时显示给用户而不是只选择其中的一个显示。另外一个限制是，当收到多个媒体流并向相同的接收器发送时，这些媒体流必须以特定的媒体方式相组合。例如：在有两个音频流的情况下，从任何一个媒体流接收的媒体都可能映射到扬声器。在这种情况下，组合操作将混合这两个媒体流。在有多个实时消息流的情况下，当接收器是显示屏幕时，组合操作应向用户显示所有媒体流。第三个限制是，如果将多个源映射到相同的流，在向流发送这些源之前必须以特定媒体方式组合这些源。虽然超越这些限制的策略具有很大的灵活性，但是代理通常不希望一个策略是从接收器向源拷贝媒体（即不应将从一个流上收到的媒体拷贝到另一个媒体流上），除非该代理是会议服务器。

使用多个相同类型媒体流的典型应用实例是预付费卡类呼叫应用，此时用户可能在呼叫过程中随时按下“#”键来挂断电话并且使用该卡发送一个新的呼叫。这要求来自用户的媒体到达两个目的地——远端网关和监视该按键的 DTMF 处理应用。这可以通过使用两个媒体流来完成，一个流到网关采用收发方式，另一个流到 DTMF 应用采用只发方式。

一旦提供者发送了提供，它就必须准备在提供描述的任何只收方式的流上接收媒体，准备在提供描述的任何收发方式的流上接收媒体，并且准备在提供描述的任何只发方式的流上发送媒体（当然，实际上还不能发送媒体，必须要等到收到对等发送的包含需要的地址的端口号信息的应答）。在使用 RTP 的情况下，即使在收到应答之前收到媒体，提供者也不能发送 RTCP 接收者报告，而必须要等到收到应答。

E.4.2 组播媒体流

如果会话描述包含组播媒体流，并且该媒体流为只收（或只发）方式，那么这意味着参加者，包括提供者和应答者，只能在该流上接收（或发送）媒体。这点和单播不同，单播中方向性指的是在提供者和应答者之媒体的流向。

除了这点之外，提供的组播媒体流的语义和 RFC 2327 中的描述相同。

E.5 产生应答

对提供的会话描述的应答基于提供的会话描述。如果应答和提供有任何不同（不同的 IP 地址、端口号等等），那么应答中的首行必须不同，因为应答是由不同的实体产生的。在这种情况下，应答中“o=”行上的版本号和提供中“o=”行上的版本号不相关。

对于提供中的每个“m=”行，在应答中必须有相应的“m=”行。应答应包含和提供中相同数目的“m=”行。这使得可以基于“m=”行的顺序来匹配媒体流。这意味着如果提供包含零个“m=”行，那么应答也必须包含 0 个“m=”行。

应答中的“t=”行应等于提供中的“t=”行。会话的时间不能协商。

出于某些原因，在应答中可以拒绝提供的媒体流。如果媒体流被拒绝，那么提供者和应答者不应对该媒体流产生媒体（或者 RTCP 分组）。要拒绝提供的媒体流，应答中相应媒体流的端口号必须置为 0。所有列出的媒体格式被忽略。至少应该提供一个媒体流，并满足 SDP 的规定。

对每个提供的媒体流产生的应答对于单播和组播应不同。

E.5.1 单播媒体流

如果媒体流被提供的是单播地址，那么对于该媒体流的应答应包含单播地址。在应答中的媒体流的

媒体类型必须和提供中的媒体流的媒体类型相匹配。

如果提供中媒体流为只发，那么在应答中对应的媒体流必须被标记为只收或未激活。如果提供中媒体流为只收，那么在应答中对应的媒体流必须被标记为只发或未激活。如果提供中媒体为收发（如果在媒体或会话描述中没有方向属性，那么在这种情况下媒体流默认为收发），那么在应答中对应的媒体流可以标记为只发、只收、收发或未激活。如果提供中媒体流为未激活，那么在应答中该媒体流必须标记为未激活。

对于应答中被标记为只收的媒体流，“m=”行应至少包含一种媒体格式，指明应答者期望接收的媒体格式，该媒体格式包含在提供中列出的媒体格式中。媒体流可以指示其他的媒体格式，该媒体格式在提供的对应媒体流中未列出，但应答者期望接收该媒体格式。对于应答中被标记为只发的媒体流，“m=”行应至少包含一种媒体格式，指明应答着期望发送的媒体格式，该媒体格式包含在提供中列出的媒体格式中。对于应答中标记为收发的媒体流，“m=”行应至少包含一个编解码，指明应答着期望发送和接收的编解码，该编解码包含在提供中列出的编解码中。媒体流也可以指示其他的媒体格式，该媒体格式在提供的对应媒体流中未列出，但应答者期望发送或接收该媒体格式（当然，此时还不能发送该媒体格式，因为该媒体格式没有列在提供中）。对于应答中标记为未激活的媒体流，根据提供形成媒体格式列表。如果提供中媒体流为只发，那么将应答中的媒体流作为只收来形成媒体格式列表。同样，如果提供中媒体流为只收，那么将应答中的媒体流作为只发来形成媒体格式列表；如果提供中媒体流为收发，那么将应答中的媒体流作为收发来形成媒体格式列表。如果提供中媒体流为未激活，那么以提供中的媒体流为收发应答中的媒体流也为收发来形成列表。

应答中的连接地址和端口号指示应答者期望从哪里接收媒体（在使用 RTP 的情况下，将在比该端口号大一的端口上接收 RTCP 分组，除非有其他明确的指示）。即使对于只发的媒体流也必须提供地址和端口号；在使用 RTP 的情况下，仍然使用较大一个的端口号来接收 RTCP 分组。

在使用 RTP 的情况下，如果提供中对于特定的载荷类型值优选某个特定的编解码，那么对于应答中对应的编解码也应该使用相同的载荷类型值。即使使用相同的载荷类型值，应答也必须包含 rtpmap 属性，定义载荷类型到动态载荷类型的映射，同时也应该包含到静态载荷类型的映射。“m=”行中的媒体格式应按照优先级的顺序列出，其中第一个媒体格式是优选的。在这种情况下，优选指的是提供者应该使用应答中具有最高优先级的媒体格式。

即使应答者可以按照期望的优选顺序来列出媒体格式，但是建议除非有特殊的原因，应答者应该以提供中这些媒体格式的排列顺序来列出这些媒体格式。也就是说，如果提供中媒体流列出的音频编解码为 8、22 和 48，而应答者只支持编解码 8 和 48，并且如果应答者没有理由改变所支持的编解码，那么建议应答中编解码的顺序应该是 8、48，而不是 48、8。这样将有助于在双向上使用相同的编解码。

提供中 fmtp 参数的解释依赖于该参数本身。在许多情况下，这些参数描述媒体格式的特殊配置，因此对这些参数应该像处理媒体格式一样处理。也就是说，如果应答中包含 fmtp 参数所描述的媒体格式，那么在应答中对于相同的媒体格式必须有相同的 fmtp 参数。如果 fmtp 参数描述其他内容，这是对于 fmtp 参数每个代理可以使用不同的值。在这种情况下，应答可以包含 fmtp 参数，并且参数的值可以和提供中这些参数的值相同，也可以不同。应答/提供中定义新参数的 SDP 扩展同时应该规定如何正确地解释这些参数。

对于任何媒体流，应答可以包含取值非 0 的 ptime 属性，该属性指示应答者希望接收分组的间隔。对

于特定的媒体流，不要求双向上分组间隔相同。

对于任何媒体流，应答者可以包含 bandwidth 属性，该属性指示应答者希望提供者用来发送媒体的带宽。该属性的值允许为 0。

如果对于提供中特定的媒体流，应答者没有相同的媒体格式，那么应答者必须通过将端口号设置为 0 来拒绝该媒体流。

如果对于所有的媒体流都没有相同的媒体格式，那么将拒绝整个提供的会话。

一旦应答者发送了应答，它就必须准备在应答中描述的所有只收媒体流上接收媒体。它必须准备在应答中描述的所有收发媒体流上发送和接收媒体，当然应答者可以立即发送媒体。它必须准备在应答中描述的所有只收或收发媒体流上使用对这些媒体流列出的任何媒体格式接收媒体，当然应答者可以立即发送媒体。当发送媒体时，如果提供中包含 ptime 属性，那么应答者应该以该属性的值为间隔发送分组。如果提供中包含 bandwidth 属性，那么应答者应该使用不大于该属性值的带宽发送媒体。应答者必须使用在提供中和应答中同时列出的媒体格式来发送媒体，并且应使用在提供中和应答中同时列出的最优选的媒体格式。在使用 RTP 的情况下，应答者必须使用提供中的载荷类型值，即使该值和应答中的不同。

E.5.2 组播媒体流

和单播不同，单播中的媒体流有双向的概念，但在组播中，媒体流只有单向的概念。因此，对于组播提供要产生应答通常包括修改媒体流的受限属性集。

如果接受组播媒体流，应答中的地址和端口号必须和提供中的相匹配。类似地，应答中的方向性信息（只发、只收或收发）也必须和提供中的相同。这是因为组播会话中的所有参加者需要使用相同的会话参数，以下关于组播的假设基于 RFC 2327。

应答中的媒体格式集必须等于提供中媒体格式集或是该格式集的子集。删除某个媒体格式表明应答者不支持该媒体格式。

如果提供中包含 ptime 和 bandwidth 属性，那么应答中的这些属性值必须等于提供中的属性值。如果提供中不包含这些属性，那么应答中可以增加值为非 0 的 ptime 属性。

E.6 提供者处理应答

当提供者收到应答时，它可以在接收的媒体流上发送媒体（假设在应答中该媒体流工作在收发或只收方式）。它必须使用应答中列出的媒体格式来发送媒体，并且应使用应答中列出的最优选的媒体格式。

提供者应根据应答中 ptime 和 bandwidth 的属性值发送媒体。

提供者可以立即停止监听初始提供中列出但应答中未列出的媒体格式。

E.7 修改会话

会话中的任何时刻，任何一个参加者都可以发送新的提供修改会话特性。这是提供/应答模式最基本的操作，和上面定义的用于修改一个已经存在的会话参数的提供/应答程序完全相同。

提供可以和最后提供给另一方的 SDP 相同（即该 SDP 曾经包含在提供或应答中），也可以不相同。称最后提供的 SDP 为“先前的 SDP”。如果提供相同，应答可以和应答者先前的 SDP 相同，当然也可以不同。如果提供的 SDP 和先前的 SDP 不同，那么在构建提供的 SDP 时将有一些限制。对限制的描述如下。

几乎会话的所有属性都可能被修改。可以增加新的媒体流，删除已经存在的媒体流，或者改变已经

存在的媒体流的参数。当发送提供来修改会话时，新 SDP 的“o=”必须和先前的 SDP 的“o=”相同，除了初始字段中的版本号必须比先前的 SDP 的版本号大一。如果初始行中的版本号不增加，则新的 SDP 必须和具有相同版本号的先前的 SDP 相同。应答者必须准备接收包含具有不变版本号的 SDP 的提供。这是非常有效的 no-op。但是，应答者必须根据第 6 节定义的程序产生一个有效的应答（可以和应答者先前的 SDP 相同，也可以不同）。

如果提供的 SDP 和先前的 SDP 不同，对于先前的 SDP 中的每个媒体流新的 SDP 必须有一个匹配的媒体流。也就是说，如果先前的 SDP 有 *N* 个“m=”行，那么新的 SDP 也必须至少有 *N* 个“m=”行。先前的 SDP 中的第 I 个媒体流（从上往下数）必须和新的 SDP 中的第 I 个媒体流（从上往下数）匹配。这种匹配是必需的，以便应答者确定新的 SDP 中的哪个媒体流和先前的 SDP 中的哪个媒体流相对应。由于这些要求，媒体流中“m=”行的数目不会减少，但可能不变或者增加。要删除先前的 SDP 中的媒体流，该媒体流也必须包含在新的 SDP 中，但是不必提供这些媒体流的属性。

E.7.1 增加媒体流

通过在存在的媒体描述之下增加新的媒体描述可以创建新的媒体流，或者通过重新使用通过设置端口号为 0 而变为不可用的媒体流的“位置”创建新的媒体流。

重新使用位置指的是用新的媒体描述代替旧的媒体描述，但是保留该媒体描述在 SDP 中的位置。新的媒体描述必须出现在所有已经存在的媒体描述的下面。排列这些媒体描述的规则和第 5 节中描述的相同。

当应答者收到比提供者先前的 SDP 包含更多媒体描述的 SDP 时，或者应答者收到在先前端口号被置为 0 的位置上包含媒体流的 SDP 时，应答者就知道 SDP 中增加了新的媒体流。可以通过在应答中包含适当的结构化的媒体描述来拒绝或接收这些媒体流。在应答中构建新的媒体描述的程序在第 6 节中描述。

E.7.2 删除媒体流

可以通过创建新的 SDP，并将其中该媒体流的端口号设置为 0 来删除存在的媒体流。媒体描述可以省略掉先前提供的所有属性，而只列出一个媒体格式。

提供中端口号为 0 的媒体流在应答中也必须标记为端口号为 0。和提供类似，应答也可以在媒体描述中省略掉先前提供的所有属性，而只列出来自提供的一个媒体格式。

删除媒体流意味着对该媒体流将不再发送媒体，所有收到的媒体也将被丢弃。在采用 RTP 的情况下，要停止发送 RTCP 分组，并且停止处理所有收到的 RTCP 分组，可以释放和该媒体流相关的所有资源。在用户接口上可以指示该媒体流已经被终止，例如通过关闭相关的 PC 窗口。

E.7.3 修改媒体流

媒体流的所有特定几乎都可以被修改。

E.7.3.1 修改地址、端口号或传输协议

可以修改媒体流的端口号。为此，提供者可以创建新的媒体描述，并且“m=”行上的端口号和先前的 SDP 中相应流的端口号不同。如果仅仅改变端口号，媒体流描述的其他部分应保持不变。一旦发送了提供，提供者必须在旧端口号和新端口号上同时准备接收媒体。在收到应答且媒体到达新端口之前，提供者不应停止对媒体的监听。但是这样处理有可能在转移的过程中丢失媒体。

在这种情况下，收到媒体指的是媒体已经传送到了媒体接收器。如果存在 playout 缓冲区，代理应继续在旧端口上监听媒体直到来自新端口的媒体到达 playout 缓冲区的顶端。此时，代理可以停止在旧端口

上监听媒体。

应答中的媒体流可以和应答者先前 SDP 中相应的媒体流相同，但也可以不同。如果应答者接受更新的媒体流，那么对于该媒体流应答者应该立即发送业务到新端口上。如果应答者改变来自先前 SDP 的端口号，那么它必须准备一发送应答就在新端口和旧端口上同时接收媒体。在媒体到达新端口之前应答者不能停止在旧端口上对媒体的监听。对于提供者也一样，当提供者发送了包含新端口号的更新的提供，在媒体到达新端口之前，提供者也不能停止在旧端口上对媒体的监听。

当然，如果提供中的媒体流被拒绝，一收到拒绝消息提供者就可以停止准备用新端口号接收媒体。

要改变媒体发往的 IP 地址，除了此时要更新连接行而不是端口号之外，其他和改变端口号的程序相同。

也可以改变媒体流的传输协议。除了要更新传输协议而不是端口号之外，其他和改变端口号的程序相同。

E.7.3.2 改变媒体格式集

可以改变会话中使用的媒体格式列表。为此，提供者可以创建新的媒体描述，其中“m=”中的媒体格式列表和先前的 SDP 中相应媒体流的媒体格式列表不同。该列表可以包含新的媒体格式，并且可以删除先前的 SDP 中的媒体格式。但是，在使用 RTP 的情况下，在会话过程中，特定动态载荷类型值到媒体流中特定编解码的映射不能改变。例如，如果 A 产生提供，并且对 G711 指定动态载荷类型值 46，那么在会话过程中，对于该媒体流的提供或应答，载荷类型值 46 必须一直映射到 G711。但是，多个载荷类型值可以映射到同一个编解码，所以更新的提供也可以对 G711 使用载荷类型值 72。

在会话过程中映射需要保持不变是因为 SDP 的信令交互和媒体流之间松散的同步

应答中相应的媒体流按照第 6 节的描述排列，并可能导致媒体格式的改变。同样，和第 6 节中的描述相同，一旦应答者发送了应答，它就必须使用同时包含在提供中和应答中的媒体格式开始发送媒体，且应该使用同时包含在提供中和应答中的最优选的格式（假设媒体流允许发送媒体），但不能使用任何提供中未包含的媒体格式发送媒体，即使这些媒体格式包含在提供者先前的 SDP 中。同样，当提供者接收到应答，它必须使用应答中的任何媒体格式开始发送媒体，且应该使用最优选的媒体格式（假设该媒体流允许发送媒体），但不能使用任何应答中未包含的媒体格式发送媒体，即使这些媒体包含在应答者先前的 SDP 中。

当代理停止使用某个媒体格式时（通过在提供或应答中不再包含该媒体格式，即使该媒体格式包含在先前的 SDP 中），代理仍然需要准备在一小段时间内接收该媒体格式的媒体。代理可以通过三种方式知道何时准备停止接收该媒体格式。第一种，代理除了改变媒体格式还可以改变端口号。当媒体到达新端口时，代理知道对等已经停止使用旧的媒体格式发送媒体，那么代理可以停止准备接收该媒体格式。这种方式利用了媒体格式的独立性。但是，改变端口号可能要求改变资源预留或者是改变安全协议的密钥。第二种方式，当丢弃一种媒体格式时，对所有的编解码使用一个完全新的动态载荷类型集。当收到使用某个新的载荷类型的媒体时，代理知道对等已经停止使用旧的媒体格式发送媒体。这种方式不影响资源预留或安全上下文，但是必须用于使用 RTP 的情况下并且会浪费少量的载荷类型空间。第三种方式是使用定时器。当从对等收到 SDP 时，启动定时器。当定时器超时时，代理停止准备接收旧的媒体格式。定时器的典型值为一分钟。在某些情况下，代理可以忽略定时器而继续准备接收旧的媒体格式。在这种情况下不必采取任何动作。

当然，如果拒绝提供中的媒体流，一旦收到拒绝消息，提供者可以停止准备接收新的媒体格式。

E.7.3.3 改变媒体类型

可以改变媒体流的媒体类型（音频、视频等等）。建议当使用不同的媒体格式传送相同的逻辑数据时改变媒体类型（相对于增加新的媒体流）。当在语音频带内传送传真和在单个媒体流上传送传真之间相互进行转移时非常有用，此时采用的是不同媒体类型。为此，提供者使用新的媒体类型创建新的媒体描述，替换要改变的先前 SDP 中的媒体描述。

应答中相应的媒体流按照第 6 节中的描述进行排列。假设接收该媒体流，那么应答者一收到提供就应开始使用新的媒体类型和新的媒体格式发送媒体。提供者必须准备同时接收旧媒体类型和新媒体类型，直到已经收到应答，同时已经收到新类型的媒体且该媒体已经到达 playout 缓冲区顶部。

E.7.3.4 改变属性

在提供或应答中可以更新媒体描述中的任何其他属性。通常，一旦收到改变的 SDP，代理必须使用新的参数发送媒体（如果媒体流允许发送媒体）。

E.8.4 设置单播媒体流处于保持状态

如果呼叫中的某一方想使另一方处于“保持”状态，即请求另一方暂时停止发送一个或多个单播媒体流，那么它应向另一方提供更新的 SDP。

如果要进入保持状态的媒体流先前为收发媒体流，那么通过将该媒体流标记为只发使对端进入保持状态。如果要进入保持状态的媒体流先前为只收媒体流，那么通过将该媒体流标为未激活使对端进入保持状态。

媒体流可以在双向上分别处于“保持”状态。每个流都可以独立地处于保持状态。对于处于保持状态的流，提供的接收者不应自动返回包含该流的应答。如果 SDP 中所有的流都处于保持状态那么该 SDP 称为保持的 SDP。

当应答者对保持的 SDP 响应保持的 SDP，那么某些第三方呼叫控制将不再起作用。

典型地，当用户启动保持时，代理将产生提供，SDP 中所有的媒体流为只发，同时本地静音，以便不向远端发送媒体，也不播放任何媒体。

当提供者在创建初始提供时想使用特定一组媒体流和媒体格式，但不知道媒体流的地址和端口号，此时可以将连接地址设置为 0.0.0.0。但此时端口号不能设置为 0。代理必须能够接收连接地址为 0.0.0.0 的 SDP，此时不应向对等发送 RTP 和 RTCP。

E.8 指示能力集

代理发送提供前，如果知道应答者是否可接受提供中的媒体格式将很有用。某些协议，如 SIP，提供一种查询能力集的方式。在对查询的响应中可以使用 SDP 指示能力集。本节描述如何创建该 SDP 消息。SDP 不能指示该消息用于能力集指示，需要从高层协议上下文中确定。SDP 指示能力集的能力非常有限。它不能表示允许的参数范围或参数值，也不能在相关的提供/应答中指示。

指示媒体能力集的 SDP 组成如下。除了可以省略“e=”和“p=”之外，它必须是有效的 SDP。“t=”行必须等于“0 0”。对于代理支持的每个媒体类型，必须有相应的媒体描述。对于每个用于指示媒体能力集的 SDP 初始字段中的会话 ID 都必须惟一。端口号必须=0，但是连接地址可以为任意值。如果该消息被当作提供或应答，端口号为 0 确保用于指示媒体能力集的 SDP 不会引起媒体流的建立。

“m=”行的传输协议部分指示媒体类型的传输协议。对于代理所支持的媒体类型的每个媒体格式，在“m=”行都应列出。在采用 RTP 的情况下，如果采用动态载荷类型，那么必须包含 rtpmtp 属性，以便将媒体类型绑定到特定的格式。这里没有办法对限制进行描述，例如对于特定的编解码可以同时支持多少媒体流，等等。

E.9 提供/应答交互示例

E.9.1 基本的交互

假设主叫 Alice 在它的提供中包含下列 SDP 描述，其中包含一个双向音频流和两个双向视频流（视频流分别使用编解码 H.261（载荷类型 31）和 MPEG（载荷类型 32）。

v=0

o=alice 2890844526 2890844526 IN IP4 host.anywhere.com

s=

c=IN IP4 host.anywhere.com

t=0 0

m=audio 49170 RTP/AVP 0

a=rtpmap：0 PCMU/8000

m=video 51372 RTP/AVP 31

a=rtpmap：31 H261/90000

m=video 53000 RTP/AVP 32

a=rtpmap：32 MPV/90000

被叫 Bob 不希望接收或发送第一个视频流，那么它将在应答中返回如下 SDP：

v=0

o=bob 2890844730 2890844730 IN IP4 host.example.com

s=

c=IN IP4 host.example.com

t=0 0

m=audio 49920 RTP/AVP 0

a=rtpmap：0 PCMU/8000

m=video 0 RTP/AVP 31

m=video 53000 RTP/AVP 32

a=rtpmap：32 MPV/90000

之后的某个时刻，Bob 想改变音频流的接收端口（从 49920 改为 65422），同时增加另外一个只收音频流，并使用 RTP 载荷类型，那么 Bob 将在提供中包含如下 SDP：

v=0

o=bob 2890844730 2890844731 IN IP4 host.example.com

s=

c=IN IP4 host.example.com

t=0 0

m=audio 65422 RTP/AVP 0

a=rtpmap：0 PCMU/8000

m=video 0 RTP/AVP 31

m=video 53000 RTP/AVP 32

a=rtpmap：32 MPV/90000

m=audio 51434 RTP/AVP 110

a=rtpmap：110 telephone-events/8000

a=recvonly

Alice 接收增加的媒体流，将产生如下应答：

v=0

o=alice 2890844526 2890844527 IN IP4 host.anywhere.com

s=

c=IN IP4 host.anywhere.com

t=0 0

m=audio 49170 RTP/AVP 0

a=rtpmap：0 PCMU/8000

m=video 0 RTP/AVP 31

a=rtpmap：31 H261/90000

m=video 53000 RTP/AVP 32

a=rtpmap：32 MPV/90000

m=audio 53122 RTP/AVP 110

a=rtpmap：110 telephone-events/8000

a=sendonly

E.9.2 从 *N* 个编解码中选择一个

嵌入式呼叫中的一个共同现象是用于压缩信号的数字信号处理器（DSP）可以同时支持多个编解码，但是一旦选择了一个编解码，就不能轻易地改变。下面的示例说明使用初始提供/应答交互如何建立对话，随后如何立即通过第二个提供/应答的交互来确定编解码。

从 Alice 发送的 Bob 的初始提供指示单个的音频流可以使用 DSP 所支持的三个音频编解码。该流被标记为未激活，因为在确定编解码之前不能接收媒体。

v=0

o=alice 2890844526 2890844526 IN IP4 host.anywhere.com

s=

c=IN IP4 host.anywhere.com

t=0 0

m=audio 62986 RTP/AVP 0 4 18

a=rtpmap：0 PCMU/8000

a=rtpmap：4 G723/8000

a=rtpmap：18 G729/8000

a=inactive

Bob 可以支持 PCMU 和 G.723 之间的动态交换，所以他发送以下应答：

v=0

o=bob 2890844730 2890844731 IN IP4 host.example.com

s=

c=IN IP4 host.example.com

t=0 0

m=audio 54344 RTP/AVP 0 4

a=rtpmap：0 PCMU/8000

a=rtpmap：4 G723/8000

a=inactive

Alice 可以选择这两个编解码中的任何一个。所以她发送更新的提供，媒体流为收发：

v=0

o=alice 2890844526 2890844527 IN IP4 host.anywhere.com

s=

c=IN IP4 host.anywhere.com

t=0 0

m=audio 62986 RTP/AVP 4

a=rtpmap：4 G723/8000

a=sendrecv

Bob 接收该编解码：

v=0

o=bob 2890844730 2890844732 IN IP4 host.example.com

s=

c=IN IP4 host.example.com

t=0 0

m=audio 54344 RTP/AVP 4

a=rtpmap：4 G723/8000

a=sendrecv

如果应答者（Bob）只能支持 *N* 个编解码中的一个，Bob 将选择这个编解码，并且将包含在应答中。此时 Alice 将重新发送 INVITE 方法来激活使用该编解码的媒体流。

一种替换方式是在初始的交互中使用“a=inactive”，并列出所有的编解码，并且一旦从 Bob 接收到媒体，Alice 就产生更新的提供，并使用接收媒体的编解码。如果 Bob 只支持 *N* 个编解码中的一个，此时就不需要重新发送 INVITE 方法来激活使用该编解码的媒体流。

E.10 安全

如果攻击者在提供或应答交互过程中能够修改它们，那么可能产生大量的攻击。通常包括转移媒体流（以便监听），使呼叫中止，以及插入不期望的媒体流。如果被叫方创建假的提供，并插入到交互过程中，也可以产生类似的攻击。即使攻击者只能简单地观察提供者和应答者，他们仍然能向存在的对话中插入媒体流。

提供/应答依赖于应用信令协议中的传输协议，如 SIP。也依赖于该传输协议提供的安全机制。为防止上述攻击，传输协议必须提供端到端的鉴权和对提供应答完整性的保护。该协议应提供对消息体的加密来防止监听。

重发攻击也是一个问题。攻击者可能重发过时的提供，如使媒体流进入保持状态的提供，从而使对话中的媒体流不能传送媒体。因此应用协议必须提供安全机制来顺序排列提供和应答，以便检测并拒绝过时的提供或应答。

附 录 F
（规范性附录）
特定事件的通知

通过该机制 SIP 实体可以请求远端实体发送通知指示某些事件已经发生。

F.1 简介

请求异步通知事件的能力在许多端实体间需要互操作的 SIP 业务中非常有用。这类业务的例子包括自动回叫业务（基于终端状态事件）、朋友列表（基于用户 presence 事件）、消息等待指示（基于邮箱状态改变事件）和 PSTN 与 Internet 互通（PINT）状态（基于呼叫状态事件）。

本文描述的方法提供了规定通知这些事件的框架。

此处定义的事件通知机制不想作为所有类型事件订阅和通知机制的基础。本文旨在对事件通知提供一种特定的 SIP 框架。基于该框架的事件包可以任意定义详细的规则，规定事件或事件类别的订阅和通知机制。

本文不描述可以直接使用的扩展，需要其他文档（本文称为“事件包”）进行扩展。

F.1.1 操作概述

对于网络中的各种资源或呼叫，网络中的实体可以订阅资源或呼叫状态，当状态改变时这些实体可能发送通知消息，如图 F1 所示。

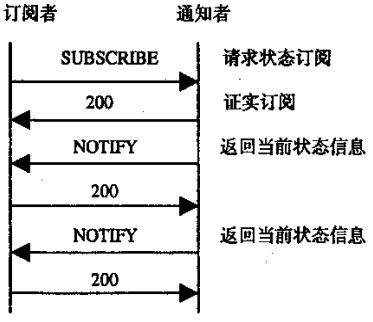


图 F.1 流程示例

订阅消息可能超时而必须使用后续的 SUBSCRIBE 消息刷新。

F.2 定义

事件包：事件包是附加的规范，用于定义通知者向订阅者报告的状态信息集。基于本文定义的框架事件包也定义用于传递状态信息的语法和语义。

事件模板包：一种特定类型的事件包，定义可用于所有可能事件包的一组状态，包括事件模板包本身。

通知：通知指的是通知者向订阅者发送 NOTIFY 消息以便通知订阅者资源的状态。

通知者：通知者是产生 NOTIFY 请求的用户代理，目的是通知订阅者资源的状态。通常通知者接收 SUBSCRIBE 请求来产生订阅。

状态代理：状态代理是发布表示资源状态信息的通知者；为此状态代理可能需要从多个源搜集相关

的状态信息。状态代理要一直为那些创建通知的资源保持完整的状态信息。

订阅者：订阅者是从通知者接收 NOTIFY 请求的用户代理。这些通知请求包含订阅者关心的资源的状态信息。通常订阅者产生 SUBSCRIBE 请求，并向通知者发送该请求以便产生订阅。

订阅：订阅是和对话相关的一组应用状态。该应用状态包含一个指向相关对话的指针和事件包名，还可能包含一个标记。事件包定义附加的订阅状态信息。通过定义，订阅可以在订阅者和通知者中同时存在。

订阅转移：订阅转移是将订阅从一个通知者转移到另一个通知者的动作。

F.3 实体行为

F.3.1 SUBSCRIBE 行为定义

SUBSCRIBE 方法用于请求远端实体的当前状态和状态更新。

F.3.1.1 订阅持续时间

SUBSCRIBE 请求应包含“Expires”头部字段。该字段的值指示订阅持续时间。为了在该持续时间之外保持该订阅有效，订阅者需要对同一个对话发送新的 SUBSCRIBE 消息周期性地刷新订阅。

如果 SUBSCRIBE 请求中不包含“Expires”头部字段，那么将使用该事件包定义的缺省值。

SUBSCRIBE 请求的 200 类响应也必须包含“Expires”头部字段。响应中该字段的值可以较小，但不能大于请求中指定的值。响应中该字段的值定义了订阅的持续时间。

对于 SUBSCRIBE 请求“Contact”头部字段中的“expires”参数没有意义，并且不必和 SUBSCRIBE 请求或响应中的“Expires”头部字段相等。

“Expires”等于 0 的 SUBSCRIBE 请求不订阅事件，但可以使对端回送状态信息。

通知者可能想取消对事件的订阅，例如订阅涉及的资源不再有用。

F.3.1.2 订阅事件或事件类别的标识

事件的标识包含三个信息：Request URI、事件类型和消息体（任选的）。

SUBSCRIBE 请求的 Request URI 最重要，其中包含足够的信息以便将请求选路到适当的实体，其中也包含足够的信息来标识订阅者期望从它获得事件通知的资源，但是不需要该 URI 必须惟一地标识事件特性（例如：“sip：adam@dynamicsoft.com”可用于订阅我的 presence 状态也可以订阅我语音信箱的当前状态）。

订阅者在 SUBSCRIBE 请求中必须包含且仅包含一个“Event”头部字段，指示订阅哪种事件或事件类型。“Event”头部字段将包含一个标记指示订阅所请求的状态类型。该标记将对应到一个事件包。“Event”头部字段也可以包含“id”参数。如果包含该参数，那么该参数中应包含标识对话中特定订阅的标记。该参数仅在单个对话中有效。

如果事件标记对应的事件包定义和 SUBSCRIBE 请求体相关的行为，应用这些语义。

事件包可以定义 Event 头部字段的参数；若定义相关的参数也必须定义参数的语义。

F.3.1.3 其他的 SUBSCRIBE 头部值

因为 SUBSCRIBE 请求创建对话，所以该请求可以包含“Accept”头部字段。该字段指示后续 NOTIFY 请求中允许的消息体格式。事件包必须定义未包含“Accept”字段的 SUBSCRIBE 请求的行为和缺省的消息类型。

F.3.1.4 订阅者 SUBSCRIBE 行为

F.3.1.4.1 请求订阅

SUBSCRIBE 是一种对话创建机制。

当订阅者想订阅某个资源的特定状态时，将形成 SUBSCRIBE 消息。如果初始 SUBSCRIBE 表示对话之外的请求，创建该消息的程序和在对话外产生 UAC 请求相同。

SUBSCRIBE 请求有最终响应确认。200 类响应指示订阅被接受，并立即发送 NOTIFY 消息。200 响应指示订阅被接受并且用户已经认可对请求资源的订阅源。202 响应仅指示订阅被接受，但并不保证用户是否认可该订阅。

200 类响应中的“Expires”头部字段指示订阅保持激活的实际持续时间（除非被刷新）。

非 200 类最终响应指示没有创建任何订制或对话，并且不发送后续 NOTIFY 消息。所有非 200 类响应有相同的意义。

SUBSCRIBE 请求可以在“Event”头部字段中包含“id”参数，以便允许在同一个对话中区分多个订阅。

F.3.1.4.2 刷新订阅

订阅超时之前的任何时刻，订阅者可以通过在同一个对话上发送另一个与当前订阅相同的 SUBSCRIBE 请求来刷新订阅的定时器，该订阅具有相同的“Event”头部字段和“id”参数（如果初始提供中包含）。除了下面的描述，对该请求的处理和对最初创建订阅的处理相同。

如果初始 SUBSCRIBE 消息在“Event”头部字段中包含“id”参数，那么刷新的订阅也必须包含相同的“id”参数；否则该订阅将被认为是对话中新的订阅。

如果用于刷新订阅的 SUBSCRIBE 请求收到“481”响应，那么表明订阅已经终止但订阅者没有收到相关的通知。在这种情况下，订阅者应认为订阅无效。如果订阅者想重新订制该状态，可以创建一个不相关的初始 SUBSCRIBE 请求，该请求包含新产生的 Call-ID 和新的且惟一的“From”标记。

如果用于刷新订阅的 SUBSCRIBE 请求收到非 481 类失败响应，那么通过 SUBSCRIBE 请求及其响应协商，或通过在“Subscription-State”头部字段中包含“expires”参数的 NOTIFY 的交互，最近确定的“Expires”持续时间之内原来的订阅仍然有效。

在网络或通知者出现故障的情况下可以收不到 NOTIFY 消息。

F.3.1.4. 未订阅

对“Expires”头部字段为 0 的未订阅的处理和对刷新订阅的处理相同。成功的未订阅可能触发最后的 NOTIFY 消息。

F.3.1.4.4 订阅创建的确认

订阅者可能期望从每个处理成功的订阅或订阅刷新的实体收到 NOTIFY 消息。在收到第一个 NOTIFY 消息之前，订阅者认为被订阅资源的状态处于中间状态。对应的事件包应对中间状态进行定义。

由于可能存在失序的消息和分支的消息，订阅者必须准备在 SUBSCRIBE 事务处理完成之间接收 NOTIFY 消息。

F.3.1.5 代理服务器 SUBSCRIBE 行为

除了要支持 SUBSCRIBE 外，代理服务器不需要其他的行为。如果对于特定的对话代理服务器想看到所有的 SUBSCRIBE 和 NOTIFY 请求，那么它必须在初始 SUBSCRIBE 请求和任何建立对话的 NOTIFY

请求中包含 record-route 字段。同样代理服务器也应在所有其他 SUBSCRIBE 和 NOTIFY 请求中包含 record-route 字段。

F.3.1.6 通知者 SUBSCRIBE 行为

F.3.1.6.1 初始 SUBSCRIBE 事务处理

SUBSCRIBE 事务处理不应比自动处理所需的时间长。特别是通知者在向 SUBSCRIBE 请求返回最终响应之前不必等待用户响应。

这主要是防止非 INVITE 事务处理超时定时器 F 在 SUBSCRIBE 事务处理期间不起作用，因为和用户交互的事务处理经常超过 64×T1 秒。

通知者应确认“Event”头部字段中规定的事件包可理解。如果不能理解，通知者应返回“489 错误事件”响应来指示不理解规定的事件/事件包。

通知者应基于本地策略执行所有必需的鉴权和认证。

通知者也应确认“Expires”头部字段中的持续时间不太小。如果超时间隔大于 0 但小于 1h 并且小于通知者配置的最小值，只有在这种情况下通知者可以返回“423 间隔太小”，并且包含“Min-Expires”头部字段。

如果通知者能够立即确认它理解该事件包，确认通过认证的订阅者有权订阅，并且确认在创建订阅前不再需要其他操作，那么通知者将创建订阅和对话（如果需要）并返回“200 OK”响应（除非这样做不期望地暴露了认证策略）。

如果通知者不能立即创建订阅（例如，它需要等待用户的输入进行认证，或者正作为另一个实体而不能接入），或者想掩饰认证策略，那么它将返回“202 Accepted”。该响应指示 SUBSCRIBE 请求已被接受和理解，但没有标明订阅是否已经经过认证。

当通知者创建了订阅，它将保存事件包名和“Event”头部字段“id”参数（如果 SUBSCRIBE 请求中包含）病最为订阅信息的一部分。

SUBSCRIBE 200 类响应中的“Expires”值和 REGISTER 响应中的“Expires”值同样处理：服务器可以缩短该间隔，但一定不能大于该间隔。

如果 SUBSCRIBE 消息中规定的持续时间太短而不能接受，通知者可能发送响应 423。

SUBSCRIBE 请求的 200 类响应除了订阅持续时间之外通常不包含任何有用信息。状态信息将通过通知者发送的后续 NOTIFY 请求进行交互。

适当地，也可以使用其他的响应代码响应 SUBSCRIBE 请求。

F.3.1.6.2 订阅创建/刷新的确认

一旦成功地接受或刷新订阅，通知者必须立即向订阅者发送 NOTIFY 消息来通知资源的当前状态。NOTIFY 消息可以在 SUBSCRIBE 响应创建的相同对话上发送。如果在处理 SUBSCRIBE 消息时资源所具有的状态没有意义，那么该 NOTIFY 消息可以包含空的或中立的消息体。

对 SUBSCRIBE 请求发送了某个 200 类响应之后总是立即发送 NOTIFY 消息，而不管该订阅是否已经经过认证。

F.3.1.6.3 SUBSCRIBE 请求的鉴权/认证

通知者用来确定特定的订阅者是否有权订阅某些事件的策略应该保密。可以通过接入控制列表或和用户进行实时交互等机制来定义相关策略。

当接收订阅和发送通知时通知者总是作为用户代理。

如果根据接入列表或某自动机制（即它能够自动进行认证确定订阅者无权进行订阅）认证失败，那么通知者应响应“403 Forbidden”或“603 Decline”，除非这样做暴露和策略相关的信息。

如果需要查询提供者的用户来确定是否允许订阅，那么将立即返回“202 Accept”响应，此时仍要发送 NOTIFY 消息。

如果订阅的认证被推迟并且通知者希望传送该订阅被拒绝的信息，那么它将发送包含“Subscription-State”头部字段的 NOTIFY 消息，该字段的值为“terminated”原因参数为“rejected”。

F.3.1.6.4 订阅的刷新

当通知者收到订阅刷新时，假设订阅者仍然有权，那么通知者将更新订阅的超时时间。和初始订阅相同，服务器可以缩短超时时间但不能增加。最终的超时时间包含在响应的“Expires”头部字段中。如果 SUBSCRIBE 消息中规定的持续时间太小而不能接受，那么通知者应响应“423 Subscription Too Brief”消息。

如果在超时时间之前通知者没有收到订阅的刷新，那么应删除订阅。当删除订阅时，通知者应发送包含“Subscription-State”头部字段且其值为“terminated”的 NOTIFY 消息，通知订阅已经被删除，并且“Subscription-State”头部字段应包含“reason=timeout”参数。

当订阅超时时，如果适当，NOTIFY 的发送允许终止相应的对话。

F.3.2 NOTIFY 行为的描述

发送 NOTIFY 消息通知订阅者它所订制的资源的状态发生改变。产生订阅一般使用 SUBSCRIBE 方法，但是也有可能使用别的方式。

如果定义了产生订阅的某些非 SUBSCRIBE 机制，那么确保 NOTIFY 消息和相应订阅之间能够进行关联是定义这些机制的实体的责任。这些机制的设计者应注意区分向意识到订阅的订阅者发送 NOTIFY 消息和向信任的实体发送 NOTIFY 消息。后面的行为是无效的，必须接收响应“481 订阅不存在”（除非其他的 400 或 500 类差错码更适当）。也就是说，订阅者和通知者必须都知道订阅才有效，即使是通过非 SUBSCRIBE 机制进行的订阅。

NOTIFY 不能终止相应的订阅，即一个 SUBSCRIBE 请求可能触发多个 NOTIFY 请求。

F.3.2.1 报告的事件、事件类别和当前状态的标识

通知消息中报告的事件的标识和订阅对事件的描述相似。

和 SUBSCRIBE 请求相同，NOTIFY 消息中的“Event”头部字段将包含一个事件包名，NOTIFY 消息就是针对该事件包产生的。“Event”头部字段中的事件包名必须和相应 SUBSCRIBE 消息中“Event”头部字段中的事件包名相匹配。如果 SUBSCRIBE 消息中包含“id”参数，那么在相应的 NOTIFY 消息中也必须包含“id”参数。

事件包可以定义和 NOTIFY 请求的消息体相关的语义；若如此将遵循事件包定义的语义。期望消息体提供和发生事件的属性及导致的资源状态相关的其他详细信息。

NOTIFY 请求中的消息体必须按照相应 SUBSCRIBE 请求中“Accept”头部字段规定的某个消息体格式进行构造。该消息体将包含订阅资源的状态或以 URI 形式表示的到该状态的指针。

F.3.2.2 通知者 NOTIFY 行为

当向 SUBSCRIBE 请求发送了 200 类响应，通知者必须立即创建并向订阅者发送 NOTIFY 请求。当

订阅的状态发生改变时，通知者应立即创建并发送 NOTIFY 请求，并考虑到认证、本地策略和取消机制。

如果响应超时，或收到非 200 类响应，该响应未包含“Retry-After”头部字段且没有暗示可以采取进一步的动作以便重新尝试该请求（例如：响应“401 要求认证”），则认为 NOTIFY 请求失败。

如果由于超时 NOTIFY 请求失败，并且是使用软状态机制（如使用 SUBSCRIBE 请求）进行的订阅，那么通知者应删除该订阅。

这种方式可以避免向已经故障或已经在网络中不存在的订阅者发送不必要的状态信息。基于重传机制该消息可能被发送多次，但是没有用户端对消息进行证实，此时连续地发送这些消息可能会导致网络的过负荷。一旦客户端开始重启或重新建立网络连接，客户端应发送 SUBSCRIBE 消息来刷新已经过时的状态信息，这些消息将在所有相关的实体中重新设置订阅。

如果由于差错响应 NOTIFY 请求失败，并且是使用软状态机制设置订阅，那么通知者必须删除相应的订阅。

通知的差错响应通常指示订阅者或者是到订阅者所经由的某个代理服务器上发生了差错。如果订阅者发生差错，一旦差错情况消除那么应允许订阅者重新进行订阅。如果代理服务器发生差错，一旦该网络问题解决，那么周期性的 SUBSCRIBE 刷新将重新设置订阅状态。

如果 NOTIFY 请求收到响应 481，那么通知者必须删除相应的订阅，即使该订阅时通过非 SUBSCRIBE 方式（如通过管理接口）进行的设置。

NOTIFY 请求必须包含“Subscription-State”头部字段，且该字段的值可以为“active”、“pending”或“terminated”。值为“active”指示已经接受订阅并已经对订阅进行了认证。值为“pending”指示已经收到订阅，但此时还没有足够的策略信息来接受或拒绝订阅。值为“terminated”指示订阅未激活。

如果“Subscription-State”头部字段的值为“active”或“pending”，那么通知者应在“Subscription-State”头部字段中包含“expires”参数指示订阅还可以持续的时间。通知者可以使用该机制来缩短订阅的持续时间，但是该机制不能用来增加订阅的持续时间。

如果“Subscription-State”头部字段的值为“terminated”，那么通知者应包含“reason”参数。如果适当通知者也可以包含“retry-after”参数。

F.3.2.3 代理服务器 NOTIFY 行为

除了要支持 NOTIFY 之外，代理服务器不需要其他行为。如果对于某个对话代理服务器想看到所有的 SUBSCRIBE 和 NOTIFY 请求，那么它必须在初始 SUBSCRIBE 和所有建立对话的 NOTIFY 请求中包含 record-route 字段。该代理在所有其他 SUBSCRIBE 和 NOTIFY 请求中也应包含 record-route 字段。

F.3.2.4 订阅者 NOTIFY 行为

一旦收到 NOTIFY 请求，订阅者应确认该请求至少和一个已经存在的订阅相匹配。如果没有相匹配的订阅，那么订阅者必须返回响应“481 不存在”，除非其他更恰当的 400 类或 500 类响应。如果通知消息和在对话内创建的订阅在相同的对话中且“Event”头部字段相匹配，那么该通知消息和该订阅相匹配。

如果由于某些原因不支持 NOTIFY 请求中“Event”头部字段指定的事件包，那么订阅者将响应“489 错误事件”。

要防止虚假事件，应使用定义的 SIP 鉴权机制对 NOTIFY 请求进行鉴权。

NOTIFY 请求必须包含“Subscription-State”头部字段来指示订阅的状态。

如果“Subscription-State”头部字段的值为“active”，那么表明该订阅已经被接受（通常）并且已经

经过认证。如果该头部字段包含“expires”参数，那么订阅者应将该参数的值作为订阅的持续时间并相应地对订阅进行调整。此时“retry-after”参数和“reason”参数没有意义。

如果“Subscription-State”头部字段的值为“pending”，那么表明通知者已经收到订阅，但还没有足够的策略信息来接受或否定该订阅。如果该头部字段包含“expires”参数，订阅者应将该参数的值作为订阅的持续时间并相应地对订阅进行调整。订阅者不需要采取进一步的动作。此时“retry-after”参数和“reason”参数没有意义。

如果“Subscription-State”头部字段的值为“terminated”，那么订阅者应认为该订阅已经终止。此时“expires”参数没有意义。如果包含原因码，那么订阅者采取下面描述的行为。如果不包含原因码或不知道原因码，订阅者应随时尝试重新进行订阅（当包含“retry-after”参数时，订阅者应该在该参数规定的时间之后尝试重新进行订阅）。定义的原因码有：

去激活：订阅已经终止，但是订阅者应该立即重试新的订阅。该原因码主要用于在实体之间进行订阅的转移。此时“retry-after”参数没有意义。

试用：订阅已经终止，但订阅者应在一段时间之后重试。如果包含“retry-after”参数，那么订阅者在尝试重新订阅之前至少应等待一段该参数规定的时间。

拒绝：由于认证策略的改变订阅已经终止。订阅者不应尝试重新进行订阅。此时“retry-after”没有意义。

超时：由于在定时器超时之前没有对订阅进行刷新而导致该订阅已经终止。订阅者可以立即重新开始订阅。此时“retry-after”没有意义。

放弃：由于通知者不能及时获得认证而导致该订阅已经终止。如果包含“retry-after”参数，订阅者在尝试重新订阅之前应至少等待一段该参数规定的时间；否则，订阅者可以立即进行尝试，但订阅可能进入 Pending 状态。

没有资源：由于要监视的资源的状态不再存在而导致订阅终止。订阅者不应尝试重新进行订阅。此时“retry-after”没有意义。

一旦订阅者确认通知可接受，那么订阅者应返回响应 200。通常，NOTIFY 的响应不包含消息体，但是如果 NOTIFY 请求包含“Accept”头部字段那么响应可以包含消息体。

如何适当，也可以返回其他响应。但是 NOTIFY 事务处理不能比自动处理所需要的时间长。特别地，在返回 NOTIFY 请求的最终响应之前不必等待用户响应。

F.3.3 综述

F.3.3.1 确定对 SUBSCRIBE 和 NOTIFY 的支持

SUBSCRIBE 和 NOTIFY 不一定要使用“Require”、“Proxy-Require”或“Supported”头部字段。如果需要，客户端可以使用 OPTIONS 请求来探测对 SUBSCRIBE 和 NOTIFY 的支持。

在消息中包含“Allow-Events”头部字段足以指示对 SUBSCRIBE 和 NOTIFY 的支持。

注册时在 Contact 字段中包含“methods”参数也可以用来指示对 SUBSCRIBE 和 NOTIFY 的支持。

F.3.3.2 CANCEL 请求

没有和取消 SUBSCRIBE 或 NOTIFY 相关的语义。

F.3.3.3 分支

要符合代理非 INVITE 请求的规则，成功的 SUBSCRIBE 请求应只收到一个 200 类响应，但是由于分

支，可能有多个实体收到订阅。因此订阅者必须准备接收“From”标记和 SUBSCRIBE 的 200 类响应中的"To"标记不同的 NOTIFY 请求。

如果在不同的对话中收到多个响应某个 SUBSCRIBE 消息的 NOTIFY 消息，那么每个对话表示 SUBSCRIBE 请求分支到的不同目的地。

F.3.3.4 对话创建和终止

如果初始 SUBSCRIBE 请求不在已经存在的对话上发送，那么订阅者将等待 SUBSCRIBE 请求的响应或者是匹配的 NOTIFY 消息。

如果 SUBSCRIBE 请求和响应中包含相同的“Call-ID”、相同的“From”标记和相同的“Cseq”，那么该响应和该请求相匹配。如果 200 类响应和该 SUBSCRIBE 请求匹配，那么该响应将创建新的订阅和新的对话（除非已经由匹配的 NOTIFY 请求创建）。

如果 NOTIFY 请求和该 SUBSCRIBE 请求包含相同的“Call-ID”、和 SUBSCRIBE 请求中“From”标记相同的“To”标记、相同的“Event”头部字段，那么该 NOTIFY 请求和该 SUBSCRIBE 请求相匹配。如果匹配的 NOTIFY 请求包含“Subscription-State”头部字段，且该字段的值为“active”或“pending”，那么该请求将创建新的订阅和新的对话（除非已经由匹配的响应创建）。

如果初始 SUBSCRIBE 请求是在已经存在的对话上发送，那么匹配的 200 类响应或成功的 NOTIFY 请求只创建和该对话相关的新的订阅。

可能有多个订阅和单个对话相关。订阅也可能存在于由 INVITE 创建的应用状态和由其他机制创建的应用状态相关的对话中。这些应用状态在对话的行为之外不进行交互（例如：路由组处理）。

当通知者发送包含“Subscription-State”字段且该字段的值为“terminated”的 NOTIFY 请求时，该订阅终止。

订阅者可以发送包含“Expires”头部字段且该字段的值为零的 SUBSCRIBE 请求来触发 NOTIFY 请求的发送。但是考虑到订阅和对话的生存时间，在发送包含“Subscription-State”字段且该字段的值为“terminated”的 NOTIFY 请求之前，将不认为订阅已经终止。

如果订阅终止后没有其他和对话相关的应用状态，那么对话也终止。如果仍然有订阅和对话相关，那么其他应用状态（如由 INVITE 请求创建的应用状态）的终止将不会终止对话。

即当收到 BYE 时不一定终止由 INVITE 创建的对话。同样，如果一个对话有多个相关的订阅，在该对话中的所有订阅终止之前不终止对话。

F.3.3.5 状态代理和通知者转移

当使用状态代理时，允许订阅在状态代理和实体（状态代理正在为这些实体提供状态汇聚，或者在多个状态代理中提供状态汇聚）间进行转移将非常有用。通过发送包含“Subscription-State”头部字段且值为“terminated”原因参数为“去活”的 NOTIFY 消息可以实现该类转移。

当收到这个 NOTIFY 消息时，订阅者应尝试重新进行订阅。该订阅将在新的对话上建立，并且不会再使用先前订阅对话中的路由集。

通过改变一个或多个服务器向实体发送 SUBSCRIBE 请求的策略（如选路策略），使另一个不同的实体来响应该 SUBSCRIBE 请求，可以产生实际的转移。此时仅简单地利用通知者内的本地策略来实现订阅的转移，此时通知者是作为代理服务器或重定向服务器。

是否执行通知者转移、什么时候执行、为什么执行转移可以在单独的事件包中进行描述；另外决定

转移是本地通知者的策略问题，将依赖于具体实现。

F.3.3.6 查询资源状态

可以通过发送包含“Expires”字段且值为 0 的 SUBSCRIBE 消息实现不建立订阅而立即返回相关的状态。

当订阅激活时，也可以通过发送包含“Expires”字段，且该字段的值等于订阅剩余持续时间的 SUBSCRIBE 消息实现立即返回相关的状态。

当收到这个 SUBSCRIBE 请求时，通知者将在相同对话中发送包含资源状态的 NOTIFY 请求。

由包含“Expires”头部字段且值为 0 的 SUBSCRIBE 消息触发的 NOTIFY 消息将包含值为“终止”的“Subscription-State”字段且其中的原因参数为“超时”。

事件状态的查询将在很大程度上增加网络和通知者的负荷，所以应该谨慎使用。特别地，当查询导致比长期运行的订阅还要多的网络消息时不应使用。

当使用事件状态的查询时，订阅者应尝试缓存查询过程中的鉴权凭证以便减少发送消息的数目。

F.3.3.7 Allow-Events 头部字段的用法

“Allow-Events”头部字段中包含一个标记列表，指示客户端（如果在请求中发送）或服务器（如果在响应中发送）所支持的事件包。也就是说，发送“Allow-Events”头部字段的实体表明它能够处理 SUBSCRIBE 请求并能对字段中列出的所有事件包产生 NOTIFY 请求。

任何实现一个或多个事件包的实体都应包含适当的“Allow-Events”头部字段，指示在所有启动对话、响应（如 INVITE）和 OPTIONS 响应的方法中所支持的所有事件。

这个信息非常有用，例如，允许用户代理根据实体是否支持需要某些特定性能的事件而决定适当地放弃特定的接口单元，此时该信息非常有用。

代理服务器不能插入“Allow-Events”头部字段。

F.3.3.8 PINT 兼容性

“Event”头部字段是必需的。但是要保证和 PINT 的兼容性，当请求订阅 PINT 事件时，服务器可能需要解释未包含“Event”头部字段的 SUBSCRIBE 请求。如果服务器不支持 PINT，那么它应对未包含“Event”头部字段的 SUBSCRIBE 消息返回“489 错误事件”。

F.4 事件包

本节描述当设计基于 SUBSCRIBE 和 NOTIFY 的事件包时需要考虑的几个问题。

F.4.1 用法的适当

当使用本文中描述的方法为事件通知设计事件包时，重点需要考虑：对于问题集 SIP 是否是一个适当的机制；是否由于 SIP 提供一些特殊的机制而选用它，或者仅仅是因为“它能够做到”。

另外在事件报告频率太快（例如：一秒多于一次）的应用中不要使用这个机制。

F.4.2 事件模板包

正常情况下，事件包定义一组应用于特定资源类型的状态，例如用户在线、呼叫状态和消息信箱状态。

事件模板包是一种特殊类型的包，它定义的是一组应用于其他包的状态，例如统计、接入策略和用户列表。事件模板包甚至可以应用于其他的事件模板包。

用于包的事件模板包的名字可以在该包名字之后加一个小圆点再加上事件模板包的名字。例如，如果事件模板包“winfo”正应用于包“presence”，那么“Event”和“Allow-Events”字段中使用的该事件标记应为“presence.winfo”。

必须定义事件模板包，以便它们可用于任何包。事件模板包不能特定地绑定到一个或几个“父”包上，这样该模板包将不能与其他包同时起作用。

F.4.3 要传送的状态的数量

在设计事件包时，重点要考虑要在通知中传送的消息类型。

一个很自然的考虑是只传送事件（例如：“收到新的语音消息”）而不传送相关的状态（例如：“共 7 个语音消息”）。这样使订阅实体的实现复杂化（因为这样它们必须维护所订制的实体的完整状态），而且易产生同步问题。

事件包可以选用两种解决方案。

F.4.3.1 全部的状态信息

包一般传送的状态信息很少（1kbit/s 左右），因此可以设计事件包使得当事件发生时发送全部的状态信息。

在有些情况下，仅传送当前状态信息对于特殊类型的事件来说可能是不够的。在这些情况下，事件包应在包含发生的事件同时包含全部的状态信息。例如，传送“没有客户服务表示可用“可能没有传送“没有客户服务表示可用；表示层 sip：46@cs.xyz.int 已注销”有用。

F.4.3.2 状态差异

当要传送的状态信息很大时，事件包可以选择采用细化的方案，在 NOTIFY 消息中包含状态差异而非全部的状态信息。

该方案可如下操作：在立即响应 SUBSCRIBE 的 NOTIFY 消息中包含所有的状态信息；由于状态改变而发送的 NOTIFY 消息仅包含改变的状态信息，订阅者将该信息合并到和资源的状态相关的当前证实中。

任何支持状态差异的事件包必须包含版本号，并且对于订阅中的每个 NOTIFY 事务处理该值递增。版本号包含在消息的消息体中，而不在 SIP 的头部。

如果收到版本号增加值大于 1 的 NOTIFY 消息，那么订阅者就知道丢失了状态差异；此时它将忽略该消息（但记录版本号，以便于检测消息丢失），并重新发送 SUBSCRIBE 消息强制对方发送包含完全状态的 NOTIFY 消息。

F.4.4 事件包的责任

不要求事件包实现本文中描述的所有行为。通常，某些事件包只用于描述对本为中的行为进行扩展或修改的行为。

F.4.4.1 事件包名

必须要定义用于指示事件包的标记名且必须包含在标记的 IANA 注册中的信息。

F.4.4.2 事件包参数

如果在“Event”头部字段中使用参数来修改事件包的行为，必须明确定义该头部字段的语法和语义。

F.4.4.3 SUBSCRIBE 体

大部分事件包而不是所有的事件包将定义 SUBSCRIBE 方法体的语法和语义，这些消息体通常要修

改、扩充、过滤、终止、和/或设置所请求的该类事件的门限。对消息体事件包的设计者应尽量使用存在的 MIME 类型。

在事件包中必须定义在 SUBSCRIBE 请求中包含的事件体的类型（或者规定不包含事件体）。对于引用的所有事件体类型应规定详细的语法和语义。

F.4.4.4 订阅持续时间

建议事件包给出订阅持续时间

F.4.4.5 NOTIFY 消息体

NOTIFY 消息体用于报告所监视的资源的状态。每个事件包必须定义在 NOTIFY 请求中包含的事件体的类型。该事件包必须对和该事件体相关的语法和语义进行详细规定或者是引用详细的规定。

事件包必须定义当 SUBSCRIBE 请求的“Accept”头部字段中没有规定 MIME 类型时所采用的默认 MIME 类型。

F.4.4.6 处理 SUBSCRIBE 请求的通知者

事件包中必须包含对通知者收到 SUBSCRIBE 请求时执行处理的描述。

包括如何对订阅者进行鉴权和对包进行认证的问题。例如，认证问题可以包括是否对该包的所有 SUBSCRIBE 请求都用 202 进行响应。

F.4.4.7 通知者产生 NOTIFY 请求

在事件包中要求包含对通知者如何产生并发送 NOTIFY 请求的描述，包括什么事件引起 NOTIFY 的发送，如何计算 NOTIFY 中的状态信息，如何产生中间的或者虚假的状态信息以便隐藏认证延迟和来自用户的判定，是否包含完整的状态信息还是包含状态差异信息。

事件包可以任选对处理后续响应的行为进行描述。

F.4.4.8 订阅者处理 NOTIFY 请求

事件包要描述当订阅者收到 NOTIFY 请求时执行的处理，包括形成相关资源状态要求的所有逻辑（如果可应用）。

F.4.4.9 分支请求的处理

每个事件包必须规定是否允许分支的 SUBSCRIBE 请求设置多个订阅。

如果不允许该行为，第一个隐含创建对话的消息将创建对话。对于所有的后续 NOTIFY 消息，该消息和 SUBSCRIBE 消息相匹配，但和对话不匹配，那么应用 481 响应拒绝该消息。有可能在收到匹配的 NOTIFY 消息之后收到 SUBSCRIBE 的 200 类响应，该响应可能不与 NOTIFY 建立的对话相关联。除了要完成 SUBSCRIBE 事务处理，该响应将被忽略。

如果允许使用分支的 SUBSCRIBE 设置多个订阅，那么订阅者可以通过对每个 NOTIFY 消息返回 200 类响应来建立到每个通知者的新的对话。每个对话处理自己所属的实体并且独立于其他对话单独进行刷新。

在允许多个订阅的情况下，事件包必须规定是否要求将多个通知合并成单独一个状态，并且要规定如何进行合并。有些事件包可能将所有对话定义为互不关联，此时必须明确申明。

F.4.4.10 通知频率

每个事件包应定义允许单个通知者产生通知频率的最大值。

每个事件包可以进一步定义允许订阅者进一步限制通知频率的终止机制。

F.4.4.11 状态代理

事件包的设计者应考虑该事件包是否能从网络汇聚点（状态代理）和/或代表其他实体的节点中获得信息（例如：当资源本身不能或不愿意提供状态信息时提供资源状态信息的实体）。该应用的一个实例是实体跟踪网络中用户在线和可用状态。

如果事件包使用状态代理，那么该事件包必须规定该状态代理如何产生信息，如何提供鉴权和认证。

事件包也可以简单规定何时进行通知者转移。

F.4.4.12 示例

事件包应包含几个用典型的、语法正确的、完整的消息进行说明的消息流图，

建议在描述事件包文档中明确指出该示例仅最为参考性示例。

F.4.4.13 使用 URI 获得状态

某些类型事件包定义的状态信息可能很大而不适合在 SIP 消息中传送。要解决这个问题，事件包可以定义传送 URI 来代理传送状态信息；然后利用该 URI 来获得实际的状态信息。

F.5 安全考虑

F.5.1 接入控制

接受订阅的能力应该由订阅者的用户直接进行控制，因为许多类型的事件都和隐私有关。同样通知者使用标准的 SIP 鉴权机制，应该能够根据订阅者标识（基于接入控制列表）有选择地拒绝订阅。

F.5.2 通知者保密机制

在某些情况下，对 SUBSCRIBE 请求仅返回 200 或某些 4xx 和 6xx 类响应，可能会暴露策略信息。在这些情况下，通知者应总是回送响应 202。虽然后续的 NOTIFY 消息可能不会传送真正的状态，但是它必须包含订阅者可以隐含获取的正确的数据，该数据不能从有效的响应中区分出。在这种情况下，关于用户是否有权订阅请求状态的信息永远不会回送到发送 SUBSCRIBE 的用户。

能够实现该操作模式的事件包及相关文档可以进一步描述如何和为什么产生隐蔽的正确数据。

F.5.3 拒绝业务攻击

当前模式对于增强的实体来说是一个可在 smurf 攻击中使用的典型模式(一个 SUBSCRIBE 请求触发一个 SUBSCRIBE 响应和一个或多个 NOTIFY 请求)。

当收到 SUBSCRIBE 请求时攻击者可能使用创建的状态来消除受害者机器上的资源，使资源看来不可用。

要减少这样的功绩，通知者应要求鉴权。

F.5.4 重发攻击

重发 SUBSCRIBE 或 NOTIFY 可能产生有害的影响。

使用 SUBSCRIBE 消息，攻击者可能设置任意多个它以前曾经设置过的订阅。重发 NOTIFY 消息可用于伪造过时的状态信息(虽然 NOTIFY 消息体的版本号可以帮助减轻这类攻击)。禁止发送 NOTIFY 消息到没有订阅事件的实体也可以帮助减轻这类攻击。

要防止这类攻击，实现时应利用反重发保护机制进行鉴权。

F.5.5 订阅设置后的攻击

即使有鉴权机制，使用 SUBSCRIBE 在订阅设置之后的攻击可用来设置任意多个订阅、冒充现有的

订阅、终止现有的订阅、或修改订阅作用的资源。要防止这种类型的攻击，实现时至少应通过 SUBSCRIBE 消息中的“Contact”、“Route”、“Expires”、“Event”和 “To”头部字段提供一套完整的保护机制。如果 SUBSCRIBE 体用于进一步定义和呼叫状态有关的信息，那么该消息体应包括在完整的保护机制中。

订阅设置后的攻击也可以使用 NOTIFY 消息冒充任意的状态信息或/和终止现有的连接。要防止这类攻击，实现时应通过“Call-ID”、“Cseq”和“Subscription-State”头部字段以及 NOTIFY 消息体提供一套完整的保护机制。

F.5.6 保密性

NOTIFY 消息中的状态信息很可能包含敏感信息。实现时可以对这些信息进行加密。

虽然很少发生，但是 SUBSCRIBE 消息也可能包含用户不希望暴露的信息。实现时可以对这些信息进行加密。

为了允许远端用户隐藏它认为敏感的信息，所有的实现应能够处理加密的 SUBSCRIBE 消息和 NOTIFY 消息。

F.6 语法

本节描述事件通知要求的语法扩展。形式语法定义使用 ABNF 格式。

F.6.1 新方法

本文定义了两个新的 SIP 方法：SUBSCRIBE 和 NOTIF。

表 F.1 头字段汇总

头部字段	位 置	SUBSCRIBE	NOTIFY
Accept	R	o	o
Accept	2xx	—	—
Accept	415	o	o
Accept-Encoding	R	o	o
Accept-Encoding	2xx	—	—
Accept-Encoding	415	o	o
Accept-Language	R	o	o
Accept-Language	2xx	—	—
Accept-Language	415	o	o
Alert-Info	R	—	—
Alert-Info	180	—	—
Allow	R	o	o
Allow	2xx	o	o
Allow	r	o	o
Allow	405	m	m
Authentication-Info	2xx	o	o
Authorization	R	o	o
Call-ID	c	m	m
Contact	R	m	m
Contact	1xx	o	o
Contact	2xx	m	o

表 F.1（续）

头部字段	位　置	SUBSCRIBE	NOTIFY
Contact	3xx	m	m
Contact	485	o	o
Content-Disposition		o	o
Content-Encoding		o	o
Content-Language		o	o
Content-Length		t	t
Content-Type		*	*
CSeq	c	m	m
Date		o	o
Error-Info	300~699	o	o
Expires		o	—
Expires	2xx	m	—
From	c	m	m
In-Reply-To	R	—	—
Max-Forwards	R	m	m
Min-Expires	423	m	—
MIME-Version		o	o
Organization		o	—
Priority	R	o	—
Proxy-Authenticate	407	m	m
Proxy-Authorization	R	o	o
Proxy-Require	R	o	o
RAck	R	—	—
Record-Route	R	o	o
Record-Route	2xx，401，484	o	o
Reply-To		—	—
Require		o	o
Retry-After	404，413，480，486 500，503，600，603	o	o
Route	R	c	c
RSeq	1xx	o	o
Server	r	o	o
Subject	R	—	—
Supported	R	o	o
Supported	2xx	o	o
Timestamp		o	o
To	c（1）	m	m
Unsupported	420	o	o
User-Agent		o	o
Via	c	m	m

表 F.1（续）

头部字段	位　置	SUBSCRIBE	NOTIFY
Warning	R	—	o
Warning	r	o	o
WWW-Authenticate	401	m	m

F.6.1.1 SUBSCRIBE 方法

SUBSCRIBE 是 SIP 中新增加的方法。

和所有的 SIP 方法名相同，SUBSCRIBE 方法名区分大小写。SUBSCRIBE 方法用于请求随后一个事件或一组事件的异步通知。

F.6.1.2 NOTIFY 方法

NOTIFY 是 SIP 中新增加的方法。

NOTIFY 方法用于通知 SIP 实体先前由 SUBSCRIBE 请求的事件已经发生。该方法也可以提供与该事件有关的更进步的详细信息。

F.6.2 新增头部字段

表 F.2 新增头字段汇总

头部字段	位置	代理	ACK	BYE	CAN	INV	OPT	REG	PRA	SUB	NOT
Allow-Events	R		o	o	—	o	o	o	o	o	o
Allow-Events	2xx		—	o	—	o	o	o	o	o	o
Allow-Events	489		—	—	—	—	—	—	—	m	m
Event	R		—	—	—	—	—	—	—	m	m
Subscription-State	R		—	—	—	—	—	—	—	—	m

F.6.2.1 “Event”头部字段

“Event”是新增加在 sip 消息头部中的定义。

为了将响应消息和 NOTIFY 消息与 SUBSCRIBE 消息进行匹配，需要逐字节地比较“Event”头部字段中的事件类型部分，并且需要逐字节地比较“id”参数标记（如果消息中包含该参数）。包含“id”参数的“Event”头部字段永远不会和未包含“id”参数的“Event”头部字段相匹配。但进行比较时不需要考虑其他参数。

例如：“Event：foo；id=1234”和“Event：foo；param=abcd；id=1234”相匹配，但是“Event：foo”（未包含“id”参数）和“Event：Foo； id=1234”不匹配。

本文没有定义事件类型的值。这些值将在单独的事件包种定义，但必须使用 IANA 注册。

事件头部字段中只能列出一个事件类型，不允许在消息中列出多个事件。

F.6.2.2 “Allow-Events”头部字段

“Allow-Events”头部字段是 SIP 协议中新定义的字段。

F.6.2.3 “Subscription-State”头部字段

“Subscription-State”头部字段是 SIP 协议请求头部中新定义的字段。

F.6.3 新的响应代码

F.6.3.1 “202 接受”响应代码

202 响应增加到“Success”头部字段定义中。“202 接受”的含义和 HTTP/1.1 中定义的含义相同。

F.6.3.2 “489 错误事件”响应代码

489事件响应增加到“Client-Error”头部字段定义中。“489 错误事件”用于指示服务器不能理解“Event”头部字段中定义的事件包。

F.6.4 ABNF 定义

对各种新的和修改过的语法单元的 ABNF 定义如下所示。

```
SUBSCRIBEm          = %x53.55.42.53.43.52.49.42.45 ; SUBSCRIBE in caps
NOTIFYm             = %x4E.4F.54.49.46.59 ; NOTIFY in caps
extension-method    = SUBSCRIBEm / NOTIFYm / token
Event               = ( "Event" / "o" ) HCOLON event-type
                        *( SEMI event-param )
event-type          = event-package *( "." event-template )
event-package       = token-nodot
event-template      = token-nodot
token-nodot         = 1*( alphanum / "-" / "!" / "%" / "*"
                        / "_" / "+" / "`" / "'" / "~" )
event-param         = generic-param / ( "id" EQUAL token )

Allow-Events        = ( "Allow-Events" / "u" ) HCOLON event-type
                    *( COMMA event-type )
Subscription-State  = "Subscription-State" HCOLON substate-value
                        *( SEMI subexp-params )
substate-value      = "active" / "pending" / "terminated"
                        / extension-substate
extension-substate  = token
subexp-params       = ( "reason" EQUAL event-reason-value )
                        / ( "expires" EQUAL delta-seconds )
                        / ( "retry-after" EQUAL delta-seconds )
                        / generic-param
event-reason-value  = "deactivated"
                        / "probation"
                        / "rejected"
                        / "timeout"
                        / "giveup"
                        / "noresource"
                        / event-reason-extension
event-reason-extension = token
```

附 录 G
（规范性附录）
SIP INFO 方法

G.1 简介

INFO 消息的目的是沿着 SIP 信令通路携带应用层消息。INFO 方法并不是用来改变 SIP 呼叫的状态，或会话的初始化状态参数。它仅是用于发送通常与会话有关的应用层的可选信息。会话中的信号信息穿过会话后的 SIP 信令通路建立是必要的。这个通路是 SIP 的 re-INVITEs、BYEs 和其他与一个独立会话联系的 SIP 请求所采用的。这允许 SIP 代理服务器接收信令并处理会话中的信令信息。

本附录通过定义新的 INFO 方法提供 SIP 的一个扩展。INFO 方法 将被用于沿着会话信令通路传送呼叫中信令信息。

G.1.1 用法举例

以下是一些 INFO 消息地可能应用场景：

—在 PSTN 网关之间传送 呼叫中 PSTN 信令消息；

—传送 SIP 会话中生成的 DTMF 数字；

—传送 无线信号强度信息以支持无线移动应用；

—传送 计算平衡信息；

—在会话的参加者之间传送 影像或其他的非流信息。

本部分不定义 INFO 方法的应用场景。

G.2 INFO方法

INFO 方法被用于沿着呼叫信令通路进行会话中信令消息间的通信。它并不是用于改变 SIP 呼叫的状态，也不是用于改变被 SIP 会话状态。然而，它提供增加的选项信息可以进一步加强 SIP 的应用程序功能。

INFO 方法的信令通路是呼叫建立之后建立的信令通路。这可以是呼叫方和被呼叫方用户代理之间的直接信令，也可以包括牵涉到呼叫建立和自己增加到初始 INVITE 信息记录路由头部的 SIP 代理服务器的信令通路。

会话中信息能够在 INFO 信息头部或作为一个消息体的一部分来进行传送。消息体和消息头部的定义被用来传送会话中信息不在本部分的范围之内。

没有与 INFO 相关的特别语法语义。语义是从定义给 INFO 用的头部或协议体那里继承来的。INFO 方法没有特别定义的语法和语义。

G.2.1 INFO 请求方法的响应

如果服务器收到一个 INFO 请求，它必须发出一个最终响应。如果 INFO 请求成功的发送到 UAS，UAS 必须发送一个不带消息体的 200 OK 响应。

表 G.1 头字段汇总

头字段	位 置	INFO
Accept	R	O
Accept-Encoding	R	O
Accept-Language	R	O
Allow	200	—
Allow	405	O
Authorization	R	O
Call-ID	Gc	M
Contact	R	O
Contact	1xx	—
Contact	2xx	—
Contact	3xx	—
Contact	485	—
Content-Encoding	E	O
Content-Length	E	O
Content-Type	E	*
Cseq	Gc	M
Date	G	O
Encryption	G	O
Expires	G	O
from	Gc	M
Hide	R	O
Max-Forwards	R	O
Organization	g	O
Priority	R	o
Proxy-Authenticate	407	o
Proxy-Authorization	R	o
Proxy-Require	R	o
Require	R	o
Retry-After	R	—
Retry-After	404, 480, 486	o
Retry-After	503	o
Retry-After	600, 603	o
Response-Key	R	o
Record-Route	R	o
Record-Route	2xx	o
Route	R	o
Server	r	o
Subject	R	o
Timestamp	g	o

表 G.1（续）

头字段	位　置	INFO
To	gc（1）	m
Unsupported	420	o
User-Agent	g	o
Via	gc（2）	m
Warning	r	o
WWW-Authenticate	401	o

如果 INFO 请求与任何现存的呼叫 leg 不匹配，那么 UAS 需要发送一个 481 响应给发送方。

如果一个服务器收到一个能够识别消息体的 INFO 请求，但是它又不知道与 INFO 过程有关的消息体规则，那么这个消息体可以被翻译并显示给用户。如果 INFO 请求包括一个服务器那时不能理解的消息体，服务器必须回应一个 415 不支持的媒体类型消息。那些在 SIP 呼叫状态中或被 SIP 初始化后的会话中完成一个改变的消息体不能被放在同一个 INFO 消息中发送。

其他请求失败（4xx）、服务器失败（5xx）和全局失败（6xx）回应将被送给 INFO 请求发起方。

G.2.3 消息体的内容

INFO 请求将包含一个消息体。

G.2.4 SIP 代理和重定向服务器的的行为

G.2.4.1 代理服务器

除非被申明，代理服务器对 INFO 请求的处理同 BYE。

G.2.4.2 分叉代理服务器

除非被申明，分叉代理服务器对 INFO 请求的处理同 BYE。

G.2.4.3 重定向服务器

除非被申明，重定向服务器对 INFO 请求的处理同 BYE。

G.3　INFO消息体

INFO 消息的目的是在 SIP 用户代理间传送会话中的信息。这类信息一般将被放在消息体中传送。对于消息体的定义或其他任何为 INFO 方法产生的新头部在本文讨论范围之外。

另外，INFO 方法并不定义确保按顺序传送的附加机制。当 CSeq 头部在传送新的 INFO 消息时递增时，就不能被用来决定 INFO 信息的顺序，这是由于在用户代理发送 re-INVITES 或其他 SIP 消息时在 INFO 消息 CSeq 的计数中将会引起错误。

G.4　INFO扩展

以下是在定义利用 INFO 方法时必须考虑的问题。

—被INFO消息传送的消息体的大小必须考虑。消息可能在UDP上传送并且可能重组一个大的消息，消息体应该保持较小。

—有一种可能是 INFO 消息被一个 SIP 代理服务器创建。完成该 INFO 消息中信息的创建过程需要考虑。

—当定义被 INFO 消息传送的消息体时，可能应用到多消息体。

—用 INFO 消息不能影响 SIP 呼叫的状态或相关会话的状态。

—本标准定义的 INFO 扩展不依赖于请求或代理请求头部的应用。

G.5 安全性考虑

如果消息体的内容是私有的，那么端到端的消息体的加密能够阻止未授权的进入去访问它的内容。

本标准正文中的安全同样适用于 INFO 方法。

附 录 H
（规范性附录）
即时消息

H.1 概述

即时消息（IM）定义了一种在一组参与者间实时交互内容的方式。这种内容通常为短文本消息而且不需要存储。IM 与电子邮件不同的是，消息的交互通常能够组成一个实时的对话，该对话中包含许多来回发送的短消息。

目前，IM 通常与 PRESENCE 业务共同提供服务，即用户可以得知朋友在线并向他发送消息。但是提供这种服务的协议通常为私有协议，这给互操作性带来了很大的困难。

本章描述了 MESSAGE 方法用于发送即时消息，这类似于双向寻呼或者手机的短消息交互，即消息之间没有明确的联系，每个即时消息都是相对独立的。这与会话方式不同，会话是有明确的初始和结束的。在 SIP 协议中，IM 会话可以是一个由 INVITE 事务初始化并以 BYE 消息结束的一个会话。

目前的即时消息通信应用了多种模式。比如用户可以发送消息给一个联系人，或者可以邀请一个或多个联系人加入对话中。此时，可以采用寻呼模式来完成发送短消息给某个联系人，或者采用会话模式来扩展对话，即加入某个聊天组。

本章仅讨论寻呼模式的即时消息，对于会话模式的即时消息不在本部分范围之内。

初始化一个对话，以模拟会话的方式在该对话中发送即时消息的方法是不能完全解决即时消息的问题的。因为这会让 MESSAGE 消息必须和 SIP 会话中的其他请求消息以同一个路径传送，即便 MESSAGE 消息可能携带媒体数据而不是信令数据。IM 应用数据一般是非常大的，如果传送路径上没有拥塞控制的话就容易引起网络拥塞。另外，MESSAGE 消息如果在一个已有的对话中传送的话就必须发送到与对话中其他消息一样的目的地。但是 SIP 中某些 IM 端点与信令端点不同。虽然在已有的对话中发送即时消息也是可行的方案，但是本标准不建议仅仅针对发送即时消息而建立会话。

H.2 基本操作

当用户需要发送即时消息给另一个用户时，发送方需要发送一个 MESSAGE 请求消息。该消息中的 Request-URI 一般为接收方的记录地址，或者接收方的当前位置的设备地址。例如，用户在 Presence 系统中使用即时消息，该系统提供某个给定的记录地址的所有的当前位置信息。用户构建 MESSAGE 消息时，其消息体部分包括需要发送的内容。该消息体可以是任何的 MIME 类型，包括 message/cpim。由于 message/cpim 格式是其他即时消息协议所支持的，采用不同的即时消息协议的终端可以在网关或者其他的中介设备不修改消息内容的情况下交换消息，这样可以增强采用不同的即时消息协议的用户的端到端的安全性。

该消息在到达目的之前可以穿越一组 SIP 代理，并使用不同的传输协议。消息的下一跳地址解析见 CPIM 协议或者本标准正文部分。在消息的穿越中，每个代理都可以根据可用的路由信息来重写消息 URI。

该消息的临时响应和最终响应需要发送给请求方，这与其他的 SIP 请求消息一样。通常，200OK 响应可以由消息的最终接收者的 UA 来发送。

MESSAGE 消息不能创建对话。

H.3 UAC对消息的处理

MESSAGE 和其响应应该与其他消息的构造方法基本相同。

所有支持 MESSAGE 消息的 UAC 都应可以发送带有文本的消息体的 MESSAGE 消息，消息体类型为 message/cpim。

MESSAGE 请求不能初始化对话。UA 不可以在 MESSAGE 消息中插入 Contact 字段。

UAC 可以在已有的对话中发送 MESSAGE 消息，如果 MESSAGE 消息在对话中发送，它与该对话相关的媒体会话或者会话关联。

如果 UAC 收到 200OK 响应，它可以假定该 MESSAGE 消息已经成功发送到对方了。它不能确定接收方是否已经读到了该即时消息。如果 UAC 收到了 202 响应，则消息已经成功发送到网关并存储转发到服务器或者其他可以将消息发送到最终用户的服务上。此时，UAC 不能确定消息是否已经发送到对方。当收到 202 响应之后确认对方是否收到消息的机制不在本标准范围之内。

下游的代理可以分叉复制一个 MESSAGE 消息，此时，该分叉代理会向上转发一个成功的响应，即使它收到了多个响应。UAC 无法检测是否有分叉代理消息。因此 UAC 不能确定某个最终响应是否代表 UAS 是否收到了该消息。

UAC 可以在消息内容中加入一个 Expires 头字段来限制其有效性。UAC 加入一个非零的 Expires 头字段，它应该在加入一个 Date 头字段表明消息的发送时间。

H.4 即时消息的URI

即时消息的收件箱一般都以 URI 的形式标识，如 “im：user@domain”。该 URI 不是一个具体的地址，最终会被翻译为一个确定的基于协议的 URI。

如果 UA 有一个 IM URI 代表即时消息的地址，则该 URI 必须能被解析为 SIP URI，并在 MESSAGE 消息未发送前，替换消息中 Request-URI 中的 URI。如果 UA 不能解析该 IM URI，它可以将该 IM URI 放于 Request-URI 中让下游的代理或者网关解析。

MESSAGE 消息可以包含发送者和接收者的逻辑标识，该标识置于 From 和 To 头字段中。这些标识必须包含 SIP/SIPS URI，但在构造消息时 SIP URI 未知的情况下，该字段也可以包含 IM URI。Record-Route 和 Route 头字段不能包含 IM URI。这些头字段应该包含明确的 SIP 或者 SIPS URI.

H.5 代理对MESSAGE的处理

对 MESSAGE 消息的路由同其他的消息类似。但是，MESSAGE 消息可能被分叉代理，即消息可以发送到用户可能位于多个地址上。代理设备对 MESSAGE 消息的分叉代理同代理其他的非 INVITE 消息类似。只是即使消息发送的多个目的地都得到了成功的响应，代理设备仍只需要转发一个最终响应。

H.6 UAS 对MESSAGE的处理

UAS 收到 MESSAGE 消息后处理与其他的消息类似。

UAS 收到 MESSAGE 消息后，应该立即发送一个最终响应。但是，UAS 此时不需要向用户显示该消息。即 200OK 响应不代表用户一定看到了消息。

MESSAGE 的 2XX 响应不能包含消息体。UAS 不可以在 2xx 响应中插入 Contact 头字段。

UAS 实际是负责存储转发的实体，它会将消息发送到非 SIP 域，并且应该返回一个 202 响应表示消

息被接收，当不能提供端到端的保证。

4xx 或者 5xx 响应表示消息没有成功转发。6xx 响应表示消息成功转发但是被对方拒绝。

如果 UAS 支持 MESSAGE 消息，则它必须能够接收并处理类型为“text/plain”的消息体，并可以支持“message/cpim”的消息体。

MESSAGE 消息中的 Expires 头字段标识的时间过期之后，消息即过期了。MESSAGE 如果没有 Expires 头字段时不会过期。如果 MESSAGE 请求带有 Expires 和 Date 头字段，UAS 应该将 Expired 作为 Date 字段的增量。如果请求不包括 Date 头字段，UAS 应该将 Expired 字段的值作为 UAS 收到该消息的时间增量。

如果 MESSAGE 消息在 UAS 显示内容给用户之前过期了，UAS 应该基于本地策略来处理消息。该策略可以是：消息不显示并删除，或者消息仍然显示给用户，或者调用其他的策略。如果消息仍然显示，UAS 应该明确的告诉用户消息已经过期。

如果 UAS 作为消息的转发者，并且不能在消息过期之前将消息转发出去，则它需要依据本地策略来决定消息的处理。它可以不转发消息并将其删除，或者将消息原样转发，或者依据其他的本次策略来处理消息。

H.7 拥塞控制

MESSAGE 消息与其他的 SIP 请求消息不同，MESSAGE 消息会携带媒体负载，而其他的 SIP 消息只是携带与媒体相关的信令信息。而且目前的即时消息的服务容量都很大。因此，当信令和即时消息共享网络时，即时消息流就会影响信令流的正常传送。拥塞控制应该是在整个 SIP 协议架构中综合考虑的问题。但由于 MESSAGE 消息与其他的消息不同，因此，对于 MESSAGE 消息流的拥塞控制应给予特殊的考虑。

如果有可能，MESSAGE 消息应该在有端到端拥塞控制的传输路径上发送，如 TCP 和 SCTP。但是，SIP 协议中并没有在 UDP 上发送的消息的下游转发保障机制。甚至，在 TCP 上传送某些请求的需求会被接收端重写。因此，UAC 仅仅使用有拥塞控制的传输协议传送 MESSAGE 消息流是不够的。

在媒体会话之外的 MESSAGE 消息的大小不能超过 1300 字节，除非 UAC 知道该消息的任何一跳都是在有拥塞保障的链路上，或者消息尺寸至少比 UAS 要求的 MTU 的最小值小 200 个字节。如果消息净荷更大的话，则该消息可以作为媒体会话的一部分，或者使用某些间接的内容类型。

SIP 协议完成选择消息长度是否比 MTU 小 200 字节，或在 MTU 长度未知的情况下限制消息小于 1300 字节，而传输策略是基于每跳的。但是，这种方式的限制就是没有上游的代理设备会在 UDP 上发送大的 MESSAGE 请求消息。由于 MESSAGE 客户端通常不知道上游设备的 MTU 限制，因此，这种基于 MTU 限制的方式某些情况下无法使用。

如果在对话外某个 URI 上已有一个未决的事务，则 UAC 不能初始化一个新的对话外的 MESSAGE 事务到该 URI 上。同样地，UAC 不能在一个对话中发起一个重叠的 MESSAGE 事务，除非该对话的路由集中每一跳都在一个有拥塞控制的路径上。

如果限制重叠的 MESSAGE 消息，可以在一定程度上保证网络不发生拥塞。请求和它响应的响应必须通过 UAC 和 UAS 间的同一个路径。因此，这所需要的时间可能会带来一定的网络拥塞。所以，这种限制可以减缓向相同的目的地发送新消息的速度。

本部分建议，对于寻呼模式的 MESSAGE 请求不使用临时响应。但由于某些代理并不能识别

MESSAGE 消息，因此这种方式有些情况下无法使用。所以，MESSAGE 消息仍然会向其他的非 INVITE 消息一样收到临时响应。

H.8 消息定义

MESSAGE 消息的 BNF 定义如下：

MESSAGEm = %x4D.45.53.53.41.47.45 ;MESSAGE in caps

表 H1 定义了可以用于 MESSAGE 消息请求和响应中的头字段。

表 H.1 头字段汇总

头字段	位 置	代 理	消 息
Accept	R		—
Accept	2xx		—
Accept	415		m*
Accept-Encoding	R		—
Accept-Encoding	2xx		—
Accept-Encoding	415		m*
Accept-Language	R		—
Accept-Language	2xx		—
Accept-Language	415		m*
Alert-Info	R		—
Alert-Info	180		—
Allow	R		o
Allow	2xx		o
Allow	r		o
Allow	405		m
Authentication-Info	2xx		o
Authorization	R		o
Call-ID	c	r	m
Call-Info		ar	o
Contact	R		—
Contact	1xx		—
Contact	2xx		—
Contact	3xx		o
Contact	485		o
Content-Disposition			o
Content-Encoding			o
Content-Language			o
Content-Length		ar	t
Content-Type			*
CSeq	c	r	m

表 H.1（续）

头字段	位 置	代 理	消 息
Date		a	o
Error-Info	300~699	a	o
Expires			o
From	c	r	m
In-Reply-To	R		o
Max-Forwards	R	amr	m
Organization		ar	o
Priority	R	ar	o
Proxy-Authenticate	407	ar	m
Proxy-Authenticate	401	ar	o
Proxy-Authorization	R	dr	o
Proxy-Require	R	ar	o
Record-Route		ar	—
Reply-To			o
Require		ar	c
Retry-After	404，413，480，486		o
	500，503，		o
	600，603		o
Route	R	adr	o
Server	r		o
Subject	R		o
Timestamp			o
To	c（1）	r	m
Unsupported	420		o
User-Agent			o
Via	R	amr	m
Via	rc	dr	m
Warning	r		o
WWW-Authenticate	401	ar	m
WWW-Authenticate	407	ar	o

MESSAGE 消息可以包括一个消息体，该消息体的类型由 MIME 头字段指定。

H.9 安全

通常，SIP 消息用于建立或者修改通信会话，而实际的通信是以媒体会话开始的，并不是以 SIP 请求开始的。MESSAGE 方法有所不同，MESSAGE 请求一般携带通信双方的通信净荷。因此 MESSAGE 消息比其他的 SIP 消息需要更多的安全保证。特别是 UA 支持 MESSAGE 消息时，它必须提供端到端的鉴权、消息体完整和机密保证机制。

H.9.1 外发鉴权

当本地代理用于外发消息时，本部分建议代理需要使用鉴权机制。这可以对消息发起方进行有效的校验并阻止来自于发起方网络的欺骗和垃圾。

H.9.2 SIPS URIs

SIPS URI 机制可以使 UA 保证每一跳都在一条安全的连接上，这在一定程度上保证了消息的完整性和私密性。但是，用户必须相信消息路径上每一个代理设备都是友好的，即代理会遵循 SIPS URI 的规则。同时，代理可以看到所有未加密的消息体内容。

另外，不需要知道 UAC 的任何信息，分叉代理设备也可以将 MESSAGE 消息发送到其他的端点上。如果使用了端到端的保证机制，用户必须相信路径上的代理设备不会将其消息转发到其他的未授权的端点上去。

H.9.3 端到端的保护

为了保护 MESSAGE 消息体的安全，MESSAGE 消息中的消息体必须采用 S/MIME 的方式保护。如果 MESSAGE 消息中携带的消息体采用其他的方式提供端到端的保护，则需要有后续的规范说明。SIP MESSAGE 的端点必须支持加密和 S/MIME 签名。

H.9.4 重放保护

为了防止 SIP 消息的重放，所有 MESSEGE 消息和响应都必须包含一个有着消息签名的 Date 头字段。MESSAGE 消息中的 Date 头字段的值如果比当前时间早几分钟或者比当前时间晚，则该消息应该以 400 响应。如果消息重复地从同一个源地址发送过来，则消息不需要响应直接丢弃。上述抵御重放攻击的方式不适用于无时钟的设备。

有些情况下，过期的 Date 也是正常的。例如，在用户离线的时候，MESSAGE 请求以存储转发的方式暂时存储在服务器上。当用户上线时，服务器将消息发送给接收者。

如果 UAS 收到一个过期的消息，该消息被证实来自于一个已知的存储转发服务器，则 UAS 应将此消息作为正常的消息接收。另外，如果有一个或者多个过期的 MESSAGE 消息在用户离线期间发送到 UAS，则 UAS 应该接收该消息，但是必须要通知用户该消息可能为一个重放消息。

H.9.5 使用 message/cpim 消息体

使用 message/cpim 格式可以利用 S/MIME 保护元数据和消息净荷。在很多情况下，元数据在 SIP 头字段中是多余的，但是 message/cpim 仍然在元数据中添加值在协议内容之外保护元数据。例如，一个签名的 message/cpim 消息体通过 From 头字段来向发送方提供鉴权，即使消息经过某个网关发送到其他的兼容即时消息的 CPIM 中。

H.9.6 IANA 定义

该方法的目录为：

http：//www.iana.org/assignments/sip-parameters/Method registry。

附 录 I
（规范性附录）
UPDATE 消息

I.1 概述

SIP 协议中定义了 INVITE 方法来初始化或者修改一个会话。但是 Invite 消息会影响到会话中已经建立的媒体流和对话的状态。因此，某些应用场景中，用 INVITE 消息更改会话会带来一些复杂的问题。某些会话在初始的 INVITE 消息得到响应之前需要更改会话中的某些部分，此时，用 INVITE 消息来完成会话的更改就比较困难。如“早媒体”（early media）的情况，即会话在 INVITE 消息得到响应之前已经建立。在这种情况下，在呼叫建立前呼叫双方是否能够修改会话参数是非常重要的。但是，re-INVITE 消息不能完成该会话的更改，因为它无法作用本次对话和会话。

在初始化 INVITE 未得到正确的响应时，呼叫双方可以使用 UPDATE 消息来完成会话的更改。UPDATE 消息可以由对话中的某一个 UA 发送，在不影响对话状态的情况下来更改会话参数。

UPDATE 方法的具体操作如下：

呼叫方发起一个 INVITE 消息来初始一个正常的会话。当对话建立之后（无论是提前建立或者得到确认之后建立），呼叫方都可以发送一个包含 SDP 协商的 UPDATE 消息来更改会话。该 UPDATE 消息的响应包括 SDP 协商的结果。同样，一旦对话建立，被叫方可以发送一个包含 SDP 的 UPDATE 消息，主叫方会在该消息的 2XX 响应中包含 SDP 的协商结果。Allow 头字段中可以指定呼叫方是否支持 UPDATE 消息。对于 UPDATE 消息使用的局限性，是根据 SDP 提供/响应模型来决定的。

I.2 如何支持UPDATE扩展消息

会话是由 INVITE 消息进行初始化的。但是，如果 UAC 支持 UPDATE 消息，需要在其 INVITE 消息的 Allow 头字段中列出 UPDATE 来表示自己能够接收 UPDATE 消息。

如果 UAS 支持 UPDATE 消息，当它收到一个新的对话的 INVITE 消息时，UAS 需要产生一个包含 SDP 的可靠的临时响应来应答该 INVITE 消息，同时，在该响应中还必须包含一个带有 UPDATE 的 Allow 头字段。这表示被叫方可以接收 UPDATE 消息。非可靠的临时响应中可以包括 Allow 字段列出 UPDATE 消息，2XX 响应中必须包括 Allow 头字段来指示是否支持 UPDATE 消息。

响应的处理和可靠性临时响应的处理见本标准的正文部分。但是此处需要指出的是，可靠的临时响应总是创建一个 UAC 的“早对话（early dialog）”。该对话的创建对于接收来自被叫的 UPDATE 请求消息是必要的。如果该响应中包括一个带有“UPDATE”的 Allow 头字段，则 UAC 之后的处理见本章 3.1 节。

I.3 UPDATE 消息处理

I.3.1 发送UPDATE消息

UPDATE 消息可以像其他的请求消息一样包含在一个已有的对话中。它可以在提前建立或者已经确认的对话中被发送，可以由主叫也可以是被叫发送。虽然 UPDATE 可以用于一个已经确认的对话中，但是建议此时最好使用 re-INVITE 消息来更改类似的会话。因为 UPDATE 消息需要被立即应答，这取决于用户对于该消息确认的可能性。

UAC 可以在 UPDATE 消息中加入可选的头字段，见表 I.1。

表 I.1 头字段汇总

头字段	位 置	代 理	UPDATE
Accept	R		o
Accept	2xx		o
Accept	415		c
Accept-Encoding	R		o
Accept-Encoding	2xx		o
Accept-Encoding	415		c
Accept-Language	R		o
Accept-Language	2xx		o
Accept-Language	415		c
Alert-Info			-
Allow	R		o
Allow	2xx		o
Allow	r		o
Allow	405		m
Allow-Events	(1)		—
Authentication-Inf	2xx		o
Authorization	R		o
Call-ID	c	r	m
Call-Info	-	ar	o
Contact	R		m
Contact	1xx		o
Contact	2xx		m
Contact	3xx	d	o
Contact	485		o
Content-Dispositio			o
Content-Encoding			o
Content-Language			o
Content-Length		ar	t
Content-Type			*
CSeq	c	r	m
Date		a	o
Error-Info	300~699	a	o
Event	(1)		—
Expires			—
From	c	r	m
In-Reply-To			—
Max-Forwards	R	amr	m
Min-Expires			—
MIME-Version		ar	o
Organization			o

表 I.1（续）

头字段	位　置	代　理	UPDATE
Priority			—
Proxy-Authenticate	407	ar	m
Proxy-Authenticate	401	ar	o
Proxy-Authorization	R	dr	o
Proxy-Require	R	ar	o
RAck	R		—
Record-Route	R	ar	o
Record-Route	2xx，18x	mr	o
Reply-To			—
Require		ar	c
Retry-After	404，413，480，486		o
	500，503		o
	600，603		o
Route	R	adr	c
RSeq	-		—
Server	r		o
Subject	-		—
Subscription-State	-1		—
Supported	R		o
Supported	2xx		o
Timestamp			o
To	c	r	m
Unsupported	420		m
User-Agent			o
Via	R	amr	m
Via	rc	dr	m
Warning	r		o
WWW-Authenticate	401	ar	m
WWW-Authenticate	407	ar	o

UPDATE 消息是一个目标刷新请求消息，即它可以更改对话中的远端目标。如果 UA 在 INVITE 事务正在处理的过程中使用 UPDATE 消息或者响应来更改远端目标，并且该目标为 INVITE 消息的 UAS，此时，必须用 UPDATE 消息中相应字段的值来替换该 INVITE 消息的 2XX 响应中的 Contact 的值。

如何在 SIP 消息中包含提供/应答（Offer/Answer）见本标准正文部分。这些规则用于保证会话状态的

连续性。

对于主叫方：

1）如果 UPDATE 消息在初始化 INVITE 事务结束前发送，并且该 INVITE 消息包含提供，如果被叫在可靠性临时响应中包含应答，并且主叫已经收到它在 PRACK 或者 UPDATE 消息中提供的应答，并且也已经在收到的 UPDATE 消息中提供产生应答时，则 UPDATE 中也可以包含提供。

2）如果 UPDATE 消息在初始化 INVITE 事务完成之后发送，而且主叫已经发送或者收到 re-INVITE 消息或者 UPDATE 消息包含提供，但该消息还未得到应答时，UPDATE 消息中不能包括提供。

对于被叫方：

1）如果 UPDATE 消息在 INVITE 事务完成之前被发送，并且初始化 INVITE 消息中包含提供，则 UPDATE 消息就不能带着提供发送，除非以下情况：被叫已经在一个可靠的临时响应中包含了提供的应答发送；或者收到了一个 PRACK 消息作为可靠的临时响应的应答；或者没有收到任何带有提供的 PRACK 或者 UPDATE 消息，该消息仍未应答；或者没有发送任何包含提供的 UPDATE 消息，该消息未应答。

2）如果 UPDATE 消息在 INVITE 事务结束前发送，并且 INVITE 消息中没有包含提供，则 UPDATE 消息不可以带着提供发送，除非下列情况：被叫已经在一个可靠的临时响应中包括提供；或者收到一个带有应答的 PRACK 消息；或者没有收到任何带着提供的 UPDATE 请求，该消息未被响应；或者没有发送任何带有提供的 UPDATE 消息，该消息未响应。

3）如果 UPDATE 消息在初始化 INVITE 事务结束后被发送，而且被叫已经发送或者接收一个带有提供的 re-INVITE 消息或者 UPDATE 消息（未应答），则该 UPDATE 消息不能包含提供发送。

I.4 接收UPDATE消息

UPDATE 消息可以作为任何的对话中的目的刷新请求来处理，参加正文 10.2 节。如果该请求被接受，则按照下文所述进行处理。本章中所述内容仅适用于 UPDATE 消息的处理。

UAS 在还未处理完上一个 UPDATE 消息并给出最终响应之前，如果接收到新的对于同一个对话的 UPDATE 消息时，它必须对其返回一个 500 响应，而且在该响应中必须包含一个 Retry-After 头字段，该字段的值为 0~10s 中的任意数值。

如果 UAS 收到了一个包含提供的 UPDATE 消息，并且 UAS 已经在 UPDATE/PRACK/INVITE 消息中携带提供但是却未得到任何应答，则 UAS 应该以 491 响应拒绝该 UPDATE 消息。同样，如果 UAS 收到一个携带提供的 UPDATE 消息，但是之前 UAS 已经收到了由 UPDATE/PRACK/INVITE 中携带的提供，但是还未对其产生响应，则 UAS 应以 500 响应拒绝该 UPDATE 消息，在该响应中应包含一个 Retry-After 头字段，该字段的值为 0~10s 中的任意数值。

如果 UA 接收到一个已有对话的 UPDATE 消息，它必须检查该消息的会话描述中的版本标识，如果没有任何版本标识，则需要检查会话描述的内容是否改变。如果会话描述部分已经改变，UAS 必须修改相应的会话参数，并以一个 2xx 消息响应。但是，UPDATE 消息必须立即响应，但是用户不能立即决定是否更改会话。如果 UAS 不能更改会话参数，它就需要以 504 响应拒绝请求。如果一个新的会话描述没有被接受，UAS 就会以 488 响应拒绝 UPDATE 消息。这个响应包含一个 Warning 头字段。

I.5 UPDATE 的处理

UPDATE 消息的处理依据本标准 10.2.1.2 中关于目的刷新请求的处理。一旦上述处理结束，该消息的

处理依据下述内容进行。该过程类似于 12.1 节所述，本章中所述内容仅适用于 UPDATE 消息。

如果 UA 接收到一个 UPDATE 的非 2xx 的响应，则该消息的会话参数不可以被改变，就像没有任何 UPDATE 消息发送一样。如果非 2xx 的最终响应为 481 或者 408，或者对于该 UPDATE 消息没有任何的响应，此时 UAC 应该终止对话。

如果 UAC 收到了一个 UPDATE 消息的 491 响应，它应该初始化一个计时器，其定时值 T 应该按如下规则选择：

1）如果 UAC 为该对话 ID 的 Call-ID 的所有者，即由它生成该值，则 T 的取值范围为 2.1~4s，时间间隔为 10ms。

2）如果 UAC 不是该对话 ID 的 Call-ID 的所有者，则 T 的取值范围为 0~2s，时间间隔为 10ms。

当定时器超时，UAC 如果想更改会话参数，它应该再次尝试发送 UPDATE 消息。

I.6 代理对UPDATE消息的处理

代理对于 UPDATE 消息的处理同其他非 INVITE 消息相同。

I.7 UPDATE消息定义

UPDATE 扩展在 BNF 方法中增加了新的值：

```
UPDATEm   =  %x55.50.44.41.54.45 ; UPDATE in caps
Method    =  INVITEm / ACKm / OPTIONSm / BYEm
             / CANCELm / REGISTERm / UPDATEm
             / extension-method
```

I.8 安全

UPDATE 消息的安全与 re-INVITE 消息的安全定义一样。但是 UPDATE 消息无论从相同的源端发送还是从对话的对端发送，都需要完整性保护和鉴权。

I.9 IANA 定义

Method Name：UPDATE

Reason Phrase：Not applicable.

附 录 J
（规范性附录）
REFER 方法

J.1 概述

REFER 方法是 SIP 协议的一个扩展方法。该方法可以实现将消息接收者转移到另外的资源上去。该位置由消息中的头字段指定。

使用 REFER 方法可以完成许多应用，如呼叫转移。

J.2 REFER 方法

REFER 用来指示接收方（以 Request-URI 标识）应该使用消息中的信息联系第三方参与呼叫。

除非特殊声明，否则发送和响应 REFER 消息的机制与 BYE 相同。不能执行 REFER 方法的 SIP 实体在正文指定。

REFER 消息可以在由 INVITE 消息创建的对话之外发送。REFER 消息会创建一个对话，而且按照记录路由来转发，因此它必须包括一个单独的 Contact 头字段。在已建立的对话中 REFER 消息必须遵循 Route/Record-Route 来发送。

J.2.1 Refer-To Header 头字段

Refer-To 仅在 REFER 请求中存在，用来指示呼叫转移的一方。

Refer-To = ("Refer-To" / "r") HCOLON (name-addr / addr-spec) *
(SEMI generic-param)

表 J.1 Refer-To 头字段定义

头字段	位置	代理	ACK	BYE	CAN	INV	OPT	REG
Refer-To	R			—	—	—	—	—

Refer-To 头字段可以按照端到端来进行加密。

Contact 头字段是 Route/Record-Route 机制中最重要的部分，它不指定呼叫转移的目的。

J.2.2 REFER 方法使用的头字段

表 J.2 为在 REFER 方法中的头字段。

表 J.2 头字段汇总

头字段	位 置	REFER
Accept	R	o
Accept	2xx	—
Accept	415	c
Accept-Encoding	R	o
Accept-Encoding	2xx	—
Accept-Encoding	415	c
Accept-Language	R	o
Accept-Language	2xx	—

表 J.2（续）

头字段	位　置	REFER
Accept-Language	415	c
Alert-Info		—
Allow	Rr	o
Allow	405	m
Authentication-Info	2xx	o
Authorization	R	o
Call-ID	c	m
Call-Info		—
Contact	R	m
Contact	1xx	—
Contact	2xx	m
Contact	3~6xx	o
Content-Disposition		o
Content-Encoding		o
Content-Language		o
Content-Length		o
Content-Type		*
CSeq	c	m
Date		o
Error-Info	3~6xx	o
Expires	R	o
From	c	m
In-Reply-To		—
Max-Forwards	R	m
Min-Expires		—
MIME-Version		o
Organization		o
Priority	R	—
Proxy-Authenticate	401	o
Proxy-Authenticate	407	m
Proxy-Authorization	R	o
Proxy-Require	R	o
Record-Route	R	o
Record-Route	2xx，18x	o
Reply-To		—
Require		c
Retry-After	404，413，480，486	o

表 J.2（续）

头字段	位　置	REFER
Retry-After	500，503	o
Retry-After	600，603	o
Route	R	c
Server	r	o
Subject	R	—
Supported	R，2xx	o
Timestamp		o
To	c（1）	m
Unsupported	420	o
User-Agent		o
Via	c（2）	m
Warning	r	o
WWW-Authenticate	401	m
WWW-Authenticate	407	o

J.2.3　消息体内容

REFER 方法可以包含消息体。消息体的定义不在本标准的范围之内。接收方可以按照消息的内容类型来处理消息。

J.2.4　SIP UA行为

J.2.4.1　构造REFER请求

REFER 是一个 SIP 请求消息。它必须包含一个 Refer-To 头字段。

J.2.4.2　处理REFER请求

当 UA 收到一个构造正确的 REFER 消息之后，它需要得到用户的认可才可以继续处理该消息。如果请求通过，UA 必须联系消息中的 Refer-To 字段指定的 URI。

如果未得到许可，则该请求应被 UA 立即拒绝。

如果请求中的 Refer-To 字段值为 0 或者包含多个 Refer-To 字段，则 UA 应该对该 REFER 消息回应一个 400 响应。

UA 可以对 REFER 消息回送 100 消息或者正确的 4xx~6xx 消息。

UA 如果不支持非 SIP URI 的处理则不能接收 REFER 消息。

如果按照上述规则没有得到最终响应，UA 必须在 REFER 事务过期之前返回一个 202 响应。

如果 REFER 消息已经接收，即得到明确的 2xx 响应。则接收者必须创建一个订阅事件（Subscription）并发送通知（Notification）上报转移的状态。

J.2.4.3　访问Referred-to指定的资源

联系 Refer-To URI 中指定的资源使用正常的 URI 机制。例如，如果该 URI 为 SIP URI 标志一个 INVITE 消息（method=INVITE URI），UA 应该发送一个新的 INVITE 消息，该消息按照正常的 INVITE 消息规则构造。

J.2.4.4　使用SIP事件上报转移结果

NOTIFY 消息用来上报 UA 关于 REFER 的转移状态。REFER 消息如果已经被一个 SUBSCRIBE 消息订阅之后，NOTIFY 消息中的对话 ID 必须与 RFFER 消息匹配。

每个 NOTIFY 消息必须包含 Event 头字段，该字段含有一个转移的值和可能的 ID 参数。每个 NOTIFY 消息必须包括类型为 message/sipfrag 的消息体。

订阅事件的创建一般会产生一个立即的 NOTIFY 请求。类似 SUBSCRIBE 的事件，发送 REFER 消息的 UA 必须可以在 REFER 事务结束前接收 NOTIFY 消息。

由 REFER 创建的订阅事件与由 SUBSCRIBE 创建的订阅事件相同。发送 REFER 消息的 UA 可以提前终止该订阅事件。无论由明确的退订方式还是通过拒绝 NOTIFY 消息的方式来终止订阅事件，都不表示该转移请求已经被取消或者丢弃了。特别是当 UA 需要对 REFER 消息作出响应的时候，它不能发送一个 CANCEL 消息给任何一个被转移的 SIP 请求，因为在转移请求结束之前 UA 发送 REFER 消息终止一个转移事件的订阅。

UA 发送 REFER 消息可以使用订阅刷新机制来延长自己的订阅事件。

REFER 消息是惟一可以创建转移事件的订阅机制。如果 UA 收到对于转移事件的 SUBSCRIBE 消息，而该订阅并不存在，则 UA 必须以 403 响应拒绝该消息。

和 SUBSCRIBE 消息不同，REFER 事务中，请求和响应都不包含订阅的周期。订阅状态的生存时间取决于转移请求的处理。接收 REFER 消息的 UA 选择订阅周期，并通过订阅事件的初始 NOTIFY 消息将订阅周期发送给 REFER 消息的发送方（使用 Subscription-State expires 参数）。如果接收 REFER 消息的 UA 不希望保持订阅状态可以用 NOTIFTY 消息结束订阅。

J.2.4.5 NOTIFY消息体

每个 NOTIFY 消息必须包含“message/sipfrag”类型的消息体。该消息体必须以 SIP 响应起始行（Response Status-Line）开始。该起始行的响应级别表示转移的状态。该消息体可以包含其他的 SIP 头字段来提供转移结果的信息。该消息体提供一个转移状态的完整声明。转移事件包不支持状态增量。

如果在订阅状态挂起（Pending）的时候产生 NOTIFY 消息，则它的消息体中必须仅由带有 100 状态码的状态行构成。

下面的例子为一个最小的完整的 NOTIFY 响应：

SIP/2.0 100 Trying

如果订阅事件挂起，则消息体为：

SIP/2.0 200 OK

如果转移成功，则消息体为：

SIP/2.0 503 Service Unavailable

如果转移失败，则消息体为：

SIP/2.0 603 Declined

如果 REFER 消息在请求得到许可时被接收，则该许可无效。

实际应用中，SIP 消息的消息体中会包含更多的内容来传递更多的信息。转移事件的响应中的 Warning 头字段包含了某些有用的信息。如果转移的位置为一个 SIP URI，则该转移动作的所有响应可以被返回到该位置，但是这样会给引发一些安全问题。因此，在选择消息体中的内容时需要慎重考虑。

如果转移位置为非 SIP URI，则 NOTIFY 中的状态码必须以 SIP 响应状态行的格式。例如，如果客

户端收到了一个带有HTTP URL的REFER消息,并且已经成功的访问到响应的资源,则该用户的NOTIFY消息可以包含状态码为“SIP/2.0 200 OK”的message/sipfrag消息体。如果通知方希望返回其他的非SIP协议的信息，它可以将其放入该消息体中。

J.2.4.6 对话中的多REFER请求

REFER消息创建了一个隐含的订阅事件，该事件与REFER消息共享一个对话ID。如果在同一个对话中有多个REFER消息发送，则该对话ID无法提供足够的信息来关联NOTIFY消息和相关的REFER消息。

因此，一个已知的对话中，UA收到的第二个REFER消息，它必须在每个NOTIFY消息Event头字段中包含一个ID参数,这些NOTIFY消息中需要包含该消息对应的REFER消息的序列号(CSeq中指定)。第一个REFER的NOTIFY消息中可以包含该ID参数，刷新或者终止该订阅事件的SUBSCRIBE消息必须包含该ID参数。

J.2.4.7 Subscription-State Header头字段

每个NOTIFY消息中必须包含Subscription-State头字段。REFER的最终的NOTIFY响应消息必须指出该订阅事件“终止”(terminated)，原因为“无资源”(noresource)。

如果NOTIFY中指出原因为正确的重订阅事件，则发送REFER消息的UA不需要重订阅。

REFER消息如果得到了202响应,但是订阅事件的许可被拒绝,则Subscription-State头字段中的reason和retry-after部分可以用来指定REFER消息是否可以被重发，并指示何时重试该消息。

J.2.5 注册/重定向服务器行为

如果注册服务器不识别REFER消息，则回送一个501响应；如果它可以识别REFER消息，则需要回送一个405响应。

对于重定向服务器对于REFER消息的处理，本部分不作要求。因此，它应该能够重定向REFER消息。

J.2.6 SIP 代理对REFER消息的处理

SIP代理设备处理REFER消息的机制应该与OPTIONS消息相同。

J.3 包定义：转移事件(Event Refer)

关于包定义见附录F。

J.3.1 事件包名称(Event Package Name)

该事件包的名字为“refer”。

J.3.2 事件包参数(Event Package Parameters)

ID参数定义见附录F。

J.3.3 SUBSCRIBE 消息体

该事件包的SUBSCRIBE 消息体无特殊含义。

J.3.4 Subscription 时长

由REFER消息创建的隐含的订阅事件的时长初始由接收REFER消息的UA决定，并用第一个NOTIFY消息的Subscription-State中的“expire”参数通知订阅的UA。该初始参数的选择依据Refer-To URI中的请求类型。该时长可以设定比转移请求给定的完成时间稍长。例如，如果Refer-To URI是一个SIP

INVITE URI，则订阅间隔应该比 INVITE 里的 Expire 值稍长，多余的时间是考虑到订阅事件的鉴权所需要的时间。订阅 UA 可以通过刷新来延长订阅时间，或者通过退订来终止订阅事件。当订阅事件在 Subscription-State 上报最终的转移结果时，接收 REFER 消息的 UA 可以终止订阅事件。

J.3.5 NOTIFY 消息体

NOTIFY 请求消息体见本附录 2.4.5。

J.3.6 通知方（Notifier）对SUBSCRIBE请求的处理

通知方（Notifier）对 SUBSCRIBE 请求的处理见本附录 2.4.4。

J.3.7 通知方（Notifier）创建NOTIFY请求

通知方（Notifier）创建 NOTIFY 请求见本附录 2.4.4。

J.3.8 订阅方（Subscriber）对NOTIFY请求的处理

订阅方（Subscriber）对 NOTIFY 请求的处理见本附录 2.4.4。

J.3.9 分叉请求的处理

已有的对话中的 REFER 消息不能够被分叉代理。REFER 消息在对话外发送可以被分叉代理。并且如果被多个 UA 接收，它可以创建一个多订阅事件。这些订阅事件每个都作为“分叉请求处理”（Handling of Forked Requests），就像 REFER 消息被订阅一样。发送 REFER 消息的 UA 处理与每个独立的订阅事件相关的状态。它不会将不同的订阅事件的状态进行合并。

J.3.10 Notification频率

转移事件的 NOTIFY 消息可以在每次转移请求可行的时候创建。例如，如果作为 SIP INVITE 的 REFER 消息，NOTIFY 必须和每个 INVITE 的临时响应和最终响应一起产生。另外，订阅事件只可以产生两个 NOTIFY 请求，即立即 NOTIFY 和带有转移最终状态的 NOTIFY。事件的 NOTIFY 消息发送频率不能小于每秒一次。

J.3.11 状态代理（State Agents）

对于转移事件不定义单独的状态代理。

J.4 安全

REFER 的安全策略见正文部分。REFER 消息对于安全的特殊考虑在于其鉴权策略和使用 message/sipfrag 来传送转移请求的结果。

J.4.1 创建 Refer-To URI

该字段用于给被转移方提供转移资源的联系信息。如果转移资源被保护，则应该考虑提供更加合适的严格的 URI。

J.4.2 REFER的鉴权

UA 可以接收带有任何 URI 方案的 Refer-To URI 字段的 REFER 消息。例如，用户可以使用 telnet URI 转移到类似于 MUD 的在线服务上去。用户服务可以使用 HTTP URI 将用户转移到一个 Web 页面上去。本标准允许 UA 在不支持消息的 URI 方案时拒绝 REFER 消息。同时，本部分也要求 UA 在使用 URI 之前必须得到用户的鉴权。当然，URI 可以为任意长度并且有时候会被恶意构造。在 URI 可以显示的时候也要避免这种事情发生。另外，在尽量暴露信息给用户的时候也需要注意减小用户错误决策的风险。例如，Refer-To 头字段可以包含一个显示名称，没有任何机制保证该显示名称的任何属性是与 URI 共享的。

某种情况下，用户在给UA策略之前可以对于REFER消息进行鉴权。例如在一个呼叫转移中，一个正常的鉴权对话中的REFER请求（指向SIP/SIPs/tel：URI）可以根据策略被接受，而不需要和用户进行交互式鉴权。同样地，如果转移方为认可的，则用户可以接收一个到HTTP URI的正常鉴权的REFER消息。在没有预先提供鉴权机制的情况下，用户必须进行交互式的鉴权方式来验证指定资源。

有时盲目地接收并处理HTTP URI是危险的。如某个Web服务器只接收某个企业防火墙后的用户的请求，该企业内部网络只做了很少的安全策略，所以该Web服务器就会造成某种恶意构造URI的攻击。如果SIP UA位于该企业网中，盲目接收来到任何HTTP URI的转移，则防火墙外的攻击者很容易危害Web Server。另外，如果UA的用户在处理URI之前作出正面的动作（如立即回送响应），则该危险就会降低到用户点击某个Web页面上的URI或者电子邮件一样的级别了。

如果UA自动的访问已知方案的URI时，则会给某些在该URI方案上有安全缺陷的主机带来攻击的危险。当主机或者UA位于一个公共的安全策略点（如防火墙），这些风险就会增加。另外，特别是每个自动访问的动作需要打开一个新的连接的时候，UA也增加了自己受到由于资源耗尽的拒绝服务的攻击的风险。

UA在处理任意第三方服务的URI方案时应该小心，尤其是当UA无法识别该URI方案或者可能的协议分支的时候。

J.4.3 message/sipfrag的用法

REFER消息中使用message/sipfrag消息体来返回转移结果和进程。但是使用该消息体时应该注意，因为不当的使用容易威胁消息的机密性和私密性。

J.5 IANA Considerations

本部分定义了一个新的SIP方法——REFER，一个新的SIP头字段——Refer-To（缩写为r）和一个新的事件包——refer。

下述内容已经加入到http：//www.iana.org/assignments/sip-parameters的子目录下：

REFER [RFC3515]

下面的内容已经加入到http：//www.iana.org/assignments/sip-parameters的头字段子目录下：

Header Name：Refer-To

Compact Form：r

Reference：RFC 3515

关于新的refer事件包的注册信息如下：

Package Name：refer

Package or Package-Template：This is a package.

Published Specification：RFC 3515

Person to Contact：Robert Sparks，rsparks@dynamicsoft.com

附 录 K
（资料性附录）
示 例

K.1 对ROUTE头字段的处理

如果没有其他的本地策略，代理服务器对于包含 Route 头字段的请求所执行的处理可被总结为如下几步：

1）检查请求的 Request-URI。如果它指示的是一个由当前代理服务器负责的资源，那么代理服务器将用从定位服务中得到的地址代替它。否则，代理服务器将不对它作改动。

2）代理服务器将检查最顶端 Route 头字段中的 URI。如果它指向当前代理服务器，那么将其从 Route 头字段中删除。

3）代理服务器将把请求转发到最顶端的 Route 头字段值中 URI 所指的资源位置，如果没有 Route 头字段出现则转发到 Request-URI 所指的资源位置。在转发请求时代理服务器通过对目标 URI 运用 RFC3263 中所描述的过程来确定所用的 IP 地址、端口和传输方式。

如果在请求消息的传输路径上没有遇到严格路由的实体，Request-URI 将总是指向请求的目的地。

K.1.1 示例

K.1.1.1 基本SIP梯形

该示例是一个基本的 SIP “梯形”结构，U1→P1→P2→U2、U1 和 U2 表示终端，P1 和 P2 表示代理服务器。两个代理服务器都要插入 record-route 头字段值。这里给出消息流程：

U1 发送请求给 P1：

INVITE sip：callee@domain.com SIP/2.0

Contact：sip：caller@u1.example.com

P1 是一个外拨代理服务器，它不负责 domain.com，所以它进行 DNS 查询并将请求发往那里。它还在请求中加了一个 Record-Route 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 收到了这个请求。它负责 domain.com，因此它运行定位服务并重写 Request-URI。它也加入了一个 Record-Route 头字段值。因为请求中没有 Route 头字段，所以它对新的 Request-URI 进行解析以确定向何处转发请求：

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>

位于 u2.domain.com 的被叫方收到了请求并以 200 OK 作答：

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>

同时，位于U2的被叫将其对话状态信息中的远端目标URI设为sip：caller@u1.example.com，并将其路由集设为：

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

这个200响应沿着P2->P1->U1的路径正常回传。现在，U1把自己对话状态信息中的远端目标URI设为sip：callee@u2.domain.com并将其路由集设为：

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

既然路由集中的所有元素都包含lr参数，U1将构造出如下的BYE请求：

BYE sip：callee@u2.domain.com SIP/2.0

Route：<sip：p1.example.com;lr>，<sip：p2.domain.com;lr>

就像任何其他的实体（包括代理服务器）一样，U1运用DNS解析最顶端Route头字段值中的URI以确定向何处发送请求。这使请求被发往P1。P1注意到自己不负责Request-URI所指的资源位置，因此它不改变其内容。它也确实发现自己是Route头字段中的第一个值，所以它将该值删除，并将请求转发到P2：

BYE sip：callee@u2.domain.com SIP/2.0

Route：<sip：p2.domain.com;lr>

P2也注意到它不负责Request-URI所指的资源（它负责的是domain.com，而不是u2.domain.com），所以它也不改变Request-URI的值。P2也确实发现自己是Route头字段的第一个值，所以把这个值删去，并通过对Request-URI进行DNS查询将如下的请求转发到u2.domain.com：

BYE sip：callee@u2.domain.com SIP/2.0

K.1.1.2 穿越一个严格路由的代理服务器

该示例中，一个对话通过4个代理服务器建立，每个代理服务器都加入了Rocord-Route头字段值。第三个代理服务器实现了严格路由过程，这些过程在RFC2543及本协议先前的诸多演进版本中规定。

U1→P1→P2→P3→P4→U2

到达U2的INVITE请求包含：

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回应以200 OK。之后，U2基于第一个Route头字段值发送如下的BYE请求给P4：

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 不负责 Request-URI 所指的资源，因此保留其值不变。它注意到自己是 Route 头字段的第一个值所指实体，所以将之删除。然后 P4 根据当前的第一个 Route 头字段值 sip：p3.middle.com 准备发送请求，但它注意到这个 URI 中没有 lr 参数，因此在发送之前，它将请求消息变为如下格式：

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 是一个严格路由的代理服务器，它将如下请求转发到 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 是自己以前放到 Record-Route 头字段中的值，所以在进一步处理之前，将请求消息改写为：

BYE sip：caller@u1.example.com SIP/2.0

Route：<sip：p1.example.com;lr>

P2 检查不负责 u1.example.com，所以它根据 Route 头字段值的解析结果将请求消息发往 P1。

P1 注意到自己被列在最顶端的 Route 头字段值中，因而它将该值删除，消息变为：

BYE sip：caller@u1.example.com SIP/2.0

既然 P1 不负责 u1.example.com，并且消息中也没有 Route 头字段，P1 根据 Request-URI 将请求消息转发到 u1.example.com。