SIP

session initiation protocol

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About this Tutorial

SIP is a signalling protocol designed to create, modify, and terminate a multimedia session over the Internet Protocol. It is an application layer protocol that incorporates many elements of the Hypertext Transfer Protocol (HTTP) and the Simple Mail Transfer Protocol (SMTP).

This tutorial covers most of the topics required for a basic understanding of SIP and to get a feel of how it works.

Audience

This tutorial has been prepared for professionals aspiring to learn the basics of SIP and make a career in telecom testing.

Prerequisites

Before proceeding with this tutorial, you should have a good grasp over preliminary networking concepts including some of the basic protocols such as TCP, UDP, HTTP, SMTP, and VoIP.

Copyright & Disclaimer

Table of Contents

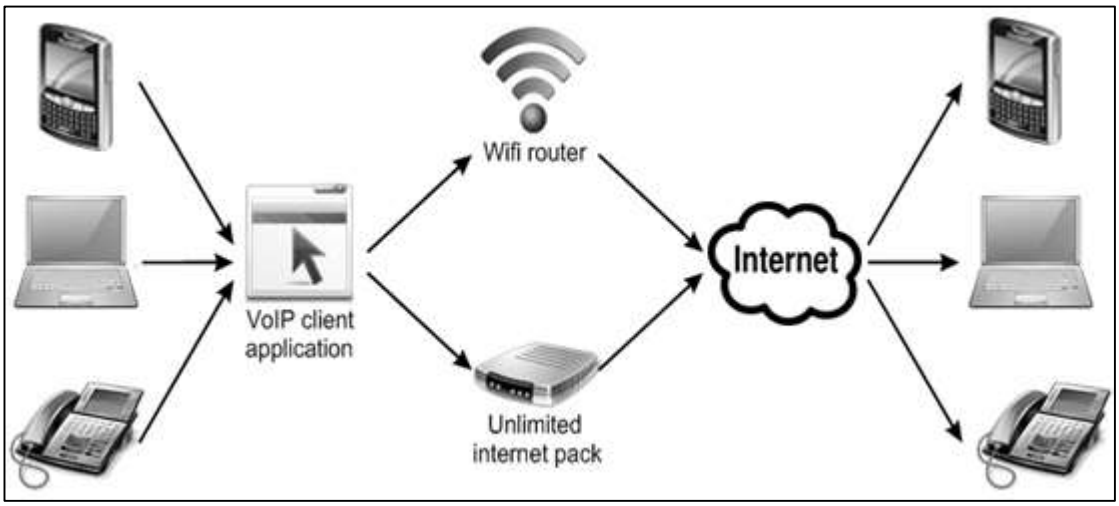
1. Introduction

Session Initiation Protocol (SIP) is one of the most common protocols used in VoIP technology. It is an application layer protocol that works in conjunction with other application layer protocols to control multimedia communication sessions over the Internet.

VoIP Technology

Before moving further, let us first understand a few points about VoIP.

- VOIP is a technology that allows you to deliver voice and multimedia (videos, pictures) content over the Internet. It is one of the cheapest way to communicate anytime, anywhere with the Internet's availability.
- Some advantages of VOIP include:
  - Low cost
  - Portability
  - No extra cables
  - Flexibility
  - Video conferencing
- For a VOIP call, all that you need is a computer/laptop/mobile with internet connectivity. The following figure depicts how a VoIP call takes place.



With this much fundamental, let us get back to SIP.

SIP – Overview

Given below are a few points to note about SIP:

- SIP is a signalling protocol used to create, modify, and terminate a multimedia session over the Internet Protocol. A session is nothing but a simple call between two endpoints. An endpoint can be a smartphone, a laptop, or any device that can receive and send multimedia content over the Internet.

- SIP is an application layer protocol defined by IETF (Internet Engineering Task Force) standard. It is defined in **RFC 3261**.

- SIP embodies client-server architecture and the use of URL and URI from **HTTP** and a text encoding scheme and a header style from **SMTP**.

- SIP takes the help of SDP (Session Description Protocol) which describes a session and RTP (Real Time Transport Protocol) used for delivering voice and video over IP network.

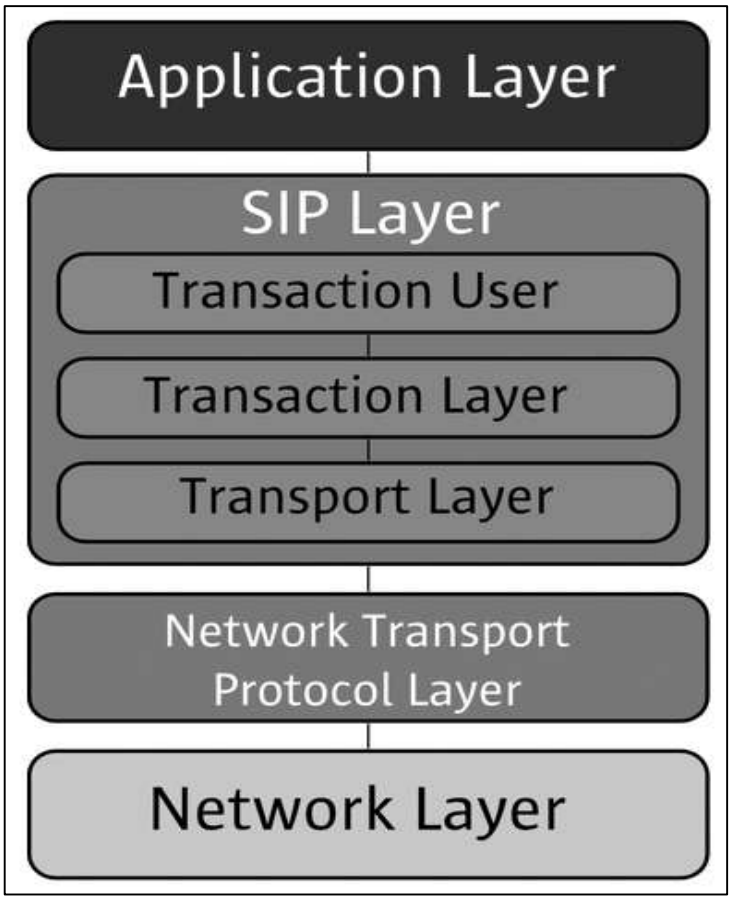
- SIP can be used for two-party (unicast) or multiparty (multicast) sessions.

- Other SIP applications include file transfer, instant messaging, video conferencing, online games, and steaming multimedia distribution.

Where Does SIP Fit In?

Basically SIP is an application layer protocol. It is a simple network signalling protocol for creating and terminating sessions with one or more participants. The SIP protocol is designed to be independent of the underlying transport protocol, so SIP applications can run on TCP, UDP, or other lower-layer networking protocols.

The following illustration depicts where SIP fits in in the general scheme of things:



Typically, the SIP protocol is used for internet telephony and multimedia distribution between two or more endpoints. For example, one person can initiate a telephone call to another person using SIP, or someone may create a conference call with many participants.

The SIP protocol was designed to be very simple, with a limited set of commands. It is also text-based, so anyone can read a SIP message passed between the endpoints in a SIP session.

2. Network Elements

There are some entities that help SIP in creating its network. In SIP, every network element is identified by a **SIP URI** (Uniform Resource Identifier) which is like an address. Following are the network elements:

- User Agent
- Proxy Server
- Registrar Server
- Redirect Server
- Location Server

User Agent

It is the endpoint and one of the most important network elements of a SIP network. An endpoint can initiate, modify, or terminate a session. User agents are the most intelligent device or network element of a SIP network. It could be a softphone, a mobile, or a laptop.

User agents are logically divided into two parts:

- **User Agent Client (UAC):** The entity that sends a request and receives a response.
- **User Agent Server (UAS):** The entity that receives a request and sends a response.

SIP is based on client-server architecture where the caller's phone acts as a client which initiates a call and the callee's phone acts as a server which responds the call.

Proxy Server

It is the network element that takes a request from a user agent and forwards it to another user.

- Basically the role of a proxy server is much like a router.
- It has some intelligence to understand a SIP request and send it ahead with the help of URI.
- A proxy server sits in between two user agents.
- There can be a maximum of 70 proxy servers in between a source and a destination.

There are two types of proxy servers:

- **Stateless Proxy Server:** It simply forwards the message received. This type of server does not store any information of a call or a transaction.

- **Stateful Proxy Server**: This type of proxy server keeps track of every request and response received and can use it in future if required. It can retransmit the request, if there is no response from the other side in time.

Registrar Server

The registrar server accepts registration requests from user agents. It helps users to authenticate themselves within the network. It stores the URI and the location of users in a database to help other SIP servers within the same domain.

Take a look at the following example that shows the process of a SIP Registration.

SIP Registration Example

Here the caller wants to register with the TMC domain. So it sends a REGISTER request to the TMC's Registrar server and the server returns a 200 OK response as it authorized the client.

Redirect Server

The redirect server receives requests and looks up the intended recipient of the request in the location database created by the registrar.

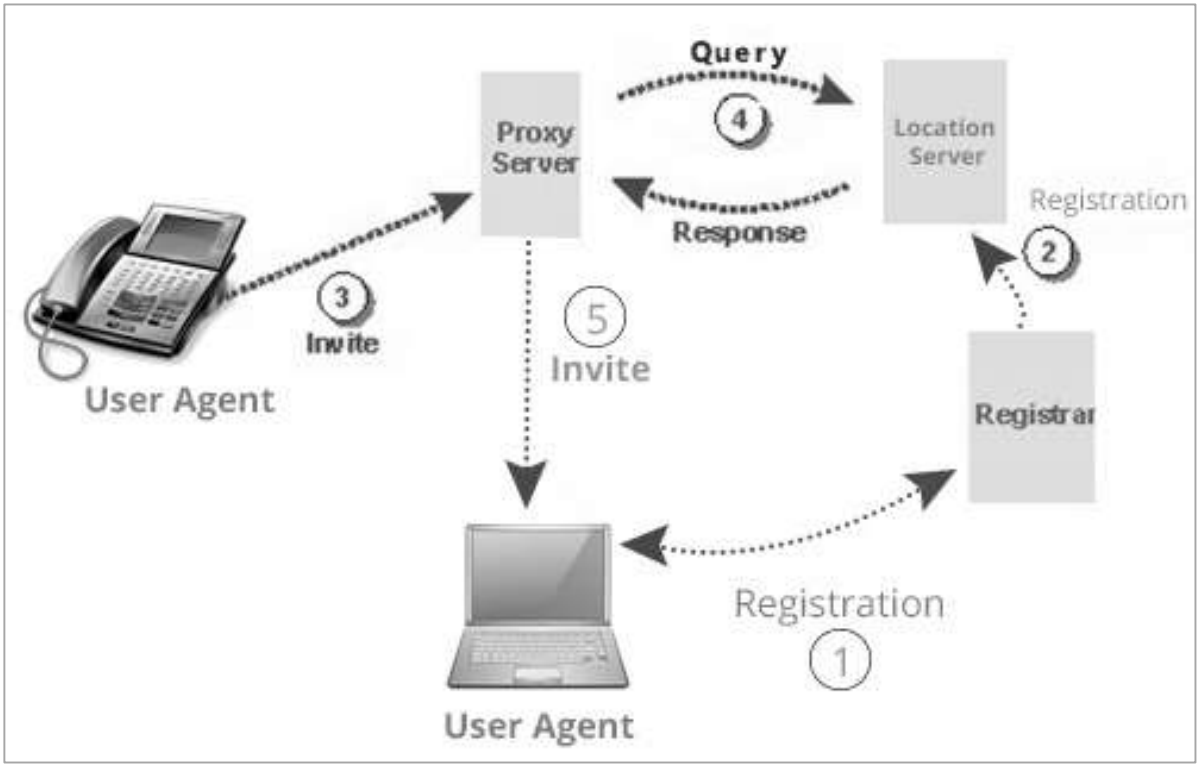
The redirect server uses the database for getting location information and responds with 3xx (Redirect response) to the user. We will discuss response codes later in this tutorial.

Location Server

The location server provides information about a caller's possible locations to the redirect and proxy servers.

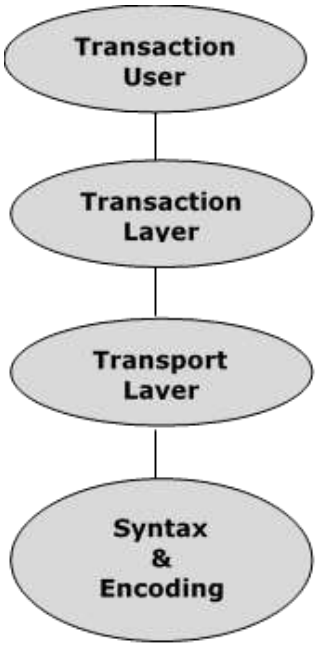
Only a proxy server or a redirect server can contact a location server.

The following figure depicts the roles played by each of the network elements in establishing a session.



SIP – System Architecture

SIP is structured as a layered protocol, which means its behavior is described in terms of a set of fairly independent processing stages with only a loose coupling between each stage.



- The lowest layer of SIP is its **syntax and encoding**. Its encoding is specified using an augmented **Backus-Naur Form grammar** (BNF).

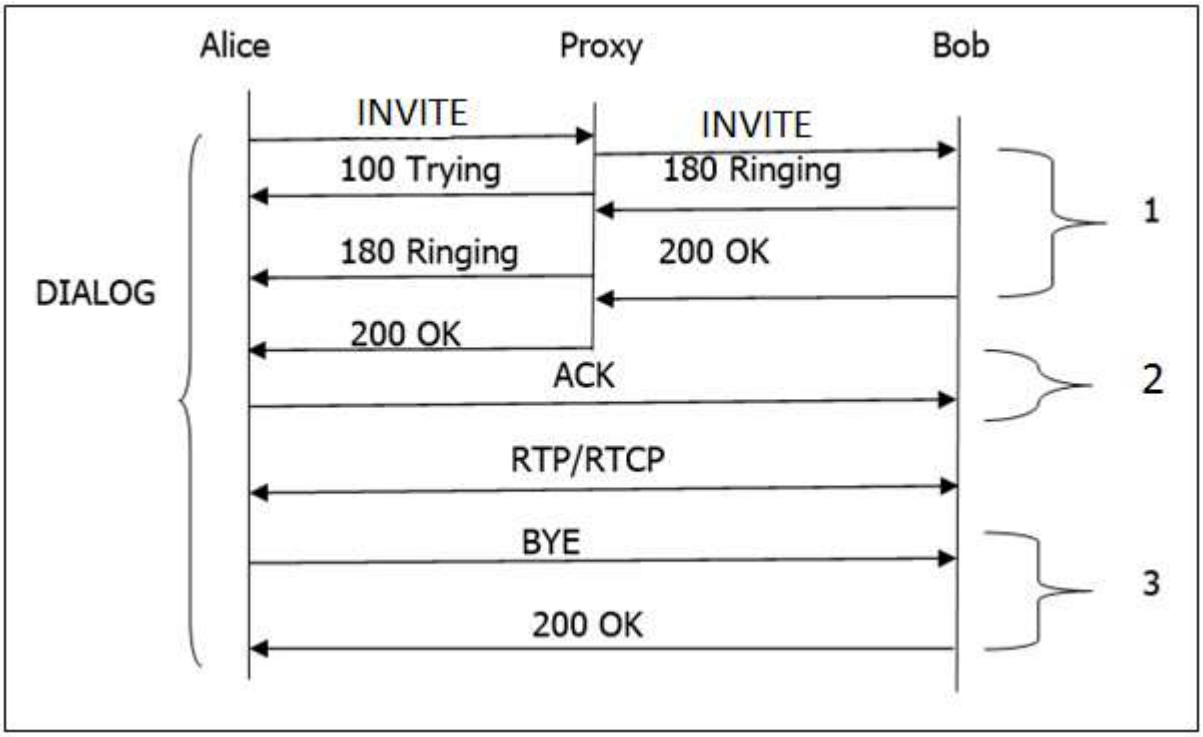
- At the second level is the **transport layer**. It defines how a Client sends requests and receives responses and how a Server receives requests and sends responses over the network. All SIP elements contain a transport layer.

- Next comes the **transaction layer**. A transaction is a request sent by a Client transaction (using the transport layer) to a Server transaction, along with all responses to that request sent from the server transaction back to the client. Any task that a user agent client (UAC) accomplishes takes place using a series of transactions. **Stateless proxies** do not contain a transaction layer.

- The layer above the transaction layer is called the **transaction user**. Each of the SIP entities, except the **stateless proxy**, is a transaction user.

3. Basic Call Flow

The following image shows the basic call flow of a SIP session.



Given below is a step-by-step explanation of the above call flow:

1. An INVITE request that is sent to a proxy server is responsible for initiating a session.

2. The proxy server sendsa**100 Trying** response immediately to the caller (Alice) to stop the re-transmissions of the INVITE request.

3. The proxy server searches the address of Bob in the location server. After getting the address, it forwards the INVITE request further.

4. Thereafter,**180 Ringing** (Provisional responses) generated by Bob is returned back to Alice.

5. A **200 OK** response is generated soon after Bob picks the phone up.

6. Bob receives an **ACK** from the Alice, once it gets **200 OK**.

7. At the same time, the session gets established and RTP packets (conversations) start flowing from both ends.

8. After the conversation, any participant (Alice or Bob) can send a **BYE** request to terminate the session.

9. **BYE** reaches directly from Alice to Bob bypassing the proxy server.

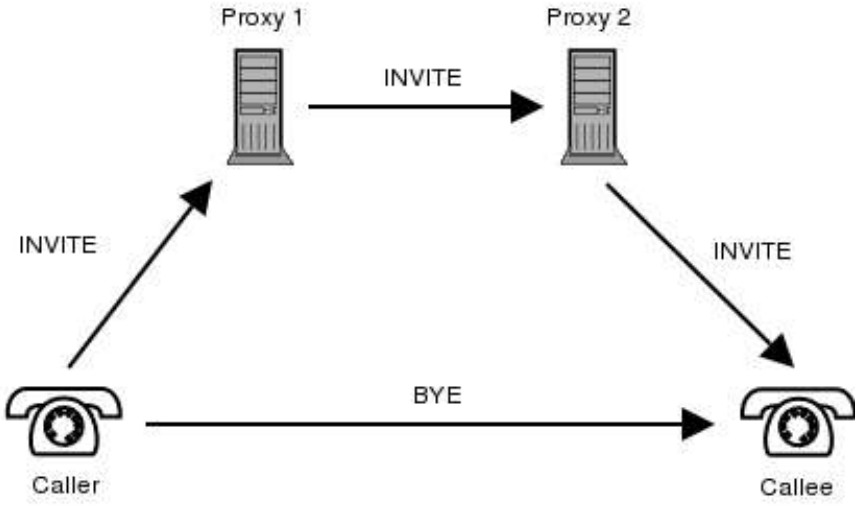
10. Finally, Bob sends a **200 OK** response to confirm the BYE and the session is terminated.

11. In the above basic call flow, three **transactions** are (marked as 1, 2, 3) available.

The complete call (from INVITE to 200 OK) is known as a **Dialog**.

SIP Trapezoid

How does a proxy help to connect one user with another? Let us find out with the help of the following diagram.



The topology shown in the diagram is known as a SIP trapezoid. The process takes place as follows:

1. When a caller initiates a call, an INVITE message is sent to the proxy server. Upon receiving the INVITE, the proxy server attempts to resolve the address of the callee with the help of the DNS server.

2. After getting the next route, caller's proxy server (Proxy 1, also known as outbound proxy server) forwards the INVITE request to the callee's proxy server which acts as an inbound proxy server (Proxy 2) for the callee.

3. The inbound proxy server contacts the location server to get information about the callee's address where the user registered.

4. After getting information from the location server, it forwards the call to its destination.

5. Once the user agents get to know their address, they can bypass the call, i.e., conversations pass directly.

4. SIP Messaging

SIP messages are of two types: **requests** and **responses**.

- The opening line of a request contains a method that defines the request, and a Request-URI that defines where the request is to be sent.
- Similarly, the opening line of a response contains a response code.

Request Methods

SIP requests are the codes used to establish a communication. To complement them, there are **SIP responses** that generally indicate whether a request succeeded or failed.

These SIP requests which are known as METHODS make SIP message workable.

- METHODS can be regarded as SIP requests, since they request a specific action to be taken by another user agent or server.
- METHODS are distinguished into two types:
  - Core Methods
  - Extension Methods

Core Methods

There are six core methods as discussed below.

INVITE

INVITE is used to initiate a session with a user agent. In other words, an INVITE method is used to establish a media session between the user agents.

- INVITE can contain the media information of the caller in the message body.
- A session is considered established if an INVITE has received a success response(2xx) or an ACK has been sent.

- A successful INVITE request establishes a **dialog** between the two user agents which continues until a BYE is sent to terminate the session.
- An INVITE sent within an established dialog is known as a **re-INVITE**.
- Re-INVITE is used to change the session characteristics or refresh the state of a dialog.

INVITE Example

The following code shows how INVITE is used.

```
INVITE sips:Bob@TMC.com SIP/2.0
      Via: SIP/2.0/TLS client.ANC.com:5061;branch=z9hG4bK74bf9
      Max-Forwards: 70
      From: Alice<sips:Alice@TTP.com>;tag=1234567
      To: Bob<sips:Bob@TMC.com>
      Call-ID: 12345601@192.168.2.1
      CSeq: 1 INVITE
      Contact: <sips:Alice@client.ANC.com>
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
      Supported: replaces
      Content-Type: application/sdp
      Content-Length: ...

      v=0
      o=Alice 2890844526 2890844526 IN IP4 client.ANC.com
      s=Session SDP
      c=IN IP4 client.ANC.com
```

```
t=3034423619 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

BYE

BYE is the method used to terminate an established session. This is a SIP request that can be sent by either the caller or the callee to end a session.

- It cannot be sent by a proxy server.
- BYE request normally routes end to end, bypassing the proxy server.
- BYE cannot be sent to a pending an INVITE or an unestablished session.

REGISTER

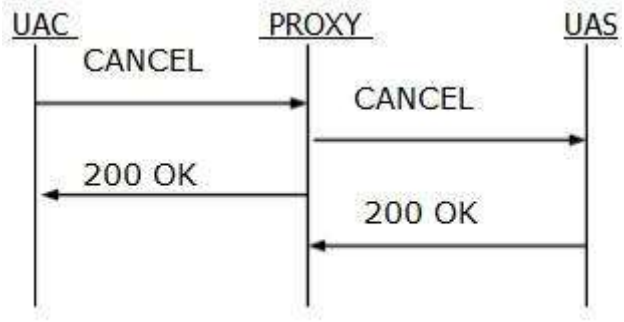
REGISTER request performs the registration of a user agent. This request is sent by a user agent to a registrar server.

- The REGISTER request may be forwarded or proxied until it reaches an authoritative registrar of the specified domain.
- It carries the AOR (Address of Record) in the **To** header of the user that is being registered.
- REGISTER request contains the time period (3600sec).
- One user agent can send a REGISTER request on behalf of another user agent. This is known as **third-party registration**. Here, the **From** tag contains the URI of the party submitting the registration on behalf of the party identified in the **To** header.

CANCEL

CANCEL is used to terminate a session which is not established. User agents use this request to cancel a pending call attempt initiated earlier.

- It can be sent either by a user agent or a proxy server.
- CANCEL is a **hop by hop** request, i.e., it goes through the elements between the user agent and receives the response generated by the next stateful element.



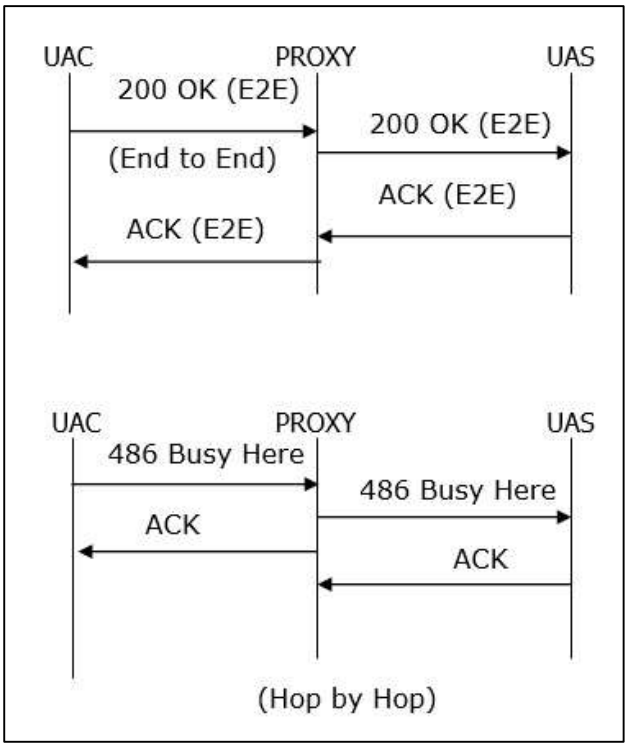
(hop by hop request)

ACK

ACK is used to acknowledge the final responses to an INVITE method. An ACK always goes in the direction of INVITE.ACK may contain SDP body (media characteristics), if it is not available in INVITE.

(SDP exchange in ACK)

- ACK may not be used to modify the media description that has already been sent in the initial INVITE.



- A stateful proxy receiving an ACK must determine whether or not the ACK should be forwarded downstream to another proxy or user agent.
- For 2xx responses, ACK is end to end, but for all other final responses, it works on hop by hop basis when stateful proxies are involved.

OPTIONS

OPTIONS method is used to query a user agent or a proxy server about its capabilities and discover its current availability. The response to a request lists the capabilities of the user agent or server. A proxy never generates an OPTIONS request.

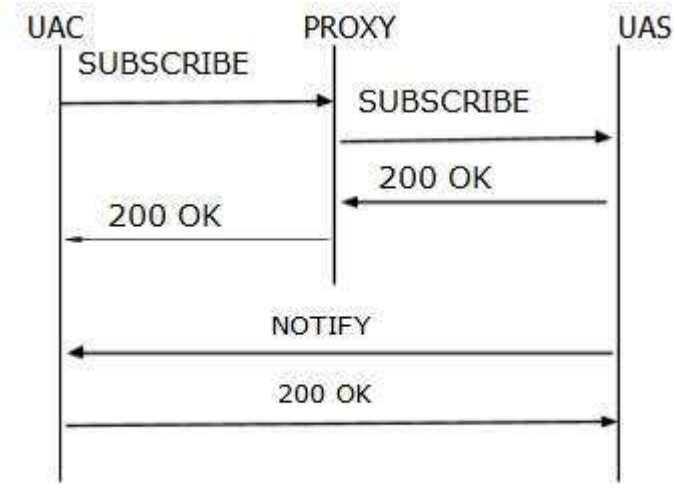
Extension Methods

Subscribe

SUBSCRIBE is used by user agents to establish a subscription for the purpose of getting notification about a particular event.

- It contains an **Expires** header field that indicates the duration of a subscription.
- After the time period passes, the subscription will automatically terminate.
- Subscription establishes a dialog between the user agents.
- You can re-subscription again by sending another SUBSCRIBE within the dialog before the expiration time.

- A 200 OK will be received for a subscription from User.
- Users can unsubscribe by sending another SUBSCRIBE method with Expires value 0(zero).



(Example of SUBSCRIBE and NOTIFY Call Flow)

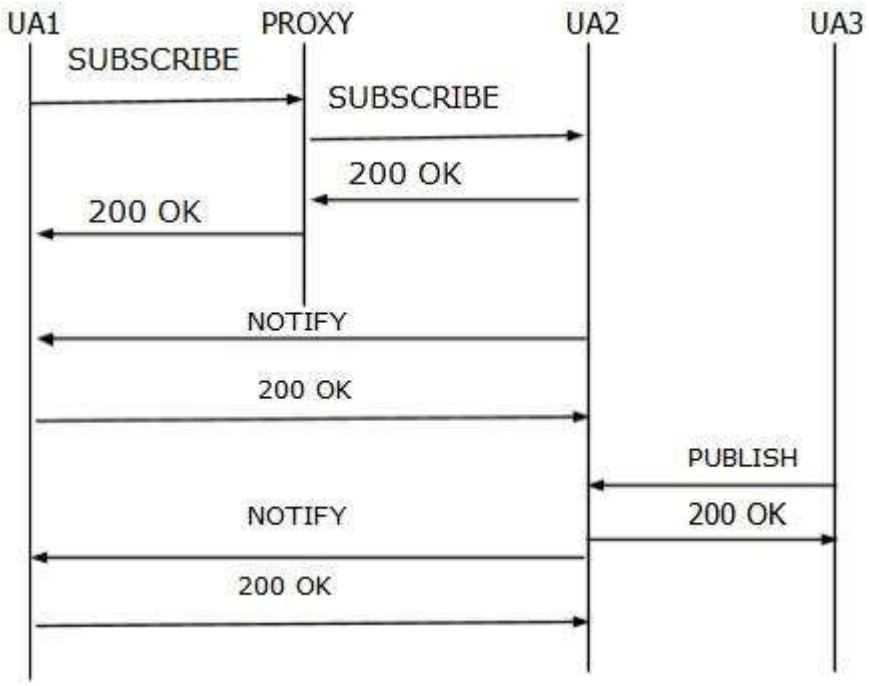
NOTIFY

NOTIFY is used by user agents to get the occurrence of a particular event. Usually a NOTIFY will trigger within a dialog when a subscription exists between the subscriber and the notifier.

- Every NOTIFY will get 200 OK response if it is received by notifier.
- NOTIFY contain an **Event** header field indicating the event and a **subscription-state** header field indicating the current state of the subscription.
- A NOTIFY is always sent at the start and termination of a subscription.

PUBLISH

PUBLISH is used by a user agent to send event state information to a server.



- PUBLISH is mostly useful when there are multiple sources of event information.
- A PUBLISH request is similar to a NOTIFY, except that it is not sent in a dialog.
- A PUBLISH request must contain an **Expires** header field and a **Min-Expires** header field.

REFER

REFER is used by a user agent to refer another user agent to access a URI for the dialog.

- REFER must contain a **Refer-To** header. This is a mandatory header for REFER.
- REFER can be sent inside or outside a dialog.
- A **202 Accepted** will trigger a REFER request which indicates that other user agent has accepted the reference.

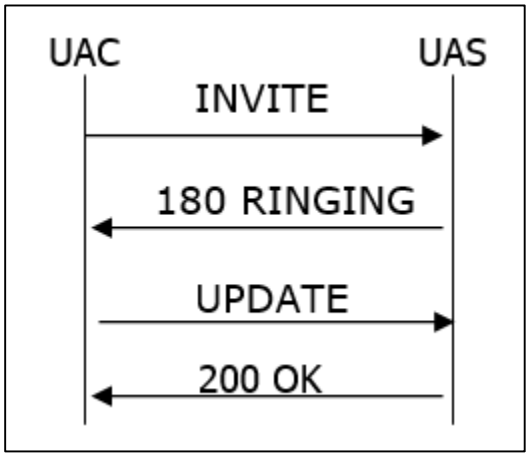
INFO

INFO is used by a user agent to send call signalling information to another user agent with which it has established a media session.

- This is an end-to-end request.
- A proxy will always forward an INFO request.

UPDATE

UPDATE is used to modify the state of a session if a session is not established. User could change the codec with UPDATE.



IF a session is established, a re-Invite is used to change/update the session.

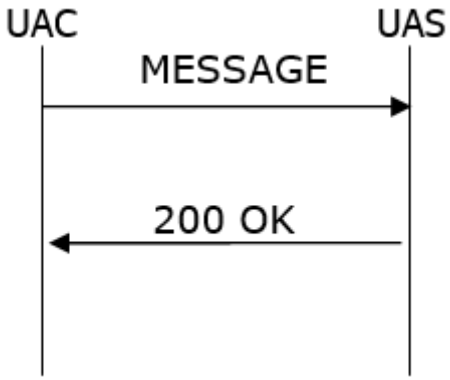
PRACK

PRACK is used to acknowledge the receipt of a reliable transfer of provisional response (1XX).

- Generally PRACK is generated by a client when it receive a provisional response containing an **RSeq** reliable sequence number and a **supported:100rel** header.
- PRACK contains (RSeq + CSeq) value in the **rack** header.
- The PRACK method applies to all provisional responses except the 100 Trying response, which is never reliably transported.
- A PRACK may contain a message body; it may be used for offer/answer exchange.

MESSAGE

It is used to send an instant message using SIP. An IM usually consists of short messages exchanged in real time by participants engaged in text conversation.



- MESSAGE can be sent within a dialog or outside a dialog.
- The contents of a MESSAGE are carried in the message body as a **MIME** attachment.
- A **200 OK** response is normally received to indicate that the message has been delivered at its destination.

5. Response Codes

A SIP response is a message generated by a user agent server (UAS) or SIP server to reply a request generated by a client. It could be a formal acknowledgement to prevent retransmission of requests by a UAC.

- A response may contain some additional header fields of info needed by a UAC.
- SIP has six responses.
- 1xx to 5xx has been borrowed from HTTP and 6xx is introduced in SIP.
- 1xx is considered as a **provisional** response and the rest are **final** responses.

1. 1xx: Provisional/Informational Responses
2. 2xx: Success Responses
3. 3xx: Redirect Responses
4. 4xx: Client Failure Responses
5. 5xx: Server Failure Responses
6. 6xx: Global Failure Responses

Informational (1xx)

Informational responses are used to indicate **call progress**. Normally the responses are end to end (except 100 Trying). The main objective of informational responses is to stop retransmission of INVITE requests.

Informational responses include the following responses:

100 Trying

- This special case response is only a **hop-by-hop** request.
- It is never forwarded and may not contain a message body.
- It is used to avoid the retransmission of **INVITE** requests.

180 Ringing

- This response is used to indicate that an **INVITE** has been received by the user agent and **alerting** is taking place.

181 Call is Being Forwarded

- This response is used to indicate that the call has been forwarded to another endpoint.

- It gives the status of the caller, as a forwarding operation may result in the call taking longer to be answered.

182 Call Queued

- This response is used to indicate that the INVITE has been received and will be processed in a queue.

183 Session Progress

- It indicates that information about the progress of a session may be present in a message body or media stream.
- Unlike a 100 Trying response, a 183 is an end-to-end response.
- It establishes early dialog.
- A typical use of this response is to allow a UAC to hear a ringtone recorded announcement in calls through a gateway into the PSTN.

Success(2xx)

This class of responses is meant for indicating that a request has been accepted. It includes the following responses:

200 OK

- 200 OK is used to accept a session invitation.
- It indicates a successful receipt of a request.

202 Accepted

- 202 Accepted indicates that the UAS has received and understood the request, but that the request may not have been authorized or processed by the server.
- It is commonly used in responses to SUBSCRIBE, REFER methods.

Redirection (3xx)

Generally these class responses are sent by redirect servers in response to INVITE. They are also known as redirect class responses. It includes the following responses:

300 Multiple Choices

- It contains multiple Contact header fields to indicate that the location service has returned multiple possible locations for the SIP URI in the Request-URI.

301 Moved Permanently

- This redirection response contains a Contact header field with the new URI of the called party.
- The address can be saved and used in future INVITE requests.

302 Moved Temporarily

- This redirection response contains a URI that is currently valid but is not permanent.
- That is, for the specified duration of time the location is valid.

305 Use Proxy

- This response points to certain proxy server which is having some authoritative information about the calling party.
- This response could be sent by a UAS issuing a proxy for incoming call screening.

380 Alternative Service

- This response returns a URI that indicates the type of service the called party would like.
- For example, a call could be redirected to a voicemail server.

Client Error (4xx)

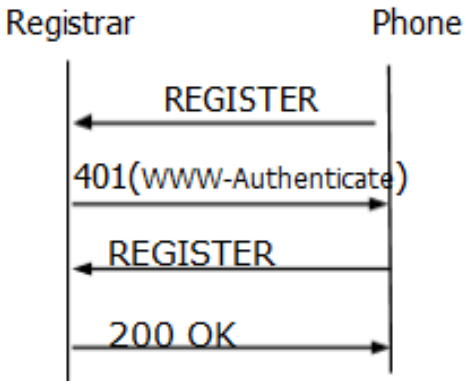
Client error responses indicate that the request cannot be fulfilled as some errors are identified from the UAC side. The response codes are generally sent by UAS. Upon receiving an error message, the client should resend the request by modifying it based on the response. Discussed below are some of the important client error responses.

400 Bad Request

- This indicates that the server could not understood the request.
- Request might be missing required header fields such as To, From, Call-ID, or CSeq.

401 Unauthorized

- It indicates that the request need to perform authentication.
- 401 Unauthorized is normally sent by a registrar server for REGISTER request.
- The response contains WWW-Authenticate header field which requests for correct credentials from the calling user agent.



- A subsequent REGISTER will trigger from the User Agent with correct credentials.

402 Payment Required

- It indicates payment is required for further processing of request.

403 Forbidden

- 403 Forbidden is sent when the server has understood the request, found the request to be correctly formulated, but will not service the request.
- This response is not used when authorization is required.

404 Not Found

- It indicates that server has not found the indicated SIP URI by the User.

405 Method Not Allowed

- It indicates that the request has contains list of methods that are not allowed.
- Example: A REGISTER request might be sent to a user agent.
- It contains an **Allow** field which inform the UAC as to what methods are acceptable.

406 Not Acceptable

- This response indicates that the request cannot be processed due to a requirement in the request message.
- The Accept header field in the request did not contain any options supported by the UAS.

407 Proxy Authentication Required

- This request sent by a proxy indicates that the UAC first has to authenticate itself with the proxy before the request can be processed.
- The response should contain **Proxy-Authenticate** header which informs about the type of credentials required by the proxy.

- The request can be resent with the proper credentials in a **Proxy-Authorization** header field.

408 Request Timeout

- When the specified time period mentioned in the Expires header field of INVITE request has passed, this response comes.
- It could be sent by a forking proxy or a user agent.
- The request can be retried at any time by the UAC.

422 Session Timer Interval Too Small

- The response is used to reject a request containing a Session-Expires header field.
- The minimum allowed interval is indicated in the required Min-SE header field.
- The calling party may retry the request without the Session-Expires header field or with a value less than or equal to the specified minimum.

423 Interval Too Brief

- The response is returned by a registrar that is rejecting a registration request because the requested expiration time on one or more Contacts is too brief.
- The response must contain a **Min-Expires** header field listing the minimum expiration interval that the registrar will accept.

480 Temporarily Unavailable

- This response indicates that the request has reached the correct destination, but the called party is not available for some reason.
- The response should contain a **Retry-After** header indicating when the request may be able to be fulfilled.

481 Dialog/Transaction Does Not Exist

- This response indicates that a response referencing an existing call or transaction has been received for which the server has no records or state information.

483 Too Many Hops

- This response indicates that the request has been forwarded the maximum number of times as set by the Max-Forwards header that is 70 in the request.
- This is indicated by the receipt of a Max-Forward: 0 header in a request.

486 Busy Here

- This indicates the user agent is busy and cannot accept the call.

487 Request Terminated

- This response can be sent by a UA that has received a CANCEL request for a pending INVITE request.
- A 200 OK is sent to acknowledge the CANCEL, and a 487 is sent to cancel the INVITE transaction.

Server Failure (5xx)

This class response is used to indicate that the request cannot be processed because of an error with the server. The server failed to fulfil an apparently valid request. The response may contain a **Retry-After** header field. The request can be tried at other locations because there are no errors indicated in the request. Some of the important server failure responses are discussed below.

500 Server Internal Error

- 500indicates that the server has experienced some kind of error that is preventing it from processing the request.
- It is one kind of server failure that indicates the client to retry the request again at this server after several seconds.

501 Not Implemented

- It indicates that the server is unable to process the request because it is not supported.
- This response can be used to decline a request containing an unknown method.

502 Bad Gateway

- This response is sent by a proxy that is acting as a gateway to another network.
- It indicates some problem in the other network is preventing the request from being processed.

503 Service Unavailable

- This response indicates that the requested service is temporarily unavailable at that time.
- The request can be retried after a few seconds, or after the expiration of the Retry-After header field.

504 Gateway Timeout

- This response comes when the request failed due to a timeout occurred in the other network to which the gateway connects.
- It is a server error class response because the call is failing due to a failure of the server in accessing resources outside the SIP network.

505 Version Not Supported

- The server denies a request when it comes with a different SIP version number. The denial is indicated in this message.
- Currently SIP version 2.0 is the only version implemented.

513 Message Too Large

- This response is used by a UAS to indicate that the request size was too large for it to process.

580 Preconditions Failure

- This response is used to reject an SDP offer in which required preconditions cannot be met.

Global Error (6xx)

This response class indicates that the server knows that the request will fail wherever it is tried. As a result, the request should not be sent to other locations.

Only a server having definitive knowledge of the user identified by the Request-URI in every possible instance should send a global error class response. Otherwise, a client error class response should be sent.

A Retry-After header field can be used to indicate when the request might be successful. Some of the important responses are discussed below:

600 Busy Everywhere

- This response indicates that the call to the specified Request-URI could be answered in other locations.

603 Decline

- This response could indicate the called party is busy, or simply does not want to accept the call.

604 Does Not Exist Anywhere

- This response is similar to the **404 Not Found** response but indicates that the user in the Request-URI cannot be found anywhere.
- This response should only be sent by a server having access to all the information about the user.

606 Not Acceptable

- This response indicates that some aspect of the desired session is not acceptable to the UAS, and as a result, the session cannot be established.
- The response may contain a **Warning header** field with a numerical code describing exactly what was not acceptable.
- The request can be retried with different media session information.

6. SIP Headers

A header is a component of a SIP message that conveys information about the message. It is structured as a sequence of header fields.

SIP header fields in most cases follow the same rules as HTTP header fields. Header fields are defined as **Header: field**, where Header is used to represent the header field name, and field is the set of tokens that contains the information. Each field consists of a field-name followed by a colon (":") and the field-value (i.e., **field-name: field-value**).

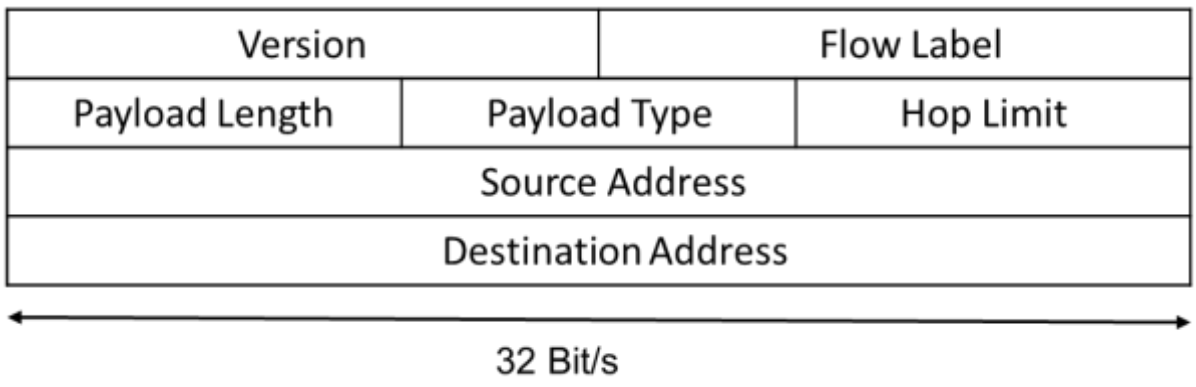
SIP Headers – Compact Form

Many common SIP header fields have a compact form where the header field name is denoted by a single lower case character. Some examples are given below:

Header	Compact form
To	T
Via	V
Call-ID	I
Contact	M
From	F
Subject	S
Content-Length	l

SIP Header Format

The following image shows the structure of a typical SIP header.



Headers are categorized as follows depending on their usage in SIP:

- Request and Response
- Request Only
- Response Only
- Message Body

Request and Response Header Fields

Accept

The Accept header field is used to indicate acceptable message Internet media types in the message body.

- The header field describes media types using the format type/sub-type commonly used in the Internet.
- If not present, the assumed acceptable message body format is **application/sdp**.
- A list of media types can have preferences set using **q** value parameters.

Accept-Encoding

The Accept-Encoding header field is used to specify acceptable message body encoding schemes.

- Encoding can be used to ensure a SIP message with a large message body fits inside a single UDP datagram.
- The use of **q** value parameters can set preferences. If none of the listed schemes are acceptable to the UAC, a 406 Not Acceptable response is returned. If not included, the assumed encoding will be **text/plain**.

To

To indicates the final recipient of the request. Any response generated by a UA will contain this header field with the addition of a tag. It is a mandatory header.

- Any response generated by a proxy must have a tag added to the **To** header field.
- The **To** header field URI is never used for routing.

From

From header field indicates the originator of the request. It is one of two addresses used to identify a dialog.

- A **From** header field may contain a tag used to identify a particular call.

- It may contain a display name, in which case the URI is enclosed in <>.
- It is a mandatory header.

Call-ID

The Call-ID header field is mandatory in all SIP requests and responses. It is used to uniquely identify a call between two user agents.

- A Call-ID must be unique across calls.
- All registrations for a user agent should use the same Call-ID.
- A Call-ID is always created by a user agent and is never modified by a server.
- It is a cryptographically random identifier.

Via

Via is used to record the SIP route taken by a request which helps to route a response back to the originator.

- A UA generating a request records its own address in a Via header field.
- A proxy forwarding the request adds a Via header field containing its own address to the top of the list of Via header fields.
- A proxy or UA generating a response to a request copies all the Via header fields from the request in order into the response, then sends the response to the address specified in the top Via header field.
- A proxy receiving a response checks the top Via header field and matches its own address.
- If it does not match, the response has been discarded.
- The top Via header field is then removed, and the response forwarded to the address specified in the next Via header field.
- Via header fields contain protocol name, version number, and transport (SIP/2.0/UDP, SIP/2.0/TCP, etc.) and may contain port numbers and parameters such as received, **rport**, **branch**, **maddr**, and **ttl**.
- A **received** tag is added to a Via header field if a UA or proxy receives the request from a different address than that specified in the top Via header field.
- A branch parameter is added to Via header fields by UAs and proxies, which is computed as a hash function of the Request-URI, and the To, From, Call-ID, and CSeq number.

CSeq

The CSeq header field is a required header field in every request. It contains a decimal number that increases for each request.

- Usually, it increases by 1 for each new request, with the exception of **CANCEL** and **ACK** requests, which use the CSeq number of the INVITE request to which it refers.
- The CSeq count is used by UASs to determine out-of-sequence requests or to differentiate between a new request (different CSeq) or a retransmission (same CSeq).
- The CSeq header field is used by UACs to match a response to the request it references.
- For example, a UAC that sends an INVITE request then a CANCEL request can tell by the method in the CSeq of a 200 OK response if it is a response to the invitation or cancellation request.

Contact

The Contact header field is used to convey the other user about the address of the request originator. Once a Contact header field has been received, the URI can be cached and used for routing future requests within a dialog.

For example, a Contact header field in a 200 OK response to an INVITE can allow the acknowledgment ACK message and all future requests during this call to bypass proxies and go directly to the called party.

Record-Route

The Record-Route header field is used to force routing through a proxy for all subsequent requests in a session (dialog) between two UAs.

Normally, the presence of a Contact header field allows UAs to send messages directly bypassing the proxy chain used in the initial request.

- A proxy inserting its address into a Record-Route header field overrides this and forces future requests to include a Route header field containing the address of the proxy that forces this proxy to be included.
- A proxy wishing to implement this inserts the header field containing its own URI, or adds its URI to an already present Record-Route header field.
- The URI is constructed so that the URI resolves back to the proxy server. The UAS copies the Record-Route header field into the 200 OK response to the request.
- The header field is forwarded unchanged by proxies back to the UAC. The UAC then stores the Record-Route proxy list plus a Contact header field if present in the200 OK for use in a Route header field in all subsequent requests.

Organization

The Organization header field is used to indicate the organization to which the originator of the message belongs.

- It can also be inserted by proxies as a message is passed from one organization to another.
- Like all SIP header fields, it can be used by proxies for making routing decisions and by UAs for making call screening decisions.

Retry-After

It is used to indicate when a resource or service may be available again.

- In 503 Service Unavailable responses, it indicates when the server will be available.
- In 404 Not Found, 600 Busy Everywhere, and 603 Decline responses, it indicates when the called UA may be available again.
- It contains time period in 'sec'.

Subject

The optional Subject header field is used to indicate the subject of the media session.

The contents of the header field can also be displayed during alerting to aid the user in deciding whether to accept the call.

```
Example:

Subject: How are you?
```

Supported

The Supported header field is used to list one or more options implemented bya UA or server.

- It is typically included in responses to OPTIONS requests.
- If no options are implemented, the header field is not included.
- If a UAC lists an option in a Supported header field, proxies or UASs may use the option during the call.
- If the option must be used or supported, the Require header field is used instead.

```
Example:

Supported: rel100
```

Expires

The Expires header field is used to indicate the time interval in which the request or message contents are valid.

- When present in an INVITE request, the header field sets a time limit on the completion of the INVITE request.
- That is, the UAC must receive a final response (non-1xx) within the time period or the INVITE request is automatically cancelled with a 408 Request Timeout response.
- Once the session is established, the value from the Expires header field in the original INVITE has no effect—the Session-Expires header field must be used for this purpose.
- If present in a REGISTER request, the header field sets the time limit on the URIs in Contact header fields that do not contain an **expires** parameter.
- Expires is also used in SUBSCRIBE requests to indicate the subscription duration.

```
Example:

Expires: 30
```

User-Agent

This header field is used to convey information about the UA originating the request.

Request Only Header

Authorization

The Authorization header field is used to carry the credentials of a UA in a request to a server.

It can be sent in reply to a **401 Unauthorized** response containing challenge information.

Event

This header field is used in a **SUBSCRIBE** or **NOTIFY** method to indicate which event package is being used by the method.

- In a SUBSCRIBE, it lists the event package to which the client would like to subscribe.
- In a NOTIFY, it lists the event package that the notification contains state information about.

Join

The Join header field is used in an INVITE to request that the dialog (session) be joined with an existing dialog (session).

- The parameters of the Join header field identify a dialog by the Call-ID, To tag, and From tag in a similar way to the Replaces header field.
- If the Join header field references a point-to-point dialog between two user agents, the Join header field is effectively a request to turn the call into a conference call.
- If the dialog is already part of a conference, the Join header field is a request to be added into the conference.

Proxy-Authorization

The Proxy-Authorization header field is to carry the credentials of a UA in a request to a server.

- It can be sent in reply to a 407 Proxy Authentication Required response containing challenge information.
- A proxy receiving a request containing a Proxy-Authorization header field searches for its own realm, and if found it processes the entry.
- If the credentials are correct, any remaining entries are kept in the request when it is forwarded to the next proxy.

Proxy-Require

The Proxy-Require header field is used to list features and extensions that a UA requires a proxy to support in order to process the request.

- A 420 Bad Extension response is returned by the proxy listing any unsupported feature in an **Unsupported header** field
- If the support of this option is desired but not required, it is listed in a **Supported** header field instead.

Max-Forwards

The Max-Forwards header field is used to indicate the maximum number of hops that a SIP request may take.

- The value of the header field is decremented by each proxy that forwards the request.
- A proxy receiving the header field with a value of zero discards the message and sends a 483 Too Many Hops response back to the originator.
- Max-Forwards is a mandatory header field in requests as per RFC 3261.

- The recommended value is 70 hops.

Priority

The Priority header field is used by a UAC to set the urgency of a request. Values are non-urgent, normal, urgent, and emergency.

Refer-To

The Refer-To header field is a mandatory header field in a REFER request, which contains the URI or URL resource that is being referenced. It may contain any type of URI from a sip or sips to a telURI.

Referred-By

The Referred-By header field is an optional header field in a REFER request and a request triggered by a REFER.

- It provides the recipient of a triggered request with information that the request was generated as a result of a REFER and the originator of the REFER.

- An unsigned Referred-By header field may be rejected with **429 Provide Referror** Identity response code.

Replaces

Replaces is used for replacing an existing call with a new call.

- A UA in an established dialog receiving another INVITE with a Replaces header field that matches the existing dialog must accept the INVITE, terminate the existing dialog with a BYE, and transfer all resources and state from the existing dialog to the newly established dialog.

- If the Replaces header field matches no dialog, the INVITE must be rejected with a 481 Dialog Does Not Exist response.

Request-Disposition

The Request-Disposition header field can be used to request servers to either proxy, redirect.

```
Example:

Request-Disposition: redirect
```

Require

The **Require** header field is used to list features and extensions that a UAC requires a UAS to support in order to process the request.

A 420 Bad Extension response is returned by the UAS listing any unsupported features in an Unsupported header field.

```
Example:
Require: rel100
```

Route

The **Route** header field is used to provide routing information for requests.

- RFC 3261 introduces two types of routing: **strict routing** and **loose routing**, which have similar meaning as the IP routing modes of the same name.
- In strict routing, a proxy must use the first URI in the Route header field to rewrite the Request-URI, which is then forwarded.
- In loose routing, a proxy does not rewrite the Request-URI, but either forwards the request to the first URI in the Route header field or to another loose routing element.
- In loose routing, the request must route through every server in the Route list before it may be routed based on the Request-URI.
- In strict routing, the request must only route through the set of servers in the Route header field with the Request-URI being rewritten at each hop.
- A proxy or UAC can tell if the next element in the route set supports loose routing by the presence of an lr parameter.

```
Example:
Route: sip:proxy@example.com;lr
```

RAck

The **RAck** header field is used within a response to a PRACK request to reliably acknowledge a provisional response that contained an RSeq header field.

- Its value is combination ofCSeq and the RSeq from the provisional response.
- The reliable sequence number is incremented for each response sent reliably.

```
Example:
RAck: 3452337 17 INVITE
```

Session-Expires

The **Session-Expires** header field is used to specify the expiration time of the session.

- To extend a session, either UA can send a re-INVITE or UPDATE with a new Session-Expires header field.
- It will come into picture once call has been established.

SIP - If - Match

The SIP-If-Match header field is part of the SIP publication mechanism. It is included in a PUBLISH request meant to refresh, modify, or remove previously published state.

- The header field contains the entity tag of the state information that was returned in a SIP-ETag header field in a 2xx response to an earlier PUBLISH.
- If the entity-tag is no longer valid, the server will return a 412 Conditional Request Failed response.

```
Example:

SIP-If-Match: 56jforRr1pd
```

Subscription-State

The Subscription-State header field is a required header field in a NOTIFY request. It indicates the current state of a subscription. Values defined include active, pending, or terminated.

```
Example:

Subscription-State: terminated;reason=rejected
```

Response Header Fields

Min-Expires

The **Min-Expires** header field is used in a 423 Interval Too Brief response from a registrar rejecting a REGISTER request in which one or more Contacts have an expiration time that is too short.

- The header field contains an integer number of seconds that represents the minimum expiration interval that the registrar will accept.
- A client receiving this header field can update the expiration intervals of the registration request accordingly and resend the REGISTER request.

Min-SE

The **Min-SE** header field is a required header field in a 422 Session Timer Interval Too Small response.

The response may also be present in an INVITE or UPDATE containing a Session-Expires header field. It contains an integer number of seconds.

Proxy-Authenticate

The **Proxy-Authenticate** header field is used in a 407 Proxy Authentication Required authentication challenge by a proxy server to a UAC.

It contains the nature of the challenge so that the UAC may formulate credentials in a Proxy- Authorization header field in a subsequent request.

SIP-ETag

The **SIP-ETag** header field is part of the SIP publication mechanism. The SIP-ETag header field is returned in a 2xx response to a PUBLISH request.

- It contains an entity tag uniquely identifying the state information that has been processed.
- This entity tag can then be used to do conditional publications on this data including refreshing, modifying, and removing.

Unsupported

The **Unsupported** header field is used to indicate features that are not supported by the server.

The header field is used in a 420 Bad Extension response to a request containing an unsupported feature listed in a Require header field.

```
Example:
Unsupported: rel100
```

WWW-Authenticate

The **WWW-Authenticate** header field is used in a 401 Unauthorized authentication challenge by a UA or registrar server to a UAC.

It contains the nature of the challenge so that the UAC may formulate credentials in a Proxy-Authorization header field in a subsequent request.

RSeq

The **RSeq** header field is used in provisional (1xx class) responses to INVITEs to request reliable transport.

The header field may only be used if the INVITE request contained the Supported: rel100 header field.

- If present in a provisional response, the UAC should acknowledge receipt of the response with a PRACK method.
- The RSeq header field contains a reliable sequence number that is an integer randomly initialized by the UAS.
- Each subsequent provisional response sent reliably for this dialog will have a monotonically increasing RSeq number.
- The UAS will retransmit a reliably sent response until a PRACK is received with a RAck containing the reliable sequence number and CSeq.

Message Body Header Fields

Content-Encoding

The **Content-Encoding** header field is used to indicate that the listed encoding scheme has been applied to the message body. It allows the UAS to determine the decoding scheme necessary to interpret the message body.

- Only those encoding schemes listed in an Allow-Encoding header field may be used.
- The compact form is **e**.

```
Examples:
Content-Encoding: text/plain
e: gzip
```

Content-Disposition

The **Content-Disposition** header field is used to describe the function of a message body. Values include session, icon, alert, and render.

The value session indicates that the message body contains information to describe a media session.

Content-Language

The **Content-Language** header field is used to indicate the language of a message body. It contains a language tag, which identifies the language.

```
Example:
Content-Language: en
```

Content-Length

The **Content-Length** is used to indicate the number of octets in the message body.

A Content-Length: 0 indicates no message body.

Content-Type

The **Content-Type** header field is used to specify the Internet media type in the message body.

- Media types have the familiar form of type/sub-type.
- If this header field is not present, application/sdp is assumed.
- If an Accept header field was present in the request, the response Content-Type must contain a listed type, or a 415 Unsupported Media Type response must be returned.

- The compact form is **c**.

Example:

```
Content-Type: application/sdp
```

MIME-Version

The **MIME-Version** header field is used to indicate the version of MIME protocol used to construct the message body.

SIP, like HTTP, is not considered MIME compliant because parsing and semantics are defined by the SIP standard, not the MIME specification. Version 1.0 is the default value.

Example:

```
MIME-Version: 1.0
```

7. SDP

SDP stands for Session Description Protocol. It is used to describe multimedia sessions in a format understood by the participants over a network. Depending on this description, a party decides whether to join a conference or when or how to join a conference.

- The owner of a conference advertises it over the network by sending multicast messages which contain description of the session e.g. the name of the owner, the name of the session, the coding, the timing etc. Depending on these information, the recipients of the advertisement take a decision about participation in the session.

- SDP is generally contained in the body part of Session Initiation Protocol popularly called SIP.

- SDP is defined in RFC 2327. An SDP message is composed of a series of lines, called fields, whose names are abbreviated by a single lower-case letter, and are in a required order to simplify parsing

Purpose of SDP

The purpose of SDP is to convey information about media streams in multimedia sessions to help participants join or gather info of a particular session.

- SDP is a short structured textual description.

- It conveys the name and purpose of the session, the media, protocols, codec formats, timing and transport information.

- A tentative participant checks these information and decides whether to join a session and how and when to join a session if it decides to do so.

- The format has entries in the form of <type>= <value>, where the <type>defines a unique session parameter and the <value>provides a specific value for that parameter.

- The general form of a SDP message is:

  x=parameter1 parameter2 ... parameterN

- The line begins with a single lower-case letter, for example, x. There are never any spaces between the letter and the =, and there is exactly one space between each parameter. Each field has a defined number of parameters.

Session Description Parameters

Session description (* denotes optional)

- v= (protocol version)
- o= (owner/creator and session identifier)
- s= (session name)
- i=* (session information)
- u=* (URI of description)
- e=* (email address)
- p=* (phone number)
- c=* (connection information -not required if included in all media)
- b=* (bandwidth information)
- z=* (time zone adjustments)
- k=* (encryption key)
- a=* (zero or more session attribute lines)

Protocol Version

The v= field contains the SDP version number. Because the current version of SDP is 0, a valid SDP message will always begin with v=0.

Origin

The o= field contains information about the originator of the session and session identifiers. This field is used to uniquely identify the session.

- The field contains:
  `o=<username><session-id><version><network-type><address-type>`
- The **username** parameter contains the originator's login or host.
- The **session-id** parameter is a Network Time Protocol (NTP) timestamp or a random number used to ensure uniqueness.
- The **version** is a numeric field that is increased for each change to the session, also recommended to be a NTP timestamp.
- The **network-type** is always IN for Internet. The address-type parameter is either IP4 or IP6 for IPv4 or IPv6 address either in dotted decimal form or a fully qualified host name.

Session Name and Information

The `s=` field contains a name for the session. It can contain any nonzero number of characters. The optional `i=` field contains information about the session. It can contain any number of characters.

URI

The optional u= field contains a uniform resource indicator (URI) with more information about the session.

E-Mail Address and Phone Number

The optional e= field contains an e-mail address of the host of the session. The optional p= field contains a phone number.

Connection Data

The c= field contains information about the media connection.

- The field contains:

```
c =<network-type><address-type><connection-address>
```

- The **network-type** parameter is defined as IN for the Internet.
- The **address-type** is defined as IP4 for IPv4 addresses and IP6 for IPv6 addresses.
- The **connection-address** is the IP address or host that will be sending the media packets, which could be either multicast or unicast.
- If multicast, the connection-address field contains:

```
connection-address=base-multicast-address/ttl/number-of-addresses
```

  where **ttl** is the time-to-live value, and number-of-addresses indicates how many contiguous multicast addresses are included starting with the base-multicast address.

Bandwidth

The optional b= field contains information about the bandwidth required. It is of the form:

```
b=modifier:bandwidth-value
```

Time, Repeat Times, and Time Zones

The t= field contains the start time and stop time of the session.

```
t=start-time stop-time
```

The optional r= field contains information about the repeat times that can be specified in either in NTP or in days (d), hours (h), or minutes (m).

The optional z= field contains information about the time zone offsets. This field is used if are occurring session spans a change from daylight savings to standard time, or vice versa.

Media Announcements

The optional `m=` field contains information about the type of media session. The field contains:

```
m=media port transport format-list
```

- The `media` parameter is either `audio`, `video`, `text`, `application`, `message`, `image`, or `control`. The `port` parameter contains the port number.
- The `transport` parameter contains the transport protocol or the RTP profile used.
- The `format-list` contains more information about the media. Usually, it contains media payload types defined in RTP audio video profiles.

Example:

```
m=audio 49430 RTP/AVP 0 6 8 99
```

One of these three codecs can be used for the audio media session. If the intention is to establish three audio channels, three separate media fields would be used.

Attributes

The optional `a=` field contains attributes of the preceding media session. This field can be used to **extend SDP to provide more information about the media**. If not fully understood by a SDP user, the attribute field can be ignored. There can be one or more attribute fields for each media payload type listed in the media field.

Attributes in SDP can be either

- session level, or
- media level.

Session level means that the attribute is listed before the first media line in the SDP. If this is the case, the attribute applies to all the media lines below it.

Media level means itis listed after a media line. In this case, the attribute only applies to this particular media stream.

SDP can include both session level and media level attributes. If the same attribute appears as both, the media level attribute overrides the session level attribute for that particular media stream. Note that the connection data field can also be either session level or media level.

An SDP Example

Given below is an example session description, taken from RFC 2327:

```
v=0
o=PPT 40467 40468 IN IP4 192.168.2.1
s=-
c=IN IP4 192.168.2.1
b=AS:49
t=0 0
b=RR:0
b=RS:0
a=rtpmap:97 AMR/8000/1
m=audio 6000 RTP/AVP 96
a=fmtp:102 0-15
a=ptime:20
a=maxptime:240
```

8. The Offer/Answer Model

The use of SDP with SIP is given in the SDP offer answer RFC 3264. The default message body type in SIP is **application/sdp**.

- The calling party lists the media capabilities that they are willing to receive in SDP, usually in either an `INVITE` or in an `ACK`.
- The called party lists their media capabilities in the `200 OK` response to the `INVITE`.

A typical SIP use of SDP includes the following fields: `version`, `origin`, `subject`, `time`, `connection`, and one or more `media` and `attribute`.

- The `subject` and `time` fields are not used by SIP but are included for compatibility.
- In the SDP standard, the `subject` field is a required field and must contain at least one character, suggested to be `s=-` if there is no subject.
- The `time` field is usually set to `t=00`. SIP uses the connection, media, and attribute fields to set up sessions between UAs.
- The `origin` field has limited use with SIP.
- The `session-id` is usually kept constant throughout a SIP session.
- The `version` is incremented each time the SDP is changed. If the SDP being sent is unchanged from that sent previously, the `version` is kept the same.
- As the type of media session and codec to be used are part of the connection negotiation, SIP can use SDP to specify multiple alternative media types and to selectively accept or decline those media types.

The offer/answer specification, RFC 3264, recommends that an attribute containing `a=rtpmap:` be used for each media field. A media stream is declined by setting the port number to zero for the corresponding media field in the SDP response.

Example

In the following example, the caller Tom wants to set up an audio and video call with two possible audio codecs and a video codec in the SDP carried in the initial `INVITE:`

```
v=0
o= John 0844526 2890844526 IN IP4 172.22.1.102
s=-
c=IN IP4 172.22.1.102
t=0 0
m= audio 6000 RTP/AVP 97 98
a=rtpmap:97 AMR/16000/1
```

```
a=rtpmap:98 AMR-WB/8000/1
m=video 49172 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

The codecs are referenced by the RTP/AVP profile numbers 97, 98.

The called party Marry answers the call, chooses the second codec for the first media field, and declines the second media field, only wanting AMR session.

```
v=0
o=Marry 2890844526 2890844526 IN IP4 172.22.1.110
s=-
c=IN IP4 200.201.202.203
t=0 0
m=audio 60000 RTP/AVP 8
a=rtpmap:97 AMR/16000
m=video 0 RTP/AVP 32
```

If this audio-only call is not acceptable, then Tom would send an `ACK` then a `BYE` to cancel the call. Otherwise, the audio session would be established and RTP packets exchanged.

As this example illustrates, unless the number and order of media fields is maintained, the calling party would not know for certain which media sessions were being accepted and declined by the called party.

The offer/answer rules are summarized in the following sections.

Rules for Generating an Offer

An SDP offer must include all required SDP fields (this includes `v=`, `o=`, `s=`, `c=`,and `t=`). These are mandatory fields in SDP.

It usually includes a media field (`m=`) but it does not have to. The media lines contain all codecs listed in preference order. The only exception to this is if the endpoint supports a huge number of codecs, the most likely to be accepted or most preferred should be listed. Different media types include audio, video, text, MSRP, BFCP, and so forth.

Rules for Generating an Answer

An SDP answer to an offer must be constructed according to the following rules:

- The answer must have the same number of `m=` lines in the same order as the answer.
- Individual media streams can be declined by setting the port number to 0.

- Streams are accepted by sending a nonzero port number.
- The listed payloads for each media type must be a subset of the payloads listed in the offer.
- For dynamic payloads, the same dynamic payload number does not need to be used in each direction. Usually, only a single payload is selected.

Rules for Modifying a Session

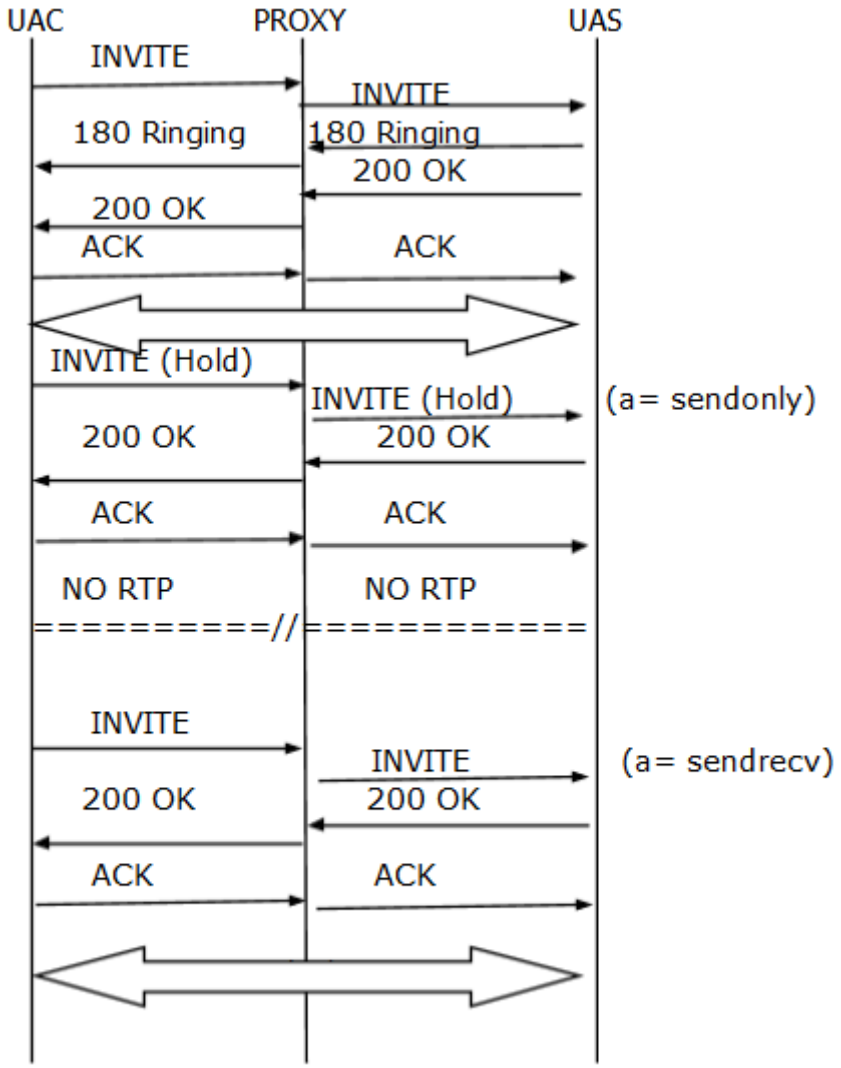
Either party can initiate another offer/answer exchange to modify a session. When a session is modified, the following rules must be followed:

- The origin (`o=`) line version number must either be the same as the last one sent, which indicates that this SDP is identical to the previous exchange, or it may be incremented by one, which indicates new SDP that must be parsed.
- The offer must include all existing media lines and they must be sent in the same order.
- Additional media streams can be added to the end of the `m=` line list.
- An existing media stream can be deleted by setting the port number to 0. This media line must remain in the SDP and all future offer/answer exchanges for this session.

Call Hold

One party in a call can temporarily place the other on hold. This is done by sending an `INVITE` with an identical SDP to that of the original `INVITE` but with **a=sendonly** attribute present.

The call is made active again by sending another `INVITE` with the **a=sendrecv** attribute present. The following illustration shows the call flow of a call hold.



(Call Flow of Call Hold)

9. SIP Mobility

Personal mobility is the ability to have a constant identifier across a number of devices. SIP supports basic personal mobility using the `REGISTER` method, which allows a mobile device to change its IP address and point of connection to the Internet and still be able to receive incoming calls.

SIP can also support **service mobility** – the ability of a user to keep the same services when mobile.

SIP Mobility During Handover (Pre-call)

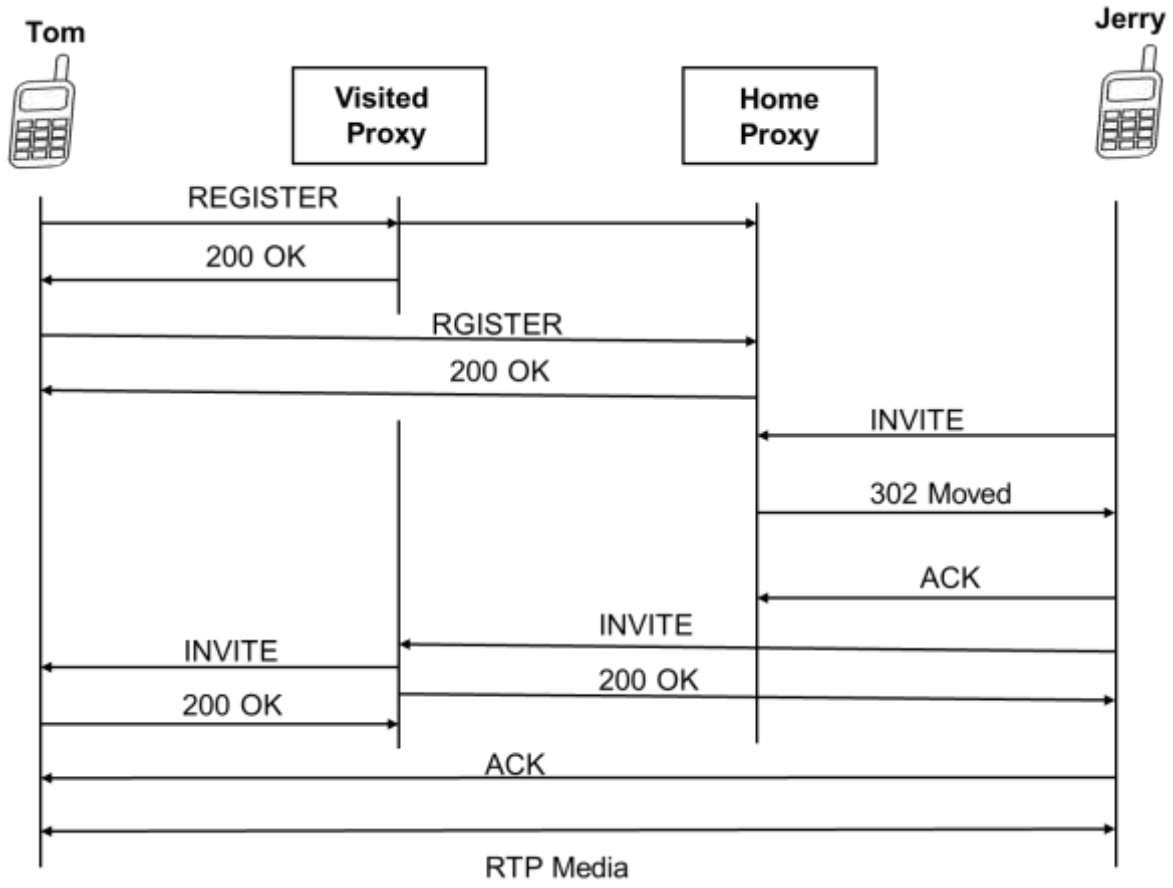
A device binds its Contact URI with the address of record by a simple sip registration. According to the device IP address, registration authorizes this information automatically update in sip network.

During handover, the User agent routes between different operators, where it has to register again with a `Contact` as an AOR with another service provider.

For example, let's take the example of the following call flow. UA which has temporarily received a new SIP URI with a new service provider. The UA then performs a double registration:

- The first registration is with the new service operator, which binds the `Contact` URI of the device with the new service provider's AOR URI.
- The second `REGISTER` request is routed back to the original service provider and provides the new service provider's AOR as the `Contact` URI.

As shown later in the call flow, when a request comes in to the original service provider's network, the `INVITE` is redirected to the new service provider who then routes the call to the user.



Precall mobility using SIP REGISTER

For the first registration, the message containing the device URI would be:

```
REGISTER sip:visited.registrar1.com SIP/2.0
Via: SIP/2.0/UDP 172.22.1.102:5060;branch=z9hG4bK97a7ea349ce0fca
Max-Forwards: 70
To: Tom <sip:UA1@registrar1.in>
From: Tom <sip:UA1@registrar1.in>;tag=72d65a24
Call-ID: 4e719d1c1fc9000803630373300@172.22.1.102
CSeq: 1 REGISTER
Contact: <sip:Tom@172.22.1.102:5060>
Expires: 600000
Content-Length: 0
```

The second registration message with the roaming URI would be:

```
REGISTER sip:home.registrar2.in SIP/2.0
Via: SIP/2.0/UDP 172.22.1.102:5060;branch=z9hG4bKah4vn2u
Max-Forwards: 70
To: Tom <sip:UA1@registrar2.in>
From: Tom <sip:UA1@registrar2.in>;tag=45375
Call-ID:87nr43i@172.22.1.102
CSeq: 6421 REGISTER
Contact: <sip:UA1@registrar2.in>
Content-Length: 0
```

The first `INVITE` that is represents in the above figure would be sent to sip:registrar2.in; the second `INVITE` would be sent to `sip:` sip:Tom@registrar2.in, which would be forwarded to sip:Tom@172.22.1.102. It reaches Tom and allows the session to be established. Periodically both registrations would need to be refreshed.

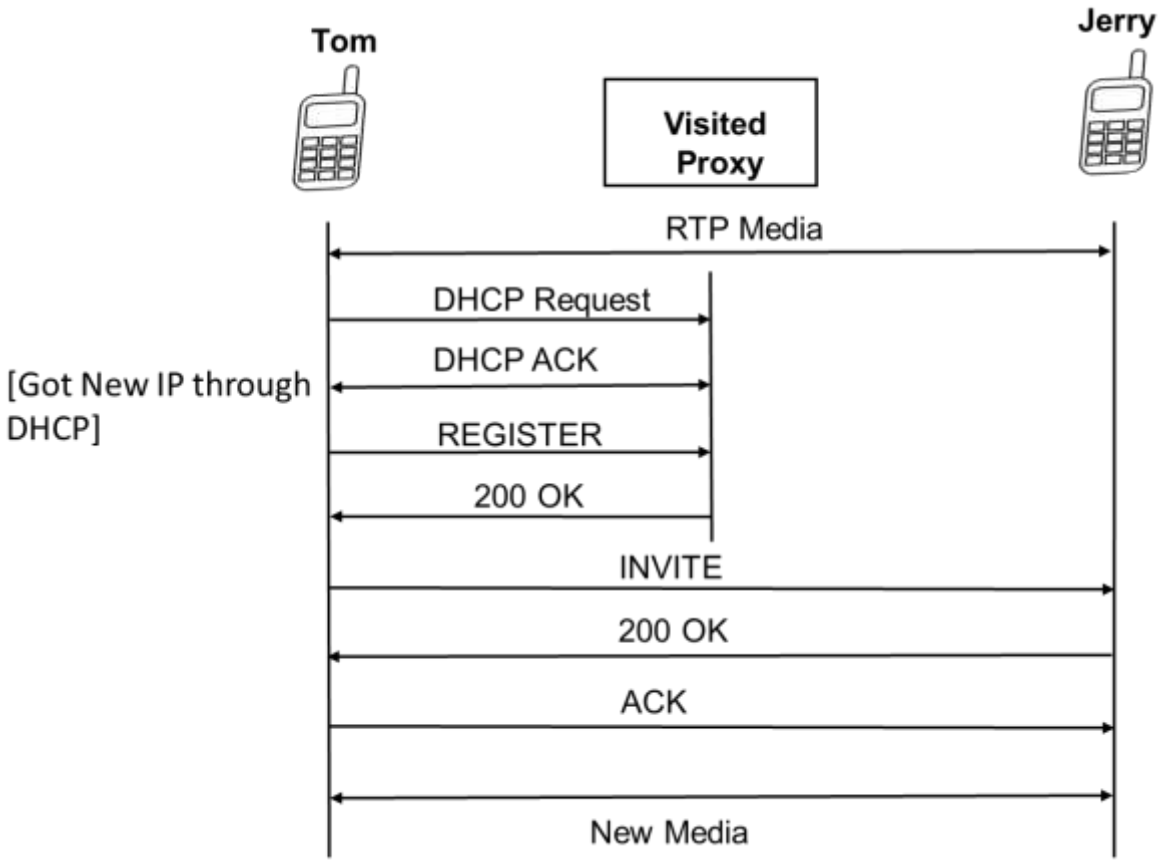
Mobility During a Call (re–Invite)

User Agent may change its IP address during the session as it swaps from one network to another. Basic SIP supports this scenario, as a re-`INVITE` in a dialog can be used to update the `Contact` URI and change the media information in the SDP.

Take a look at the call flow mentioned in the figure below.

- Here, Tom detects a new network,
- Uses DHCP to acquire a new IP address, and
- Performs a re-`INVITE` to allow the signalling and media flow to the new IP address.

If the UA can receive media from both networks, the interruption is negligible. If this is not the case, a few media packets may be lost, resulting in a slight interruption to the call.



Midcall mobility using a re-INVITE

The re-`INVITE` would appear as follows:

```
INVITE sip:Jerry@TTP.com SIP/2.0
Via: SIP/2.0/UDP 172.22.1.102:5060;branch=z9hG4bK918f5a84fe6bf7a
Max-Forwards: 70
To: <sip:Harry@TTP.com>
From: sip:Tom@PPT.com;tag=70133df4
Call-ID: 76d4861c19c
CSeq: 1 INVITE
Accept: application/sdp
Accept-Language: en
Allow: INVITE,ACK,CANCEL,BYE,INFO,OPTIONS,REFER,NOTIFY,SUBSCRIBE
Contact: <sip:172.22.1.102:5060>;
Content-Type: application/sdp
Content-Length: 168
v=0
```

```
o= PPT 40467 40468 IN IP4 192.168.2.1
s=-
c=IN IP4 192.168.2.1
b=AS:49
t=0 0
b=RR:0
b=RS:0
a=rtpmap:97 AMR/8000/1
m=audio 6000 RTP/AVP 96
a=fmtp:102 0-15
a=ptime:20
a=maxptime:240
```

The re-INVITE contains Bowditch's new IP address in the `Via` and `Contact` header fields and SDP media information.

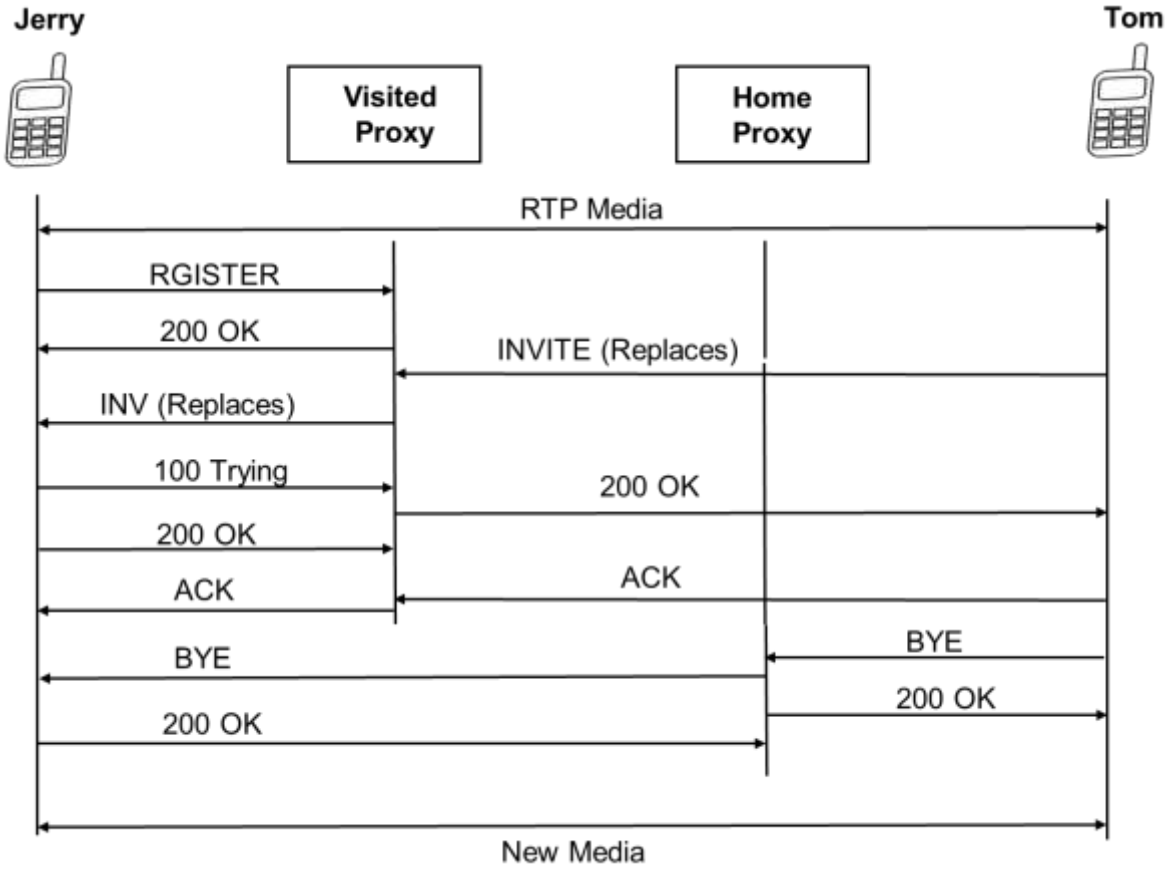
Mobility in Midcall (With replace Header)

In midcall mobility, the actual route set (set of SIP proxies that the SIP messages must traverse) must change. We cannot use a re-`INVITE` in midcall mobility.

For example, if a proxy is necessary for NAT traversal, then `Contact` URI must be changed — a new dialog must be created. Hence, it has to send a new `INVITE` with a `Replaces` header, which identifies the existing session.

Note: Suppose A & B both are in a call and if A gets another INVITE (let's say from C) with a replace header (should match existing dialog), then A must accept the INVITE and terminate the session with B and transfer all resource to newly formed dialog.

The call flow is shown in the following Figure. It is similar to the previous call flow using re-INVITE except that a `BYE` is automatically generated to terminate the existing dialog when the `INVITE` with the `Replaces` is accepted.



Midcall mobility using INVITE with Replaces

Given below are the points to note in this scenario:

- The existing dialog between Tom and Jerry includes the old visited proxy server.
- The new dialog using the new wireless network requires the inclusion of the new visited proxy server.
- As a result, an `INVITE` with `Replaces` is sent by Tom, which creates a new dialog that includes the new visited proxy server but not the old visited proxy server.
- When Jerry accepts the `INVITE`, a `BYE` is automatically sent to terminate the old dialog that routes through the old visited proxy server that is now no longer involved in the session.
- The resulting media session is established using Tom's new IP address from the SDP in the `INVITE`.

Service Mobility

Services in SIP can be provided in either proxies or in UAs. Providing service mobility along with personal mobility can be challenging unless the user's devices are identically configured with the same services.

SIP can easily support service mobility over the Internet. When connected to Internet, a UA configured to use a set of proxies in India can still use those proxies when roaming in Europe. It does not have any impact on the quality of the media session as the media always flows directly between the two UAs and does not traverse the SIP proxy servers.

Endpoint resident services are available only when the endpoint is connected to the Internet. A terminating service such as a call forwarding service implemented in an endpoint will fail if the endpoint has temporarily lost its Internet connection. Hence some services are implemented in the network using SIP proxy servers.

10. SIP Forking

Sometime a proxy server forwards a single SIP call to multiple SIP endpoints. This process is known as forking. Here a single call can ring many endpoints at the same time.

With SIP forking, you can have your desk phone ring at the same time as your softphone or a SIP phone on your mobile, allowing you to take the call from either device easily.

Generally, in an office, suppose boss unable to pick the call or away, SIP forking allow the secretary to answer calls his extension.

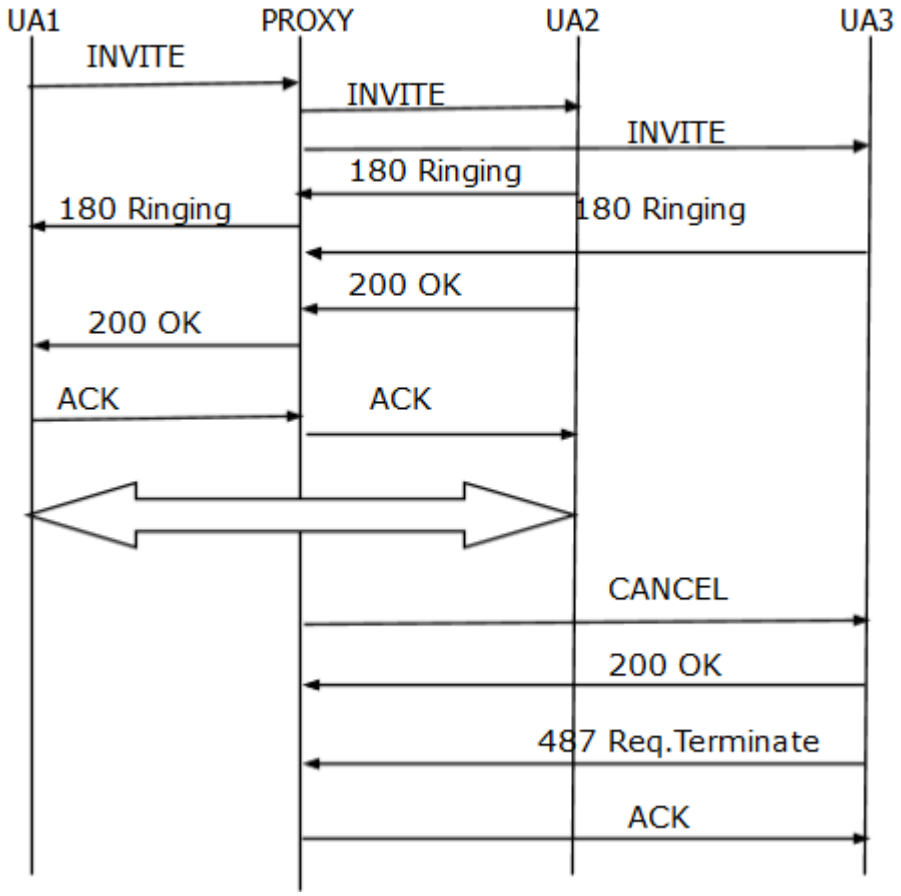
Forking will be possible if there is a stateful proxy available as it needs to perform and response out of the many it receives.

We have two types of forking:

- Parallel Forking
- Sequential Forking

Parallel Forking

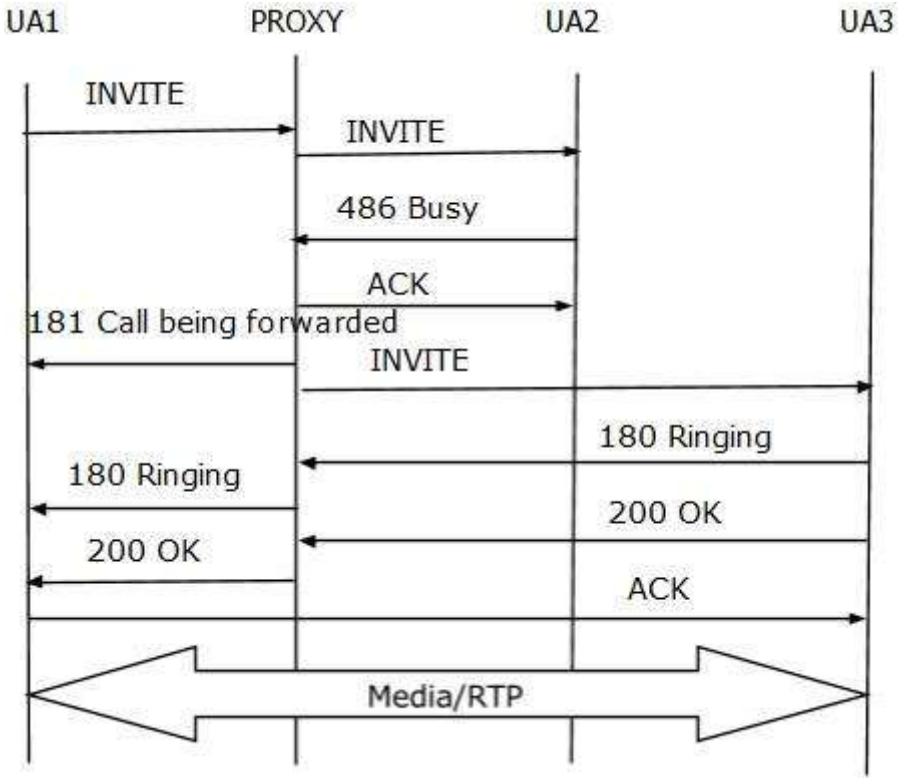
In this scenario, the proxy server will fork the INVITE to, say, two devices (UA2, UA3) at a time. Both the devices will generate 180 Ringing and whoever receives the call will generate a 200 OK. The response (suppose UA2) that reaches the Originator first will establish a session with UA2. For the other response, a CANCEL will be triggered.



If the originator receives both the responses simultaneously, then based on q-value, it will forward the response.

Sequential Forking

In this scenario, the proxy server will fork the INVITE to one device (UA2). If UA2 is unavailable or busy at that time, then the proxy will fork it to another device (UA3).



Branch ID & Tag

Branch IDs help proxies to match responses to forked requests. Without Branch IDs, a proxy server would not be able to understand the forked response. Branch-id will be available in Via header.

Tags are used by the UAC to distinguish multiple final responses from different UAS. A UAS cannot resolve whether the request has been forked or not. Therefore, it need to add a tag.

Proxies also can add tags if it generates a final response, they never insert tags into requests or responses they forward.

It may be possible that a single request can be forked by multiple proxy servers also. So the proxy which would fork shall add its own unique IDs to the branches it created.

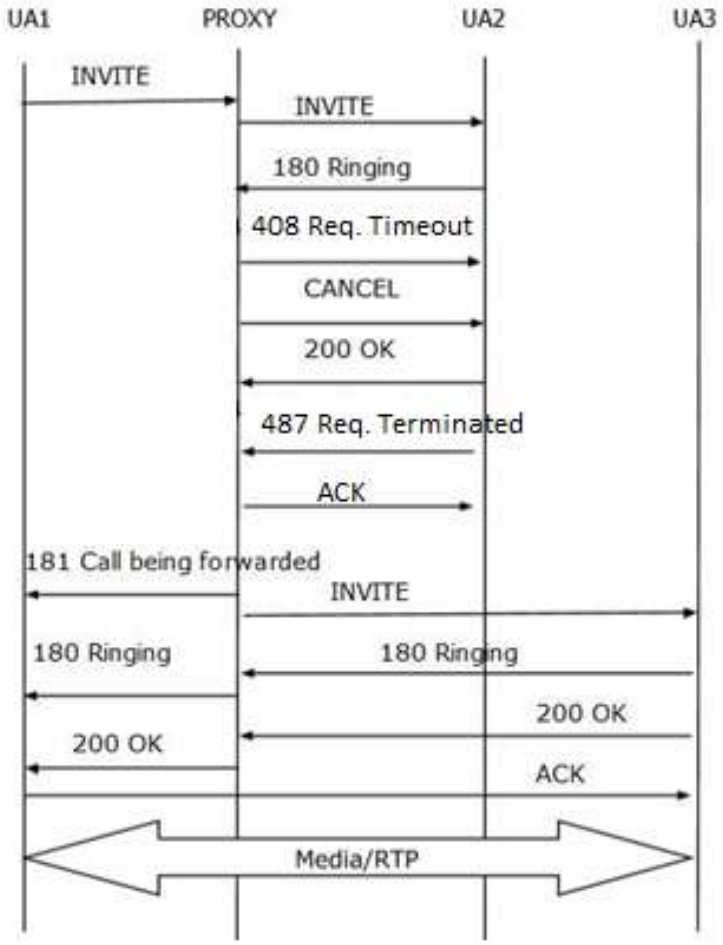
Call leg & Call ID

A call leg refers to one to one signalling relationship between two user agents. The call ID is a unique identifier carried in SIP message that refers to the call. A call is a collection of call legs.

A UAC starts by sending an INVITE. Due to forking, it may receive multiple 200 OK from different UAs. Each corresponds to a different call leg within the same call.

A call is thus a group of call legs. A call leg refers to end-to-end connection between UAs.

The CSeq spaces in the two directions of a call leg are independent. Within a single direction, the sequence number is incremented for each transaction.



Call Forward – No Answer

Voicemail

Voicemail is very common now-a-days for enterprise users. It's a telephone application. It comes to picture when the called party is unavailable or unable to receive the call, the PBX will announce to calling party to leave a voice message.

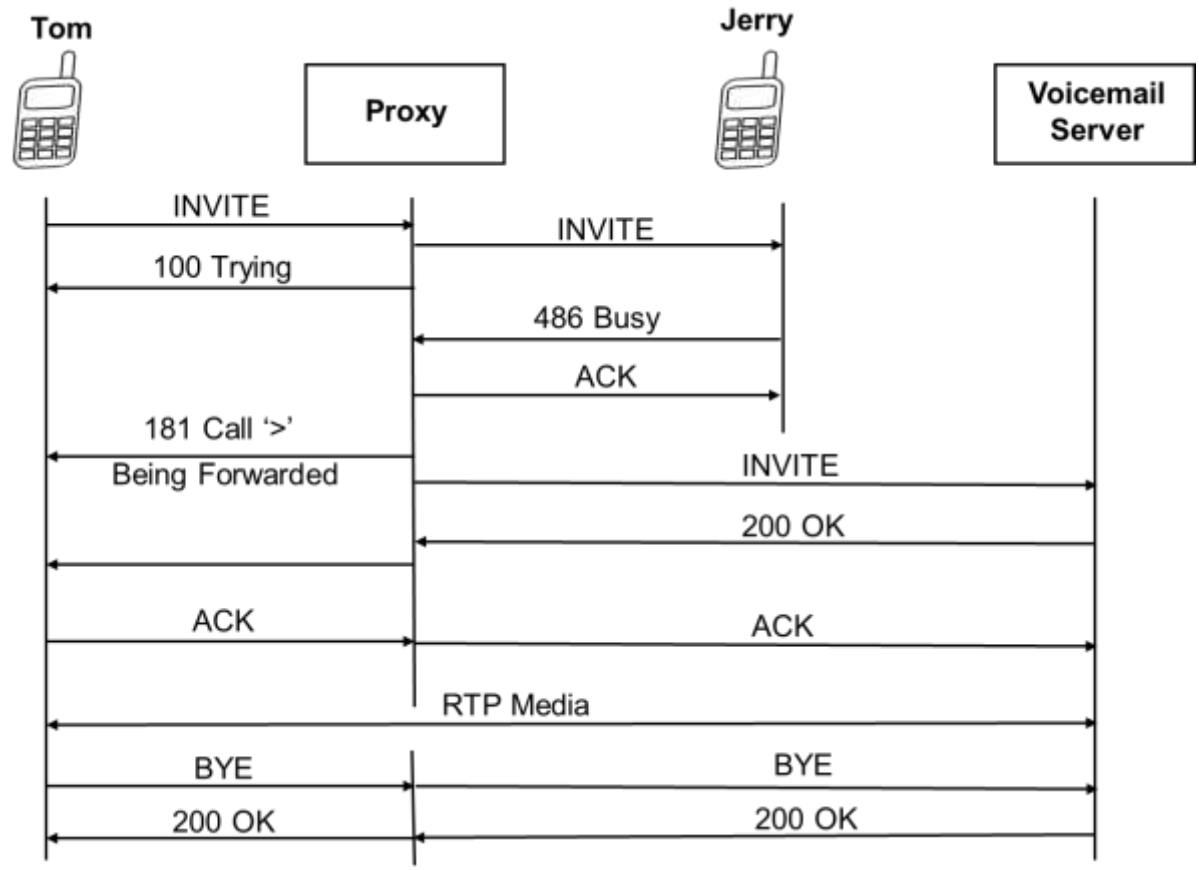
User agent will either get a 3xx response or redirect to voicemail server if the called party's number is unreachable. However, some kind of SIP extension is needed to indicate to the voicemail system which mailbox to use—that is, which greeting to play and where to store the recorded message. There are two ways to achieve this:

- By using a SIP header field extension
- By using the Request-URI to signal this information

Suppose for the user sip:Tom@tutorialspoint.com has a voicemail system at sip:voicemail.tutorialspoint.com which is providing voicemail, the Request-URI of the INVITE when it is forwarded to the voicemail server could look like:

```
sip:voicemail.tutorialspoint.com;target=sip:Tom@tutorialspoint.com;cause=486
```

The following illustration shows how the Request-URI carries the mailbox identifier and the reason (here 486).



SIP Voicemail Call Flow

11. Proxies and SIP Routing

As we know, a proxy server can be either stateless or stateful. Here, in this chapter, we will discuss more on proxy servers and SIP routing.

Stateless Proxy Server

A stateless proxy server simply forwards the message it receives. This kind of server does not store any information of the call or transaction.

- Stateless proxies forget about the SIP request once it has been forwarded.
- Transaction will be fast via stateless proxies.

Stateful Proxy Server

A stateful proxy server keeps track of every request and response that it receives. It can use the stored information in future, if required. It can retransmit the request if it does not receive a response from the other side.

- Stateful proxies remember the request after it has been forwarded, so they can use it for advance routing. Stateful proxies maintain *transaction* state. *Transaction implies transaction state,* ***not*** *call state.*
- Transaction is not as fast with stateful proxies as stateless.
- Stateful proxies can fork and retransmit if required.(e.g.: call forward busy, for example).

Via & Record-route

Record-Route

The Record-Route header is inserted into requests by proxies that wanted to be in the path of subsequent requests for the same call-id. It is then used by the user agent to route subsequent requests.

Via

Via headers are inserted by servers into requests to detect loops and to help responses to find their way back to the client. This is helpful for only responses to reach their destination.

- A UA himself generate and add its own address in a Via header field while sending request.
- A proxy forwarding the request adds a Via header field containing its own address to the top of the list of Via header fields.

- A proxy or UA generating a response to a request copies all the Via header fields from the request in order into the response, then sends the response to the address specified in the top Via header field.

- A proxy receiving a response checks the top Via header field and matches its own address. If it does not match, the response has been discarded.

- The top Via header field is then removed, and the response forwarded to the address specified in the next Via header field.

Via header fields contain `protocolname`, `versionnumber`, and `transport` (SIP/2.0/UDP, SIP/2.0/TCP, etc.) and contain `portnumbers` and `parameters` such as `received`, `rport`, `branch`.

- A `received` tag is added to a Via header field if a UA or proxy receives the request from a different address than that specified in the top Via header field.

- A `branch` parameter is added to Via header fields by UAs and proxies, which is computed as a hash function of the Request-URI, and the `To`, `From`, `Call-ID`, and `CSeq` number.

12. SIP to PSTN

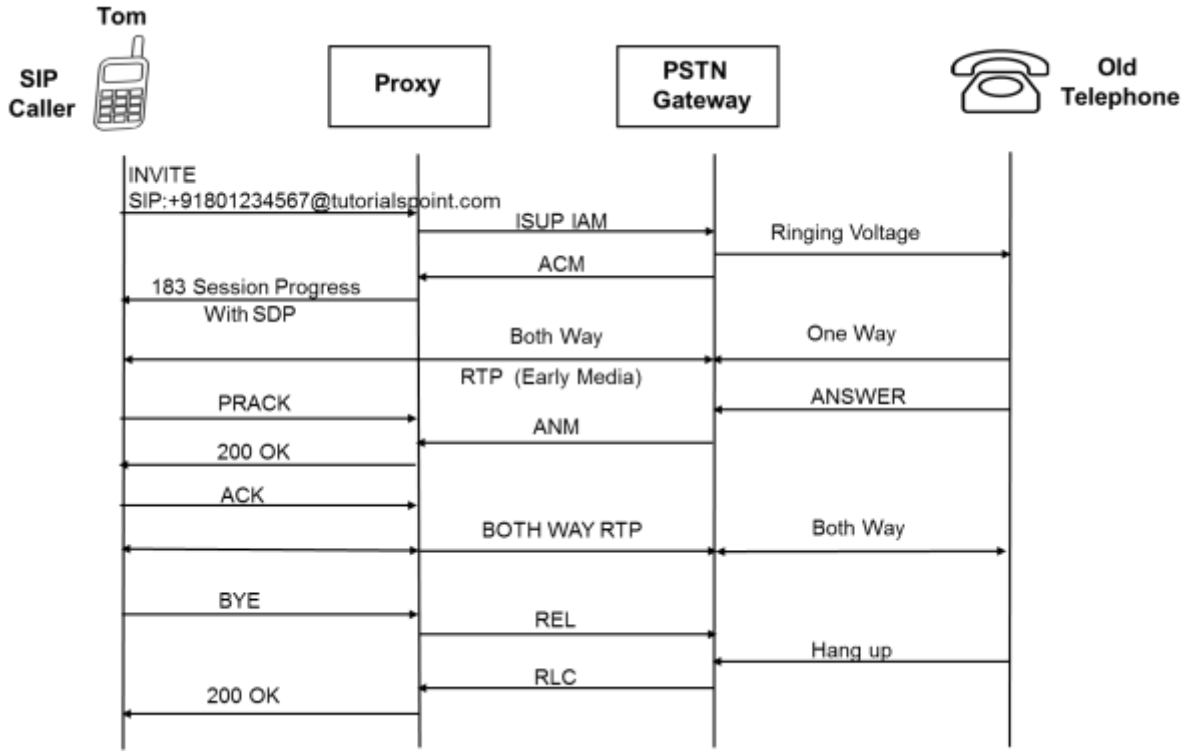
SIP (Softphone) and PSTN (Old telephone) both are different networks and speaks different languages. So we need a translator (Gateway here) to communicate between these two networks.

Let us take an example to show how a SIP phone places a telephone call to a PSTN through PSTN gateway.

In this example, Tom (sip:tom@tutorialspoint.com) is a sip phone and Jerry uses a global telephone number +91401234567.

SIP to PSTN through Gateways

The following illustration shows a call flow from SIP to PSTN through gateways.



SIP to PSTN Call Flow

Given below is a step-by-step explanation of all the process that takes place while placing a call from a SIP phone to PSTN.

1. First of all, (Tom)SIP phone dials the global number +91401234567 to reach Jerry. SIP user agent understands it as a global number and converts it into request-uri using DNS and trigger the request.

2. The SIP phone sends the INVITE directly to gateway.

3. The gateway initiates the call into the PSTN by selecting an SS7 ISUP trunk to the next telephone switch in the PSTN.

4. The dialled digits from the `INVITE` are mapped into the ISUP IAM. The ISUP address complete message (ACM) is sent back by the PSTN to indicate that the trunk has been created.

5. The telephone generates ringtone and it goes to telephone switch. The gateway maps the ACM to the `183 Session Progress` response containing an SDP indicating the RTP port that the gateway will use to bridge the audio from the PSTN.

6. Upon reception of the `183`, the caller's UAC begins receiving the RTP packets sent from the gateway and presents the audio to the caller so they know that the callee progressing in the PSTN.

7. The call completes when the called party answers the telephone, which causes the telephone switch to send an answer message (ANM) to the gateway.

8. The gateway then cuts the PSTN audio connection through in both directions and sends a `200 OK` response to the caller. As the RTP media path is already established, the gateway replies the SDP in the `183` but causes no changes to the RTP connection.

9. The UAC sends an `ACK` to complete the SIP signalling exchange. As there is no equivalent message in ISUP, the gateway absorbs the `ACK`.

10. The caller sends BYE to gateway to terminates. The gateway maps the `BYE` into the ISUP release message (REL).

11. The gateway sends the `200OK` to the `BYE` and receives an RLC from the PSTN.

13. SIP CODECS

A codec, short for coder-decoder, does two basic operations:

- First, it converts an analog voice signal to its equivalent digital form so that it can be easily transmitted.
- Thereafter, it converts the compressed digital signal back to its original analog form so that it can be replayed.

There are many codecs available in the market – some are free while others require licensing. Codecs vary in the sound quality and vary in bandwidth accordingly.

Hardware devices such as phones and gateways support several different codecs. While talking to each other, they negotiate which codec they will use.

Here, in this chapter, we will discuss a few popular SIP audio codecs that are widely used.

G.711

G.711 is a codec that was introduced by ITU in 1972 for use in digital telephony. The codec has two variants: **A-Law** is being used in Europe and in international telephone links, **u-Law** is used in the U.S.A. and Japan.

- G.711 uses a logarithmic compression. It squeezes each 16-bit sample to 8 bits, thus it achieves a compression ratio of 1:2.
- The bitrate is 64 kbit/s for one direction, so a call consumes 128 kbit/s.
- G.711 is the same codec used by the PSTN network, hence it provides the best voice quality. However it consumes more bandwidth than other codecs.
- It works best in local area networks where we have a lot of bandwidth available.

G.729

G.729 is a codec with low bandwidth requirements; it provides good audio quality.

- The codec encodes audio in frames of 10 ms long. Given a sampling frequency of 8 kHz, a 10 ms frame contains 80 audio samples.
- The codec algorithm encodes each frame into 10 bytes, so the resulting bitrate is 8 kbit/s in one direction.
- G.729 is a licensed codec. End-users who want to use this codec should buy a hardware that implements it (be it a VoIP phone or gateway).
- A frequently used variant of G.729 is G.729a. It is wire-compatible with the original codec but has lower CPU requirements.

G.723.1

G.723.1 is the result of a competition that ITU announced with the aim to design a codec that would allow calls over 28.8 and 33 kbit/s modem links.

- We have two variants of G.723.1. They both operate on audio frames of 30 ms (i.e. 240 samples), but the algorithms differ.
- The bitrate of the first variant is 6.4 kbit/s, while for the second variant, it is 5.3 kbit/s.
- The encoded frames for the two variants are 24 and 20 bytes long, respectively.

GSM 06.10

GSM 06.10 is a codec designed for GSM mobile networks. It is also known as GSM Full Rate.

- This variant of the GSM codec can be freely used, so you will often find it in open source VoIP applications.
- The codec operates on audio frames 20 ms long (i.e. 160 samples) and it compresses each frame to 33 bytes, so the resulting bitrate is 13 kbit/.

14. B2BUA

A back-to-back user agent(B2BUA) is a logical network element in SIP applications. It is a type of SIP UA that receives a SIP request, then reformulates the request, and sends it out as a new request.

Unlike a proxy server, it maintains dialog state and must participate in all requests sent on the dialogs it has established. A B2BUA breaks the end-to-end nature of SIP.

B2BUA –How it Works?

A B2BUA agent operates between two endpoints of a phone call and divides the communication channel into two **call legs**. B2BUA is a concatenation of UAC and UAS. It participates in all SIP signalling between both ends of the call, it has established. As B2BUA available in a dialog service provider may implement some value-added features.

In the originating call leg, the B2BUA acts as a *user agent server* (UAS) and processes the request as a *user agent client* (UAC) to the destination end, handling the signalling between end points back-to-back.

A B2BUA maintains the complete state for the calls it handles. Each side of a B2BUA operates as a standard SIP network element as specified in RFC 3261.

Functions of B2BUA

A B2BUA provides the following functions:

- Call management (billing, automatic call disconnection, call transfer, etc.)
- Network interworking (perhaps with protocol adaptation)
- Hiding of network internals (private addresses, network topology, etc.)

Often, B2BUAs are also implemented in media gateways to bridge the media streams for full control over the session.

Example of B2BUA

Many private branch exchange (PBX) enterprise telephone systems incorporate B2BUA logic.

Some firewalls have built in with ALG (Application Layer Gateway) functionality, which allows a firewall to authorize SIP and media traffic while still maintaining a high level of security.

Another common type of B2BUA is known as a Session Border Controller (SBC).