SIP - QUICK GUIDE

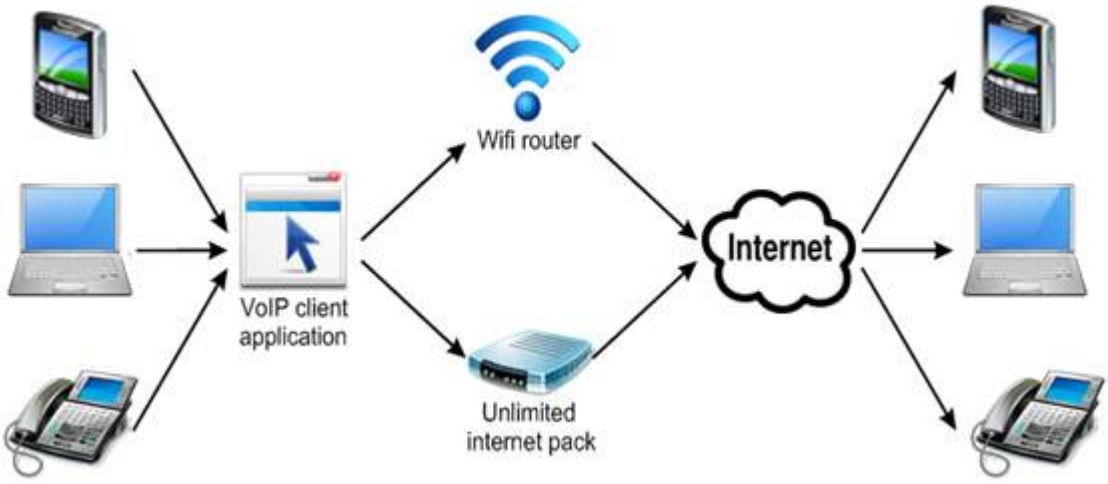
SESSION INITIATION PROTOCOL - INTRODUCTION

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 transmit multimedia content over the Internet.
- SIP is an application layer protocol defined by IETF *InternetEngineeringTaskForce* standard. It is defined in **RFC 3261.**
- SIP is incorporated with two widely used internet protocols: **HTTP** for web browser and **SMTP** used for email. From HTTP, SIP borrowed the client-server architecture and the use of URL

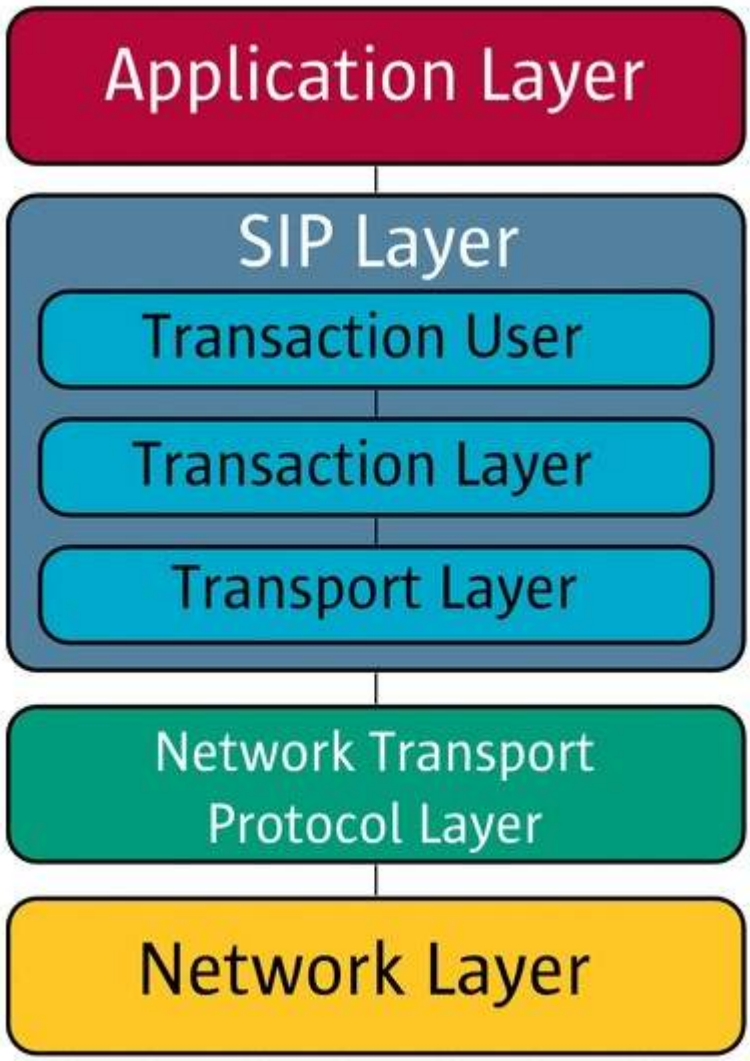
and URI. From SMTP, it borrowed a text encoding scheme and a header style.

- SIP takes the help of SDP *SessionDescriptionProtocol* which describes a session and RTP *RealTimeTransportProtocol* 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?

SIP 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.

SIP - NETWORK ELEMENTS

There are some entities that help SIP in creating its network. Inside SIP, every network element is identified by a **SIP URI** *UniformResourceIdentifier* which is like an address or identification. 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.

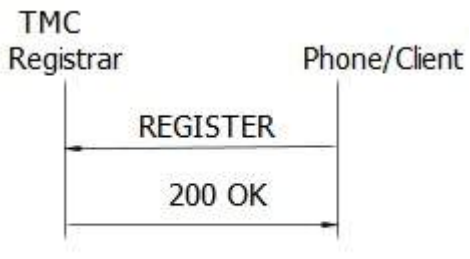
There are two types of proxy servers:

- **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.

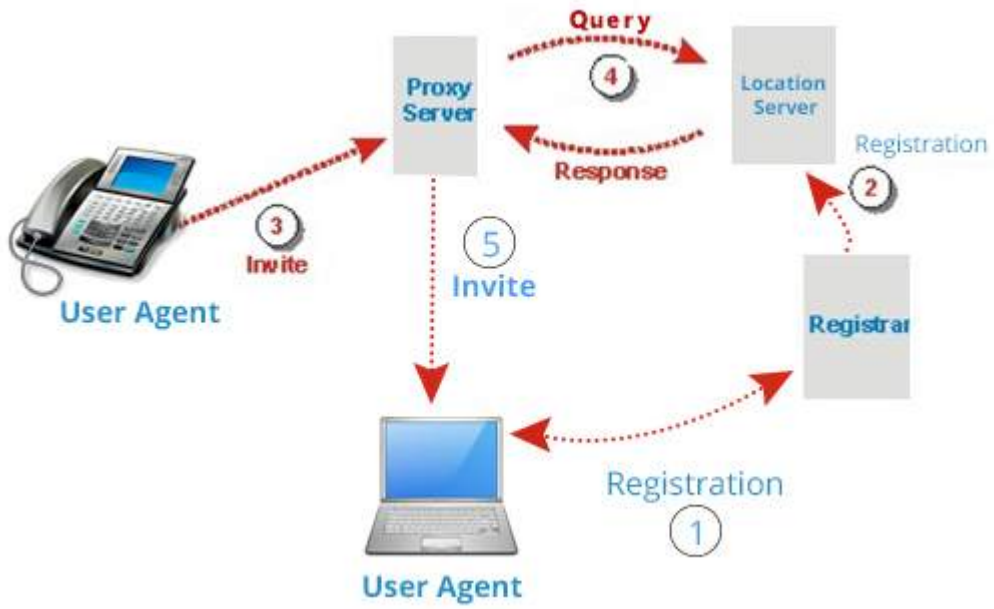
The redirect server uses the database for getting location information and responds with 3xx *Redirectresponse* to the user. We will discuss response codes later in this tutorial.

Location Server

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

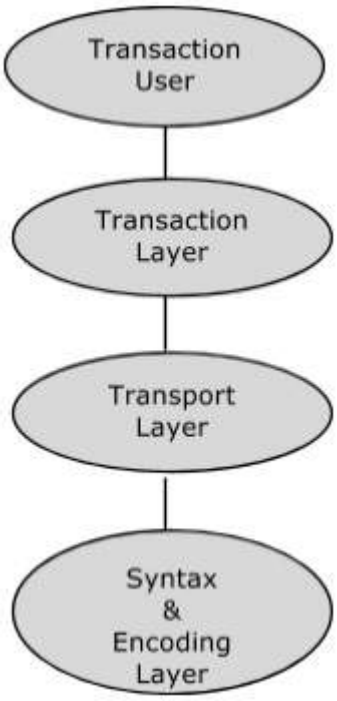
Only a proxy server or a redirect server can contact a location server.

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



SIP – System Architecture

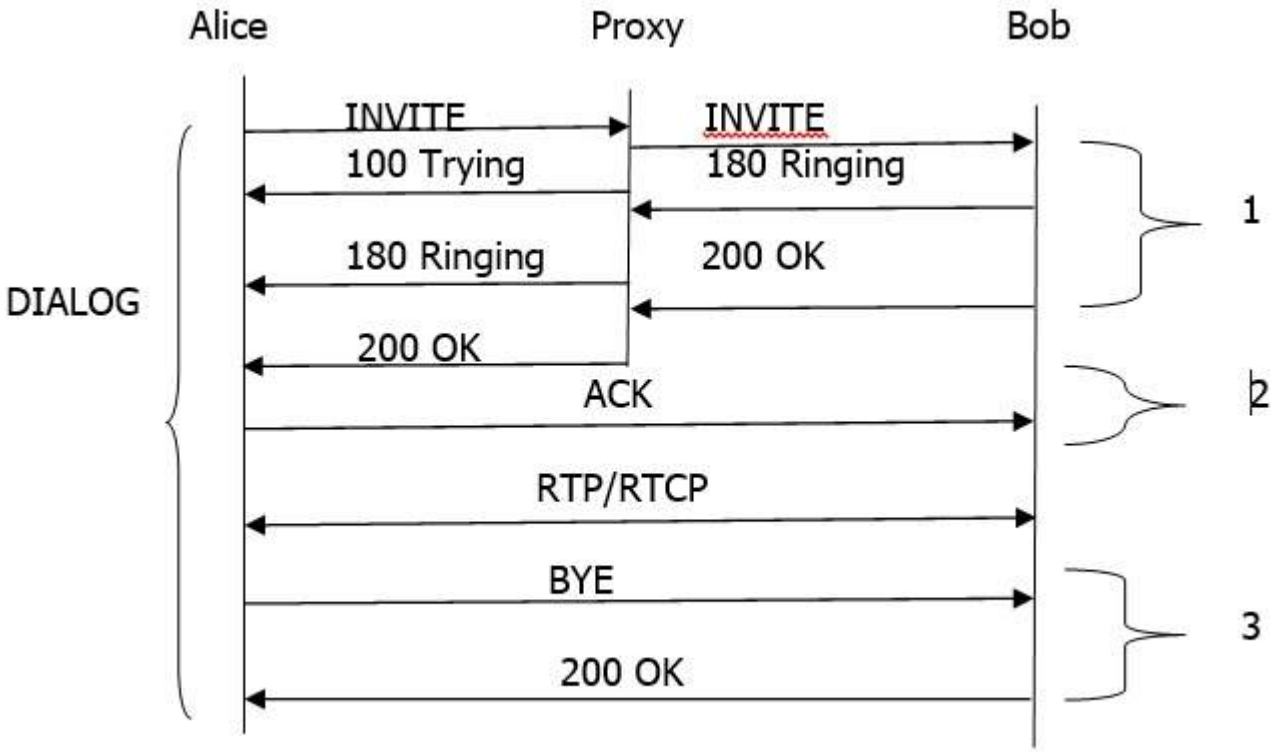
SIP is structured as a layered protocol, which means its behavior is described in terms of a set of fairly independent processing stages with only a loose coupling between each stage.



- 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 *usingthetransportlayer* 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.

SIP - 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:

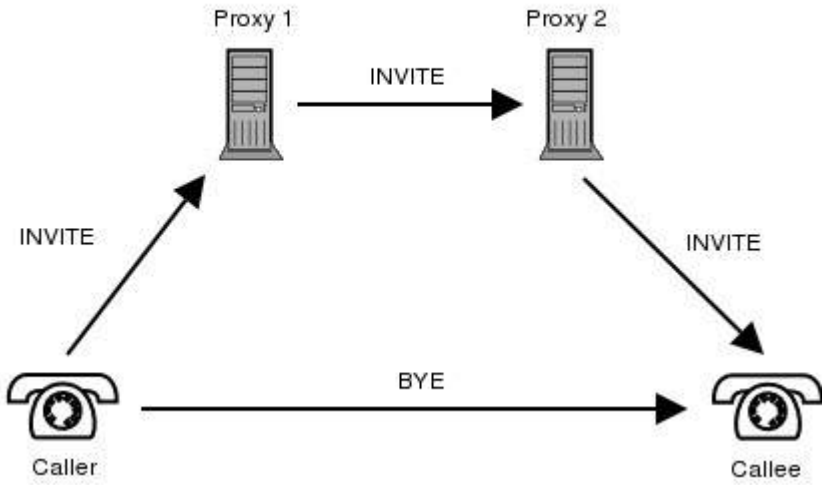
- An INVITE request that is sent to a proxy server is responsible for initiating a session.
- The proxy server sends a **100 Trying** response immediately to the caller *Alice* to stop the re-transmissions of the INVITE request.
- The proxy server searches the address of Bob in the location server. After getting the address, it forwards the INVITE request further.
- Thereafter, **180 Ringing** *Provisionalresponses* generated by Bob is returned back to Alice.
- A **200 OK** response is generated soon after Bob picks the phone up.
- Bob receives an **ACK** from the Alice, once it gets **200 OK**.
- At the same time, the session gets established and RTP packets *conversations* start flowing from both ends.

- After the conversation, any participant *AliceorBob* can send a **BYE** request to terminate the session.
- **BYE** reaches directly from Alice to Bob bypassing the proxy server.
- Finally Bob sends a **200 OK** response to confirm the BYE and the session is terminated.
- In the above basic call flow, three **transactions** are *markedas*1, 2, 3 available.

The complete call *fromINVITEto*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:

- 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.
- After getting the next route, caller's proxy server *Proxy*1, *alsoknownasoutboundproxyserver* forwards the INVITE request to the callee's proxy server which acts as an inbound proxy server *Proxy*2 for the callee.
- The inbound proxy server contacts the location server to get information about the callee's address where the user registered.
- After getting information from the location server, it forwards the call to its destination.
- Once the user agents get to know their address, they can bypass the call, i.e., conversations pass directly.

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.

There are commands known as METHODS that make a 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@atlanta.com> ;tag = 1234567
To: Bob <sips:Bob@TMC.com>
Call-ID: 12345601@ANC.com
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

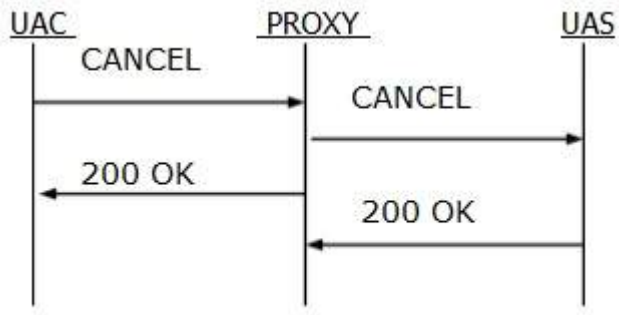
REGISTER request performs the registration of a user agent. This request is sent by a user agent to a registrar server.

- The REGISTER request may be forwarded or proxied until it reaches an authoritative registrar of the specified domain.
- It carries the AOR *AddressofRecord* in the **To** header of the user that is being registered.
- REGISTER request contains the time period 3600*sec*.
- 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 an unestablished session. 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.



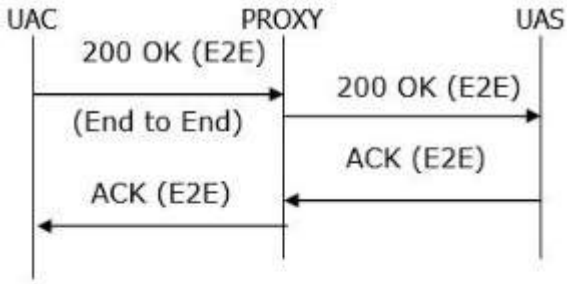
(hop by hop request)

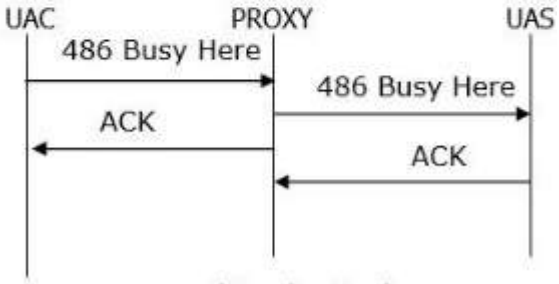
ACK

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 *mediacharacteristics*, 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.





(Hop by Hop)

- 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.

OPTION

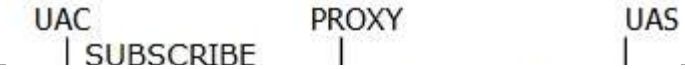
OPTIONS method is used to query a user agent or a proxy server about its capabilities and discover its current availability. The response to a request lists the capabilities of the user agent or server. A proxy never generates an OPTIONS request.

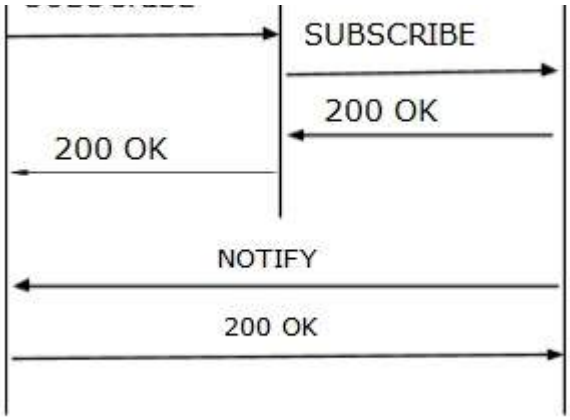
Extension Methods

Subscribe

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

- It has a time period in the **Expires** header field that indicates the desired duration of existence of a subscription.
- After the specified time period passes, the subscription is automatically terminated.
- A successful subscription establishes a dialog between the user agents.
- A subscription can be refreshed by sending another SUBSCRIBE within the dialog before the expiration time.
- The server accepting a subscription returns a 200 OK.
- Users can unsubscribe by sending another SUBSCRIBE method with Expires value 0 *zero*.





(Example of SUBSCRIBE and NOTIFY Call Flow)

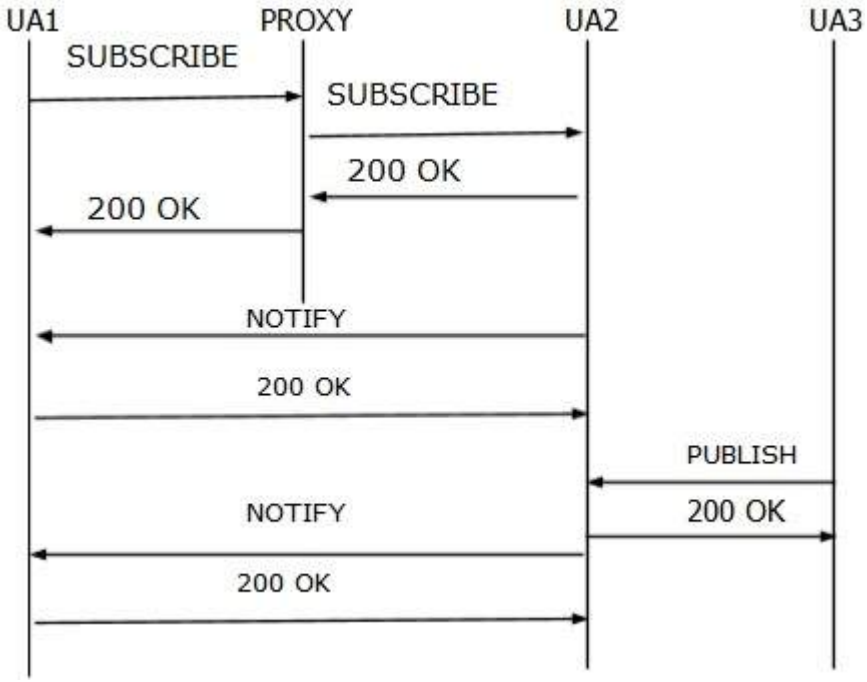
NOTIFY

NOTIFY is used by user agents to convey the occurrence of a particular event. A NOTIFY is always sent within a dialog when a subscription exists between the subscriber and the notifier.

- A 200 OK response is received for every NOTIFY to indicate that it has been received.
- NOTIFY requests contain an **Event** header field indicating the package and a **subscription-state** header field indicating the current state of the subscription.
- A NOTIFY is always sent at the start of a subscription and at the termination of a subscription.

PUBLISH

PUBLISH is used by a user agent to send event state information to a server known as an **event state compositor**.



- 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 and never generate by proxies. A proxy will always forward an INFO request.

UPDATE

UPDATE is used to modify the state of a session without changing the state of the dialog. UPDATE is used if a session is not established and the user wants to change the codec.

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

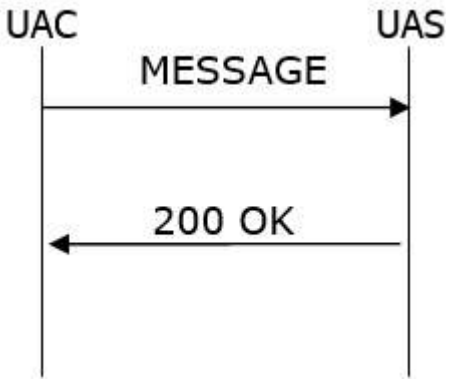
PRACK

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

- PRACK is generated by a user agent client when a provisional response has been received containing an **RSeq** reliable sequence number and a **supported:100rel** header.
- PRACK contains RSeq + CSeq value in the **rack** header.
- A PRACK may contain a message body; it may be used for offer/answer exchange.

MESSAGE

It is used to send an instant message or **IM** 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.

SIP - 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.

Class	Description	Action
1xx	Informational	This indicates the status of the call prior to completion also known as provisional response.
2xx	Success	The request has succeeded. If it was for an INVITE, Ack should be sent; otherwise, stop the retransmission of the request.
3xx	Redirection	The server has returned possible locations. The client should retry the request at another server.
4xx	Client Error	The request has failed due to an error by the client. The client may retry the request if it is reformulated according to the response.
5xx	Server Failure	The request has failed due to an error by the server. The request may be retired at another server.
6xx	Global Failure	The request has failed. The request should not be tried again at this or other servers.

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 is sent when the information may be of use to the caller.
- 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 and establishes a dialog.
- A typical use of this response is to allow a UAC to hear a ringtone, busy tone, or 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 completion or 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 permanent 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, the location is valid for the duration of the time specified.

305 Use Proxy

- This response contains a URI that points to a proxy server having 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*

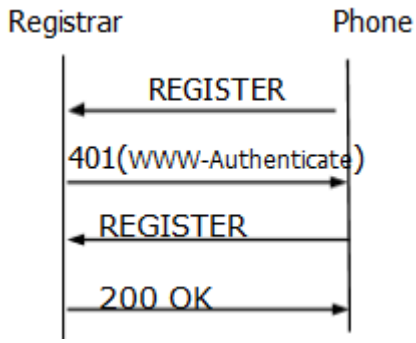
Client error responses indicate that the request cannot be fulfilled as some errors are identified from the UAC side. The response codes are generally sent by UAS. Upon receiving an error message, the client should resend the request by modifying it based on the response. Discussed below are some of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

400 Bad Request

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

401 Unauthorized

- It indicates that the request requires the user 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.

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

- 404 Not Found indicates that the user identified by the SIP URI in the Request-URI cannot be located by the server or that the user is not currently signed on with the user agent.

405 Method Not Allowed

- It indicates that the server or user agent has received and understood a request but is not willing to fulfil the request.
- Example: A REGISTER request might be sent to a user agent.

- An **Allow** field must be present to 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 must first authenticate itself with the proxy before the request can be processed.
- The response should contain information about the type of credentials required by the proxy in a **Proxy-Authenticate** header field.
- The request can be resubmitted with the proper credentials in a **Proxy-Authorization header** field.

408 Request Timeout

- This response is sent when an Expires header field is present in an INVITE request and the specified time period has passed.
- 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 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

- 500 indicates 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.

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

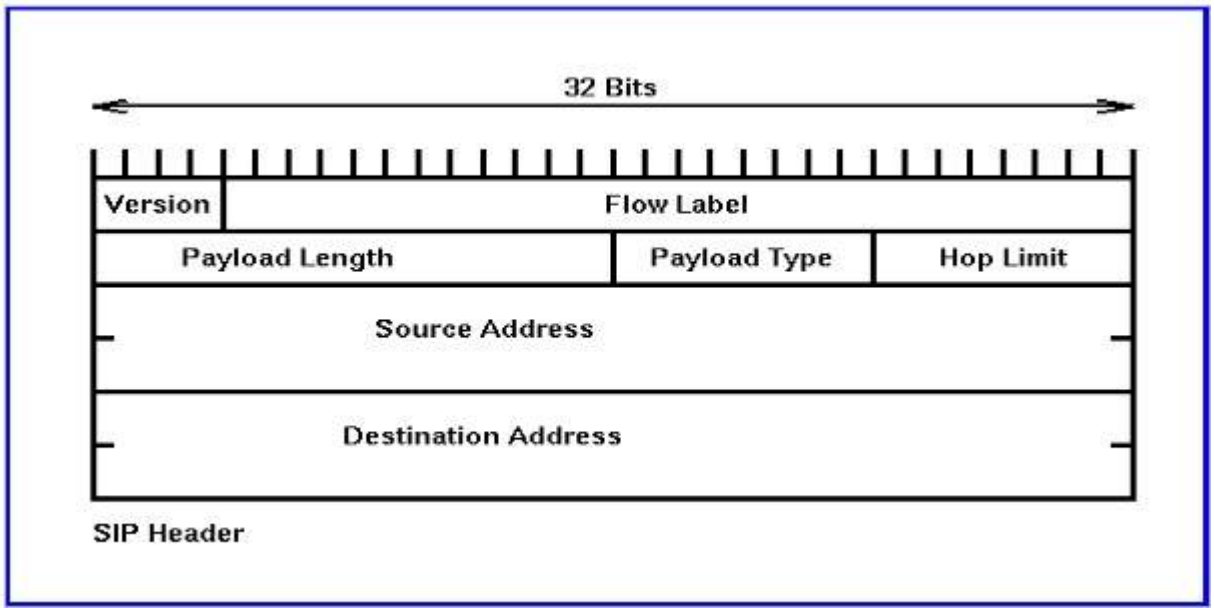
Many common SIP header fields have a compact form where the header field name is denoted by a single lowercase character.

Compact Forms of SIP Header Fields

Header Field	Compact Form
Accept-Contact	a
Allow-Event	u
Call-ID	i
Contact	m
Content-Encoding	e
Content-Length	l
Content-Type	c
Event	o
From	f
Refer-To	r
Referred-By	b
Reject-Contact	j
Subject	s
To	t
Via	v

SIP Header Format

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



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

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

SIP - SESSION DESCRIPTION PROTOCOL

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 it is 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 = mhandley2890844526 2890842807 IN IP4 126.16.64.4
s = SDP Seminar
i = A Seminar on the session description protocol
u = http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e = mjh@isi.edu(Mark Handley)
c = IN IP4 224.2.17.12/127
t = 2873397496 2873404696
a = recvonly
m = audio 49170 RTP/AVP 0
m = video 51372 RTP/AVP 31
```

```
m = application 32416udp wb
a = orient:portrait
```

SDP Extensions

There are a number of SDP extensions that have been defined. Common ones are summarized in the following table.

Common SDP Extensions

Attribute	Name	Reference
a=rtcp	Port and IP address for RTCP [6]	RFC 3605
a=mid a=group	Media session identifier and grouping of media streams [7]	RFC 3388
a=setup a=connection	Connection-oriented media using as TCP transport [8]	RFC 4145
a=key-mgt	Key management for MIKEY [9]	RFC 4567
a=crypto	Key management for SRTP [10]	RFC 4568
a=floorctrl a=confid a=userid a=floorid	Binary Floor Control Protocol (BFCP) information [11]	RFC 4583
a=fingerprint	Connection-oriented media using TLS [12]	RFC 4572
a=label	Media label [13]	RFC 4574
a=accept-types a=accept-wrapped-types a=max-size a=path	Message Session Relay Protocol (MSRP) information [14]	RFC 4975
a=ice-pwd a=ice-ufrag a=ice-lite a=ice-mismatch a=ice-options	Interactive connectivity establishment (ICE) [15]	[15]
a=chatroom	Chat room name for MSRP	[16]

- The a = setup and a = connection attributes are used for connection oriented media, such as TCP.
- The first m= media line is for a BFCP stream running over TLS over TCP.
- The a=connection:new indicates that a new TCP connection needs to be opened and that this endpoint will do a passive open the other endpoint will do the active open.
- The a = fingerprint contains a fingerprint of the certificate to be exchanged during the TLS handshake.
- The a = confide and a = userid attributes contain the conference ID and user ID of the user.
- The a = floorid attributes indicate that floor 1 is associated with a = label:1, which is associated with the m = audio stream while floor 2 is associated with a = label:2, which is associated with the m = video stream.

SIP - 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 Tesla 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 = Tesla 2890844526 2890844526 IN IP4 lab.high-voltage.org
s =-
c = IN IP4 100.101.102.103
t = 0 0
m = audio 49170 RTP/AVP 0 8
a = rtpmap:0 PCMU/8000
a = rtpmap:8 PCMA/8000
m = video 49172 RTP/AVP 32
a = rtpmap:32 MPV/90000
```

The codecs are referenced by the RTP/AVP profile numbers 0, 8, and 32.

The called party Marconi answers the call, chooses the second codec for the first media field, and declines the second media field, only wanting a PCM A-Law audio session.

```
v = 0
o = Marconi 2890844526 2890844526 IN IP4 tower.radio.org
s =-
c = IN IP4 200.201.202.203
t = 0 0
m = audio 60000 RTP/AVP 8
a = rtpmap:8 PCMA/8000
m = video 0 RTP/AVP 32
```

If this audio-only call is not acceptable, then Tesla 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=.

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

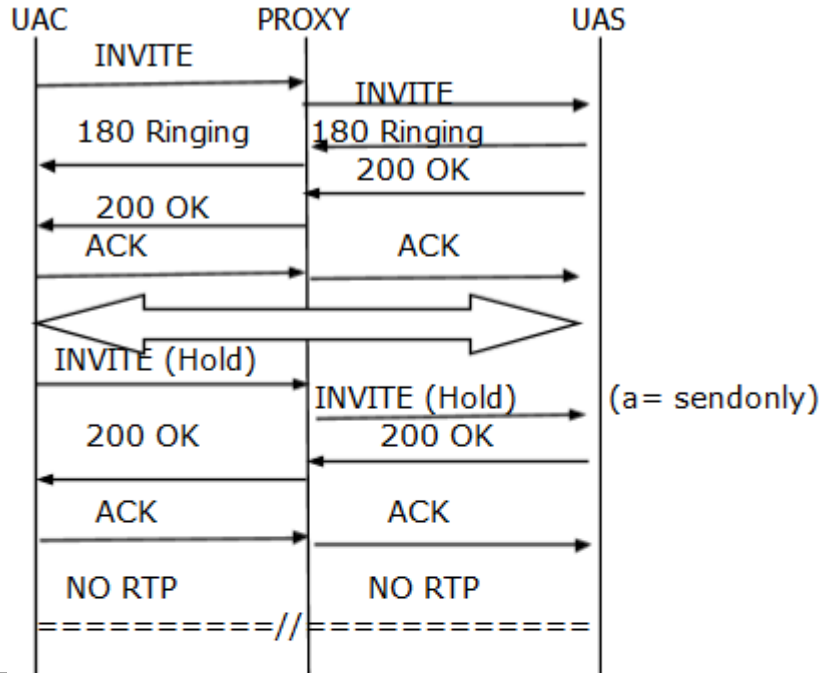
Either party can initiate another offer/answer exchange to modify a session. When a session is modified, the following rules must be followed:

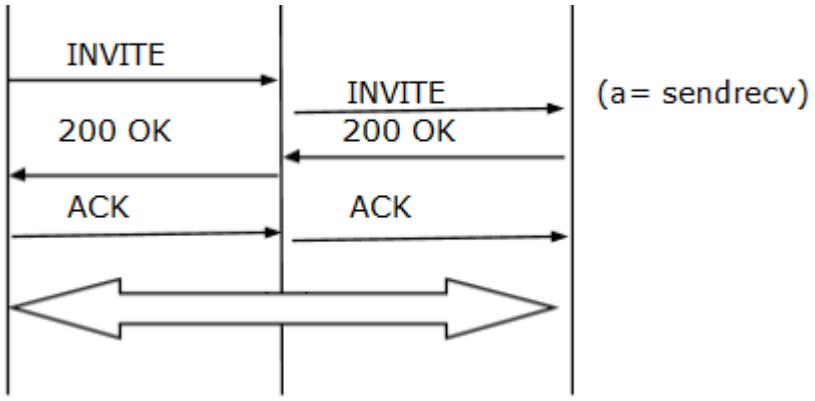
- The origin o= line version number must either be the same as the last one sent, which indicates that this SDP is identical to the previous exchange, or it may be incremented by one, which indicates new SDP that must be parsed.
- The offer must include all existing media lines and they must be sent 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)

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 HandoverPre-call

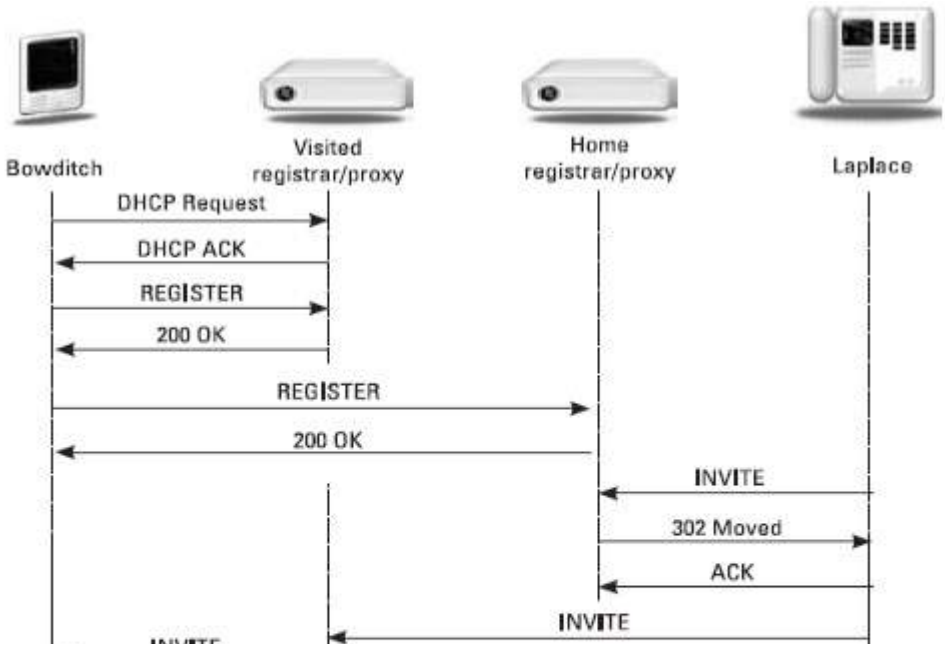
Registration in SIP temporarily binds a user's AOR Address of Record URI with a Contact URI of a particular device. As a device's IP address changes, registration allows this information to be automatically updated in the SIP network.

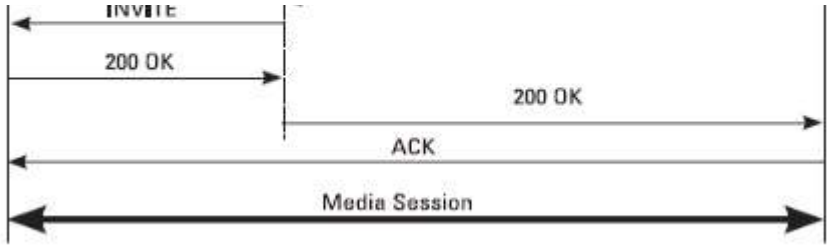
An end device can also move between service providers using multiple layers of registrations, in which a registration is actually performed with a Contact as an AOR with another service provider.

For example, consider the UA in the Figure below, which has temporarily acquired a new SIP URI with a new service provider. The UA then performs a double registration:

- The first registration is with the new service provider, 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:registrar.capetown.org SIP/2.0
Via: SIP/2.0/TLS 128.5.2.1:5060;branch=z9hG4bK382112
Max-Forwards: 70
To: Nathaniel Bowditch <sip:bowditch321@capetown.org>
From: Nathaniel Bowditch <sip:bowditch321@capetown.org>
;tag = 887865
Call-ID: 54-34-19-87-34-ar-gr
CSeq: 3 REGISTER
Contact: <sip:nat@128.5.2.1>
Content-Length: 0
```

The second registration message with the roaming URI would be:

```
REGISTER sip:registrar.salem.ma.us SIP/2.0
Via: SIP/2.0/TLS 128.5.2.1:5060;branch=z9hG4bK1834
Max-Forwards: 70
To: Nathaniel Bowditch <sip:n.bowditch@salem.ma.us>
From: Nathaniel Bowditch <sip:n.bowditch@salem.ma.us>
;tag=344231
Call-ID: 152-45-N-32-23-W3-45-43-12
CSeq: 6421 REGISTER
Contact: <sip:bowditch321@capetown.org>
Content-Length: 0
```

The first INVITE that is depicted in the Figure would be sent to sip:n.bowditch@salem.ma.us; the second INVITE would be sent tosip:bowditch321@capetown.org, which would be forwarded to sip:nat@128.5.2.1. It reaches Bowditch and allows the session to be established. Both registrations would need to be periodically refreshed.

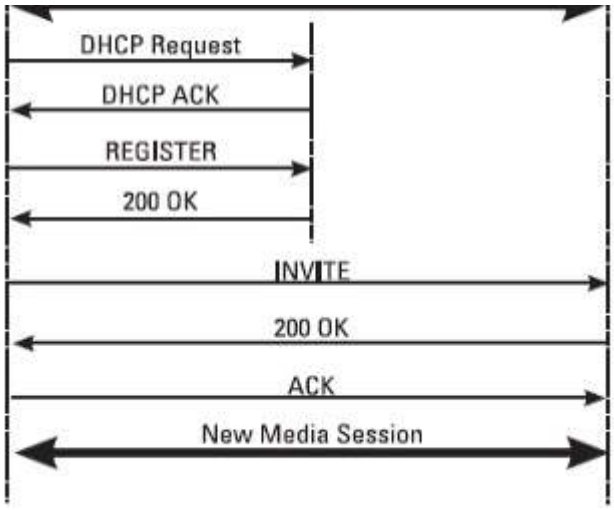
Mobility During a Callre-Invite

During a session, a mobile device may also change its IP address as it switches between one wireless 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 shown in the figure below.

- Here, Bowditch detects a new wireless network,
- Uses DHCP to acquire a new IP address, and
- Performs a re-INVITE to allow the signaling 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:laplace@client.mathematica.org SIP/2.0
Via: SIP/2.0/UDP 65.32.21.2:5060;branch=z9hG4bK34213
Max-Forwards: 70

To: Marquis de Laplace <sip:laplace@mathematica.org>
;tag=90210

From: Nathaniel Bowditch <sip:n.bowditch@salem.ma.us>
;tag = 4552345

Call-ID: 413830e4leoi34ed4223123343ed21
CSeq: 5 INVITE
Contact: <sip:nat@65.43.21.2>
Content-Type: application/sdp
Content-Length: 143

v = 0
o = bowditch 2590844326 2590944533 IN IP4 65.32.21.2
s = Bearing
c = IN IP4 65.32.21.2
t = 0 0
m = audio 32852 RTP/AVP 96
a = rtpmap:96iLBC/8000
```

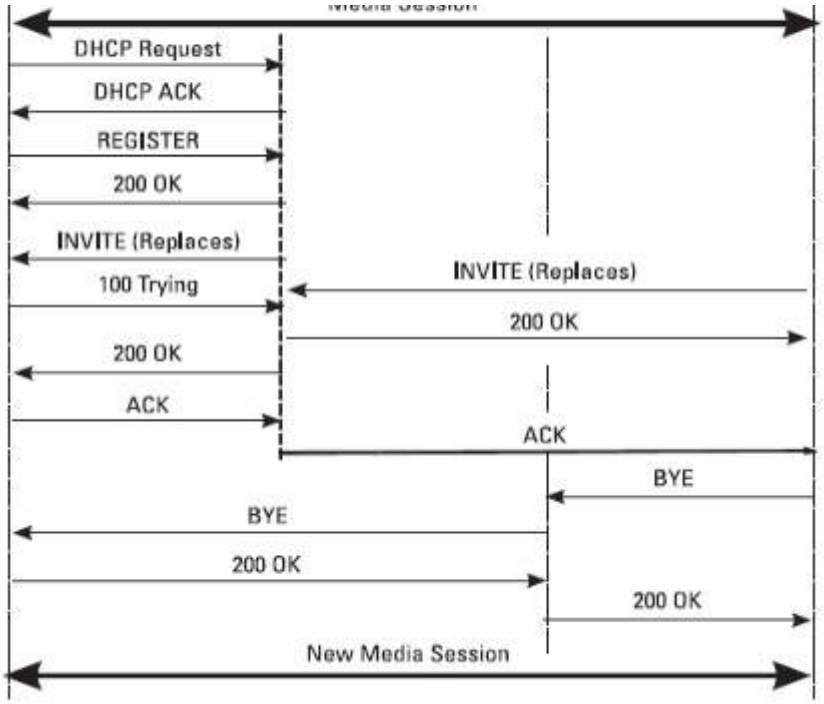
The re-INVITE contains Bowditch's new IP address in the Via and Contact header fields and SDP media information.

Mobility in MidcallWith 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/firewall traversal, then more than just the Contact URI must be changed — a new dialog must be created. The solution to this is to send a new INVITE with a Replaces header, which identifies the existing session.

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 Bowditch and Laplace 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 Bowditch, which creates a new dialog that includes the new visited proxy server but not the old visited proxy server.
- When Laplace 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 Bowditch'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.

SIP - FORKING

SIP forking refers to the process of "forking" a single SIP call to multiple SIP endpoints. This is a very powerful feature of SIP. 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. No forwarding rules would be necessary as both devices would ring.

In the same manner, SIP forking can be used in an office and allow the secretary to answer calls to the extension of his/her boss when he is away or unable to take the call.

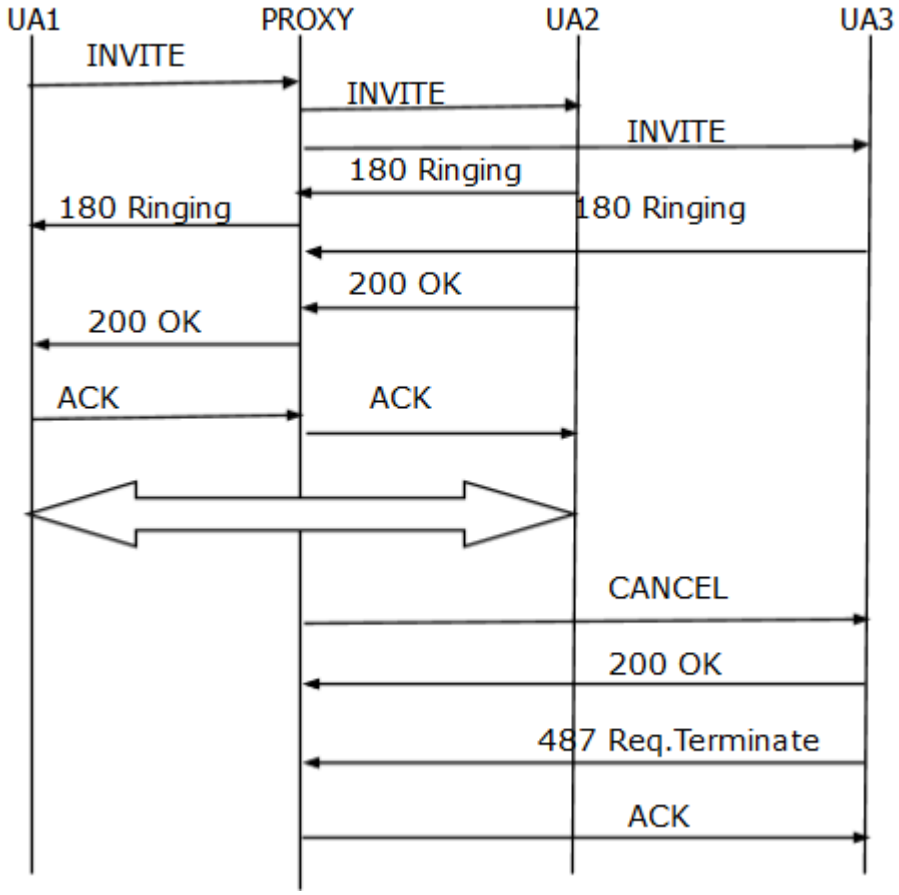
Note that a forking SIP proxy cannot be stateless because it needs to perform a filtering operation, returning one response out of the many it receives.

We have two types of forking:

- Parallel Forking
- Sequential Forking

Parallel Forking

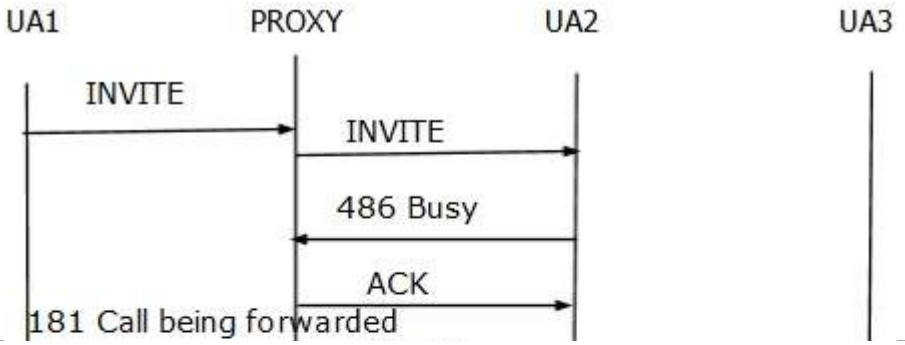
In this scenario, the proxy server will fork the INVITE to, say, two devices UA2, UA3 at a time. Both the devices will generate 180 Ringing and whoever receives the call will generate a 200 OK. The response suppose UA2 that reaches the Originator first will establish a session with UA2. For the other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

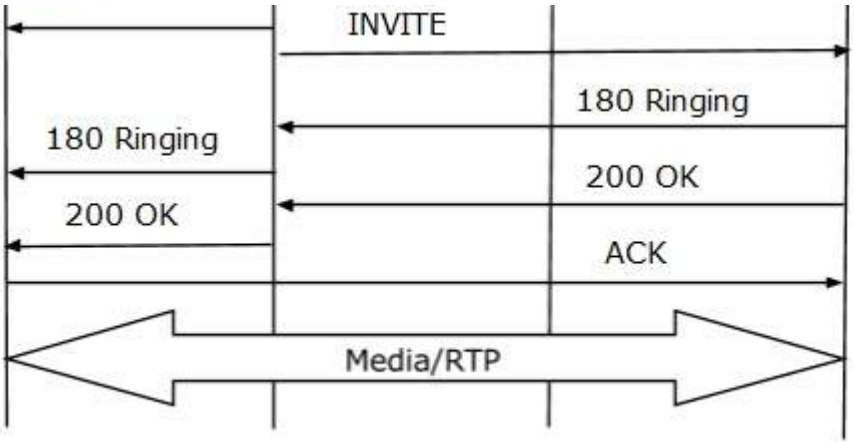


If the originator receives both the responses simultaneously, then based on q-value, it will forward the response.

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 & TagID

Branch IDs allow proxies to match responses to forked requests. Without them, a proxy wouldn't be able to tell which branch a response corresponds to.

Tags are used by the UAC to distinguish multiple final responses from different UAS. A UAS has no reliable way of determining if the request has been forked or not. To be safe, it needs to add a tag. Proxies only insert tags into the final responses they generate themselves; they never insert tags into requests or responses they forward.

Since a request can be forked several times on its way to UAS, a single "tag" added to the request by one of the proxies is not sufficient for the next forking proxy along the chain to match responses on its own branches; every proxy that forked the request would need to add its own unique IDs to the branches it created.

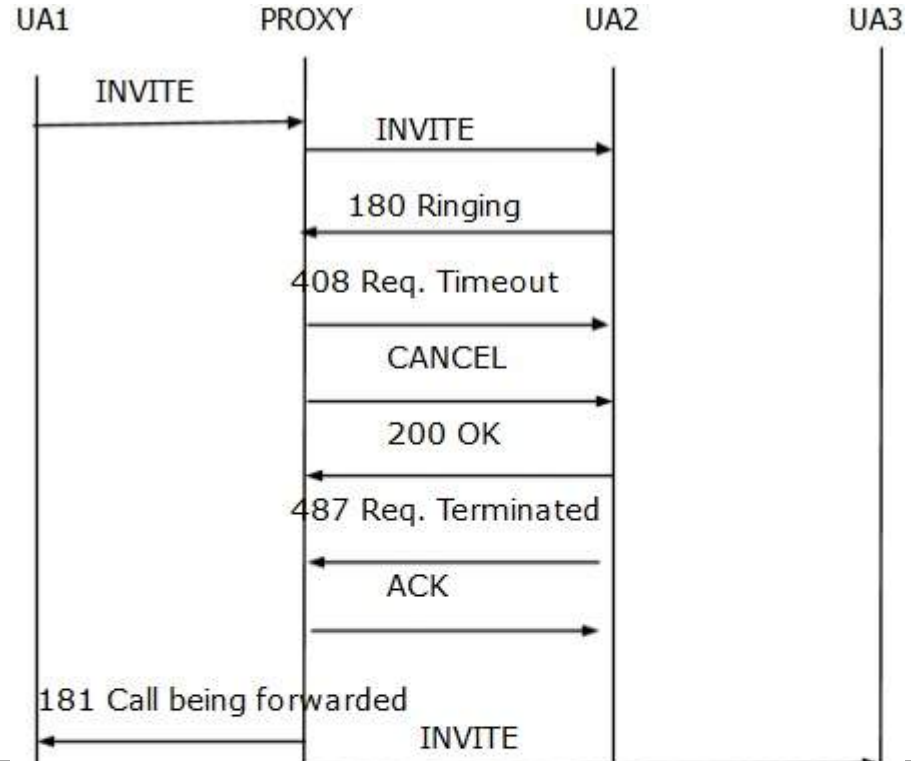
Call leg & Call ID

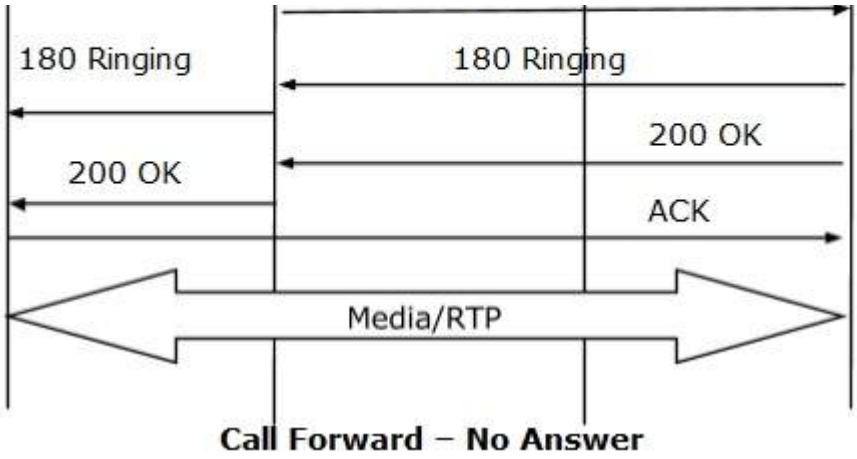
A call leg refers to one to one signalling relationship between two user agents. The call ID is an identifier carried in SIP message that refers to the call. A call is a collection of call legs.

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 a messaging service commonly associated with telephony applications. It can be implemented as a service in a network as provided by mobile phone providers, in a separate device such as a home answering machine, or incorporated in a telephony device such as an enterprise PBX or key system.

Voicemail involves call forwarding no answer/busy/unavailable to a storage device which plays a customizable greeting. The user is then alerted by some means that a message is waiting, and can then retrieve the message by dialling into the system.

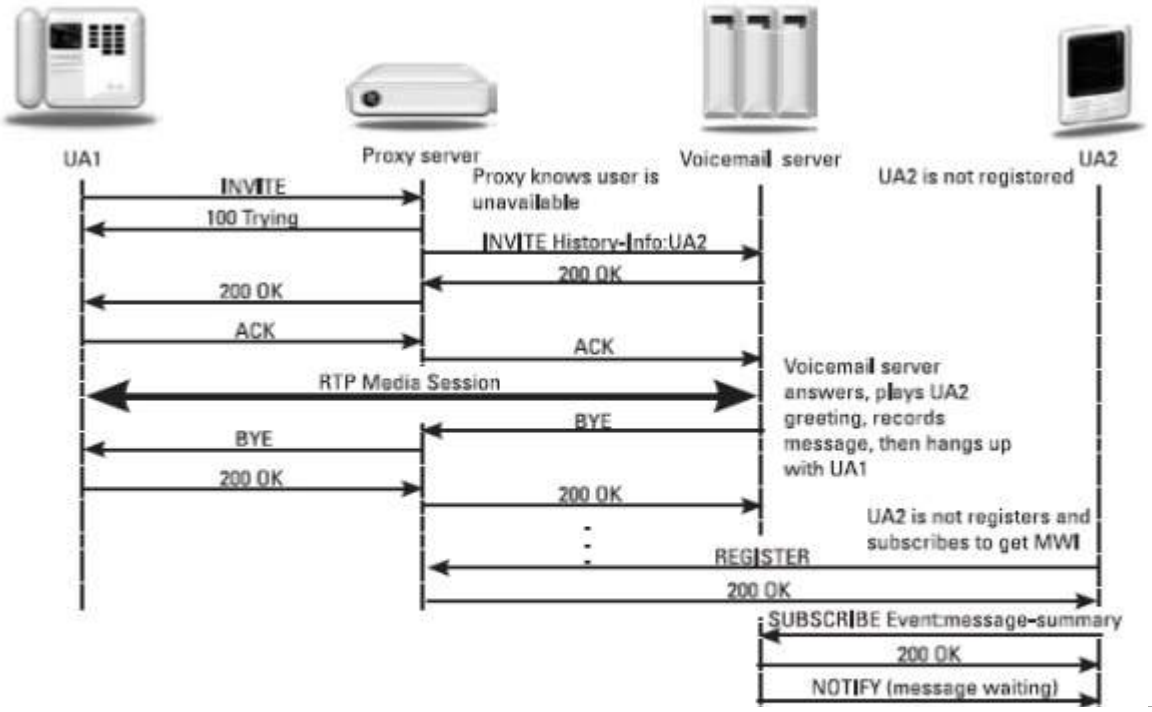
In SIP terms, the call forwarding is straightforward, with either a proxy forwarding or endpoint redirection 3xx response used to send the call to the voicemail server. However, some kind of SIP extension is needed to indicate to the voicemail system which mailbox to use—that is, which greeting to play and where to store the recorded message. We have two ways to do this:

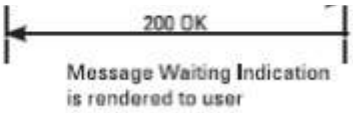
- Use a SIP header field extension
- Use the Request-URI to signal this information

For a voicemail system at sip:voicemail.example.com, which is being used to provide voicemail for sip:alice@example.com, the Request-URI of the INVITE when it is forwarded to the voicemail server could look like:

```
sip:voicemail.example.com; target = sip:alice@example.com; cause = 486
```

In this way, the Request-URI carries the mailbox identifier as well as the reason the call is being forwarded to the voicemail.





SIP Voicemail Callflow

SIP - PROXIES & 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 type of server does not store any information of the call or transaction.

- Stateless proxies forget about the SIP request once it has been forwarded.
- Stateless proxies scale very well, and can be very fast. They are good for network cores.

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 associate the response with some internal state. In other words, stateful proxies maintain *transaction* state. Stateful *implies* transaction state, **not** call state.
- Stateful proxy servers don't scale as much as stateless ones.
- Stateful proxies can fork and provide services that stateless ones can't call forward busy, for example.

Neither stateful nor stateless proxies need to maintain call state, although they can.

Via & RecordVia-route

Record-Route

The Record-Route header is inserted into requests by proxies that want 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. The mechanism is similar to a source-route, copying the Record-Route information into a set of Route headers. The Request-URI is set to the first Route header.

Via

Via headers are inserted by servers into requests to detect loops and to allow responses to find their way back to the client. They have no influence on the routing of future requests or responses.

- 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.

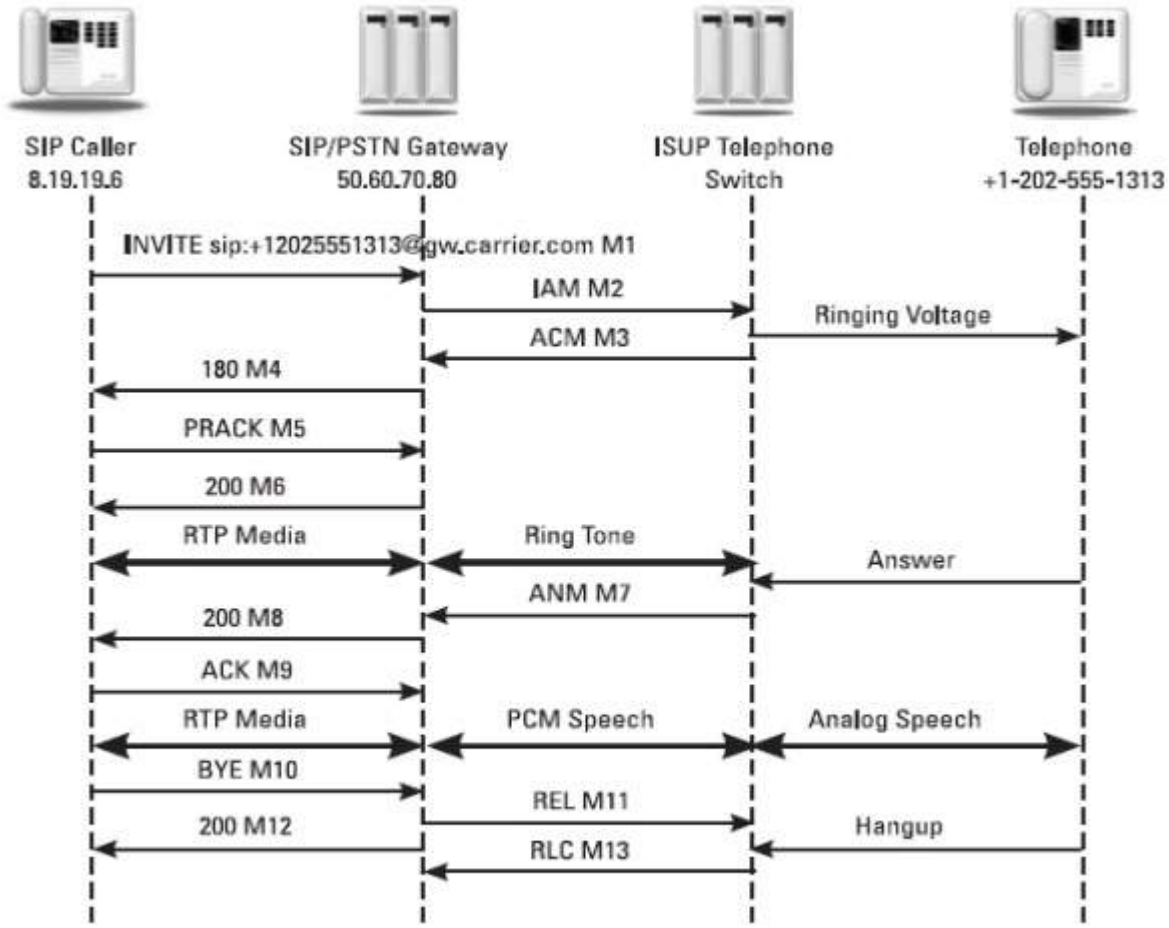
- A received tag is added to a Via header field if a UA or proxy receives the request from a different address than that specified in the top Via header field.
- A branch parameter is added to Via header fields by UAs and proxies, which is computed as a hash function of the Request-URI, and the To, From, Call-ID, and CSeq number.

SIP TO PSTN

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

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.

- First of all, SIP phone collects the dialled digits and puts them into a SIP URI used in the Request-URI and the To header. The caller may have dialled either the globalized phone number 1-202-555-1313 or a local number 555-1313, and the SIP phone added the assumed country code and area code to produce the globalized URI using the built-in dial plan.
- The SIP phone has been preconfigured with the IP address of the PSTN gateway, so it is able to send the INVITE directly to gw.carrier.com.
- The gateway initiates the call into the PSTN by selecting an SS7 ISUP trunk to the next telephone switch in the PSTN.

- 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 seized.
- In this example, ringtone is generated by the far-end 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.
- 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 call is progressing in the PSTN.
- The call completes when the called party answers the telephone, which causes the telephone switch to send an answer message ANM to the gateway.
- 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 echoes the SDP in the 183 but causes no changes to the RTP connection.
- The UAC sends an ACK to complete the SIP signalling exchange. As there is no equivalent message in ISUP, the gateway absorbs the ACK.
- The call terminates when the caller sends the BYE to the gateway. The gateway maps the BYE to the ISUP release message REL.
- The gateway sends the 200 OK to the BYE and receives an RLC from the PSTN.

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, the bandwidth required, the computational requirements, etc.

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/.

SIP - 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.

B2BUA – How it Works?

A B2BUA agent operates between two endpoints of a phone call and divides the communication channel into two **call legs**. The B2BUA agent mediates all SIP signalling between both ends of the call, from call establishment to termination. For each call, all the control messages flow through the B2BUA, hence a service provider may implement value-added features available during the call.

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.

A B2BUA breaks the end-to-end nature of SIP.

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 ALG functionality built in, which allows a firewall to permit 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.