**sip**

**CONTENTS:**

|  |
| --- |
| **SIP (BASICS)** |
| **SIP (SERVERS)** |
| **SIP (HEADER FIELDS)** |
| **SIP (REQUEST-METHODS)** |
| **SIP (CALL-FLOW)** |
| **SIP (RESPONSE-CODES)** |
| **SIP (DEFINITIONS)** |

**SIP (SESSION INITIATION PROTOCOL):**

The **Session Initiation Protocol (SIP)**, is an application layer control (Signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet Telephone Calls, Multimedia Distribution, and Multimedia Conferences. SIP clients traditionally use TCP and UDP(port 5060) to connect to SIP servers and other SIP endpoints. SIP is primarily used in setting up and tearing down voice or video calls. SIP is a peer to peer protocol.

In SIP, a **session** is considered an exchange of data between associations of participants. SIP can also invite participants to already existing sessions, such as multi cast conferences. Media can be added to an existing session.

**SIP ARCHITECTURE:**

SIP is a signaling protocol that handles the set-up, modification, and tears down of multimedia sessions. Media is normally exchanged by using RTP as the transport protocol. Signaling is passed via one or more proxy or redirect servers while the media stream takes a more direct path.

**SIP NETWORK ENTITIES:**

SIP has two basic classes of network entities.

1.Clients

2.Servers

**Clients:** Often termed as User Agent Clients(UAC) is an application program that sends SIP requests.

**Servers:** Often termed as User Agent Servers(UAS) is an entity that responds to requests.

Thus, SIP is a Client-Server protocol.

**Structure of SIP Protocol:**

SIP is structured as a layered protocol, which means that its behavior is described in terms of a set of fairly independent processing stages.

The lowest layer of SIP is its **Syntax and Encoding**. Its encoding is specified using **ABNF** grammar.

The second layer 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.

The third layer is the **Transaction Layer**. A Transaction is a request sent by a client to a server, along with all responses to that request sent from the server back to the client. The transaction layer handles application layer retransmission, matching of responses to requests, and application layer timeouts.

The layer above the transaction layer is called the **Transaction User** (TU). Each SIP entities except the state less proxy, is a Transaction User. When a TU wishes to send a request, it creates a client transaction instance and passes it the request along with the destination IP address, port, and transport to which to send the request. A TU, that creates a client transaction can also Cancel it. When a client Cancels a transaction, it requests that the server stops further processing; revert to the state that existed before the transaction was initiated, and generate a specific error response to that transaction. This is done with a Cancel request, which constitutes its own transaction.

**Types of SERVERS:**

In SIP, there are four types of servers.

PROXY

REDIRECT

USER AGENT SERVER

REGISTRAR

**PROXY SERVER:** Clients send requests to the proxy, and the proxy either handles those requests itself, or forwards them to other servers; after performing some transaction. A proxy receives requests and sends responses, thus it acts both as a client and as a server. A proxy server primarily plays the role of routing. Proxies can be of two types.

Stateful Proxy.

Stateless Proxy.

**STATEFUL PROXY SERVER:** A stateful proxy is a logical entity that maintains the client and server transaction state machines during the processing of a request. A stateful proxy server keeps track of requests and responses received in the past and use that information in processing future requests and responses.

**STATELESS PROXY SERVER:** A stateless proxy is a logical entity that does not maintain the client or server transaction state machines when it processes a request. A stateless proxy forwards every requests it receives downstream and every response it receives upstream.

**REDIRECT SERVER:** It is a server that accepts SIP requests, maps the destination address to zero or more new addresses, and returns the translated address to the originator of the request. Thereafter, the originator of the request can send requests to the address returned by the redirect server. A redirect server does not initiate any SIP requests of its own. A redirect server generates 3xx responses to requests it receives.

**USER AGENT SERVER:** A User Agent Server accepts SIP requests and contacts the user. A response from the user to the user agent server results in a SIP response on behalf of the user. In reality, a SIP device functions as both a User Agent Client and as a User Agent Server. Acting as a User Agent Client, the SIP device can initiate SIP requests. Acting as a User Agent Server, the device can receive and respond to SIP requests. In practical terms, the device can initiate and receive calls, which enable SIP to be used for peer-to-peer communication.

**REGISTRAR SERVER:** This server accepts SIP REGISTER requests. SIP includes the concept of user registration, whereby a user indicates to the network that he or she is available at a particular address. Typically, a registrar is combined with a proxy or redirect server.

**SYNTAX OF SIP REQUEST:**

Request-line=<Method Name>space<Request URI>space<Protocol Version>

**SYNTAX OF SIP RESPONSE:**

Response-line=<SIP Version>space<Status Code>space<Reason Phrase>

**Various Types of SIP Requests:**

INVITE

REGISTER

BYE

ACK

CANCEL

OPTION

REFER

SUBSCRIBE

NOTIFY

MESSAGE

INFO

PRACK

UPDATE

Every SIP message consists of some mandatory Header Fields.

**HEADER:** A Header is a component of a SIP message that conveys information about the message.

**HEADER FIELD:** A Header Field is a component of the SIP message header. A header field can appear as one or more header field rows. Header Field rows consist of a header field name and zero or more header field values.

There are six mandatory Header Fields, which are present in all SIP requests. They are:

TO

FROM

VIA

MAX-FORWARDS

C-SEQ

CALL-ID

**TO:** It is a required header field in every SIP message. It is used to indicate the recipient of the request. Any responses generated by a user agent contains this header field with the addition of a tag. Any response generated by a proxy, must have a tag added to the “To” header field. A tag added to the header field in a 200 OK response is used throughout the call.

**FROM:** The “From” header field indicates the originator of the request. It identifies the call originator's address. A “From” header field may contain a tag, which is used to identify a particular call. It may contain a display name, where the URI is enclosed in <>. If there is both the URI parameter and a tag, then the URI including any parameters is enclosed in <>.

**VIA:** VIA header field is used to record the SIP route taken by a request and is used to route a response back to the originator. A user agent generating a request records its own address in a VIA header field in the request. VIA is used to route responses. 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 adding a VIA header field always includes a branch tag. A proxy receiving a response checks the top VIA header field to ensure that it matches its own address. The top VIA header field is then removed, and the response forwarded to the address specified in the next VIA header field.

VIA header field contains protocol name and version number and transport(SIP/2.0/UDP or TCP)

and may contain port numbers and parameters such as received, maddr, and ttl. **The VIA header field MUST contain a branch parameter. The branch parameter is used to identify the transaction created by that request. This parameter is used by both the client and the server.** The branch parameter value MUST be unique across space and time for all requests sent by the UA. The exceptions to this rule are CANCEL and ACK for non-2xx responses. A CANCEL request will have the same value of the branch parameter as the request it cancels. As discussed in Section 17.1.1.3, an ACK for a non-2xx response will also have the same branch ID as the INVITE whose response it acknowledges.

The “maddr” and “ttl” parameters are used for multi cast transport and have a similar meaning as the equivalent SIP URI parameters. A received tag is added to a VIA header field if a user agent or proxy receives the request from a different address than that specified in the top of the VIA header field. This indicates that a firewall proxy is in the message path.

**MAX-FORWARDS:** This 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. The suggested initial value is 70 hops. 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.

**C-SEQ:** The C-Seq header field serves as a way to identify and order transactions. It consists of a sequence number and a method. The method MUST match that of the request. For non REGISTER requests outside of a dialog, the sequence number value is arbitrary. Its value MUST be expressible as a 32 bit unsigned integer and MUST be less than 2\*\*31. Usually, it increases by 1 for each new request, with the exception of CANCEL and ACK request; which use the c-seq number of the INVITE request to which it refers.

**CALL-ID:** The Call-ID header field acts as a unique identifier to group together a series of messages. It MUST be the same for all requests and responses sent by either user agent in a dialog. It SHOULD be the same in each registration from a user agent. Call-ID could be a local-id, the @ symbol, and a host name or IP address. If a proxy challenges for authentication then the retried requests are not considered new requests, and therefore do not need new Call-ID header fields.

**INVITE:** INVITE method is used to establish media sessions between user agents. INVITE usually has a message body containing the media information of the caller. A media session is considered established when the INVITE, 200 OK, and ACK messages have been exchanged between the UAC and the UAS. Until a final response to the initial INVITE has been received, a UAC MUST NOT sent a re-INVITE request. When two User Agents send re-INVITE simultaneously to each other, it is handled in the “Retry After” header. This condition is called as “glare” condition. An INVITE request consists of all the six mandatory header fields plus an additional “Contact” header field, which is mandatory in INVITE message.

**CONTACT:** The CONTACT header field is used to convey a URI that identifies the resource requested or the resource originator; depending on whether it is present in a request or response. Once a Contact header field has been received, that URI is catched and used for routing future requests within a dialog. There are three additional parameters defined for use in Contact header fields: q, action, and expires. They are placed at the end of the URI and separated by semicolons. The “q” value parameter is used to indicate relative preference, which is represented by a decimal number in the range 0 to 1. The “q” value is not a probability, and there is no requirement that the “q” values for a given list of Contacts add up to 1. The “Expires” parameter indicates how long the URI is valid and is also only used in registrations. The parameter either contains an integer number of seconds or a date in SIP form.

**REGISTER:** A REGISTER request is used by a User Agent to notify a SIP network of its current contact URI (IP Address). Registration is not required to enable a user agent to use a proxy server for outgoing calls, but for receiving incoming calls from proxies. Depending on the use of the Contact and Expires headers in the Register request, the REGISTRAR server takes different actions. The Request-URI contains only the domain of the registrar server with no user portion. A user agent sending a REGISTER request may receive a 3xx redirection or 4xx failure response containing a contact header of the location to which registrations should be sent. A third party registration is possible through REGISTER request. In this case the “From” header will contain the URI of the party submitting the registration on be-half of the party identified in the “To” header.

Register requests add, remove, and query bindings. A Register request can add a new binding between an address of record and one or more contact addresses. A Register request does not establish a dialog. The Record-Route Header Field has no meaning in REGISTER requests or responses. A “Contact” header field MAY be included in a REGISTER request. The User Agents MUST NOT send a new registration until they have received a final response from the REGISTRAR for the previous one or the previous REGISTER request has timed out.

In a REGISTER request, the value of C-Seq is incremented. The Request-URI contains only the domain of the REGISTRAR server with no user portion. The “To” header contains the SIP URI of the AOR of the user agent that is being registered. The “From” header contains the SIP URI of the sender of the request, usually the same as the “To” header. It is recommended that the same Call-ID be used for all registrations by a user agent.

**BYE:** The BYE method is used to terminate an established media session. A session is considered established if an INVITE has received a success class response(2xx), or an ACK has been sent. A BYE is sent only by user agents participating in the session, never by proxies or other third parties. It is an end-to-end method. BYE is not forked like an INVITE.

**ACK:** ACK method is used to acknowledge final responses to INVITE requests. Final responses to all other requests are never acknowledged. Final responses are defined as 2xx,3xx,4xx,5xx, and 6xx class responses. The C-Seq number is never incremented for an ACK, but the C-Seq method is changed to ACK. This is so that a UAS can match the C-Seq number of the ACK with the number of the corresponding INVITE.

The ACK request constructed by the client transaction MUST contain values for the Call-ID,From,and Request-URI that are equal to the values of those header fields in the request passed to the transport by the client transaction. The “To” header field in the ACK MUST equal the “To” header field in the response being acknowledged, and therefore will usually differ from the “To” header field in the original request by the addition of the tag parameter. The ACK MUST contain a single via header field, and this MUST be equal to the top via header field of the original request. If the INVITE request whose response is being acknowledged had Route header fields, those header fields MUST appear in the ACK.

An ACK may contain a SDP message body. SDP in an ACK is used in inter working scenarios with other protocols where the media characteristics may not be known when the initial INVITE is generated and sent.

For 2xx responses, the ACK is end to end, but for all other final responses it is done on a hop by hop basis when stateful proxies are involved. A hop by hop ACK reuses the same branch ID as the INVITE since it is considered part of the same transaction. An end to end ACK uses a different branch ID,as it is considered a new transaction.

**CANCEL:** The CANCEL method is used to terminate pending searches or call attempts. It can be generated by either user agents or proxy servers provided that a 1xx response containing a tag has been received, but no final response has been received. CANCEL is a hop by hop request and receives a response generated by the next state full element. The C-Seq is not incremented for this method so that proxies and user agents can match the C-Seq of the CANCEL with the C-Seq of the pending INVITE to which it corresponds. The branch ID for a CANCEL matches the INVITE that it is canceling.

A proxy receiving a CANCEL, forwards the CANCEL to the same set of locations with pending requests that the initial INVITE was sent to. A proxy does not wait for responses to the forwarded

CANCEL requests, but responds immediately. A User Agent confirms the cancellation with a 200

OK response to the CANCEL and replies to the INVITE with a 487 Request Terminated response. A CANCEL may not contain a message body.

**OPTIONS:**  The OPTIONS method is used to query a user agent or server about its capabilities and discover its current availability. The response to the request lists the capabilities of the user agent or server. A proxy never generates an OPTION request. A success class response(2xx) can contain ALLOW, ACCEPT, ACCEPT-ENCODING,ACCEPT-LANGUAGE, and SUPPORTED headers indicating its capabilities.

**REFER:** A REFER method is used by a user agent to request another user agent to access a URI or URL resource. The resource is identified by a URI or URL in the required Refer-To header field. This method is being used to implement a call transfer service. A REFER request can be sent either inside or outside an existing dialog. REFER uses the non-INVITE method state machine, which requires an immediate final response, unlike an INVITE which may take several seconds to complete.

The UAC sends an INVITE setting the Request-URI to the Refer-To URI. This INVITE is successful since it receives a 200 OK response. This successful outcome is communicated back to the UAC using a NOTIFY method. The message body of the NOTIFY contains a partial copy of the final response to the triggered request.

A REFER and the SIP request triggered by the REFER may contain the Referred-By header field, which contains information about who requested the request.

REFER method uses a “Refer-To” header field which is a mandatory header field.

**REFER-TO:** It is a required header field in a REFER request, which contains the URI or URL resource that is being referenced. It may contain any type of URI, like SIP or SIPS URI, Tel URI, http URI etc.

**SUBSCRIBE:** The subscription method is used by a user agent to establish a subscription for the purpose of receiving notifications( via the NOTIFY method) about a particular event. A successful subscription establishes a dialog between the UAC and the UAS. The subscription request contains an Expires header field, which indicates the time duration of the existence of the subscription. After this time period, the subscription automatically terminated. The subscription can be refreshed by sending another SUBSCRIBE within the dialog before the expiration time. A server accepting a subscription returns a 200 OK response also contains a Expires header field. The expiration timer can be the same as the request, or the server may shorten the interval.

The method SUBSCRIBE with Expires:0 requests the termination of a subscription and the dialog. A terminal subscription will result in a final NOTIFY indicating that the subscription has been terminated. A 202 response, does not indicate whether the subscription has been authorized- it merely means it has been understood by the server.

The client sends a SUBSCRIBE, which is successful, and receives NOTIFYS as the requested events occur at the server. Before the expiration of the subscription time, the client re-SUBSCRIBES to extend the subscription and hence receives more notifications.

A client must be prepared to receive a NOTIFY before receiving a 200 OK response to the SUBSCRIBE. Due to forking, a client must be prepared to receive NOTIFYS from multiple servers, although only one 200 OK response to the SUBSCRIBE may be received.

In addition to the six mandatory header fields, SUBSCRIBE method has two more mandatory header fields. They are “EVENTS” and “ALLOW EVENTS”.

**EVENT:** The EVENT 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.

**ALLOW-EVENTS:** The ALLOW-EVENTS header field is used to list the support event packages that are supported. A user agent that supports SIP Events then knows that it may send a SUBSCRIBE to that event package.

**NOTIFY:** The NOTIFY method is used by a user agent to convey information about the occurrence of a particular event. A NOTIFY is always sent within a dialog. Since it is sent within a dialog, the NOTIFY contains a “To” tag, “From” tag, and existing Call-ID. A NOTIFY request normally receives a 200 OK response 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. The Event header contains the package name used in the subscription. The SUBSCRIPTION-STATE header field can either be active, pending, or terminated.

A NOTIFY is always sent at the start of a subscription and at the termination of a subscription.

**SUBSCRIPTION-STATE:**  The subscription-state header field is a required header field in a NOTIFY request. It indicates the current subscription state. Values defined include active, pending, or terminated. Additional parameters include expires, reason, and retry-after. Values defined for

the reason parameter include deactivated, give up, probation, no resource, rejected, and time out.

**MESSAGE:** The MESSAGE method is used to transport Instant Messages(IM) using SIP. IM usually consists of short message exchanged in near real time by participants engaged in a “Conversation”. MESSAGES may be sent within a dialog or outside a dialog, but they do not establish a dialog by themselves. The actual message content is carried in the message body as a MIME attachment. A MESSAGE request normally receives a 200 OK response to indicate that the message has been delivered to the final destination. An Instant Message response should not be sent in the message body of a 200 OK, but rather a separate MESSAGE request sent to the original sender.

**INFO:** It is used by user agent to send all signaling information to another user agent with which it has an established media session. This is different from a re-INVITE since it does not change the media characteristics of the call. The request is end to end, and is never initiated by proxies. A proxy only forwards an INFO request, it is the UAS who checks whether the dialog is valid. An INFO method contains a message body. The contents may be signaling information, a mid call event, or some sort of stimulus. The INFO method always increments the C-Seq.

**PRACK:** The PRACK method is used to acknowledge receipt of provisional responses(1xx). The reliability of 2xx,3xx,4xx,5xx,6xx responses to the INVITE is achieved using the ACK method. In cases, where a provisional response, such as 180 ringing, is critical in determining the call state, it is necessary for the receipt of a provisional response to be confirmed. The PRACK method applies to all provisional responses except the 100 trying response, which is never reliably transported. The PRACK method always increments the C-Seq. A PRACK may contain a message body.

**R-Ack:**  The R-Ack header field is used within a response to a PRACK request to reliably acknowledge a provisional response that contained a R-Seq header field. The R-Ack header field echoes the C-Seq and the R-Seq from the provisional response.

**UPDATE:** The UPDATE method is used to modify the state of a session without changing the state of the dialog. In an established session, a re-INVITE is used to update session parameters. However, neither party in a pending session( INVITE sent but no final response received) may re-INVITE- instead, the UPDATE method is used.

Possible uses of UPDATE include muting or placing on hold pending media streams, performing QoS or other end to end attribute negotiation prior to session establishment.

**BASIC CALL FLOW IN SIP:**

**ALICE's Phone**   **PROXY-1**  **PROXY-2**  **BOB's Phone**

**SIP RESPONSE MESSAGES:**

1XX PROVISIONAL Request received, continuing to process the request.

2XX SUCCESS The action was successfully received.

3XX REDIRECTION Further action needs to be taken in order to complete the

request.

4XX CLIENT-ERROR The request contains bad syntax or can not be fulfilled at

this server.

5XX SERVER-ERROR The server failed to fulfill an apparently valid request.

6XX GLOBAL FAILURE The request can not be fulfilled at any server.

**PROVISIONAL RESPONSE:**

100 Trying: This response is only a hop by hop request. It is never forwarded and may not contain

a message body. A forking proxy must send a 100 trying response. This response can

be generated either a proxy or a user agent. It only indicates that some kind of action

is being taken to process the call-it does not indicate that the user has been located.

It does not contain a “To” tag. This response stops retransmission of the INVITE.

180 Ringing:This indicate that the INVITE has been received by the user agent and that alerting

is taken place. When the user agent answers immediately, a 200 OK is sent without

a 180 ringing; this scenario is called the “fast answer” case in telephony. This could

contain a message body.

181 Call Is Being Forwarded: This indicate that the call has been handed off to another end-point.

182 Call Queued: This indicate that the INVITE has been received, and will be processed in a

queue. A message body in this response can be used to carry music on hold.

183 Session Progress: It is used to convey information about the progress of the call. It is an

end to end session and does establish a dialog(must contain a To tag and Contact).

It allows a UAC to hear ring tone, busy tone, or a recorded announcement in calls

through a gateway into the PSTN. This is call progress information is carried in the

media stream in the PSTN.

**SUCCESSFUL RESPONSE:**

200 OK: The request has succeeded. The information returned with the response depends on the

method used in the request.

202 Accepted: This indicates that the UAS has received and understood the request, but the

request may not have been authorized or processed by the server. It is commonly

used in responses to SUBSCRIBE,REFER, and MESSAGE methods.

**REDIRECTION RESPONSE:**

300 Multiple Choices: This response contains multiple Contact header fields. The order of the

Contact header fields is assumed to be significant. That is, they should be

tried in the order in which they were listed in the response.

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 that

is not permanent.

305 Use Proxy: his redirection response contains a URI that points to a proxy server who has

authoritative information about the calling party. The caller should resend

the request to the proxy for forwarding. This response is generated by the

UASs.

380 Alternative Service: This response returns a URI that indicates the type of service that the

called party would like. An example might be a redirect to a voicemail

server.

**CLIENT-ERROR RESPONSE:**

400 Bad Request: This response indicate that the request was not understood by the server. An

example might be a request that is missing required header fields such as

To,From,Call-ID, or C-Seq. This response is also used if a UAS receives

multiple INVITE requests for the same Call-ID.

401 Unauthorized: This response indicates that the request requires the user to perform

authentication. His response is issued by UASs and registrars.

402 Payment Required: Reserved for future use.

403 Forbidden: The server understood the request, but is refusing to fulfill it.

404 Not Found: This response indicate that the user identified by the SIP or SIPS URI in the

Request-URI can not be located by the server, or that the user is not currently

signed on with the user agent.

405 Method Not Allowed: The method specified in the Request-line is understood, but not allowed

for the address identified by the Request-URI. This response MUST include an

ALLOW header field containing a list of valid methods for the indicated address.

406 Not Acceptable: This response indicates that the request can not 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 is sent by a proxy. It indicate that the UAC must

first authenticate itself with the proxy before the request can be processed. The

response should contain a Proxy Authenticate header field. The request is

resubmitted with Proxy-Authorization header field.

408 Request Timeout: The server could not produce a response within a suitable amount of time.

410 Gone: This response is similar to the 404 Not Found response but contains the hint that the

requested user will not be available at this location in the future.

413 Request Entity Too Large: The sever is refusing to process a request because the request

entity-body is larger than the server is willing or able to process.

414 Request-URI Too Long: The server is refusing to service the request because the Request-URI

is longer than the server is willing to interprete.

415 Unsupported Media Type: The server is refusing to service the request because the message

body of the request is in a format not supported by the server for the requested

method.

416 Unsupported URI Scheme: The server can not process the request because the scheme of the

URI in the Request-URI is unknown to the server.

420 Bad Extension: This response indicates that the extension specified in the Require header

field is not supported by the proxy or user agent. The response should contain a

Supported header field listing the extensions that are supported.

421 Extension Required: This response indicates that a server requires an extension to process

the request that was not present in a Supported header field in the request. The

required extension should be listed in a Required header field in the response.

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 Call/Transaction Does Not Exist: It indicate that the UAS has received a request that does

not match any existing dialog or transaction.

482 Loop Detected: The server has detected a loop.

483 Too Many Hops: The server received a request that contains a Max-Forwards header field

with the value zero.

484 Address Incomplete: The server received a request with a Request URI that was incomplete.

485 Ambiguous: Indicates that the Request URI is ambiguous.

486 Busy Here: The callee's end system was contacted successfully, but the callee is currently

not willing or able to take additional calls at this end system.

487 Request Terminated: The request was terminated by a BYE or CANCEL request. This response

is never returned for a CANCEL request itself.

488 Not Acceptable Here: This response indicates that some aspect of the proposed session is

not acceptable and may contain a Warning header field indicating the exact

reason.

491 Request Pending: The request was received by a UAS that had a pending request within the

same dialog.(“glare” situations).

**SERVER-ERROR RESPONSE:**

500 Server Internal Error: The server encountered an unexpected condition that prevented it from

fulfilling the request.

501 Not Implemented: The server does not support the functionality required to fulfill the request.

502 Bad Gateway: The server, while acting as a gateway or proxy, received an invalid response

from the downstream server it accessed in attempting to fulfill the request.

503 Service Unavailable: Server's maintenance error.

504 Server Time-Out: The server did not receive a timely response from an external server it

accessed in attempting to process the request.

505 Version Not Supported: The server does not support the SIP protocol version that was used

in the request.

513 Message Too Large: The server was unable to process the request since the message length

exceeded its compatibilities.

**GLOBAL-FAILURE RESPONSES:**

600 Busy Everywhere: Callee is busy everywhere.

603 Decline: The callee's machine was successfully contacted but the user explicitly does not

wish to or can not participate.

604 Does Not Exist Anywhere: The server has authoritative information that the user indicated in

the Request-URI does not exist anywhere.

606 Not Acceptable: The user agent was contacted successfully but some aspect of the session

description such as the requested media, bandwidth, or addressing style were

not acceptable.

**Definitions:**

**Address of Records:** An Address of Record (AOR) is a SIP or SIPS URI that points to a domain with a location service that can map the URI to another URI where the user might be available. The location service is populated through registrations. We can also called the AOR as the “public address” of the user.

**Back-to-Back User Agent:** A back to back user agent is a logical entity that receives a request and processes it as a user agent server(UAS). In order to determine how the request should be answered, it acts as a user agent client(UAC) and generates requests. Unlike a proxy server, it maintains dialog state, and MUST participate in all requests sent on the dialogs it has established.

**Dialog:** A dialog is a peer to peer SIP relationship between two user agents that persists for some time. A dialog is established by SIP messages, such as a 2xx response to an INVITE request. A dialog is identified by Call-ID, remote tag, and local tag.

**Loop:** When a request arrives at a proxy, is forwarded, and later arrives back at the same proxy. When it arrives the second time the Request-URI is identical to the first time, and other header fields that affect proxy operation are unchanged, so that the proxy would make the same processing decision on the request it made the first time. Looped requests are errors.

**Spiral:** A spiral is a SIP request that is routed to a proxy, forwarded onwards, and arrives once again at that proxy, but this time differs in a way that will result in a different processing decision than the original request. This means that the request's Request URI differs from its previous arrival. A spiral is not an error condition, unlike a loop. We can call it as call forwarding.

**Loose Routing:** A proxy is said to be loose routing if it follows the procedures defined in this specification for processing of the Route header field. These procedures separate the destination of the request (present in the Request URI) from the set of proxies that need to be visited along the way (present in the Route header field).

**Strict Routing:** A proxy is said to be strict routing if it follows the Route processing rule. That rule caused proxies to destroy the contents of the Request URI when a Route header field was present. The call proceeds through all the proxies mandatorily the manner they are present in the

Route header field.

**Downstream:** A direction of message forwarding within a transaction that refers to the direction that requests flow from the user agent client to user agent server.

**Upstream:** A direction of message forwarding within a transaction that refers to the direction that responses flow from the user agent server back to the user agent client.

**Transaction:** A transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non 1xx) response sent from the server to the client. If the request is INVITE and the final response is a non 2xx, the transaction also include an ACK to the response. The ACK for a 2xx response to an INVITE request is a separate transaction.

**Forking:** Proxy servers can make flexible “routing decisions” to decide where to send a request.

Ex: If Bob's SIP phone returned a 486 Busy Here response, then the proxy server could direct the INVITE to Bob's voicemail server.

Thus a proxy server can also send an INVITE to a number of locations at the same time. This is callees as “Forking”. There are two types of forking. 1. Parallel 2. Sequential.

**Record Routing:**  Mechanism by which a proxy can inform user agents that it wishes to stay on the path of all further messages is called “Record Routing”. Such a proxy would insert Record Route header field into SIP messages which contain address of the proxy.

**Authentication:** Authentication takes two general forms. 1. Authentication of a user agent by a proxy, redirect, or registrar server. 2. Authentication of a user agent by another user agent. Mutual authentication between proxies or a proxy and a user agent is also possible using Certificates.

Using HTTP Digest authentication, a proxy requiring authentication replies to an unauthenticated INVITE with a 407 Proxy Authorization Required response containing a Proxy-Authenticate header field with the form of the challenge. After sending an ACK for the 407, the user agent can then resend the INVITE with a Proxy-Authorization header field containing the credentials. The request is usually sent using the same

Call ID but an incremented C-Seq number. User Agent, redirect, or registrar servers use 401 Unauthorized response to challenge authentication containing a www-Authenticate header field, and expect a new INVITE to contain an Authorization header field containing the user agent's credentials.

**TAGS:** A tag is a random number with at least 32 bits of randomness. Tags are added to “To” and “From” header to uniquely identify a dialog. The “To” header in the initial INVITE does not contain a tag. Although a caller must include a tag in the “From” header. Excluding 100 trying, all other responses contain a tag to the “To” header.

Tag of 200 OK response is called as **Dialog Identifier**.

Tag of 200 OK response is used in all future requests (with the call-id).

Any response generated by a proxy have a tag added by the proxy.

An ACK generated by either a user agent or a proxy, will always copy the “From” tag of the response in the ACK request.