RFC 3261 Annotated

Jay G. Nam

Open Stack, Inc.

January 11, 2008

Contents

1 Video Phone Overview **3**
- 1.1 Two Kinds of Phone . . . 3
- 1.2 Components of Vide Phone . . . 3
  - 1.2.1 UI . . . 3
  - 1.2.2 Signaling . . . 4
  - 1.2.3 Codec . . . 4
  - 1.2.4 OS . . . 5
  - 1.2.5 Hardware . . . 5
- 1.3 Market Players . . . 5
- 1.4 Questions . . . 5

2 SIP Overview **6**
- 2.1 RFC 3261 Anatomized . . . 6
- 2.2 General Information . . . 7
  - 2.2.1 Three Types of Entity in the SIP World . . . 7
  - 2.2.2 The Structure of the Protocol . . . 7
- 2.3 Question . . . 8

3 The Most Basic and Often Used Call Scenario **8**
- 3.1 Actors . . . 8
- 3.2 Entity . . . 8
- 3.3 Actions . . . 9
  - 3.3.1 When Alice's phone boots, it registers itself to the reigstrar . . . 9
  - 3.3.2 Alice calls to Bob . . . 9
  - 3.3.3 Bob answers the phone . . . 10
  - 3.3.4 They are chatting . . . 10
  - 3.3.5 Bob hangs on the phone . . . 10

4 Methods Introduced in RFC 3261 **10**
- 4.1 REGISTER (chapter 10) . . . 11
- 4.2 INVITE and ACK (chapter 13, 14) . . . 11
- 4.3 BYE (chapter 15) . . . 12
- 4.4 CANCEL (chapter 9) . . . 12
- 4.5 OPTIONS (chapter 11) . . . 13

5 General User Agent Behavior **13**
5.1 UAC Behavior . . . 13
5.1.1 Mandatory 6 Headers for All Requests . . . 13
5.1.2 How to Determine the Destination for a Request . . . 15
5.2 UAS Behavior . . . 15

6 Transport **16**
6.1 UAC . . . 16
6.1.1 Sending Request . . . 16
6.1.2 Receiving Response . . . 17
6.2 UAS . . . 17
6.2.1 Receiving Requests . . . 17
6.2.2 Sending Responses . . . 17
6.3 Framing . . . 17
6.4 Error Handling . . . 17

7 Transaction **18**
7.1 Type of Transactions . . . 18
7.2 Transaction Lifetime . . . 18
7.3 The Components of Transaction . . . 18
7.3.1 non-INVITE Transaction . . . 18
7.3.2 INVITE Transaction . . . 18
7.4 Purpose of Transactions . . . 19
7.5 The Basic Principle of How Transaction Works . . . 19
7.6 Analysis of Transactions . . . 20
7.6.1 INVITE Client and Server Transactions . . . 21
7.6.2 non-INVITE Client and Server Transactions . . . 21
7.7 Matching Responses to Client Transactions . . . 21
7.8 Matching Requests to Server Transactions . . . 21

8 Dialog **21**

9 INVITE **24**

10 PRACK (RFC3262) **24**

11 Offer/Answer Model **27**

12 Proxy Behavior **28**
12.1 Request Processing . . . 28
12.2 Response Processing . . . 29

13 Header Fields **29**

14 Response Codes **30**

15 SUBSCRIBE/NOTIFY (RFC3265) **30**

16 REFER (RFC3515) **33**

17 UPDATE (RFC 3311) **34**

18 INFO (RFC 2976) **34**

19 MESSAGE (RFC 3428) **34**

1 Video Phone Overview

1.1 Two Kinds of Phone

1. soft phone
   software-only imementation of an IP telephone
   softphone is software that simulates a real phone and runs on a general purpose computer, rather than a dedicated device. It is usually used with a headset connected to the sound card of the PC or USBphone.

   Examples

   - XTen
   - Adore Softphone
   - SJphone
   - Microsoft Messenger

2. hard phone
   an appliance specially targeted for Video Phone
   a physical IP telephone
   hardware + softphone

   Examples

   - Inocova SM7000
   - Wooksung WVP 2100
   - C&S CIP 6000
   - Samsung i8000
   - Motorola Ojo Personal Video Phone
   - Leadtek BVP 8770

1.2 Components of Vide Phone

Main components of video phone

1. UI(User Interface)
2. Signaling
3. Codec
4. OS, device drivers and system softwares
5. Hardware

1.2.1 UI

1. Qt in Linux
2. Microwindows in Linux
3. Windows CE

1.2.2 Signaling

VoIP Signaling Protocols

1. SIP – RFC 3261
2. H.323 – ITU-T recommendation
3. MGCP – RFC 3435

Three Pillars of VoIP

1. SIP(Session Initiation Protocol) : how to manage session.
2. SDP(Session Description Protocol): describe the content of session
3. RTP(Realtime Transport Protocol) : how to transport the multimedia data in real time

Fundamental RFCs for VoIP

1. SIP RFC 3261
2. SDP RFC 2327, 3264, 4566
3. RTP RFC 3550, 3551

1.2.3 Codec

raison d'etre: voice and video data is big.

- raw voice data: 64 KBS (8,000 sampling x 8 bit per sample)
- raw video data: 60 MBS (CIF size, 25 fps: 352 width x 288 height x 24 bits per pixel x 25 fps)

We have to encode the data before sending and decode after receiving.
captured raw data → encode → network → decode → resored raw data

Voice codec

1. G.711 μ-law, A-law
2. G.723.1
3. G.729
4. G.722.1 (wideband codec)
5. iLBC(internet Low Bitrate Codec)

Video Vodec

1. H.261 the codec of past
2. H.263 the dominant codec of current
3. H.264 the codec of future
4. MPEG4 not for real-time streaming

1.2.4 OS

1. Linux
2. WinCE

1.2.5 Hardware

1. PC for softphone
2. SoC
   - DaVinci from Texas Instrument; DSP
   - i.MX27 from Freescale ; hardware codec
   - 8x8 VCP for WVP2100 video codec
   - Equator BSP-16 for WVP2200

1.3 Market Players

1. Standard Creator
   - IETF – RFC
   - ITU-T – ITU-T recommendataion
2. Vendors of hard phone
   Inocova, C&S Technology, Leadtech, AddPac, Samsung, LG
3. Carriers
   - KT(Korea Telecom)
   - Dacom
   - NTT
   - Telecom Italia
   - BT(British Telecom)
   - Deutsche Telecom
   - Comcast
   - Lucent Technolgy
4. Comsumers

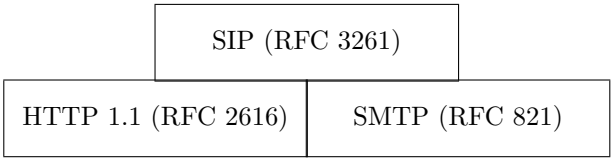
1.4 Questions

Alice is going to make a softphone. She will use ffmpeg as the codec. Tell me other three parts she should know to achieve her mission? What RFCs are explaining the 3 parts?

Answer: 3 parts she should know are: SIP, SDP, RTP
RFCs explaining them: SIP: 3261, SDP: 4566, 3264, RTP: 3550, 3551

2 SIP Overview

1. SIP: Session Initiaion Protocol
2. *SIP is a signalling protocol for creating, modifying, and terminating session.*
3. RFC 2543 (March 1999) → RFC 3261 (June 2002)
4. SIP is based on HTTP and SMTP.



The main characteristics of SIP

1. TEXT based → easy to debug and implement (cf: H.323 uses ASN.1 encoding.)
2. REQUEST/RESPONSE struture like HTTP
3. uses SIP url which is similar to email url of SMTP.

2.1 RFC 3261 Anatomized

We can divide RFC 3261 into three parts:

1. Informational chapters; do not require deep thinking.
   1, 2, 3, 4, 5, 6, 7, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
2. Essential chapters for understanding SIP; the logic of SIP
   - chapter 8: general user agent behavior
   - chapter 12: dialogs
   - chapter 16: proxy behavior
   - chapter 17: transactions
   - chapter 18: transport
3. Method explanation chapters; the action of SIP
   - CANCEL chapter 9
   - REGISER chapter 10
   - OPTIONS chapter11
   - INVITE, ACK chapter 13, 14
   - BYE chapter 15
   - INFO RFC 2976
   - MESSAGE RFC 3428
   - PRACK RFC 3262
   - REFER RFC 3515
   - UPDATE RFC 3311
   - SUBSCIBE/NOTIFY RFC 3265

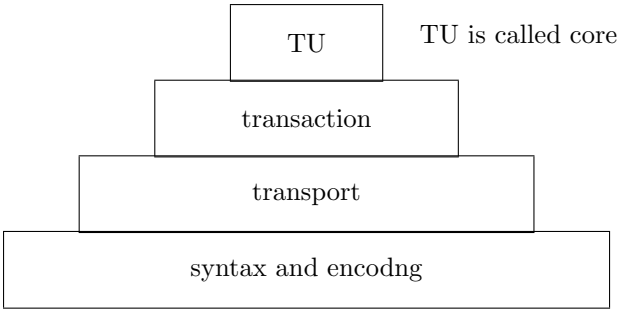
2.2 General Information

2.2.1 Three Types of Entity in the SIP World

Threre are only three types of entity in the SIP world.

1. User Agent
2. Proxy server
3. Registar

2.2.2 The Structure of the Protocol



Clearing Points

- Transaction User: the entity which utilize transaction. this term is from the viewpoint of transaction.
- Core: the entity of SIP. that is, it can be user agent, stateful proxy, , stateless proxy and registar. this term is from the viewpoint of SIP entity.

Clearing Points

- Address-of-Record(AOR) : the logical address
  (ex) sip: "Jay Nam" yaluruns@example.com
  Request-URI, From, and To header uses AOR.
  AOR is the superset of Address-of-Device.
- Address-of-Device : the physical address. only one device for a device address.
  (ex) sip: 58.72.14.171:5060
  Contact uses Address-of-Device.

Clearing Points

- Call an informal term. it includes session, dialiogs.
- Session multimedia session. that is, the session of multimedia stream
- Dialog the relationship from the viewpoint of SIP signalling. Session and dialog are independent. For exampe, BYE method is used to terminate not the dialog but the session. Therefore, BYE does not necessarily terminate dialog.

2.3 Question

Question: How many methods are defined by the RFCs? Enumerate all the methods and RFCs corresponding to them.

3 The Most Basic and Often Used Call Scenario

*A*lice calls to Bob.

Bob answers the phone.

They are chatting.

Bob hangs up the phone.

Let's analyze this scenario.

3.1 Actors

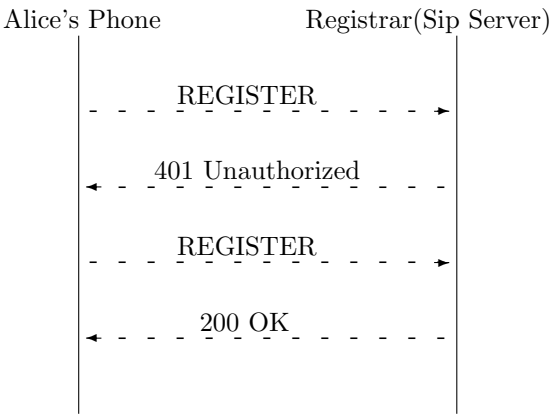
1. Alice
2. Alice's phone
3. Bob
4. Bob's phone
5. Proxy
6. Registar

3.2 Entity

1. Alice
2. Alice's phone
3. The Phone number of Alice
4. Bob
5. Bob's phone
6. The Phone number of Bob
7. Proxy
8. Registar

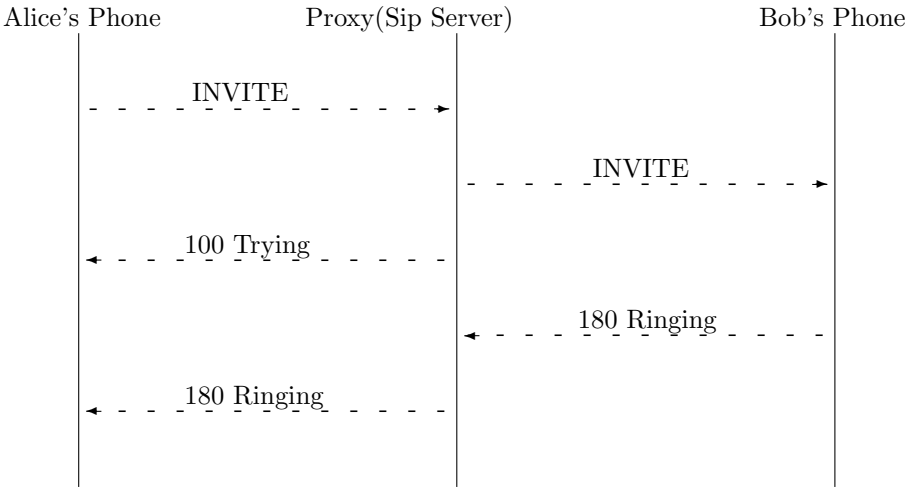
3.3 Actions

3.3.1 When Alice's phone boots, it registers itself to the reigstrar



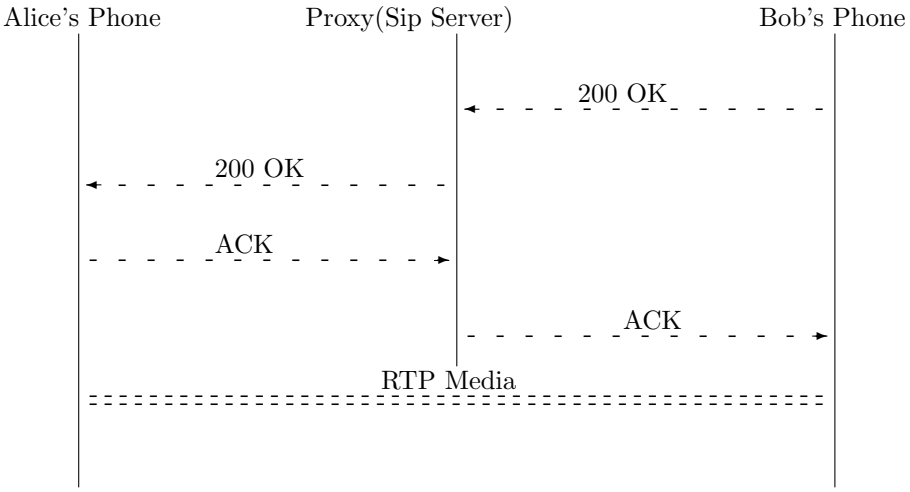
See section 2.1 in RFC 3665 for details.

3.3.2 Alice calls to Bob



See section 3.2 in RFC 3665 for details.

3.3.3 Bob answers the phone

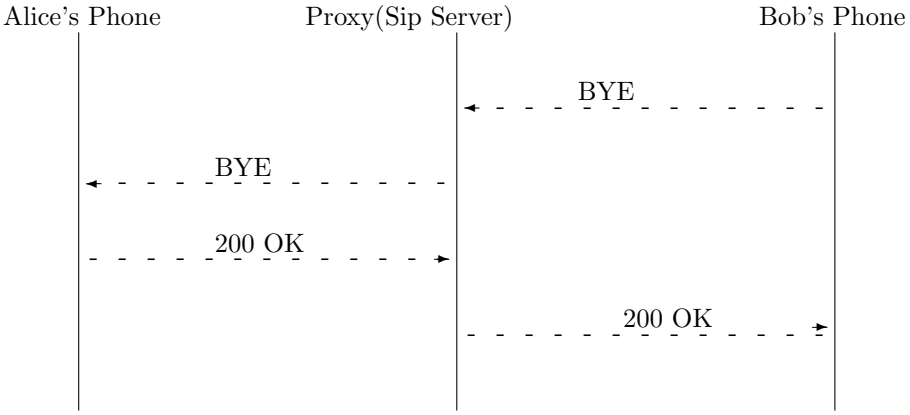


See section 3.2 in RFC 3665 for details.

3.3.4 They are chatting

RTP Media carris voice and video data.

3.3.5 Bob hangs on the phone



See section 3.2 in RFC 3665 for details.

4 Methods Introduced in RFC 3261

RFC 3261 introduces six methods:

1. REGISTER
2. INVITE
3. ACK
4. BYE
5. CANCEL

6. OPTIONS

4.1 REGISTER (chapter 10)

Epitome of REGISTER

> Reqeust from UA to Registrar: map AOR in **To** header to AOD in **Contact** header for the seconds in **Expire** header.
>
> Response from Registrar to UA: Show *all* the registration mappings that is currently valid in the **Contact** header.

REGISTER maps Address-of-Record to Address-of-Device. For example, Alice's AOR(i.e. her phone number) is *sip:010-9500-8170@example.com* and her AOD(i.e. her phone IP) is *sip:58.72.14.171*.

The mapping is $1 - n$ relationship. That is, there can be many physical phones to a phone number. For example, Alice has one AOR(sip:010-9500-8170@example.com). But she has three AOD; sip:58.72.14.171 for her office phone, sip:222.121.94.123 for her home phone, and sip:64.167.312.15 for her cellular phone.

Registrar has the mapping information and proxy uses this information to proxy the calls from UAs.

Stack Writer's Attention Point

1. **To** header is AOR and **Contact** header is AOD in registration (**From** header does not have big meaning).
2. The same **Call-ID** is used for a registration.
3. The **Branch-ID** is different for each registration request because a request is a new transaction.
4. The sequence number of **CSeq** is incremented by one for each registration request.
5. Default registration time is 3,600 seconds.
6. 3GPP recommends 60,000 seconds for registration time(ref: 3GPP TS 24.229).
7. 3GPP recommends to update the registration after the half of registration time has elapsed.

4.2 INVITE and ACK (chapter 13, 14)

Epitome of INVITE

> Reqeust from UA to Proxy or UA: I want to talk to the AOR in the **Request-URI** in the such way described in the SDP content.
>
> Response from Proxy or UA to UA: Please Wait, or I Accept or Decline your request.

Epitome of ACK

> Callee also wants to know whether his response delivered well to the caller.

INVITE is used not just to create a session or dialog but also to modify an existing session or dialog.
(cf: UPDATE modifies only an existing session.)

$$\text{INVITE} \overset{\text{creates or modifies}}{\Longrightarrow} \begin{cases} \text{dialog} \\ \text{session} \end{cases}$$

ACK is used only for INVITE request.

Stack Writer's Attention Point

1. ACK handling depends on whether the final response is 2XX or else.
2. The sequence number in **CSeq** in the ACK is same with that of the INVITE.
3. re-INVITE must not sent within a dialog while another INVITE transaction is in progress.

4.3 BYE (chapter 15)

Epitome of BYE

> I want to hang up the call.

BYE is used to terminate not the dialog but the session as the title of chapter 15 in RFC 3261 is *Terminating a Session*.

If there is no other activity in the dialog after the session is terminated, the dialog is also terminated. That is, the dialog can be terminated on the BYE in some circumstances.

Stack Writer's Attention Point

1. BYE can be sent only if he can be sure that there is a dialog. That, is, in caller's case, on a early or confirmed dialog and in callee's case, on the confirmed dialog after he gets the ACK.

4.4 CANCEL (chapter 9)

Epitome of BYE

> I want to cancel the INVITE request I have sent.

The difference between CANCEL and BYE

1. CANCEL is hob-by-hop and BYE is end-to-end. hop-by-hop means that proxy decides on its own and end-to-end means that only UA decides and proxy merely forwards the response. End-to-end is to a dealer as hop-by-hop is to a broker.
2. BYE can be sent only if there exists a dialog.
3. CANCEL has no effect once a final respose is sent.
4. Only UAC can send CANCEL. But both the UAC and UAS can send BYE.

Thre Relationship among INVITE, BYE and CANCEL

$$\text{CANCEL} \overset{\text{cancels}}{\Longrightarrow} \text{INVITE} \overset{\text{creates or modifies}}{\Longrightarrow} \begin{cases} \text{dialog} \\ \text{session} \end{cases} \overset{\text{terminates}}{\Longleftarrow} \text{BYE}$$

Question:
Alice calls to Bob. Bob's phone is ringing. Suddenly Alice hangs up her phone. As a stack writer, which of CANCEL and BYE shall we send when she hangs up?
Answer: BYE is better than CANCEL. If Bob hooks up his phone at the moment we send CANCEL, the CANCEL has no effect. But if we send BYE, BYE still has the effect to terminate the call.

4.5 OPTIONS (chapter 11)

Epitome of OPTIONS

> I want to know your current state and capability.

The name of OPTIONS is kind of misleading. It has nothing to do with option. State-Capability might be a better name

Stack Writer's Attention Point

1. The response code of OPTION request must be same that would have been chosed had the request been an INVITE.

5 General User Agent Behavior

UA behavior depends on two factors:

1. inside or outside of a dialog
2. method

5.1 UAC Behavior

Request-URI should be set to the value of the URI in the **To** header.

Contact header provides a AOD of the UA.

5.1.1 Mandatory 6 Headers for All Requests

To, From, CSeq, Call-ID, Max-Forwards, and Via are the mandaroty headers for all the requests.

See Table 2 in Section 20.1 to exactly know which header fields can be used in which context.

To The recipient's AOR of the request.
Except the REGISER request, **To** header except its tag does not have much role from sipstack's viewpoint.

Example:
```
To: Alice <sip:alice@sip.com>
```

From The initiator's AOR of the request.
From header except its tag does not have much role from sipstack's viewpoint.

Example:
```
To: "Bob Doe" <sip:alice@sip.com>
```

Call-ID The id of a call.
Rigorously speacking, there does not exist the concept of call in RFC 3261. (Ony dialog and session are meaningful in RFC 3261) Then what is call id ?

One INVITE request results in the creation of *multiple* dialogs through a parallel forking by a proxy.

$$\text{INVITE} \overset{\text{creates}}{\Longrightarrow} \begin{cases} \text{dialog } 1 \\ \text{dialog } 2 \\ \dots \\ \text{dialog } n \end{cases}$$

Informally, we can say a call is the set of all the dialogs creates by the INVITE request. That is,

The call created by the INVITE = {dialog 1, dialog 2, ..., dialog n}

Example:
`Call-ID: gfh323GsdfS`

In RFC 3261, call id alone has no role. Call id has its meaning as it is the part of the dialog id that is the combination of local tag, remote tag, and call id.
Call-ID is just byte-string. It does not necessarily have the host name part.

CSeq the sequence number of request.
The sequence number has meaning only in two contexts. One is the REGISTER and the other is the dialog. That is, its scope is within a registration or a dialog.

Example:
`CSeq: 5 INVITE`

1. sequence number in the REGISER request increases by one for each new request.
2. sequence number in a dialog increases by one for each new reqest except ACK and CANCEL.
3. the sequence number in ACK is that of the INVITE the ACK is going to acknowledge.
4. the sequence number in CANCEL is that of the INVITE the CANCEL is going to cancel.

Max-Forwards serves to limit the numbef of hops a request can transit on the way to the destination.
Initial value should be 70.
Each proxy in the route decreases the Max-Forwards value by one.
If it becomes 0, proxy responses with 483 Too Many Hops.

Example:
`Max-Forwards: 70`

Via The Role of **Via**

1. from where the request was sent.
2. to where the response shall be sent.
3. transaction id

Example:
`Via:  SIP/2.0/UDP 10.1.23.15:5070;branch=z9hG4bK123456;received=58.72.14.171`

Let's analyze this example:

1. SIP/2.0 : says the protocol used.
2. UDP : says the transport. it could be UDP, TCP or TLS.
3. 10.1.23.15:5070 : UAC says the host and port number from which the request was sent. this is called sent-by parameter.
4. branch=z9hG4bK123456: says the transaction id of the request. That is, via branch is a transaction id.
5. received=58.72.14.171 : UAS says the IP address from which the request was received. This could be different from the sent-by paramerter.

5.1.2 How to Determine the Destination for a Request

It is easily decided where to send the response through the information in the **Via** header. However, where to send the request is not so simple.

The Case of Not Using Proxy
URI in the To header → Request-URI → the Destination to send the request

Example:
Alice's phone does not use proxy.
She types 64.23.57.132, the IP address of Bob's phone to call to him.

To header: To: sip:64.23.57.132
Request-Line: INVITE sip:64.23.57.132 SIP/2.0
destination: 64.23.57.132

The case of Using Proxy
URI in the To header → Request-URI
URI of proxy → URI in the Route header → the Destination to send the request

Exampel:
Alice's phone uses proxy, of which name is proxy.example.com and of which domain is example.com.
She types 010-9500-8170, the phone number of Bob to call to him.

To header: To: sip:010-9500-8170@example.com
Request-Line: INVITE sip: 010-9500-8170@example.com SIP/2.0
Route header: Route: sip: <proxy@example.com;lr>
Destination: proxy.example.com

5.2 UAS Behavior

The six mandatory headers in the request becomes as follows in the response:

1. Call-ID header from the request → Call-ID header in the response
2. To header from the request → To header in the response (could add to tag)
3. From header from the request → From header in the response
4. Via header from the request → Via header in the resposne
5. CSeq header from the request → CSeq header in the response
6. Response does not have Max-Forwards header.

The Contact header contains the AOD of the UAS.

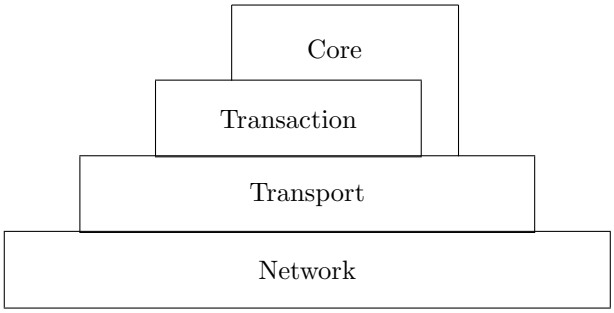


Figure 1: The Relationship among Network, Transport, Transaction, and Core

6 Transport

The relationship among tranport, transaction, and core is depicted in the above picture.

$$\left.\begin{array}{l}\text{Core}\\ \text{Transaction}\end{array}\right\} \overset{\text{uses}}{\Longrightarrow} \text{Transport} \overset{\text{uses}}{\Longrightarrow} \text{Network}$$

The Role of Transport

1. transmit and receive requests and responses via the network.
2. hand out the requests or response to the transaction or the core.

Transports introduced in RFC 3261

1. UDP
2. TCP
   must be used when the message size is larger than 1,300 bytes.
3. TLS
   for secure communication.
   must be used when the hop is addressed by SIPS.

6.1 UAC

UAC has two aspects regarding the transport. One is sending requests and the other is receiving responses for the requests.

6.1.1 Sending Request

1. the user of transport layer computes {address, port, transport}. In this case, if the message size is larger than 1,300 bytes, TCP or TLS must be used.
2. the user of transport layer passes {request, IP, port, transport} to the transport.
3. the transport layer adds **sent-by** field into the Via header.
4. the transport layer sends the request to the network using the above information.

6.1.2 Receiving Response

1. check the **sent-by** field. it is not its ip, discard the response.
2. find the matching transaction according to the section 17.1.3.
3. if there is a transaction matching the response, it is passed to the transaction.
4. if not, the response is passed to the core.

6.2 UAS

UAS has two aspects regarding the transport. One is receiving requests and the other is sending responses for the requests.

6.2.1 Receiving Requests

1. add **received** parameter if **sent-by** parameter in the top Via is a domain name or if it contains an IP address that differs from the packet source address.
2. find a matching transaction according to the secion 17.2.3.
3. if there is a matching transaction, pass the request to it.
4. if not, pass the request to the core.

6.2.2 Sending Responses

In short, use the **received** or **send-by** parameter to decide where to send the response.

6.3 Framing

UDP The Content-Length header field is optional.

1. if there is no Content-Length header, it is ok. take the entire content as the message body.
2. if there is a Content-Length header and its value is larger than the real content, this is an error.
3. if there is a Content-Length header and its value is smaller that the real content, take the first Content-Length bytes as the message body.

TCP The Content-Length header field is a MUST.

6.4 Error Handling

In a Transaction

1. inform the TU that a transport error has occurred.
2. terminate the transaction.

Not In a Transaction Inform the core that a transport error has occurred.
(cf: The only case that a message is not sent in a transaction context in a UA is the ACK request for a 200 OK response and 200 OK response retransmission for an INVITE message)

7 Transaction

7.1 Type of Transactions

Transaction can be seen from two viewpoints. One is whether it is INVITE or non-INVITE, and the other is whether it is Client or Server. Therefore, there are four types of transaction in SIP.

1. INVITE Client Transacton
2. INVITE Server Transacton
3. non-INVITE Client Transacton
4. non-INVITE Server Transacton

7.2 Transaction Lifetime

- ICT: its lifetie is at most Timer B(64*T1) + Timer C(at least 3 minutes) + Timer D(at least 32s).
  (We assume that we use Timer C)
- NICT: its lifetime is at most Timer F(64*T1).
- IST
  - it can stay Proceeding state forever.
  - once it sends a final response, its lifetime is at most Timer H(64*T1).
- NIST
  - it can stay either Trying or Proceeding state forever if it does not get a final response from the TU.
  - once it gets a final response, its lifetime is Timer J(64*T1).

7.3 The Components of Transaction

7.3.1 non-INVITE Transaction

The transaction consists of a single request an any responses to that request, which include zero or more provisional responses and one or more final responses.

7.3.2 INVITE Transaction

if the final response is 2XX, it consists of

1. INVITE requests
2. any provisional responses
3. the first 2XX response

The 2xx responses from the second one and the ACK requests are not part of the transaction. The ACK request and the 2XX response from the second one is not part of any transaction.

if the final response is non-2XX, it consists of

1. INVITE requests
2. any provisional responses
3. 2XX responses
4. ACK requests

Why does RFC 3261 differentiate these cases depending on whther the final response is 2xx or not? The reason is in the ACK. The ACK request can contain SDP for 2xx response. This SDP can be created only by the UAC core. That is, it can be created neither by the proxy nor by the UA Transaction. Only the core knows the context and can create the SDP. Therefore, ACK cannot be the part of the transaction. If it is part of the transaction, the proxy or transaction should be able to create the SDP mechanically.

7.4 Purpose of Transactions

Client Transaction

1. receive a request from TU.
2. reliably deliver the request to the server transaction.
3. receive responses and deliver them to the TU.
4. filter out any response retransmissions or disallowed responses.
5. in the case of an INVITE request, generate the ACK request for a non-2xx final response.

Server Transaction

1. receive requests from the transport layer.
2. deliver them to the TU.
3. filter out any request retransmissions.
4. accept responses from the TU and delivers them to the transport layer.
5. in the case of an INVITE transaction, absorb the ACK request for any non-2xx final response.

7.5 The Basic Principle of How Transaction Works

1. UDP transport: retransmit the request or the response if they do not work as expected.
2. TCP transport: no retransmission

non-INVITE Transaction The UAC takes the responsibility to achieve the transaction. In other words, UAC cares and UAS does not care the transaction.
UAC retransmits the request if it does not receive the response. UAS sends response only when it receives a request and does nothing more.

INVITE Transaction The UAS takes the responsibility to achieve the transaction. In other words, UAS cares and UAC does not care the transaction.
UAS retransmits the final response if it does not receive ACK. UAC sends ACK only when it receives a final response.

7.6 Analysis of Transactions

Basic Time Intervals
There are 3 elementary time intervals in SIP. All other time interval are derived from them.

1. T1: an estimate of the round trip time (RTT).
   default value is 500 ms.

2. T2: the maximum retransmit interval for non-INVITE requests and INVITE responses.
   default value is 4 seconds.

3. T4: maximum duration a message will remain in the network.
   default value is 5 seconds.

Timers (Table 4 of RFC 3261)
Transaction State Machines are controlled by following 11 timers.

1. Timer A: INVITE request retransmit timer for UDP only
   initially T1
   1st (0, 0) → 2nd (500ms, 0.5s) → 3rd (1s, 1.5s) → 4th (2s, 3.5s) → 5th (4s, 7.5s) → 6th (8s, 15.5s) → 7th (16s, 31.5s)

2. Timer B: INVITE request timeout timer
   64*T1 (32 seconds if T1 is 500 ms)

3. Timer C: proxy INVITE transaction timeout
   > 3min

4. Timer D: Wait time for response retransmits in INVITE client transaction
   > 32s for UDP
   0s for TCP/SCTP

5. Timer E: non-INVITE request retransmit interval for UDP only
   initially T1
   1st (0, 0) → 2nd (500ms, 0.5s) → 3rd (1s, 1.5s) → 4th (2s, 3.5s) → 5th (4s, 7.5s) → 6th (4s, 11.5s) → 7th (4s, 15.5s) → 8th (4s, 19.5s) → 9th (4s, 23.5s) → 10th (4s, 27.5s) → 11h (4s, 31.5s)

6. Timer F: non-INVITE transaction timeout timer
   64*T1 (32 seconds if T1 is 500 ms)

7. Timer G: INVITE response retransmit interval for UDP only
   initially T1
   firing behavior is same with Timer E.
   1st (0, 0) → 2nd (500ms, 0.5s) → 3rd (1s, 1.5s) → 4th (2s, 3.5s) → 5th (4s, 7.5s) → 6th (4s, 11.5s) → 7th (4s, 15.5s) → 8th (4s, 19.5s) → 9th (4s, 23.5s) → 10th (4s, 27.5s) → 11h (4s, 31.5s)

8. Timer H: Wait time for ACK receipt
   64*T1

9. Timer I: Wait time for ACK retransmits
   T4 for UDP
   0s for TCP/SCTP

10. Timer J: Wait time for non-INVITE request retransmits
    64*T1 for UDP
    0s for TCP/SCTP

11. Timer K: Wait time for response retransmits in non-INVITE client transaction
    T4 for UDP
    0s for TCP/SCTP

Role	INVITE Client	non-INVITE Client	INVITE Server	non-INVITE Server
retransmit request	Timer A	Timer E		
request timeout	Timer B	Timer F		
retransmit response			Timer G	
ACK timeout			Timer H	
absorb response	Timer D	Timer K	Timer I	Timer J

Table 1: Relationship Between Timers

7.6.1 INVITE Client and Server Transactions

INVITE transaction is three-way handshake.

1. is the final response 2xx or non-2xx ?
2. if the answer is no, ACK belongs to the orignal INVITE transaction.
3. if the answer is yes,
   (a) the INVITE Server Transaction is destroyed as soon as the 2xx response is transmitted.
   (b) ACK is a new pseudo-transaction by itself.
   (c) UAS retransmitts the 2xx response if it does not receive ACK message.

7.6.2 non-INVITE Client and Server Transactions

INVITE transaction is two-way handshake.

7.7 Matching Responses to Client Transactions

1. branch parameter
2. If the method parameter in the CSeq header field matches the method of the request that created the transaction. The method is needed since a CANCEL request constitutes a different transaction, but shares the same value of the branch parameter.

7.8 Matching Requests to Server Transactions

1. branch parameter
2. the sent-by value in the top Via of the request is equal to the one in the request that created the transaction
3. the method of the request matches the one that created the transaction, except for ACK, where the method of the request that created the transaction is INVITE.

8 Dialog

Definition A dialog represents a peer-to-peer *SIP relationship* between two user *agents* that persists for some time.

Role of Dialog

1. facilitates *sequencing of messages* between the user agents and proper *routing of requests* between both of them.
2. represents a *context* in which to interpret SIP messages.

Dialog State

1. early
   when dialog is created by 101-199 response, it is called in early state.
2. confirmed
   when dialog is created by or got 2xx response, it is called in confirmed state.

Dialog State Dialog State consists of following 8 pieces of information.

1. dialog id = { Call-ID, local tag, remote tag }
2. local sequence number
3. remote sequence number
4. local URI
5. remote URI
6. remote tatget URI
7. route set
8. secure flag

Explanation of the Above Terms

1. tag
   tag exists in To or From header.
   Why does tag exist? From tag does not have any role as far as I know. But To tag does.

   An INVITE request can results in creating multiple dialogs through parallel forking. These dialogs have the same Call-ID, the Call-ID of INVITE message, and From tag. But each UAS will add different to tags to the To header. So the role of tag, more precisely, the role of To tag, is to differentiate multiple dialogs that are created through an INVITE message.
2. local tag
   What is *local*? *Local* means me. So local tag means my tag, the tag I created by myself.
   - UAC it adds a tag to the From header. Hence, the tag in From is local tag to the UAC.
   - UAS it adds a tag to the To header. Hence, the tag in To is local tag to the UAS.
3. remote tag
   What is *remote*? *Remote* means you. So remote tag means your tag, the tag you created.
   - UAC the tag in To header is remote tag to the UAC.
   - UAS the tag in From header is remote tag to the UAS.
4. local sequence number
   the sequence number I maintain in my side.

5. remote sequence number
   the sequence number you maintain in your side.
6. local URI
   - UAC the uri in From header
   - UAS the uri in To header
7. remote URI
   - UAC the uri in To header
   - UAS the uri in From header
8. remote target URI
   the URI in the Contact header
9. Route Set
   the set of routes every message in the dialog must follow.
   proxy adds Record-Route header if it wants to be in the middle of the dialog for all messages.
   Record-Route by proxy → Route in the dialog of UA
   - UAC the route set must be set to the list of URIs in the Record-Route header field from the response, taken in reverse order and preserving all the URI parameters.
   - UAS the route set must be set to the list of URIs in the Record-Route header field from the requests, taken in order and preserving all URI parameters.

Creating a Request within Dialog

1. Call-ID → Call-ID of the request
2. local sequence number + 1 → the sequence number of a new request except ACK and CANCEL method
3. local URI → the URI in From header
4. remote URI → the URI in To header
5. local tag → the tag in From header
6. remote tag → the tag in To header

Special Attention to How to Compute Request-URI and Route headers

1. Alice type 100 number to call Bob.
2. UI of WVP2100 interprets Alice's intention as `sip:100@proxy.com`.
3. `sip:100@proxy.com` → initial Request-URI
4. Now we compute the final Request-URI, Route headers, and the target address to send the message from the initial Request-URI and route set of the dialog. That is,

$$\left\{ \begin{array}{l} \text{Request-URI from the user} \\ \text{route set of the dialog} \end{array} \right\} \text{computes according} \xrightarrow{\text{to section 12.2.1}} \left\{ \begin{array}{l} \text{final Request-URI} \\ \text{Route headers} \\ \text{target address} \end{array} \right\}$$

   The Rule in Section 12.2.1

   (a) is the route set empty?

(b) if answer is no

i. no Route header

ii. intial Request-URI → final Requset-URI

iii. final Request-URI → target address

(c) if answer is yes, is the first Route URI in the route set of the dialog is loose-routing? that is, does it have a lr parameter?

i. if the answer is yes

A. intial Request-URI → final Requset-URI

B. the URI of the first Route of the route set → target address

C. route set → Route headers

ii. if the answer is no

A. intial Request-URI → the last Route header of the new request

B. the URI of the first Route of the route set → final Request-URI

C. Route headers = { the Routes in the route set except the first one, the initial Request-URI }

9 INVITE

200 Response Retransmit: same with Timer G and Timer H. But applies to all transports.

If the server retransmits the 2xx response for 64*T1 seconds without receiving an ACK, the dialog is confirmed, but the session SHOULD be terminated. This is accomplished with a BYE.

10 PRACK (RFC3262)

The 4 RFCs published together:

1. RFC 3261 (SIP)
2. RFC 3262 (PRACK)
3. RFC 3263 (Locating SIP Servers)
4. RFC 3264 (Offer/Answer Model)

Key Points

- The Mission of PRACK: make the unreliable provisional responses reliable.
- RFC 3262 (PRACK) modified the INVITE Server Transaction behavior in RFC 3261.
- RFC 3262 provides a new Offer/Answer model in addition to RFC 3261.

Types of Response

$$\text{response} = \begin{cases} \text{provisional} & \text{sent unreliably (no retransmission)} \quad \leftarrow \text{PRACK makes it reliable} \\ & \text{sent reliably by the retransmission} \\ \text{final} & \text{and ack mechanism} \end{cases}$$

bug: the second line in the third paragraph in page 2: Those *requests* → Those *responses*

Terms Introduced in RFC 3262

1. PRACK (method): **P**rovisional **R**esponse **AC**Knowledgement
2. 100rel (option tag)
3. RSeq (header): **R**esponse **Seq**uence
4. RAck (header): **R**esponse **Ack**nowledgement

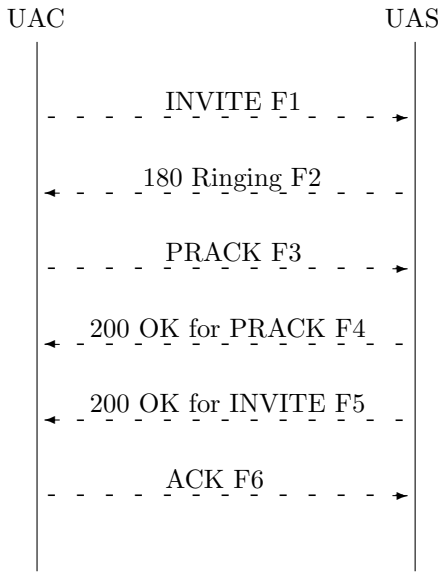


Figure 2: PRACK Example

Content of Figure 2 Call Flow

```
F1 INVITE UAC -> UAS

INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Supported: 100rel
CSeq: 1 INVITE
Content-Length: 0


F2 180 Ringing UAS -> UAC

SIP/2.0 180 Ringing
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
```

```
Call-ID: 3848276298220188511@atlanta.example.com
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Require: 100rel
CSeq: 1 INVITE
RSeq: 5236
Content-Length: 0
```

F3 PRACK UAC -> UAS

```
PRACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK012345
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
CSeq: 2 PRACK
RAck: 5236 1 INVITE
Content-Length: 0
```

F4 200 OK UAS -> UAC

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK012345
    ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 PRACK
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Length: 0
```

F5 200 OK UAS -> UAC

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Length: 0
```

F6 ACK UAS -> UAC

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bd5
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
```

```
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

Analysis of Figure 2 Call Flow

1. UAC sends INVITE F1 request with 100rel in Supported or Require header.
2. UAS send 180 Ringing F2 with 100rel in Require header and RSeq header.
3. 180 Ringing F2 is retransmitted at $T1$, $2*T1$, ... for $64*T1$ seconds if UAS does not receive PRACK, of which behavior is same with Timer A in RFC 3261.
4. UAC send PRACK F3 with RAck header.
5. UAS stops retransmitting of 180 Ringing F2 once it receives PRACK.
6. UAS sends 5xx response for the original INVITE F1 request if it does not receive PRACK request for $64 * T1$ seconds.

11 Offer/Answer Model

The sources of Offer/Answer Model

1. page 80 in section 13.2.1 of RFC 3261.
2. page 9 in section 5 of RFC 3262
3. RFC 3264 (Offer/Answer Model)
4. RFC 2329 (SDP)

Case 1: INVITE does have an Offer

1. unreliable 1xx
   (a) does not have an answer → 2xx MUST have an answer
   (b) has an answer → 2xx MUST have an answer, which must be same with the one in 1xx.
2. reliable 1xx
   (a) has an answer → 2xx MAY have an answer, which must be same with the one in 1xx.
   (b) does not have an answer → 2xx MUST have an answer.
3. 2xx MUST has an answer

Case 2: INVITE does not have an Offer

1. the first reliable 1xx or 2xx MUST have an offer.
2. the first method after the reliable response MUST have an answer.
   it can be ACK or PRACK.

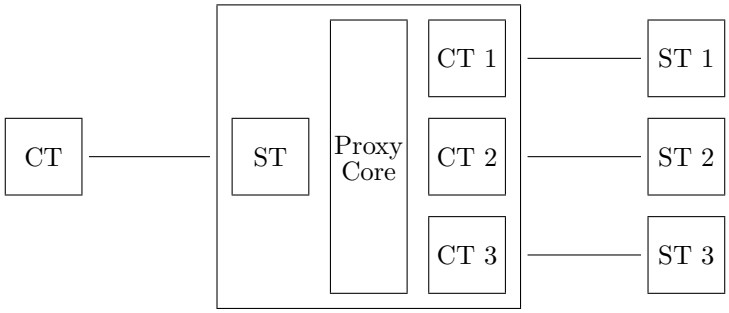


Figure 3: Stateful Proxy Model

12 Proxy Behavior

Proxy's Role

1. route SIP requests to user agent servers.
2. route SIP responses to user agent clients.

12.1 Request Processing

1. create a server transaction.
2. validate the request.
   (a) reasonable syntax
   (b) URI scheme
   (c) Max-Forwards
   (d) Loop Detection
   (e) Proxy-Require
   (f) Proxy-Authorization
3. preprocess the routing information.
   remove the Route that points to this proxy and adjust Requst-URI if necessary.
4. determine targets for the request.

$$\text{target set} = \begin{cases} \{\text{Request-URI}\} \text{ if Request-URI is not the domain that the proxy is responsible.} \\ \{\text{anything the proxy wants to add at the beggining or in the middle}\} \text{ if else} \end{cases}$$

5. create a *response context*.
6. forward the request to each target
   (a) Make a copy of the received request.
   (b) Update the Request-URI.
   (c) Update the Max-Forwards header field.
   (d) Optionally add a Record-route header field.
   (e) Optionally add additional header fields.
   (f) Postprocess routing information.
   (g) Determine the next-hop address, port, and transport.

(h) Add a Via header field.

(i) Add a Content-Length header field if necessary.

(j) Forward the new request.

(k) Set timer C.

12.2 Response Processing

1. Find the matching transaction.
2. if not found,
   $$\begin{cases} \text{forward the response if it is 2xx INVITE response.} \\ \text{discard the response in all other case.} \end{cases}$$
3. if found, do followings.
4. Find the response context.
5. Update timer C for provisional responses.
6. Remove the topmost Via.
7. Add the response to the response context.
8. until the final reposne is forwarded, forward the reponse immediately if it is
   (a) 1xx response
   (b) 2xx response
9. after the final response is forwarded, forward the response immediately if it is
   (a) 2xx INVITE response
10. Generate any necessary CANCEL requests if the proxy forwarded a final response.
11. If no final response has been forwarded after every client transaction associated with the response context has been terminated, choose the best final response from the response context.

13 Header Fields

We should be familiar to how to read Table 3: Summary of header fields in RFC 3261.

header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Accept	R		-	o	-	o	m*	o

Table 2: a fragment of Table 3 in RFC 3261

How to read the table

- where: empty(everywhere), **R**equest, **r**esponse, **c**opied, 1xx, 2xx, etc.
- proxy: **a**dd, **m**odify, **d**elete, **r**ead
- method header: **c**onditional, **m**andatory, **m***andatory, **o**ptional, s**t**ream, *****, **-**

A few headers of attention

- Supported: list all the extensions supported by the UAC or the UAS.
- Require: UA tells to the UA about the options that it should support in order to process the request.
- Unsupported: list the extensions not supported by the UAS.
  Require $\longleftrightarrow$ Unsupported
- Allow: list the set of methods supported by the UA.

14 Response Codes

- Provisional 1xx (Informational)
- Successful 2xx
- Redirection 3xx
- Request Failure 4xx (Client-Error)
- Server Failure 5xx
- Global Failure 6xx

15 SUBSCRIBE/NOTIFY (RFC3265)

The 5 RFCs published together:

1. RFC 3261 (SIP)
2. RFC 3262 (PRACK)
3. RFC 3263 (Locating SIP Servers)
4. RFC 3264 (Offer/Answer Model)
5. RFC 3265 (SUBSCRIBE/NOTIFY)

Terms Introduced in RFC 3265

1. SUBSCRIBE (method)
2. NOTIFY (method)
3. Allow-Events (header)
4. Subscription-State (header)
5. Event (header)
6. 202 Accepted (response code)
7. 489 Bad Event (response code)

Key Points of Subsribe/Notify

1. SUBSCRIBE is a dialog creating method.
2. dialog is created upon receiving 2xx response for the SUBSCRIBE or the first NOTIFY.

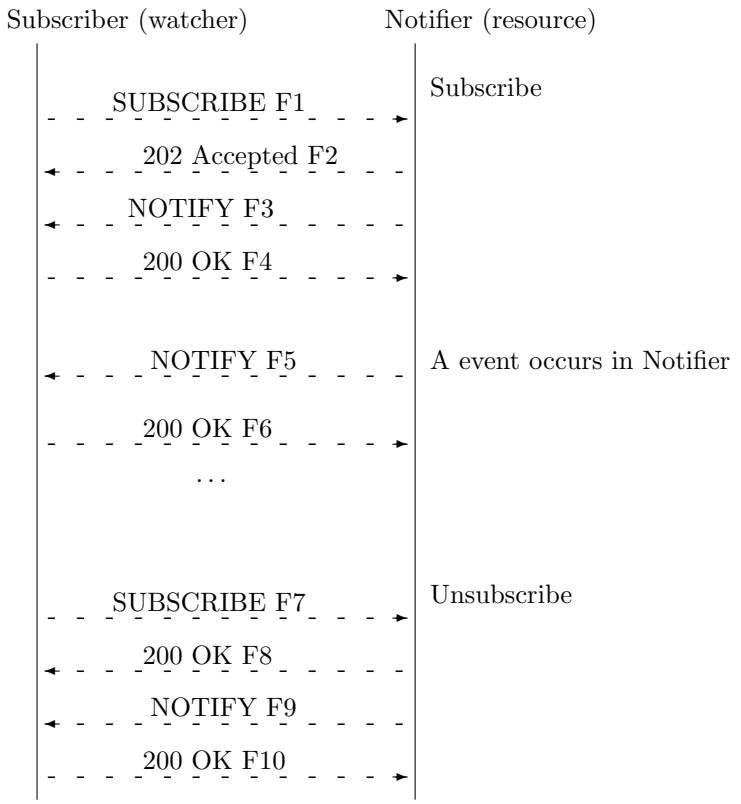


Figure 4: SUBSCRIBE/NOTIFY Prototype

3. Subscription is terminated upon receiving NOTIFY with `Subscription-State: terminated` header.

Explanation of Figure 4

```
F1 SUBSCRIBE

SUBSCRIBE sip:resource@proxy.com SIP/2.0
Event: {event name}
Accept: {MIME type}
Expires: 1000
...

F2 202 OK

SIP/2.0 202 Accepted
Expires: 600


F3 Initial NOTIFY

NOTIFY sip:user@watcher.example.com SIP/2.0
Event: {event name}
Subscription-State: pending;expires=599
...
```

```
F4 200 OK


F5 NOTIFY ; an event occurs in the resource.

NOTIFY sip:user@watcher.example.com SIP/2.0
Event: {event name}
Subscription-State: active;expires=500
...

F6 200 OK


F5 and F6 can be repeated.


F7 SUBSCRIBE ; unsubscribe

SUBSCRIBE sip:resource@proxy.com SIP/2.0
Event: {event name}
Accept: {MIME type}
Expires: 0
...

F8 200 OK


F9 Final NOTIFY
NOTIFY sip:user@watcher.example.com SIP/2.0
Event: {event name}
Subscription-State: terminate;reason=timeout
...

F10 200 OK
```

Some Event Packages (source: `http://www.iana.org/assignments/sip-events`)

Package Name	Type	Contact	Reference
message-summary	package	Rohan Mahy <rohan@cisco.com>	[RFC3842]
presence	package	Jonathan Rosenberg <jdrosen@jdrosen.net>	[RFC3856]
reg	package	Jonathan Rosenberg <jdrosen@jdrosen.net>	[RFC3680]
refer	package	Robert Sparks <rsparks@dynamicsoft.com>	[RFC3515]
spirits-INDPs	package	Vijay K. Gurbani <vkg@lucent.com>	[RFC3910]
spirits-user-prof	package	Vijay K. Gurbani <vkg@lucent.com>	[RFC3910]
winfo	template-package	Jonathan Rosenberg <jdrosen@jdrosen.net>	[RFC3857]

16 REFER (RFC3515)

Terms Introduced in RFC 3515

1. REFER (method)
2. Refer-To (header)
3. refer (package name)

Key Points of REFER

1. REFER is a special type of SUBSCRIBE.
2. Therefore, REFER is a dialog creating method like SUBSCRIBE.
3. REFER is often used in call transfering.

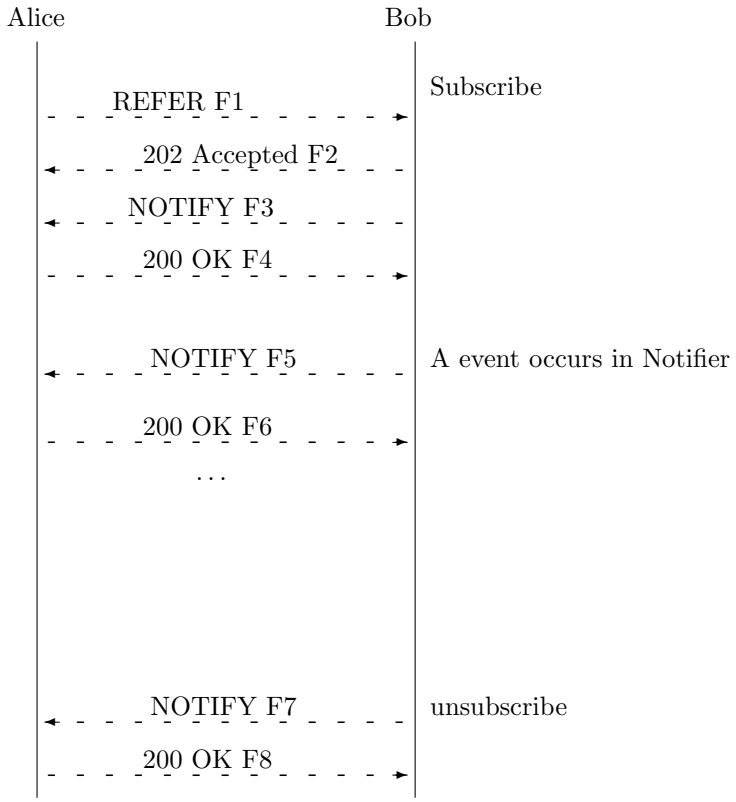


Figure 5: REFER Prototype

Further Reading

- RFC 3420 (sipfrag)
- RFC 3892 (Referred-By header)
- RFC 3891 (Replace header)
- draft-ietf-sipping-cc-transfer-05
- draft-ietf-sipping-service-example-09

17 UPDATE (RFC 3311)

The Difference between UPDATE and INVITE

- INVITE
    1. creates or modifies dialog and session.
    2. cannot be used in early dialog.
- UPDATE
    1. modifies a session.
    2. can be used to modify session in early dialog.

18 INFO (RFC 2976)

1. carries application layer information in the body.
2. changes neither the dialog nor the session.

19 MESSAGE (RFC 3428)

used to send instant message.

- EOF -