

Session Initiation Protocol

Author: *James Wright, MSc*

“This paper is a technical overview of the Session Initiation Protocol and is designed for IT professionals, managers, and architects who want a quick low-level guide to the workings and details of the protocol.”

Session Initiation Protocol

Session Initiation Protocol (SIP) was designed from the bottom up to connect people and devices whenever and wherever they are in order to engage in a (possibly lengthy) exchange of information. Existing protocols, such as HTTP and SMTP, were not purpose-built for this essential human activity, and so SIP was born to fill the gap. However, SIP borrows from these two other protocols heavily. From using HTTP's message exchange pattern, message format and encoding, to SMTP's URI scheme.

In 2002 a revised version of the SIP standard was formalised into the Internet Engineering Task Force's (IETF) standardisation process as RFC3261. Because of the open nature of the IETF standards process, the fact SIP is text based and shares many features with existing specifications, it has been readily understood, extended and implemented.

Entities

A SIP environment consists of a number of connected entities.:

- A User Agent (UA) is the entity which represents an end user in a client device. It usually operates in two modes: a User Agent Client (UAC) sends the initial request messages and processes responses; and a User Agent Server (UAS) accepts requests and sends responses.
- Proxy Servers are involved in routing the SIP messages to the correct endpoint. Stateful proxies sometime make use of User Agents in a logical entity called a Back-To-Back-User-Agent.
- Redirect Servers provide a new address or different route path to the recipient. The server may make use of a Location Server to persist location information.
- A Registrar acts as current repository of a client's attachment to the network.

It is the User Agent that tends to reside on the end user's device. The other entities provide essential support services in many scenarios.

Messages

SIP messages come in two flavours:

- **Request:** sent from client to a server and define the operation sought by the client.
- **Response:** sent from server to a client and provide the status of that request.

Requests

A SIP request is characterised as a method much like HTTP, and is considered a 'verb', since it requests actions to be performed by other User Agents or servers. RFC3261 defines six methods (the first six in Table 1) with subsequent standards defining the remaining extension methods (from INFO onwards).

Table 1. *SIP Methods*

Method	Description
INVITE	Used to set up a SIP session. Session parameters are negotiated.
REGISTER	Authenticates the User Agent and provides a current location to the network.
BYE	Terminates an open session.
ACK	Confirms a success response to an INVITE. The third part to a three-way-handshake.
CANCEL	Cancels an open request. BYE should be used to cancel (tear down) an existing request.
OPTIONS	Queries the capabilities of correspondents.
[Extension Methods]	
INFO	Provides mid-call session-related information. It is rarely used.
MESSAGE	Used to transfer Instant Messages.
NOTIFY	Publishes the outcome of events. Used in combination with SUBSCRIBE requests.
PRACK	A Provisional Response ACKnowledgment. Confirms receipt of a provisional response.
PUBLISH	Publishes status information. Used for Instant Messaging presence services.
REFER	Mechanism to pass a request to someone more appropriate to deal with it.
SUBSCRIBE	Used to request receipt of future NOTIFY or PUBLISH requests.
UPDATE	Modifies session parameters in mid-call.

Response

SIP Response messages are always sent in reply to a request. They convey status updates, confirmations, directions and error codes back to the UAC originating the request. Responses are characterised as either provisional or final; and every response must be identified by a 3-digit code.

Response Types

Six classes of response have been defined and are categorised using a 3-digit code. The first five are borrowed from HTTP; the sixth is new to SIP.

Table 2. *Response Classes*

Class	Description
1xx Provisional	Confirms receipt of request and that processing is continuing. Provisional responses to INVITEs are never ACKed.
2xx Success	The request was received, processed and accepted.
3xx Redirection	Provides location information or alternative services to try.

Class	Description
4xx Request Failure	The request contained an error or cannot be processed by the server.
5xx Server Failure	The server is unable to fulfil the request because of an internal error.
6xx Global Failure	No service can be found to fulfil the request.

Within each class, numerous response codes have been predetermined — some copied from HTTP. Table 3 provides a sample of the most important responses.

Table 3. *Sample Response Codes*

#	Method	Description
100	Trying	The next hop received the request.
180	Ringing	Attempting to alert the user.
182	Queued	Temporarily unavailable and request is in a queue (not rejected).
200	OK	The request has succeeded.
301	Moved Permanently	User is no longer available at the address given in the Request URI.
302	Moved Temporarily	Retry the request at a new address given in the Contact header.
400	Bad Request	Could not understand or process correctly the request.
401	Unauthorised	The request either failed authentication or needs more information.
403	Forbidden	The server is refusing to process the request. Do not retry.
404	Not Found	The server cannot identify the user in its domain.
408	Request Timeout	The server could not process the request in a reasonable time.
415	Unsupported Media	The format is not supported by the server.
480	Temporarily Unavailable	The called party is currently unavailable.
485	Ambiguous	The Request URI is ambiguous.
486	Busy Here	The called party is currently not willing or able to take the call.
500	Server Internal Error	The server encountered an unexpected condition.
513	Message Too Large	The message length exceeded a determined limit.
603	Decline	The user explicitly refused to accept the request.

Warning Header Field

The Warning header field is used to carry additional information about the status of the response. The header defines a 3-digit code between 300 and 399, the host name and a warning text.

Warning: 307 isi.edu "Session parameter 'foo' not understood"

Anatomy of a Message

Each SIP message begins with a Start-Line, is followed by a sequence of headers, and separated from the message body by a carriage-return line-feed sequence (CRLF).

- **Start-Line:** formatted as a Request-Line for Requests or a Status-Line for Responses.
- **Headers:** Named attributes that provide additional information about the message.
- **Separator Line:** Separator between header and body.
- **Body:** binary or textual payload. Typically Session Description Protocol (SDP) or a message text.

The start line, each header line and the separator line is terminated by a [CRLF] sequence.

Start Line

The start-line conveys the type of message and protocol version. For both Request (Request-Line) and Responses (Status-Line), the start-line has three elements separated by spaces.

- **Request-Line:** Contains a method, URI and ends with the protocol version ("SIP/2.0").
- **Status-Line:** Starts with the protocol version, followed by a numeric status code and is completed with a short textual reason.

Headers

Headers follow the same generic header format as HTTP. Each consisting of a case-insensitive ASCII encoded name and colon followed by a value which is sometimes UTF8 encoded and usually case-sensitive. Each header can have one or more semi-colon separated parameters appended to the value, providing additional tags and features.

header-name: header-value(;parameter-name=parameter-value)[CRLF]*

Each header can be separated on to different lines using a [CRLF] followed by a [TAB or SPACE] sequence (known as folding). What's more, multiple headers with the name same (e.g. Contact) can appear on separate lines, or, can be placed on the same line separated by commas. For example:

Contact: <sip:alice@atlanta.com>

Contact: <sip:alice1@chicago.com>

Can be represented as:

Contact: <sip:alice@atlanta.com>, <sip:alice1@chicago.com>

Or using folding:

*Contact: <sip:alice@atlanta.com>,
<sip:alice1@chicago.com>*

Body

A message body describes the session (using SDP) or contains opaque text or binary body parts containing the payload related to the session (e.g. MIME or Message formats). Bodies can appear in request or response messages.

Header Fields

The following message breakdown is an example of an INVITE request containing an SDP message being responded to with a "200" OK response. In the example, Alice makes a call to Bob using his SIP URI, 'sip:bob@897s.sydney.com'. Bob answers Alice with a success Response.

Request Message

Table 4. Request Message Example

Line	Description
INVITE sip:bob@897s.sydney.com SIP/2.0	An INVITE method request addressed to bob at 897s.Sydney.com using SIP version 2.0
Via: SIP/2.0/UDP 124.191.8.8:11506	The address where Alice expects to receive responses to the request
Max-Forwards: 70	Limits the number of network hops and identifies loops.
To: Bob <sip:bob@897s.sydney.com>	The display name and address of the original recipient.
From: Alice <sip:alice@melbourne.com>;tag=769122	The display name and address of the original sender
Call-ID: afh7989asdfhf@mel99.melbourne.com	A globally unique identifier for the call. The To, From, and Call-ID tuple provides a unique key for a call.
CSeq: 3434534 INVITE	An incremented command sequence number and original method.
Contact: <sip:alice@mel99.melbourne.com>	Further contact information for Alice.
Content-Type: application/sdp	A description of the message body (SDP).
Content-Length: 136	The byte count of the message length.
	[SEPERATOR LINE]
v=0	The SDP version number.
o=Alice 4352354234 523542345 IN IP4 124.191.9.1	The originator of the session plus session identifier and version.
s=Going to lunch	A textual session name
c=IN IP4 224.2.18.12/222	Address and network connection data.
t=278956487 789456775	Session start and stop time (seconds since 1900)
m=audio 49170 RTP/AVP 0	The description of the session media: type, port, protocol and formats.

Request Message

Table 5. Request Message Example

Line	Description
SIP/2.0 200 OK	The response code and reason phrase.
Via: SIP/2.0/UDP 124.191.8.8:11506	Copied from the INVITE request.
To: Bob <sip:bob@897s.sydney.com>;tag=abgj67	Copied from the INVITE request. A new tag parameter is added.
From: Alice <sip:alice@melbourne.com>;tag=769122	Copied from the INVITE request.
Call-ID: afh7989asdfhf@mel99.melbourne.com	Copied from the INVITE request.
CSeq: 3434534 INVITE	Copied from the INVITE request.
Contact: <sip:bob@897s.sydney.com>	Contact information for Bob
Content-Type: application/sdp	A description of the message body (SDP).
Content-Length: 132	The byte count of the message length.
	[SEPERATOR LINE]
v=0	The SDP version number.
o=Bob 4352354234 523542345 IN IP4 124.191.10.165	The originator of the session plus session identifier and version.
s=Going to lunch	A textual session name
c=IN IP4 224.120.9.98/222	Address and network connection data.
t=278956487 789456775	Session start and stop time (seconds since 1900)
m=audio 49170 RTP/AVP 0	The description of the session media: type, port, protocol and formats.

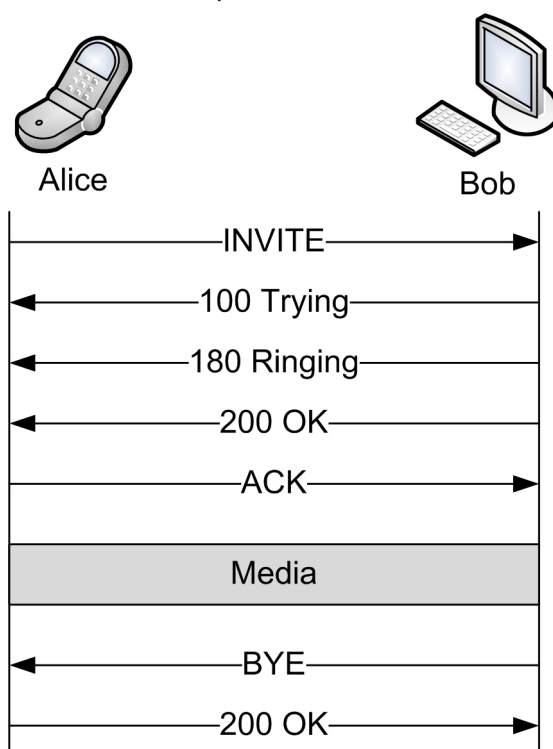


Figure 1. Alice completes a SIP call with Bob, exchanges media packets, then Bob terminates the call.

Call Flow Example

Call Flow: Simple Successful SIP to SIP

This section details a call flow between that same two SIP User Agents as above (see Figure 1) and could use the same message structures. The successful calls show the initial signalling, the establishment of the media session, then finally the termination of the call.

Session Setup

- Alice's UA sends an INVITE message to Bob's SIP address (i.e. <sip:bob@897s.sydney.com>). The message contents are a Session Description Protocol message describing the expected media exchange.
- Bob's UA receives the INVITE and responds with a 100 Trying message.
- The UA then attempts to attract the attention of Bob, and simultaneously sends a 180 Ringing message to Alice.
- Bob respond and is UA sends a 200 OK message. The 200 OK contains the SDP message Bob is agreeing to.
- Finally, Alice's UA acknowledges receipt of the OK with an ACK request.
- Media streams are established directly between Alice and Bob.

Session Tear Down

- Bob hangs up and his UA initiates a session termination by sending a BYE request to Alice.
- Alice's UA response with a 200 OK.

Further Reading

H. Schulzrinne. (2010) Henning Schulzrinne. [Online]. <http://www.cs.columbia.edu/~hgs/>

A. B. Johnston, SIP: Understanding the Session Initiation Protocol, 3th ed. Boston, MA, USA: Artech House, 2007.

J. Rosenberg et al., "SIP: Session Initiation Protocol," RFC 3261 2002.

IETF. (2010) Multiparty Multimedia Session Control (mmusic). [Online]. <http://datatracker.ietf.org/wg/mmusic>

IETF. (2010) Session Initiation Protocol (sip). [Online]. <http://datatracker.ietf.org/wg/sip/>

Konnetic Website. <http://www.konnetic.com>

Other Papers In the Series

IMS - Technologies and Architecture

<http://www.konnetic.com/documents/KonneticIMSTechnologiesAndArchitecture.pdf>

IMS - Services and Revenue

<http://www.konnetic.com/documents/KonneticIMSServicesAndRevenue.pdf>

SIP - An Introduction

<http://www.konnetic.com/documents/KonneticSIPIntroduction.pdf>

SDP - Technical Overview

<http://www.konnetic.com/documents/KonneticSDPTechnicalOverview.pdf>

James Wright MSc is the founder of Konnetic, a specialist in providing SIP and IMS based software to the .NET (C#, VB.NET, F# and more) community.