Internet Engineering Task Force SIP WG
Internet Draft J.Rosenberg,H.Schulzrinne
draft-rosenberg-sip-guidelines-00.txt dynamicsoft,Columbia U.
March 10, 2000
Expires: September, 2000

Guidelines for Authors of SIP Extensions

STATUS OF THIS MEMO

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as work in progress.

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

The Session Initiation Protocol (SIP) is a flexible, yet simple tool for establishing interactive connections across the Internet. Part of this flexibility is the ease with which it can be extended. In order to facilitate effective and interoperable extensions to SIP, some guidelines need to be followed when developing SIP extensions. This document outlines a set of such guidelines for authors of SIP extensions.

1 Introduction

The Session Initiation Protocol (SIP) [1] is a flexible, yet simple tool for establishing interactive connections across the Internet. Part of this flexibility is the ease with which it can be extended. SIP can be extended in numerous ways. New methods can be defined, new

headers can be added, new body types can be used, and new parameters for existing headers can be added. This flexibility also means that caution should be exercised when defining extensions, in order to ensure interoperability.

In order to facilitate interoperability, this document serves as a set of guidelines for authors of SIP extensions. It points out issues to consider when deciding whether a SIP extension is the right answer for a specific problem. It then points out issues which extensions should deal with from within the specification. Finally, it discusses common interactions with existing SIP features which often cause difficulties in extensions.

2 Should I define a SIP Extension?

The first question to be addressed when defining a SIP extension is: is a SIP extension the best solution to my problem? SIP has been proposed as a solution for numerous problems, including mobility, configuration and management, QoS control, call control, caller preferences, device control, third party call control, and MPLS path setup, to name a few. Clearly, not every problem can be solved by a SIP extension. More importantly, some problems that could be solved by a SIP extension, probably shouldn't.

To assist engineers in determining whether a SIP extension is an appropriate solution to their problem, we present two broad criteria. First, the problem should fit into the general purvey of SIPs solution space. Secondly, the solution must conform to the general SIP architectural model.

While the first criteria might seem obvious, we have observed that numerous extensions to SIP have been proposed because some function is needed in a device which also speaks SIP. The argument is generally given that "I'd rather implement one protocol than many". As an example, user agents, like all other IP hosts, need some way to obtain their IP address. This is generally done through DHCP [2]. SIPs multicast registration mechanisms might supply an alternate way to obtain an IP address. This would eliminate the need for DHCP in clients. However, we do not believe such extensions are appropriate. We believe that protocols should be defined to provide specific, narrow functions, rather than being defined based on all communications requirements between a pair of devices. The latter approach to protocol design yields modular protocols with broad application. It also facilitates extensibility and growth; single protocols can be removed and changed without affecting the entire system. We observe that this approach to protocol engineering mirrors object oriented software engineering.

Our second criteria, that the extension must conform to the general SIP architectural model, ensures that the protocol remains manageable and broadly applicable.

2.1 SIP's Solution Space

In order to evaluate the first criteria, it is necessary to define exactly what SIPs solution space is, and what it is not.

SIP is a protocol for initiating, modifying, and terminating interactive sessions. This process involves the discovery of a user wherever they may be located, so that a description of the session can be delivered to the user. SIP itself is independent of the session, and the session description is delivered as an opaque body. Much of SIP focuses on this discovery component. Its ability to fork, its registration capabilities, and its routing capabilities are all present for the singular purpose of finding the called user wherever they may be. As such, features and capabilities such as personal mobility, automatic call distribution, and follow-me are well within the SIP solution space.

Session initiation also depends on the ability of the called party to have enough information about the session itself in order to make a decision on whether to join or not. That information includes data about the caller, the purpose for the invitation, and parameters of the session itself. For this reason, SIP includes this kind of information.

Part of the process of session initiation is the communication of progress and the final results of establishment of the session. SIP provides this information as well.

There are many functions that SIP explicitly does not provide. It is not a session management protocol or a conference control protocol. The particulars of the communications within the session are outside of SIP. This includes features such as media transport, voting and polling, virtual microphone passing, chairman election, floor control, and feedback on session quality.

SIP is not a resource reservation protocol for sessions. This is fundamentally because (1) SIP is independent of the underlying session it establishes, and (2) the path of SIP messages is completely independent from the path that packets for a session may take. The path independence refers to paths within a providers network, and the set of providers itself. For example, it is perfectly reasonable for a SIP message to traverse a completely different set of autonomus systems than the audio in a session SIP establishes.

SIP is not a transfer protocol. It is not meant to send large amounts of data unrelated to SIPs operation. It is not meant as a replacement for HTTP. This is for numerous reasons, one of which is that SIP's recommended mode of operation is over UDP. Sending large messages over SIP can lead to fragmentation at the IP layer and thus poor performance in even mildly lossy networks. This is not to say that carrying payloads in SIP messages is never a good thing; in many cases, the data is very much related to SIPs operation. However, SIP is not meant to carry large amounts of data unrelated to SIPs general function.

2.2 SIP Architectural Model

We describe here some of the primary architectual assumptions which underly SIP. Extensions which violate these assumptions should be examined more carefully to determine their appropriateness for SIP.

Session independence: SIP is independent of the session it establishes. This includes the type of session, be it audio, video, game, chat session, or virtual reality. SIP operation should never be dependent on some characteristic of the session.

SIP and Session Path Independence: We have already touched on this once, but it is worth noting again. The set of routers and/or networks and/or autonomous systems traversed by SIP messages and the packets in the session are unrelated. They may be the same in some cases, bit it is fundamental to SIPs architecture that they need not be the same. Extensions which only work under some assumption of overlap are not generally applicable to SIPs operation and should be scrutinized carefully.

Multi-provider and Multi-hop: SIP assumes that its messages will traverse the "Big I". That is, SIP works through multiple networks administered by different providers. It is also assumed that SIP messages traverse many hops (where each hop is a proxy). Extensions which only work in single hop or single provider networks may not be appropriate for SIP.

Transactional: SIP is a request/response protocol. Many of the rules of operation in SIP are based on general processing of requests and responses. This includes the reliability mechanisms, routing mechanisms, and state maintenance rules. Extensions which add new messages that are not within the request-response model will likely break many aspects of SIP.

Proxies can ignore bodies: In order for proxies to scale well, they must be able to operate with minimal message processing. SIP has been engineered so that proxies can always ignore bodies. Extensions which require proxies to examine bodies in order to work will likely lead to serious scaling problems.

Proxies don't need to understand the method: Processing of requests in proxies does not depend on the method, except for the well known methods INVITE, ACK, and CANCEL. This allows for extensibility. Extensions that define new methods which must be understood by proxies are NOT RECOMMENDED.

INVITE messages carry full state: An initial INVITE message for a session is nearly identical (the exception is the tag) to a re-INVITE message to modify some characteristic of the session. This is strongly coupled to the idempotency of SIP requests, but is a different characteristic. Extensions which modify INVITE processing such that data spanning multiple INVITEs must be collected in order to perform some feature, are frowned upon.

Generality over efficiency: Wherever possible, SIP has favored general purpose components rather than narrow ones. If some capability is added to support one service, but a slightly broader capability can support a larger variety of services (at the cost of complexity or message sizes), the broader capability is generally preferred.

3 Issues to be Addressed

Given an extension has met the litmus tests in the previous section, there are several issues that all extension should take into consideration.

3.1 Backwards Compatibility

One of the most important issues to consider is whether the new extension is backwards compatible with baseline SIP. This is tightly coupled with how the Require, Proxy-Require, and Supported [3] headers are used.

If an extension consists of new headers inserted by a user agent in a request, and the request cannot be processed reasonably by a proxy and/or user agent without understanding the headers, the extension MUST mandate the usage of the Require and/or Proxy-Require headers in the request. These extensions are not backwards compatible with SIP.

The result of mandating usage of these headers means that requests cannot be serviced unless the entities being communicated with also understand the extension. If some entity does not understand the extension, the request will be rejected. The UAC can then handle this in one of two ways. In the first, the request simply fails, and the service cannot be provided. This is basically an interoperability failure. In the second case, the UAC retries the request without the extension. This will preserve interoperability, at the cost of a "dual stack" implementation in a UAC (processing rules for operation with and without the extension). As the number of extensions increases, this leads to an exponential explosion in the sets of processing rules a UAC may need to implement. The result is excessive complexity.

Because of the possibility of interoperability and complexity problems that result from the usage of Require and Proxy-Require, we believe the following guidelines are appropriate:

- The usage of these headers in requests for basic SIP services (in particular, session initiation and termination) is NOT RECOMMENDED. The less frequently a particular extension is needed in a request, the more reasonable it is to use these headers.

- The Proxy-Require header SHOULD be avoided at all costs. As the number of hops for a request increases, the likelihood a particular proxy doesn't support some extension increases exponentially. On the other hand, the Require header only mandates that a single entity, the UAS, support the extension. Usage of Proxy-Require is thus considered exponentially worse than usage of the Require header.

Extensions which define new methods do not need to use the Require header. SIP defines mechanisms which allow a UAC to know whether a new method is understood by a UAS. This includes both the OPTIONS request, and the 405 (Method Not Allowed) response with the Accept header. It is fundamental to SIP that proxies do not need to understand the semantics of a new method in order to process it. If an extension defines a new method which must be understood by proxies in order to be processed, a Proxy-Require header is needed. As discussed above, these kinds of extensions are frowned upon.

In order to achieve backwards compatibility for extensions that define new methods, a "probing" mechanism SHOULD generally be defined as an integral component of the extension. In this mechanism, some header is included by the UAC in a standard SIP request. The UAS places some information in the response if it understands this header (and thus, the extension). If the UAC sees this information in the

response, it knows it is safe to send a request with the new method.

Another type of extension are those which require a proxy to insert headers into a request as it traverses the network, or for the UAS to insert headers into a response. Some extensions can simply insert these headers. If the UAC or UAS does not understand them, the message can still be processed correctly. These extensions are completely backwards compatible.

Most other extensions of this type will need to make use of the Supported request header mechanism. This mechanism allows a server to determine if the client can understand some extension applied to the response. If an extension is such that it requires a server to insert information into a response which must be understood in order for the response to be correctly processed, that extension SHOULD make use of [3]. By their nature, these extensions may not always be able to be applied to every response.

If an extension requires a proxy to insert a header into a request, and this header needs to be understood by both UAC and UAS to be executed correctly, a combination of the probing mechanism above, and the Supported mechanism will need to be used. An example of such an extension is the SIP Session Timer [4].

Yet another type of extension are those which define new body types to be carried in SIP messages. If the body type is to be conveyed in a request without usage of multipart, the compatibility issues mirror those of new methods. A probing mechanism is RECOMMENDED to determine if the body type is understood. If a body type is to be conveyed in a response, that type MUST only be sent if support for it was indicated in an Accept header in the request. If the body type is to be conveyed in a request with multipart, that body can either be mandatory or optional. Mandatory implies that the request cannot be processed unless the body is understood. Optional implies that the request can be processed if the body is understood. It is RECOMMENDED that extensions specify optional bodies if at all possible.

> We note that there is no defined way right now through MIME headers to indicate whether a body is mandatory or optional. This can be accomplished through a Require header, but a MIME parameter somehow seems more appropriate

3.2 Security

Security is an important component of any protocol. SIP extensions SHOULD consider how (or if) they affect usage of the general SIP security mechanisms. Most extensions should not require any new

security capabilities beyond general purpose SIP. If they do, it is likely that the security mechanism has more general purpose application, and should be considered an extension in its own right.

3.3 Usage Guidelines

All SIP extensions MUST contain guidelines defining when the extension is to be used.

For new headers, the extension MUST define the request methods the header can appear in, and what responses it can be used in. It is RECOMMENDED that this information be represented as a new row of Table 4 of RFC 2543 [1]. The extension SHOULD specify which entities (UAC, UAS, proxy, redirect, registrar) are allowed to insert the header.

3.4 Syntactic Issues

Extensions that define new methods SHOULD use all capitals for the method name. Method names SHOULD be less than 10 characters, and SHOULD attempt to convey the general meaning of the request.

Extensions that define new headers SHOULD define a compact form representation if the non-compact header is more than four characters.

Case sensitivity of parameters and values is a constant source of confusion. SIP extensions MUST clearly indicate the case sensitivity or insensitivity of every parameter, value or field they define. In general, case sensitivity is preferred because of the reduced processing requirements.

Extensions which contain freeform text MUST allow that text to be UTF-8. This ensures that SIP remains an internationalized standard. As a general guideline, freeform text is never needed by programs in order to perform protocol processing. It is usually entered by and displayed to the user. If an extension uses a parameter which can contain UTF-8 encoded characters, and that extension requires a comparison to be made of this parameter to other parameters, the comparison MUST be case sensitive. Case insensitive comparison rules for UTF-8 text are extremely complicated and are to be avoided.

Extensions which make use of dates and times MUST use the SIP-Date BNF defined in RFC 2543. No other date formats are allowed.

Extensions which include network layer addresses SHOULD permit dotted quad IPv4 addresses, IPv6 addresses in the format described in , and domain names.

Extensions which have headers containing URLs SHOULD allow any URI, not just SIP URLs.

3.5 Semantics, Semantics, Semantics

Developers of protocols often get caught up in syntax issues, without spending enough time on semantics. The semantics of a protocol are far more important. SIP extensions MUST clearly define the semantics of the extensions. Specifically, the extension MUST specify the behaviors expected of a UAC, UAS and proxy in processing the extension. This is often best described by having separate sections for each of these three elements. Each section SHOULD step through the processing rules in temporal order of the most common messaging scenario.

Processing rules generally specify actions to take (in terms of messages to send, variables to store, rules to follow) on receipt of messages and expiration of timers. If an action requires transmission of a message, the rule SHOULD outline requirements for insertion of headers or other information in the message.

The extension SHOULD specify procedures to take in exceptional conditions. This usually includes receipt of messages that are not expected, expiration of timers that handle timeouts, and presence of headers in messages when they are not expected.

3.6 Examples Section

Presence of sections in the extension giving examples of call flows and message formatting is RECOMMENDED. Extensions which define substantial new syntax SHOULD include examples of messages containing that syntax. Examples of message flows SHOULD be given to cover common cases and at least one failure or unusual case.

3.7 Overview Section

Too often, extension documents dive into detailed syntax and semantics without giving a general overview of operation. This makes understanding of the extension harder. It is RECOMMENDED that extensions have a protocol overview section which discusses the basic operation of the extension. Basic operation usually consists of the message flow, in temporal order, for the most common case covered by the extension. The most important processing rules for the elements in the call flow SHOULD be mentioned. Usage of the RFC 2119 [5] terminology in the overview section is RECOMMENDED.

3.8 Additional Considerations for New Methods

Extensions which define new methods SHOULD take into consideration, and discuss, the following issues:

- Can it contain bodies? If so, what is the meaning of the presence of those bodies? What body types are allowed?
- Can a transaction with this request method occur while another transaction, in the same and/or reverse direction, is in progress?
- What headers are allowed in requests of this method? It is recommended that this information be presented through a column of Table 4 in RFC 2543 [1].
- All SIP requests can generally be cancelled. However, an extension MAY mandate that a new method not be cancelled. In either case, handling of CANCEL SHOULD be described. In particular, the rules a UAS should follow upon cancellation of an unanswered request SHOULD be described.
- Can the request be sent within a call or not? In this context, within means that the request is sent with the same Call-ID, To and From field as an INVITE that was sent or received previously. For, example, the REGISTER method is not associated with a call, whereas the BYE method is.

3.9 Additional Considerations for New Headers or Header Parameters

The most important issue for extensions that define new headers is backwards compatibility. See Section 3.1 for a discussion of the issues. The extension MUST detail how backwards compatibility is addressed.

It is often tempting to avoid creation of a new method by overloading an existing method through a header. Headers are not meant to fundamentally alter the meaning of the method of the request. A new header SHOULD NOT change the basic semantic and processing rules of a method.

3.10 Additional Considerations for New Body Types

Because SIP can run over UDP, extensions that specify the inclusion of large bodies are frowned upon. If at all possible, the content should be included indirectly through an http URL.

4 Interactions with SIP Features

We have observed that certain capabilities of SIP continually

interact with extensions in unusual ways. Writers of extensions SHOULD consider the interactions of their extensions with these SIP capabilities, document any unusual interactions if they exist. The most common causes of problems are:

Forking: Forking by far presents the most troublesome interactions with extensions. This is generally because it can cause (1) a single transmitted request to be received by an unknown number of UASs, and (2) a single request to have multiple responses.

Tags: Tags are used to uniquely identify call legs. Their presence is neccesitated as a result of forking. They are an unfortunate exception to many SIP processing rules. Extensions should carefully consider their effect.

CANCEL and ACK: CANCEL and ACK are "special" SIP requests, in that they are exceptions to many of the general request processing rules. That is because CANCEL and ACK are always associated with another request. New methods SHOULD consider the meaning of cancellation. New headers in INVITE requests SHOULD consider whether they also need to be included in ACK.

Routing: The Route, Record-Route and Via headers are used to support message routing. New request methods SHOULD carefully consider how these headers are used.

Stateless Proxies: SIP allows a proxy to be stateless. Stateless proxies are unable to retransmit messages and cannot execute certain services. Extensions which depend on some kind of proxy processing SHOULD consider how stateless proxies affect that processing.

5 Security Considerations

The nature of this document is such that it does not introduce any new security considerations.

6 Authors Addresses

Jonathan Rosenberg
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052

email: jdrosen@dynamicsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: schulzrinne@cs.columbia.edu

7 Bibliography

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments (Proposed Standard) 2543, Internet Engineering Task Force, Mar. 1999.

[2] R. Droms, "Dynamic host configuration protocol," Request for Comments (Draft Standard) 2131, Internet Engineering Task Force, Mar. 1997.

[3] J. Rosenberg and H. Schulzrinne, "Mandating SIP extension support by servers," Internet Draft, Internet Engineering Task Force, Jan. 2000. Work in progress.

[4] S. Donovan, "SIP session timer," Internet Draft, Internet Engineering Task Force, Oct. 1999. Work in progress.

[5] S. Bradner, "Key words for use in RFCs to indicate requirement levels," Request for Comments (Best Current Practice) 2119, Internet Engineering Task Force, Mar. 1997.

Full Copyright Statement

Copyright (c) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of

developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.