Moving the Session Initiation Protocol (SIP) Towards Draft Standard
draft-sparks-sip-steps-to-draft-00

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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.

This Internet-Draft will expire on January 3, 2009.

Abstract

This document is intended to stimulate discussion and progress towards advancing SIP to Draft Standard. It points to some of the issues the working group will need to work through and proposes an approach for creating an interoperability statement.
Table of Contents

1. What will be required to take SIP to Draft Standard ........ 3
   1.1. Deciding on the size of the package to try to progress ... 3
   1.2. Understanding and adjusting to the dependencies ........ 4
   1.3. Choosing to advance rather than revise ................ 4
   1.4. Prepare one or more implementation reports ........... 5
2. Survey of RFC3261 Normative References .................. 5
3. A strawman implementation report format for RFC3261 ...... 7
4. Strawman discussion points ................................. 10
5. Security Considerations ................................. 11
6. Informative References .................................. 11
   Author's Address ...................................... 12
   Intellectual Property and Copyright Statements ........... 14
1. What will be required to take SIP to Draft Standard

[RFC2026] defines IETF standards maturity levels and what's required to progress between them. In short, to progress a standard from Proposed (where [RFC3261] is now) to Draft, we will need to show that it is reasonably stable, implemented, and interoperable. At the highest level, the activities required to advance SIP to Draft Standard will involve:

- Creating one or more RFCs reflecting the subset of the standard that actually is stable, implemented, and provably interoperable.
- Preparing an implementation report evidencing that implementation and interoperability.
- Establishing that the protocols SIP builds on are also of at least Draft quality.

Each of these activities involve a number of decisions and issues to discuss in addition to a significant writing and testing effort.

A competing, necessary, and conflicting activity is adding to and refining the specifications. Changes other than removing what we know is not working and interoperating adds additional effort before we can progress those changes to Draft.

1.1. Deciding on the size of the package to try to progress

The level of effort required to take some part of the body of SIP documents to Draft Standard depends significantly on how much of that body we take on at a time. We have a wide range of options:

- Trying to progress everything we can build an interoperability report against.
- Trying to progress everything we think is "core" (as indicated in the hitch-hiker's guide [I-D.ietf-sip-hitchhikers-guide]).
- Trying to progress everything we can agree is "core" to the "core" (3261 plus extensions like [RFC3262]).
- Trying to progress 3261 and dependencies including widely implemented small updates (such as rport [RFC3581] and the non-INVITE fixes [RFC4320]).
Trying to progress the absolute minimum of things required to take as much of 3261 itself to Draft as possible.

The optimal choice probably lies somewhere near the option labeled with the "->" in the above list. It is a portion of work that we can reasonably complete and the result would actually be useful to the overall community.

1.2. Understanding and adjusting to the dependencies

RFC 2026 states:

Note: Standards track specifications normally must not depend on other standards track specifications which are at a lower maturity level or on non standards track specifications other than referenced specifications from other standards bodies.

In other words, any normative references from a Draft Standard will need to be to other Draft or full Standard documents, or come with a strong explanation of why a "downref" or an external reference is acceptable [RFC3967].

RFC 3261 normatively depends on SDP (literally [RFC2327], but we'll be working against [RFC4566]). This is likely to be one of the harder dependencies to resolve. We will either have to progress SDP as a prerequisite, find a way to construct the documents to avoid a normative reference to SDP, or discover some sort of compromise between those extremes.

RFC 3261 also normatively depends on TLS (again the literal reference, [RFC2246], no longer makes sense - we'll be working against [RFC4346]). There is precedent for a downref in cases where the parts of the referenced documents that are used by the standard are believed to be of adequate quality. In this case, we can make the argument that the parts of TLS that are used by this specification are well enough specified and of adequate quality to qualify for draft standard. Furthermore, we anticipate being able to demonstrate interoperability of SIP's use of TLS with entirely discrete implementations (including the implementation of TLS).

Section 2 surveys the normative references in RFC3261.

1.3. Choosing to advance rather than revise

To succeed in advancing SIP to draft, we will have to be very careful to concentrate on identifying and documenting what is, and isn't, currently implemented and interoperable. There will be very strong pressure to "fix" or improve the protocol along the way. There may
be similar strong pressure to rewrite, reorganize, and otherwise clarify the existing requirements without changing them. If we decide that giving in to those pressures is the right thing to do, we have evidence that what we really need to be doing is revising with a goal of recycling at Proposed. We can optimize for progressing to Draft at the next cycle, but we need to recognize doing so is a different activity than attempting to progress to draft now. If we choose to push to Draft at this point, the changes to the specification will be primarily to remove things, not fix or clarify them. (Some fixes and clarifications, specifically those that are already provably reflected in implementation will, of course, make sense - but we will need to be wary of being at the top of a long and very slippery slope).

1.4. Prepare one or more implementation reports

RFC 2026 requires a report on the implementation and interoperability of the features in a standard before advancing it to Draft. [I-D.dusseault-impl-reports] clarifies and refines that requirement. SIP has a very large number of normative statements in its specification (RFC 3261 alone has over one thousand [RFC2119] keyword-based requirements). Most of those statements interact. It is simply infeasible to produce a detailed checklist of every combination of every requirement along with a pair of implementations that have proven that configuration works. Rather, following the guidelines in [I-D.dusseault-impl-reports], what we can do is capture a set of higher level feature descriptions and agree that testing to those features will cover the full set of combinations of requirements. The interoperability report would capture pairs of implementations where that feature worked and call out any exceptions (requirements that were not met). A strawman example of what such a set of features for RFC3261 would look like appears in Section 3. A similar approach was taken for the interoperability report for RTP/RTCP (see http://www.ietf.org/IESG/Implementations/RTP-RTCP-Implementation.txt).

Additional reports may be required for some of the normatively referenced documents. It might be possible to cover some of those (particularly [RFC3263]) simply by including them in this report.

2. Survey of RFC3261 Normative References

RFC 3261 has 26 direct normative references in the categories detailed below. We will, as part of this exercise, also need to understand the indirect references (the normative references in the documents below that are not at Draft or above, and the normative references those references contain and so on), but this version of
this document does not attempt to enumerate that closure.

- **Items already at Draft or above**
  - **RFC 2396**: (as RFC 3986 aka STD 66) : Uniform Resource Identifier (URI): Generic Syntax
  - **RFC 2279**: (as RFC 3629 aka STD 63) : UTF-8
  - **RFC 2616**: (at Draft) : HTTP
  - **RFC 2234**: (as RFC 5234 aka STD 68) : ABNF
  - **RFC 2046**: (at Draft) : MIME Part Two
  - **RFC 768**: STD 6 : UDP
  - **RFC 761**: (Did we mean to point at RFC793 : STD 7?): TCP
  - **RFC 2617**: (at Draft) : Digest authentication
  - **RFC 1123**: STD 3 : Requirements for Internet Hosts - Application and Support

- **Items at Proposed**
  - **RFC 2327**: SDP
  - **RFC 2822**: Internet Message Format - this may be on its way to draft - see draft-resnick-2822upd-06.txt</t>
  - **RFC 3263**: Locating SIP Servers
  - **RFC 3268**: Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)
  - **RFC 2806**: (now RFC 3966) : tel URIs
  - **RFC 3264**: Offer/Answer
  - **RFC 2960**: (now RFC 4960) : SCTP
  - **RFC 2183**: Content-Disposition
  - **RFC 3294**: MIME for ISUP and QSIG
  - **RFC 1847**: Security Multiparts for MIME
  - **RFC 2630**: (now RFC 3370 and 3852) : Cryptographic Message Syntax
  - **RFC 2633**: (now RFC 3851) : S/MIME Message Specification
  - **RFC 2246**: (now RFC 4346) : TLS
  - **RFC 2401**: (now RFC 4301) : IPSec Architecture

- **Items at BCP**
  - **RFC 2119**: BCP 14: Key words for use in RFCs
  - **RFC 1750**: (now RFC 4086 aka BCP 106) : Randomness Recommendations for Security
  - **RFC 2277**: BCP 18 : IETF Policy on Character Sets and Languages
3. A strawman implementation report format for RFC3261

WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING

    The following list is an initial strawman. It has not received any review.
    DO NOT USE
    This list for any purpose other than discussing what should be in this list.
    It is incomplete, probably wrong and it WILL change.
    If you think you have a use for this list outside this draft, please participate
    in the upcoming discussion and use some future version of the list that results.
    Do not use this one.

WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING

- Basic mechanics
  - Interoperable completion of a non-INVITE transaction over an unreliable transport
  - Interoperable completion of an INVITE transaction over a reliable transport
  - Interoperable completion of an INVITE/200 through proxies with a mix of reliable and unreliable transport hops
  - Interoperable completion of a SIP transaction over
    - UDP
    - TCP
    - TLS
    - SCTP
  - Interoperable completion of SIP transactions over IPv6
  - Interoperable completion of a SIP transaction using non-default transaction state machine timers
  - Demonstrate correct implementation of handling "stray" responses
  - Demonstrate correct implementation of CANCEL at an endpoint (UAS and UAC)
  - Interoperable completion of a SIP transaction with messages using non-ascii UTF8
  - Demonstrate correct implementation of detection and recovery from INVITE glare
  - Demonstrate completion of a SIP transaction with messages using a mix of multiple comma-separated values after a header
name vs multiple instances of the header name
- Demonstrate successful completion of a SIP transaction using the "received" and "rport" mechanisms
- Demonstrate correct recognition and handling of merged requests at an endpoint
- Demonstrate correct recognition and handling of transport errors occurring during a transaction
- Demonstrate interoperable implementation of the entity type negotiation mechanisms (Accept, Accept-encoding, 415 Unsupported Media Type)
- Mechanisms for locating next destination
  - Demonstrate correct implementation of RFC3263 "locating a SIP server"
  - Demonstrate correct implementation of sending to a URI that utilizes explicit or default address, port, or transport attributes.
- Redirection mechanisms
  - Demonstrate successful redirection of a request (including having the redirected request succeed).
- Dialog mechanisms
  - Interoperable establishment and termination of a session-usage dialog
  - Interoperable establishment and termination of a session-usage dialog set up through a forking proxy.
  - Interoperable establishment and termination of multiple session-usages dialogs resulting from a single INVITE forking at a forking proxy.
  - Demonstrate ability to update a remote target using a re-INVITE
  - Demonstrate correct implementation of Route/Record-Route mechanism.
- Authentication mechanisms
  - Demonstrate successful authentication using SIP Digest
    - Correct implementation of 2069/2617 negotiation
    - Successful use of auth and auth-int using MD5
  - Correct implementation of rejecting unknown algorithms
  - Demonstrate successful authentication through multiple proxies, placed both in series and in parallel on different branches of a forked request
  - Demonstrate successful exchange of S/MIME protected entities
  - Demonstrate correct implementation of identifying a SIP peer using a TLS connection (mutual and server only auth)
- Extensibility mechanisms
  - Demonstrate correct implementation of rejecting unknown methods at endpoints
  - Demonstrate correct implementation of forwarding unknown methods at proxies
  - Demonstrate correct implementation of rejecting unknown
"Require"d extensions
- Demonstrate successful negotiation and exercise of some "Require"d extension
- Demonstrate correct implementation of ignoring unknown header fields at endpoints
- Demonstrate correct implementation of forwarding unknown header fields at proxies
- Demonstrate correct implementation of rejecting unknown SIP versions
- Demonstrate correct implementation of handling requests with unknown schemes in the RURI
- Demonstrate correct handling of unknown responses
- Demonstrate correct implementation of ignoring unknown header field parameters at endpoints, and forwarding at proxies.

- URI specific features
  - Demonstrate successful transaction using a Request-URI containing escaped and non-ascii UTF8 characters
  - Demonstrate correct implementation of request generation based on a SIP URI containing embedded header fields
  - Demonstrate correct implementation of request generation based on a SIP URI containing an embedded entity (body)
  - Demonstrate correct implementation of SIP URI comparison
  - Demonstrate correct implementation of recommendations for constructing sip: URIs from tel: URIs

- Application-level features
  - Demonstrate successful offer/answer exchange using SIP
    - in INVITE/200
    - in 200/ACK
  - Demonstrate successful registration of an endpoint (creation, refreshing, and removal of a binding)
  - Demonstrate correct implementation of a "query" register
  - Demonstrate correct implementation of "soft-state" removal of unrefreshed bindings
  - Demonstrate correct manipulation of a single binding at an AoR with several bindings in place
  - Demonstrate correct implementation of wildcard de-registration
  - Demonstrate that someone will cut and paste this list into some inappropriate place without reading it before it's had review
  - Demonstrate successful query for capabilities (correct implementation of OPTIONS)

- Proxy application-level features
  - Demonstrate correct implementation of specified behavior upon receiving a CANCEL request
  - Demonstrate correct implementation of the Timer-C based mechanism for terminating an INVITE transaction
  - Demonstrate correct implementation of the Max-Forwards mechanism
  - Demonstrate correct implementation of loop-detection
Demonstrate correct implementation of merging challenges received from multiple responses to a forked request

- Demonstrate correct implementation of merging multiple contacts received in redirect responses to a forked request
- Demonstrate correct implementation of recursing on redirections responses at proxies

- 2543-compatibility (that we might be able to drop?)
  - Successful exchange of messages without a Content-Length header over UDP
  - Successful transaction without a z9Gh4bK style transaction identifier

- Other unusual things (will we be able to find implementations?)
  - Demonstrate correct implementation of stateless UASs
  - Demonstrate correct implementation of stateless proxies
  - Demonstrate successfully setting an internal clock based on a Date header field returned in a REGISTER response.
  - Demonstrate successful discovery of a registrar through multicast to (224.0.1.72)

4. Strawman discussion points

Choosing the correct level of detail for the interoperability statements in Section 3 is a balancing act. As [I-D.dusseault-impl-reports] recommends, some of these statements make other statements by implication. For instance, we are not (yet) explicitly calling out testing case-sensitivity, multipart-mime, provisional responses, or exercise of the registration duration negotiation mechanisms. Instead, those are each already contained in one or more of the statements made above and it is expected that the report will call out any problems with these features at the appropriate statement. Thus, we need to review the statements we plan to make carefully to make sure they are fine enough to convince ourselves that we have an opportunity to capture when things don't work without becoming so fine that the sheer level of detail hides the real statement of interoperability that we're trying to make. It is certain that the list as presented so far is not complete.

Here are a few seed questions for exploring how well the statements so far cover the specification (this is not intended to be exhaustive):

- Right now there is nothing covering sips:.  What do those statements need to look like?
o What kind of statement will capture that we've ensured more than one implementation exercises 413 Request Entity too large and that clients do the right thing on receiving it?

o Do we want to dive deeper into q-value processing at registrars and proxies?

o How do we make a statement that captures we've found two implementations that discover their registrar through configuration rather than using URI resolution.

o Do we dive any deeper than the statements above into proxy rewriting of Record-Route header field values in responses? (I propose no, that this a contained case as discussed above, but its less obvious that we'll be sure to remember to check and comment on problems if we can't find two independent implementations).

o How do we make a statement that captures that we've verified independent implementation of the correct behavior for request timeouts at endpoints and proxies?

The list here _is_ intended to be mostly complete, however. Please review it carefully from that perspective and help identify what needs to be added and removed to take it all the way to complete.

5. Security Considerations

This document is not known to introduce any new security issues for consideration.

6. Informative References

[I-D.dusseault-impl-reports]
Dusseault, L. and R. Sparks, "Guidance on Interoperation and Implementation Reports",
draft-dusseault-impl-reports-00 (work in progress), July 2008.

[I-D.ietf-sip-hitchhikers-guide]


Author's Address

Robert Sparks
Tekelec
17210 Campbell Road
Suite 250
Dallas, Texas  75254-4203
USA

Email: RjS@nostrum.com