Best Current Practices for SIP Interoperability
draft-kaplan-sipping-interop-bcp-01

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and 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 September 9, 2009.

Copyright and License Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info).
Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This document identifies several commonly found interoperability issues with SIP, and provides guidance to implementers for how to avoid them. This is an initial set of commonly found problems.
1. Introduction

SIP has grown both in terms of vendor/customer adoption and protocol complexity, with numerous implementations, and differing assumptions, leading to numerous interoperability issues. Unlike some other protocols, it suffers from a lack of either a single dominant vendor, or of a single autocratic standards body. The large number of vendors involved, from different regions of the World, and the differences in needs and wants of the customers of those vendors, has led to a complicated interoperability problem space.

This document lists some of the more common interoperability issues encountered in deployments, and provides BCP recommendations for how to avoid or resolve them.

2. Applicability

This draft is focused on SIP interoperability issues only.
3. Warning for Readers

WARNING WARNING WARNING WARNING WARNING WARNING WARNING

The following list is an initial strawman. It is still under 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.

Specifically, some Recommendations in this draft may well exacerbate problems, or lead to MORE or WORSE interoperability issues.

! You have been warned.!
4. General Interoperability Issues and Recommendations

The following sections detail interoperability issues found, and general guidelines and recommendations for implementers to follow to avoid interoperability problems. This is by no means an exhaustive/comprehensive list.

4.1. Response code issues

There are numerous reasons a given response code may be sent, and in some cases more than one response code may be appropriate, which has led to differing expectations and behaviors. The need to resolve such conflicts between domains of proxies has led to middle-boxes changing the response codes, which may well exacerbate the problem in the future.

In general the interoperability problems that arise are where the upstream proxies or UAC perform automatic re-attempts to alternate paths for certain response codes but not others, and such action cannot be known in advance to the downstream device. For example a 404 Not Found or 480 Temporarily Unavailable are commonly returned by a proxy when it cannot find a route to the target for any number of reasons, and this response causes some upstream nodes to try alternate paths and some not to. Because a 404/480 can be returned for a variety of reasons, some of which should cause a re-route and some not, some vendors send different response codes than 404/480 for those conditions: response codes which are more explicit about whether a re-route is the appropriate action.

Another example is 503, which seems to cover everything from temporary overload conditions, administrative-down state, permanent failure, and as a catch-all for anything not easily identified by other codes. Some devices treat this response code as a semi-permanent condition for the next-hop, and avoid sending any subsequent requests to the next-hop for a sustained period of time, which may or may not be the correct action to take. Unfortunately the upstream nodes have no idea which downstream proxy actually generated the 503.

Interoperability problems also arise due to SIP response code reason-phrases, which are supposed to be ignored from a response processing perspective. They are useful for logs and to show the user, but only the response code itself should be used for response processing. Some vendors use unique reason-phrase string values for different events of the same code number, to aid in troubleshooting, while other vendors need the reason-phrase to be the default examples from the RFC’s.
See "REGISTER response behavior" section for related problems.

4.1.1  Recommended Behavior

[Can we recommend any specific behavior about response codes here? It’s not clear we can.]

UA’s, B2BUA’s, and Proxies MUST ignore the received SIP response reason-phrase from a response processing perspective. The reason-phrase is an opaque string value useful for logging, troubleshooting, and displaying to human users. The example reason-phrases in the RFC’s are only examples, and are non-normative. Many SIP nodes generate unique reason-phrase string values to aid in troubleshooting, which is a significant value for SIP operators and users.

4.2  SIP field and message lengths

While the RFCs do not define any maximum lengths for SIP header fields (values, parameters, etc.) or SIP messages, the reality of computing technology is such that vendors often do impose maximum lengths for received fields and messages. Whether it’s due to security concerns, product architecture, logging constraints, etc., the fact is there are many systems which cannot or will not handle fields as large as other systems can generate. Although [RFC3261] does define some specific response codes (413/414/513) for this case, it does not fix the underlying interoperability issue. Devices cannot simply stop sending larger fields based on a SIP response code.

This issue has been appearing more frequently lately, with the use of embedded cookies in URIs and parameter growth.

4.2.1  Recommended Behavior

It is RECOMMENDED that a UA, B2BUA or Proxy not generate any SIP field longer than 256 bytes. This includes any header parameters, URI, and header value. It is RECOMMENDED that a Proxy, B2BUA, and UA be able to process SIP fields of at least 1024 bytes in length.

It is RECOMMENDED that a UA or B2BUA not generate any SIP message size larger than 64,000 bytes. Note this is not 65,535, in order to accommodate message growth due to header insertion in Proxies. It is RECOMMENDED that a Proxy, B2BUA, or UA be able to process SIP messages of at least 65,535 bytes.

It is RECOMMENDED that a UA or B2BUA not generate any SIP message bodies larger than 32,768 bytes. It is RECOMMENDED that a B2BUA or
UA be able to process SIP message bodies of at least 64,000 bytes in length.

4.3. SIP and TEL URI formats

Despite all RFC wording to the contrary, the SIP URI format has seen widespread use as essentially the semantic equivalent of the TEL URI, albeit with different syntax. Many provider systems treat sip:16035551212@example.net as logically equivalent to tel:+16035551212, even though the former has local scope to example.net only, and the latter has global scope. Part of the reason for this, I believe, is that originating UA’s have no real way of knowing when a URI should be one or the other – the user pressed digit buttons and hit "send", and all the UAC can do is send the request to sip:[digits]@[local-domain]. It doesn’t know the numbers pressed were global in scope, or even E.164 numbers. Only the routing proxies know this, and even then they only know the numbers they’re each responsible for. Thus we see the domain portion of SIP URIs getting replaced by middle-boxes at provider boundaries, if the username portion looks like an E.164 number.

Furthermore, many systems have either been designed or provisioned to handle only one scheme type: SIP URIs. This has led to cases where requests are rejected unless the appropriate URI scheme is used, and frequently that single common scheme needs to be used in more than just the request-URI (e.g., To and From URI’s as well). This wholesale replacement of schemes and domain names in URIs leads to interop issues when the same URIs are expected to be used for end-to-end purposes, in headers or XML bodies the middle-boxes do not or cannot change. The most recent example is [RFC4474] sip-identity.

4.3.1 Recommended Behavior

It is RECOMMENDED that a UA or B2BUA only use SIP URIs for any SIP header field values which need a URI; for example the request URI, To, From, Contact, P-Asserted-Identity, P-Called-Party-ID, P-Associated-URI, History-Info, and so on. It is RECOMMENDED that a UA, B2BUA, or Proxy be able to process/support TEL and SIPS URI’s per their associated RFC’s.

It is RECOMMENDED that SIP domains not allocate SIP URI AoR usernames which match the following regular expression syntax, unless they are actually E.164 assigned numbers for them:

^+[0-9]{11,16}\$

Note this is after removing visual-separators, and any user parameters.
Providing AoR usernames which match E.164 syntax but which are not truly E.164 assigned numbers will lead to significant interoperability issues and user confusion, if such domain interconnect with other SIP domains. One SIP Service Provider is known to currently use such usernames.

It is RECOMMENDED, that the SIP Working Group address the fundamental issue of E.164 or telephone number scope in SIP URI’s. The TEL URI also has specific syntactic and usage issues beyond simply lack of broad industry support, which may be a reason for not supporting it.

4.4. Tel URI’s and their parameters in SIP URI’s

Although the Tel URI has not seen much usage itself using the "tel" scheme, it is quite common to see SIP URI’s containing Tel-URI parameters - i.e., Tel URI’s in the form of SIP URI’s. There are many URI parameters defined for Tel-URI that are popular, and thus frequently appear in SIP URI’s, for example in the request-URI.

The most common interoperability problem in this area is the placement of the Tel URI parameters, for example the "tgrp" and "trunk-context" parameters from [RFC4904], and the "rn", "npdi", and "cic" parameters from [RFC4694]. RFC 3261 section 19.1.6 is very clear that the entire telephone-subscriber portion, including parameters, is placed in the userinfo portion of a SIP URI. Thus the Tel-URI parameters become user parameters for SIP, not SIP URI parameters.

An example, therefore, of a SIP URI containing some of the aforementioned Tel-URI parameters would be:
sip:+12125551212;npdi;tgrp=foo;trunk-context@example.com@example.net

Another common interoperability problem concerns the visual-separators. The Tel-URI format allows for visual-separators such as dashes and parenthesis to appear in Tel-URI’s, and thus in SIP URI usernames, even though they are not truly part of the username from a processing perspective. The URI username "+12125551212" is semantically equivalent to "+1(212)555-1212". For humans and IETF document examples, this provides a nice visual representation of telephone numbers. However several systems cannot accept such usernames, or do not process/route them in the same way they do ones without the visual-separators. [And in this author’s opinion, the SIP protocol should not have allowed a human-presentation-layer format to be incorporated into a machine-session-layer URI on the wire!]
4.5. Recommended Behavior

UA’s, B2BUA’s, or Proxies which convert Tel URI’s to SIP URI’s, or insert Tel URI parameters into SIP URI’s, such as those based on [RFC4904] or [RFC4694], MUST insert them as SIP URI userinfo parameters, not URI parameters.

It is RECOMMENDED that UA’s, B2BUA’s, or Proxies which process received SIP URI’s and support Tel URI parameters, such as those based on [RFC4904] or [RFC4694], be able to recognize and process such parameters regardless of whether they are SIP URI userinfo parameters or just URI parameters.

It is RECOMMENDED that UA’s, B2BUA’s, or Proxies which generate SIP or Tel URI’s containing telephone numbers do not include visual-separators in their usernames.

UA’s, B2BUA’s, or Proxies MUST be able to process SIP or Tel URI usernames containing visual-separators, in the same manner and with the same results as they do ones without visual-separators.

5. Specific Interoperability Issues

5.1. Offer-less Invites and Re-Invites

Although this is clearly a device implementation issue (i.e., a "bug"), we have seen numerous devices from different vendors have trouble handling Invites or re-Invites that do not contain SDP. For initial Invites without SDP, often the root cause for failure is that specific request routing or admission decision logic of intermediate devices depends on the SDP; for example devices which route calls based on codec, or bandwidth allocation devices, or 3PCC transcoding devices which themselves send out offer-less Invites but didn’t expect to receive such. (Apparently they never considered that a call could cross two such systems!)

For re-Invites, the delayed SDP offer model is performed for very specific use cases which are common, but were simply not envisioned by the developers of the UA’s.

5.1.1 Recommended Behavior

It is RECOMMENDED that a UAC provide the SDP offer with its INVITE requests.

A UAS or B2BUA MUST support receiving an INVITE without an SDP offer.
5.2. REGISTER response behavior

Another form of the interop problems that arise from responses is the behavior of UA’s with regard to Registration and Subscribe response handling. For example, only a minority of UA’s properly support 3xx redirects for REGISTER, even though it would be a useful mechanism for load-balancing. For REGISTER requests specifically, it would be beneficial if there was explicit documentation of what actions should be performed by the UAC.

To reinforce this point, consider that UA’s perform Registrations and Subscriptions in a fairly automatic fashion with little user interaction, and so the way in which they treat specific response codes can have dramatic consequences. For example, it is not well-defined what a UA should do when its REGISTER is rejected with a 404, or even 503, and hardly any UA’s honor the Retry-After header. A very few UA’s will give up altogether and wait for user input; some UA’s will wait a few minutes and try again, indefinitely; some will re-attempt their Registration almost immediately, even faster, and never give up. This creates numerous problems in large network deployments, and has led SBC vendors to implement various protection schemes – from dynamic hardware ACLs, to even sending a 200 ok just to shut the UA up.

[What can we say here that UA’s will actually do?]

5.3. Call-hold signaling

The legacy mechanism defined in [RFC2543] for call-hold by setting the SDP connection address to 0.0.0.0 is unfortunately far from obsolete in usage, despite the superior direction attribute (a=inactive/recvonly/sendonly) concept of [RFC3264]. To increase interoperability, some devices send both types in the re-Invite, which defeats the purpose of using a direction attribute (e.g., keeping RTCP flowing). Other vendors send the direction attribute first, and if the SDP answer does not mirror it they use the legacy approach, which leads to extraneous signaling overhead. An IETF recommendation/BCP for this is probably warranted. In hindsight [RFC3264] should have been backwards compatible (e.g., still using the 0.0.0.0 syntax with some new attribute for on-hold connection address, which would be ignored by legacy devices but used by newer ones). [note: I recognize this is SDP not SIP, but it’s a big deal and was caused by rfc2543]

5.3.1 Recommended Behavior

It is RECOMMENDED that a UA include both a direction attribute and 0.0.0.0 connection address in an SDP offer.
It is RECOMMENDED that a UA support both the direction attribute of (a=inactive/recvonly) and the 0.0.0.0 connection address semantic for on-hold, when processing received SDP offers, such that the existence of *either* one, or both, triggers on-hold behavior. It is also RECOMMENDED that the UA continue to send RTCP to the same connection address used previously, despite the connection line address being changed to 0.0.0.0. [Note: Is that legit?]

If a UA receives an SDP offer with both attributes, it is RECOMMENDED that the UA use only the direction attribute in its SDP answer, and leave the connection address line set to whatever IP Address it expects to receive RTCP on.

UA’s MUST support the direction attribute of "a=sendrecv" being an implicit default attribute, such that if another direction attribute does not exist in the SDP, sendrecv is implied. Note that a connection line of 0.0.0.0 means there is no remote address to send RTP/RTCP to, and thus implying "sendrecv" does not conflict with stopping RTP.

It is RECOMMENDED that UA’s include an "a=sendrecv" attribute in SDP, instead of relying on implicit behavior.

5.4. Contact-URI Handling

Contact URI’s represent a significant role in SIP routing behavior. Record-Route URI’s form a portion of the route set for in-dialog requests, but the Contact-URI is the final route hop to reach the UA, and many case a B2BUA. They also provide reachability information through REGISTER requests, along with the Path header fields. The common interoperability problems found with Contact-URI handling are around "unexpected" usernames, username and URI parameters, and transport information.

In particular, some Registrars/Proxy-Registrars expect the Contact URI username for REGISTER requests to match the AoR username. This behavior is generally unsafe, because it makes guessing a UA’s Contact-URI relatively easy. Conversely, some UA’s constantly generate random Contact-URI’s for every Registration refresh and out-of-dialog request they generate, in the misguided belief that it is more secure to do so.

Interoperability issues have also been found with Contact URI username and URI parameters, whereby the parameters are not reflected/copied into the request-URI for upstream requests. This is incorrect behavior (i.e., a "bug") – the entire Contact-URI MUST be reflected/copied, per RFC 3261.
Lastly, interoperability issues have been found with upstream request routing using Contact-URI’s for transport information. This is particularly true for TCP-based scenarios, where in some cases a UA is not actually reachable at its indicated TCP server port, due to Firewalls or NATs. Some UA’s will use their TCP client socket information in the URI, in the hopes of making upstream requests use the same TCP or TLS connection.

5.5. Require header values

SIP offers a means of indicating extension use, in the Require, Proxy-Require, and Supported header fields. Some SDO’s and vendors make assumptions about the capabilities of other UA’s, and do not take user perception or expectations into account. Inserting an option tag into a Require or Proxy-Require header literally implies "I don’t want this request to succeed, unless every possible recipient of this in the Universe supports this extension".

For example, the 100rel option tag for supporting PRACK is sometimes inserted in the Require header of an INVITE request. This is fundamentally unsound in actual usage. The support for PRACK is not universal, by any measure, and inserting this tag in the Require header leads to failed calls. No user or operator wants a failed call, unless it is literally impossible to succeed. Such is not the case with PRACK. There is no condition under which not getting a provisional response should ever lead to call failure, because in fact there may be no provisional responses ever sent.

The Require header field makes much more sense for requests which do not create sessions, such as REGISTER or OPTIONS; but again only if the client (and ultimately the human user) truly wants the request to fail if the other end cannot support the extension. Implementors should take note that requiring an extension may mean many failure cases, and so are advised to not implement things in such a way that they are actually required.

5.6. Rare SIP usages

Although they have not led to interoperability problems yet, there are some incredibly rare SIP usages that will probably lead to interoperability problems if anyone actually tries to use them. In this author’s opinion they should probably be deprecated or moved into an experimental RFC if they were published more than 2 years ago without success. In the mean-time, the ones which are 2 years old or older and have seen little use are noted here:

- S/MIME in [RFC3261]
- S/MIME AES [RFC3853]
- The SIPS URI scheme (sips:...) in [RFC3261]
. SDP ANAT Parameters [RFC4091] - this one *is* used by IPv6 nodes, but has issues and is rarely supported by IPv4 nodes, which basically defeats using it to begin with
. AIB Authenticated Identity Body [RFC3893]
. Callee Capabilities [RFC3840]
. Caller Preferences [RFC3841]
. The following SIP Header fields in [RFC3261]:
  Accept-Language, Call-Info, Content-Language, Error-Info, In-
  Reply-To, Mime-Version, Organization, Priority, Reply-To

[NOTE: Obviously the above list is the author’s opinion for a starting volley. It may well be some of these are wildly popular, unbeknownst to this author, but we will never find out unless we start with some list and find differently. So please don’t get upset if your favorite extension is in this list – simply email on the SIP mailing list that you’ve seen it really used in an interoperable fashion.]

6. IANA Considerations

This document makes no request of IANA.

7. Acknowledgments

Thanks to Robert Sparks for the Warning Label text idea, and Marco Lamberto’s sunnyspot.org for the skull-n-crossbones ascii art.

8. Informative References


Author’s Address

Hadriel Kaplan
Acme Packet
71 Third Ave.
Burlington, MA 01803, USA
Email: hkaplan@acmepacket.com