Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 3, 2014

P. Saint-Andre
Cisco Systems, Inc.
A. Houri
IBM
J. Hildebrand
Cisco Systems, Inc.
September 30, 2013

Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Architecture, Addresses, and Error Handling

draft-ietf-stox-core-06

Abstract

As a foundation for the definition of bidirectional protocol mappings between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP), this document specifies the architectural assumptions underlying such mappings as well as the mapping of addresses and error conditions.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The IETF has worked on two signalling technologies that can be used for multimedia session negotiation, messaging, presence, capabilities discovery, notifications, and other application-level functionality:

- o The Session Initiation Protocol [RFC3261], along with various SIP extensions developed within the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) Working Group.
- o The Extensible Messaging and Presence Protocol [RFC6120], along with various XMPP extensions developed by the IETF as well as by the XMPP Standards Foundation.

Because these technologies are widely deployed, it is important to clearly define mappings between them for the sake of interworking. This document inaugurates a series of SIP-XMPP interworking specifications by defining the architectural assumptions underlying such mappings as well as the mapping of addresses and error conditions.

The discussion venue for this document is the mailing list of the STOX WG; visit https://www.ietf.org/mailman/listinfo/stox for subscription information and discussion archives.

2. Terminology

A number of terms used here are explained in [RFC3261] and [RFC6120].

Several examples use the "XML Notation" from the IRI specification [RFC3987] to represent Unicode characters outside the ASCII range (e.g., the string "ř" stands for the Unicode character LATIN SMALL LETTER R WITH CARON).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Architectural Assumptions

Protocol translation between SIP and XMPP could occur in a number of different entities, depending on the architecture of real-time communication deployments. For example, protocol translation could occur within a multi-protocol server (which uses application-specific connection managers to initiate traffic to and accept traffic from clients or other servers natively using SIP/SIMPLE, XMPP, etc.),

within a multi-protocol client (which enables a user to establish connections natively with various servers using SIP/SIMPLE, XMPP, etc.), or within a gateway that acts as a dedicated protocol translator (which takes one protocol as input and provides another protocol as output).

This document assumes that the protocol translation will occur within a gateway. (This assumption is not meant to discourage protocol translation within multi-protocol clients or servers; instead, this assumption is followed mainly to clarify the discussion and examples so that the protocol translation principles can be more easily understood and can be applied by client and server implementors with appropriate modifications to the examples and terminology.) Specifically, we assume that the protocol translation will occur within an "XMPP-to-SIP gateway" that translates XMPP syntax and semantics on behalf of an XMPP service when communicating with SIP services and/or within a "SIP-to-XMPP gateway" that translates SIP syntax and semantics on behalf of a SIP service when communicating with XMPP services (naturally, these logical functions could occur in one and the same actual translator).

This document assumes that a gateway will translate directly from one protocol to the other. For the sake of the examples, we further assume that protocol translation will occur within a gateway in the source domain, so that information generated by the user of an XMPP service will be translated by a gateway within the trust domain of that XMPP service, and information generated by the user of a SIP service will be translated by a gateway within the trust domain of that SIP service. However, nothing in this document ought to be taken as recommending against protocol translation at the destination domain.

An architectural diagram for a possible gateway deployment is shown below, where the entities have the following significance and the "#" character is used to show the boundary of a trust domain:

o romeo@example.net -- a SIP user.
o example.net -- a SIP service with a gateway ("GW") to XMPP.
o juliet@example.com -- an XMPP user.
o example.com -- an XMPP service with a gateway ("GW") to SIP.

```
##@#########################################################
#                           #                              #
#    +-------------+----+   #   +----+-------------+     #
#    | example.net | GW |---#---| GW | example.com |     #
#    +-------------+----+   #   +----+-------------+     #
#          |                #              |               #
#     romeo@example.net     #       juliet@example.com  #
#                           #                              #
############################################################
```

4. Interdomain Federation

The architecture assumptions underlying this document imply that communication between a SIP-based service and an XMPP-based service will take place using interdomain federation.

When an XMPP service receives an XMPP stanza whose 'to' address specifies or includes a domain other than the domain of the XMPP service, it needs to determine whether the destination domain offers an XMPP service or a SIP service. To do so, it performs one or more DNS SRV lookups [RFC2782] for "_xmpp-server" records as specified in [RFC6120]. If the response returns a hostname, the service can attempt XMPP communication. If not, the service can attempt to locate a SIP service for that domain using the procedures specified in [RFC3263].

Similarly, when a SIP service receives a SIP message whose Request-URI specifies or includes a domain other than the domain of the SIP service, it needs to determine whether the destination domain offers a SIP service or an XMPP service. To do so, it uses the procedures specified in [RFC3263]. If that response returns a hostname, the service can attempt SIP communication. If not, the service can perform one or more DNS SRV lookups [RFC2782] for "_xmpp-server" records as specified in [RFC6120].

In both cases, the service in question might have previously determined that the foreign domain is a SIP service or an XMPP service, in which case it would not need to perform the relevant DNS lookups. The caching of such information is a matter of implementation and local service policy, and is therefore out of scope for this document.

Because [RFC6120] specifies a binding of XMPP to TCP, a gateway from SIP to XMPP will need to support TCP as the underlying transport protocol. By contrast, as specified in [RFC3261], either TCP or UDP can be used as the underlying transport for SIP messages, and a given SIP deployment might support only UDP; therefore, a gateway from XMPP

to SIP might need to communicate with a SIP service using either TCP or UDP.

5. Address Mapping

5.1. Overview

The basic SIP address format is a 'sip' or 'sips' URI as specified in [RFC3261]. When a SIP entity supports extensions for instant messaging it might be identified by an 'im' URI as specified in the Common Profile for Instant Messaging [RFC3860] (see [RFC3428]) and when a SIP entity supports extensions for presence it might be identified by a 'pres' URI as specified in the Common Profile for Presence [RFC3859] (see [RFC3856]). SIP entities typically also support the 'tel' URI scheme [RFC3966] and might support other URI schemes as well.

The XMPP address format is specified in [RFC6122] (although note that XMPP URIs [RFC5122] are not used natively on the XMPP network); in addition, [RFC6121] encourages instant messaging and presence applications of XMPP to also support 'im' and 'pres' URIs as specified in [RFC3860] and [RFC3859] respectively, although such support might simply involve leaving resolution of such addresses up to an XMPP server.

In this document we primarily describe mappings for addresses of the form <user@domain>; however, we also provide guidelines for mapping the addresses of specific user agent instances, which take the form of Globally Routable User Agent URIs (GRUUs) in SIP and "resourceparts" in XMPP. Mapping of protocol-specific identifiers (such as telephone numbers) is out of scope for this specification. In addition, we have ruled the mapping of domain names as out of scope for now since that is a matter for the Domain Name System; specifically, the issue for interworking between SIP and XMPP relates to the translation of fully internationalized domain names (IDNs) into non-internationalized domain names (IDNs are not allowed in the SIP address format, but are allowed in the XMPP address via Internationalized Domain Names in Applications, see [RFC6122] and [I-D.ietf-xmpp-6122bis]). Therefore, in the following sections we focus primarily on the local part of an address (these are called variously "usernames", "instant inboxes", "presentities", and "localparts" in the protocols at issue), secondarily on the instance-specific part of an address, and not at all on the domain-name part of an address.

The sip:/sips:, im:/pres:, and XMPP address schemes allow different sets of characters (although all three allow alphanumeric characters

and disallow both spaces and control characters). In some cases, characters allowed in one scheme are disallowed in others; these characters need to be mapped appropriately in order to ensure interworking across systems.

5.2. Local Part Mapping

The local part of a sip:/sips: URI inherits from the "userinfo" rule in [RFC3986] with several changes; here we discuss the SIP "user" rule only:

```
user             =  1*( unreserved / escaped / user-unreserved )
user-unreserved  =  "&" / "=" / "+" / "$" / "," / ";" / "?" / "/"
unreserved       =  alphanum / mark
mark             =  "-" / "_" / "." / "!" / "~" / "*" / "'"
                    / "(" / ")"
```

Here we make the simplifying assumption that the local part of an im:/pres: URI inherits from the "dot-atom-text" rule in [RFC5322] rather than the more complicated "local-part" rule:

```
dot-atom-text =  1*atext *("." 1*atext)
atext         =  ALPHA / DIGIT /    ; Any character except
                 "!" / "#" / "$" /  ; controls, SP, and
                 "%" / "&" / "'" /  ; specials. Used for
                 "*" / "+" / "-" /  ; atoms.
                 "/" / "=" / "?" /
                 "^" / "_" / "`" /
                 "{" / "|" / "}" /
                 "~"
```

The local part of an XMPP address allows any ASCII character except space, controls, and the " & ' / : < > @ characters.

To summarize the foregoing information, the following table lists the allowed and disallowed characters in the local part of identifiers for each protocol (aside from the alphanumeric, space, and control characters), in order by hexadecimal character number (where each "A" row shows the allowed characters and each "D" row shows the disallowed characters).

Table 1: Allowed and disallowed characters

```
+---+-----------------------------------+
| SIP/SIPS CHARACTERS                   |
+---+-----------------------------------+
| A | !  $ &'()*+,-./ ; = ?     _    ~ |
| D |  "# %          : < > @[\]^ `{|}  |
+---+-----------------------------------+
| IM/PRES CHARACTERS                    |
+---+-----------------------------------+
| A | ! #$%&'  *+ - /   = ?    ^_`{|}~ |
| D |  "     ()  , . :;< > @[\]         |
+---+-----------------------------------+
| XMPP CHARACTERS                       |
+---+-----------------------------------+
| A | ! #$%  ()*+,-.  ; = ? [\]^_`{|}~ |
| D |  "   &'       /: < > @            |
+---+-----------------------------------+
```

When transforming the local part of an address from one scheme to another, an application SHOULD proceed as follows:

1. Unescape any escaped characters in the source address (e.g., from SIP to XMPP unescape "%23" to "#" per [RFC3986] and from XMPP to SIP unescape "\27" to "'" per [XEP-0106]).
2. Leave unmodified any characters that are allowed in the destination scheme.
3. Escape any characters that are allowed in the source scheme but reserved in the destination scheme, as escaping is defined for the destination scheme. In particular:
   * Where the destination scheme is a URI (i.e., an im:, pres:, sip:, or sips: URI), each reserved character MUST be percent-encoded to "%hexhex" as specified in Section 2.5 of [RFC3986] (e.g., when transforming from XMPP to SIP, encode "#" as "%23").
   * Where the destination scheme is a native XMPP address, each reserved character MUST be encoded to "\hexhex" as specified in [XEP-0106] (e.g., when transforming from SIP to XMPP, encode "'" as "\27").

5.3. Instance-Specific Mapping

The meaning of a resourcepart in XMPP (i.e., the portion of a JID after the slash character, such as "foo" in "user@example.com/foo") matches that of a Globally Routable User Agent URI (GRUU) in SIP [RFC5627]. In both cases, these constructs identify a particular device associated with the bare JID ("localpart@domainpart") of an XMPP entity or with the Address of Record (AOR) of a SIP entity.

Therefore, it is reasonable to map the value of a "gr" URI parameter to an XMPP resourcepart, and vice-versa.

The mapping described here does not apply to temporary GRUUs, only to GRUUs associated with an Address of Record.

The "gr" URI parameter in SIP can contain only characters from the ASCII range (although characters outside the ASCII range can be percent-encoded in accordance with [RFC3986], whereas an XMPP resourcepart can contain nearly any Unicode character [UNICODE]. Therefore Unicode characters outside the ASCII range need to be mapped to characters in the ASCII range, as described below.

5.4. SIP to XMPP

The following is a high-level algorithm for mapping a sip:, sips:, im:, or pres: URI to an XMPP address:

1. Remove URI scheme.
2. Split at the first '@' character into local part and hostname (mapping the latter is out of scope).
3. Translate any percent-encoded strings ("%hexhex") to percent-decoded octets.
4. Treat result as a UTF-8 string.
5. Translate "&" to "\26", "'" to "\27", and "/" to "\2f" respectively in order to properly handle the characters disallowed in XMPP addresses but allowed in sip:/sips: URIs and im:/pres: URIs as shown in Table 1 above (this is consistent with [XEP-0106]).
6. Apply Nodeprep profile of Stringprep [RFC3454] or its replacement (see [RFC6122] and [I-D.ietf-xmpp-6122bis]) for canonicalization (OPTIONAL).
7. Recombine local part with mapped hostname to form a bare JID ("localpart@domainpart").
8. If the (SIP) address contained a "gr" URI parameter, append a slash character "/" and the "gr" value to the bare JID to form a full JID ("localpart@domainpart/resourcepart").

Several examples follow, illustrating steps 3, 5, and 8 described above (the percent-encoded string "%C3%BC" and XML Notation string "&#00FC;" both represent the Unicode character LATIN SMALL LETTER U WITH DIAERESIS).

SIP URI	XMPP Address
sip:f%C3%BC@sip.example	f&#00FC;@sip.example
sip:o'malley@sip.example	o\27malley@sip.example
sip:foo@sip.example;gr=bar	foo@sip.example/bar

5.5. XMPP to SIP

The following is a high-level algorithm for mapping an XMPP address to a sip:, sips:, im:, or pres: URI:

1. Split XMPP address into localpart (mapping described in remaining steps), domainpart (hostname; mapping is out of scope), and resourcepart (specifier for particular device or connection, for which an OPTIONAL mapping is described below).
2. Apply Nodeprep profile of [RFC3454] or its replacement (see [RFC6122] and [I-D.ietf-xmpp-6122bis]) for canonicalization of the XMPP localpart (OPTIONAL).
3. Translate "\26" to "&", "\27" to "'", and "\2f" to "/" respectively (this is consistent with [XEP-0106]).
4. Determine if the foreign domain supports im: and pres: URIs (discovered via [RFC2782] lookup as specified in [RFC6121]), else assume that the foreign domain supports sip:/sips: URIs.
5. If converting into im: or pres: URI, for each byte, if the byte is in the set (),.;[\] or is a UTF-8 character outside the ASCII range then percent-encode that byte to "%hexhex" format. If converting into sip: or sips: URI, for each byte, if the byte is in the set #%[\]^`{|} or is a UTF-8 character outside the ASCII range then percent-encode that byte to "%hexhex" format.
6. Combine resulting local part with mapped hostname to form local@domain address.
7. Prepend with 'im:' scheme (for XMPP <message/> stanzas) or 'pres:' scheme (for XMPP <presence/> stanzas) if foreign domain supports these, else prepend with 'sip:' or 'sips:' scheme according to local service policy.
8. If the XMPP address included a resourcepart and the destination URI scheme is 'sip:' or 'sips:', optionally append the slash character '/' and then append the resourcepart (making sure to percent-encode any UTF-8 characters outside the ASCII range) as the "gr" URI parameter.

Several examples follow, illustrating steps 3, 5, and 8 described above (the percent-encoded string "%C3%BC" and XML Notation string "&#00FC;" both represent the Unicode character LATIN SMALL LETTER U WITH DIAERESIS).

XMPP Address	SIP URI
tsch&#00FCss@xmpp.example	sip:tsch%C3%BCss@xmpp.example
m\26m@xmpp.example	sip:m&m@xmpp.example
baz@xmpp.example/qux	sip:baz@xmpp.example;gr=qux

6. Error Mapping

Various differences between XMPP error conditions and SIP response codes make it hard to provide a comprehensive and consistent mapping between the protocols:

- o Whereas the set of XMPP error conditions is fixed in the core XMPP specification (and supplemented where needed by application-specific extensions), the set of SIP response codes is more open to change, as evidenced by the IANA registry of SIP response codes.
- o XMPP has defined fewer error conditions related to stanza handling (22 are defined in [RFC6120]) than SIP has defined response codes related to message handling (at the date of this writing, 71 SIP response codes are registered with IANA as defined in [RFC3261] and numerous SIP extensions).
- o In many cases, the SIP response codes are more specific than the XMPP error conditions (e.g., from an XMPP perspective the SIP codes "413 Request Entity Too Large" and "414 Request-URI Too Long" are just two forms of a bad request, and the SIP codes "415 Unsupported Media Type" and "416 Unsupported URI Scheme" are just two forms of a request that is not acceptable).
- o SIP differentiates between responses about a particular endpoint or resource (the 4xx series) and responses about a user, i.e., all of a user's endpoints or resources (the 6xx series). There is no such distinction in XMPP, since the same error condition can be returned in relation to the "bare JID" (localpart@domainpart) of a user or the "full JID" (localpart@domainpart/resourcepart) of a particular endpoint or resource, depending on the 'to' address of the original request.

As a result of these and other factors, the mapping of error conditions and response codes is more of an art than a science. This document provides suggested mappings, but implementations are free to deviate from these mappings if needed. Also, because no XMPP error conditions are equivalent to the provisional (1xx) and successful (2xx) response codes in SIP, this document suggests mappings only for the SIP redirection (3xx), request failure (4xx), server failure (5xx), and global failure (6xx) response code families.

Supplementary information about SIP response codes can be expressed in the "Reason-Phrase" in the Status-Line header, and detailed information about XMPP error conditions can be expressed in the <text/> child of the <error/> element. Although the semantics of these constructs are specified in a slightly different way, it is reasonable for a gateway to map these constructs to each other if they are found in a SIP response or XMPP error stanza.

6.1. XMPP to SIP

The mapping of specific XMPP error conditions to SIP response codes SHOULD be as described in the following table.

Table 2: Mapping of XMPP error conditions to SIP response codes

XMPP Error Condition	SIP Response Code
<bad-request/>	400
<conflict/>	400
<feature-not-implemented/>	405 or 501 (1)
<forbidden/>	403 or 603 (2)
<gone/>	410
<internal-server-error/>	500
<item-not-found/>	404 or 604 (2)
<jid-malformed/>	400
<not-acceptable/>	406 or 606 (2)
<not-allowed/>	403
<not-authorized/>	401
<policy-violation/>	403
<recipient-unavailable/>	480 or 600 (2)
<redirect/>	302
<registration-required/>	407
<remote-server-not-found/>	404 or 408 (3)
<remote-server-timeout/>	408
<resource-constraint/>	500
<service-unavailable/>	see note (4) below
<subscription-required/>	407
<undefined-condition/>	400
<unexpected-request/>	491 or 400

1. If the error relates to a "full JID" (localpart@domainpart/resourcepart), the SIP 405 response code is RECOMMENDED. If the error relates to a "bare JID" (localpart@domainpart), the SIP 501 response code is RECOMMENDED.

2. If the error relates to a "full JID" (localpart@domainpart/resourcepart), the SIP response code from the 4xx series is RECOMMENDED. If the error relates to a "bare JID" (localpart@domainpart), the SIP response code from the 6xx series is RECOMMENDED.
3. The XMPP <remote-server-not-found/> error can mean either that the remote server (a) does not exist or (b) cannot be resolved. SIP has two different response codes here, 404 to cover (a) and 408 to cover (b).
4. The XMPP <service-unavailable/> error condition is widely used to inform the requesting entity that the intended recipient does not support the relevant feature, to signal that a server cannot perform the requested service either generally or in relation to a particular user, and to avoid disclosing whether a given account exists at all. This is quite different from the semantics of the SIP 503 Service Unavailable response code, which is used to signal that communication with a server is impossible (e.g., even if the XMPP <service-unavailable/> error condition is returned in relation to a specific user, the SIP 503 response code will be interpreted as applying to the entire domain, not only the specific user). Therefore, mapping the XMPP <service-unavailable/> error condition to the SIP 503 Service Unavailable response code is NOT RECOMMENDED. Although no precise mapping is available, the SIP 403 Forbidden and 405 Method Not Allowed response codes are closest in meaning to the XMPP <service-unavailable/> error condition.

6.2. SIP to XMPP

The mapping of SIP response codes to XMPP error conditions SHOULD be as described in the following table. If a gateway encounters a SIP response code that is not listed below, it SHOULD map a 3xx-series code to <redirect/>, a 4xx-series code to <bad-request/>, a 5xx-series code to <internal-server-error>, and a 6xx-series code to <recipient-unavailable/>.

Table 3: Mapping of SIP response codes to XMPP error conditions

SIP Response Code	XMPP Error Condition
3xx	<redirect/>
300	<redirect/>
301	<gone/>
302	<redirect/>
305	<redirect/>
380	<not-acceptable/>
4xx	<bad-request/>

400	<bad-request/>
401	<not-authorized/>
402	see note (1) below
403	<forbidden/> (2)
404	<item-not-found/> (3)
405	<feature-not-implemented/>
406	<not-acceptable/>
407	<registration-required/>
408	<remote-server-timeout/> (4)
410	<gone/>
413	<policy-violation/>
414	<policy-violation/>
415	<not-acceptable/>
416	<not-acceptable/>
420	<feature-not-implemented/>
421	<not-acceptable/>
423	<resource-constraint/>
430	<recipient-unavailable/> (5)
439	<feature-not-implemented/> (5)
440	<policy-violation/> (6)
480	<recipient-unavailable/>
481	<item-not-found/>
482	<not-acceptable/>
483	<not-acceptable/>
484	<item-not-found/>
485	<item-not-found/>
486	<recipient-unavailable/>
487	<recipient-unavailable/>
488	<not-acceptable/>
489	<policy-violation/> (7)
491	<unexpected-request/>
493	<bad-request/>
5xx	<internal-server-error/>
500	<internal-server-error/>
501	<feature-not-implemented/>
502	<remote-server-not-found/>
503	see note in Section 6.1
504	<remote-server-timeout/>
505	<not-acceptable/>
513	<policy-violation/>
6xx	<recipient-unavailable/>
600	<recipient-unavailable/>
603	<recipient-unavailable/>
604	<item-not-found/>
606	<not-acceptable/>

1. The XMPP <payment-required/> error condition was removed in [RFC6120].
2. Depending on the scenario, other possible translations for SIP 403 are <not-allowed/> and <policy-violation/>.
3. Depending on the scenario, another possible translation for SIP 404 is <remote-sever-not-found/>.
4. Depending on the scenario, another possible translation for SIP 408 is <remote-server-not-found/>.
5. Codes 430 and 439 are defined in [RFC5626].
6. Code 440 is defined in [RFC5393].
7. Code 489 is defined in [RFC6665].

7. IANA Considerations

This document makes no requests of IANA.

8. Security Considerations

Detailed security considerations for SIP are given in [RFC3261] and for XMPP in [RFC6120].

As specified in Section 26.4.4 of [RFC3261] and updated by [RFC5630], a To header or a Request-URI containing a SIPS URI is used to indicate that all hops in a communication path need to be protected using Transport Layer Security [RFC5246]. Because XMPP lacks a way to signal that all hops need to be encrypted, if the To header or Request-URI of a SIP message is a SIPS URI then the SIP-to-XMPP gateway MUST NOT translate the SIP message into an XMPP stanza and MUST NOT route it to the destination XMPP server.

A gateway between SIP and XMPP (in either direction) effectively acts as a SIP back-to-back user agent ("B2BUA"). The amplification vulnerability described in [RFC5393] can manifest itself with B2BUAs (see also [I-D.ietf-straw-b2bua-loop-detection]), and a gateway SHOULD implement the loop-detection methods defined in that specification to help mitigate the possibility of amplification attacks. Note that, although it would be possible to signal the Max-Forwards and Max-Breadth SIP headers over XMPP using the Stanza Headers and Internet Metadata (SHIM) extension [XEP-0131], that extension is not widely implemented; therefore, defenses against excessive looping and amplification attacks when messages pass back and forth through SIP and XMPP networks is out of scope for this document. However, it ought to be addressed in the future, and implementations are strongly encouraged to incorporate appropriate counter measures wherever possible.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5393] Sparks, R., Lawrence, S., Hawrylyshen, A., and B. Campen, "Addressing an Amplification Vulnerability in Session Initiation Protocol (SIP) Forking Proxies", RFC 5393, December 2008.

[RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.

[RFC5630] Audet, F., "The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)", RFC 5630, October 2009.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, March 2011.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.2", 2012, <http://www.unicode.org/versions/Unicode6.2.0/>.

9.2. Informative References

[I-D.ietf-straw-b2bua-loop-detection]
Kaplan, H. and V. Pascual, "Loop Detection Mechanisms for Session Initiation Protocol (SIP) Back-to- Back User Agents (B2BUAs)", draft-ietf-straw-b2bua-loop-detection-02 (work in progress), September 2013.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("STRINGPREP")", RFC 3454, December 2002.

[RFC3856] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[RFC3859] Peterson, J., "Common Profile for Presence (CPP)", RFC 3859, August 2004.

[RFC3860] Peterson, J., "Common Profile for Instant Messaging (CPIM)", RFC 3860, August 2004.

[RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004.

[RFC5122] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)", RFC 5122, February 2008.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

[RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.

[RFC6665] Roach, A., "SIP-Specific Event Notification", RFC 6665, July 2012.

[I-D.ietf-xmpp-6122bis] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", draft-ietf-xmpp-6122bis-07 (work in progress), April 2013.

[XEP-0106] Saint-Andre, P. and J. Hildebrand, "JID Escaping", XSF XEP 0106, May 2005.

[XEP-0131] Saint-Andre, P. and J. Hildebrand, "Stanza Headers and Internet Metadata", XSF XEP 0131, July 2006.

Appendix A. Acknowledgements

The authors wish to thank the following individuals for their feedback: Mary Barnes, Mike De Vries, Fabio Forno, Adrian Georgescu, Philipp Hancke, Saul Ibarra Corretge, Markus Isomaki, Olle Johansson, Paul Kyzivat, Salvatore Loreto, Daniel-Constantin Mierla, Tory Patnoe, and Robert Sparks.

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

Avshalom Houri
IBM
Rorberg Building, Pekris 3
Rehovot 76123
Israel

Email: avshalom@il.ibm.com

Joe Hildebrand
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: jhildebr@cisco.com