Internet Engineering Task Force (IETF)
Request for Comments: 6140
Updates: 3680
Category: Standards Track
ISSN: 2070-1721

A.B. Roach
Tekelec

March 2011

Registration for Multiple Phone Numbers in the Session Initiation Protocol (SIP)

Abstract

This document defines a mechanism by which a Session Initiation Protocol (SIP) server acting as a traditional Private Branch Exchange (PBX) can register with a SIP Service Provider (SSP) to receive phone calls for SIP User Agents (UAs). In order to function properly, this mechanism requires that each of the Addresses of Record (AORs) registered in bulk map to a unique set of contacts. This requirement is satisfied by AORs representing phone numbers regardless of the domain, since phone numbers are fully qualified and globally unique. This document therefore focuses on this use case.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6140.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. One of SIP's primary functions is providing rendezvous between users. By design, these rendezvous have been provided through a combination of the server look-up procedures defined in RFC 3263 [4] and the registrar procedures described in RFC 3261 [3].

The intention of the original protocol design was that any user's AOR (Address of Record) would be handled by the authority indicated by the hostport portion of the AOR. The users would register individual reachability information with this authority, which would then route incoming requests accordingly.

In actual deployments, some SIP servers have been deployed in architectures that, for various reasons, have requirements to provide dynamic routing information for large blocks of AORs, where all of the AORs in the block were to be handled by the same server. For purposes of efficiency, many of these deployments do not wish to maintain separate registrations for each of the AORs in the block. Thus, an alternate mechanism to provide dynamic routing information for blocks of AORs is desirable.

Although the use of SIP REGISTER request messages to update reachability information for multiple users simultaneously is somewhat beyond the original semantics defined for REGISTER requests by RFC 3261 [3], this approach has seen significant deployment in certain environments. In particular, deployments in which small to medium SIP-PBX servers are addressed using E.164 numbers have used this mechanism to avoid the need to maintain DNS entries or static IP addresses for the SIP-PBX servers.

In recognition of the momentum that REGISTER-based approaches have seen in deployments, this document defines a REGISTER-based approach. Since E.164-addressed UAs are very common today in SIP-PBX environments, and since SIP URIs in which the user portion is an E.164 number are always globally unique, regardless of the domain, this document focuses on registration of SIP URIs in which the user portion is an E.164 number.

2. Constraints

Within the problem space that has been established for this work, several constraints shape our solution. These are defined in the MARTINI requirements document [22] and are analyzed in Appendix A. In terms of impact to the solution at hand, the following two

constraints have the most profound effect: (1) The SIP-PBX cannot be assumed to be assigned a static IP address; and (2) No DNS entry can be relied upon to consistently resolve to the IP address of the SIP-PBX.

3. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [2].

Further, the term "SSP" is meant as an acronym for a "SIP Service Provider," while the term "SIP-PBX" is used to indicate a SIP Private Branch Exchange.

> Indented portions of the document, such as this one, form non-normative, explanatory sections of the document.

Although SIP is a text-based protocol, some of the examples in this document cannot be unambiguously rendered without additional markup due to the constraints placed on the formatting of RFCs. This document uses the <allOneLine/> markup convention established in RFC 4475 [17] to avoid ambiguity and meet the RFC layout requirements. For the sake of completeness, the text defining this markup (Section 2.1 of RFC 4475 [17]) is reproduced in its entirety below:

> Several of these examples contain unfolded lines longer than 72 characters. These are captured between <allOneLine/> tags. The single unfolded line is reconstructed by directly concatenating all lines appearing between the tags (discarding any line feeds or carriage returns). There will be no whitespace at the end of lines. Any whitespace appearing at a fold-point will appear at the beginning of a line.
>
> The following represent the same string of bits:
>
> Header-name: first value, reallylongsecondvalue, third value
>
> <allOneLine>
> Header-name: first value,
> reallylongsecondvalue
> , third value
> </allOneLine>

```
<allOneLine>
Header-name: first value,
 reallylong
second
value,
 third value
</allOneLine>
```

Note that this is NOT SIP header-line folding, where different strings of bits have equivalent meaning.

4. Mechanism Overview

The overall mechanism is achieved using a REGISTER request with a specially formatted Contact URI. This document also defines an option tag that can be used to ensure that a registrar and any intermediaries understand the mechanism described herein.

The Contact URI itself is tagged with a URI parameter to indicate that it actually represents multiple phone-number-associated contacts.

We also define some lightweight extensions to the Globally Routable UA URIs (GRUU) mechanism defined by RFC 5627 [20] to allow the use of public and temporary GRUUs assigned by the SSP.

Aside from these extensions, the REGISTER request itself is processed by a registrar in the same way as normal registrations: by updating its location service with additional AOR-to-Contact bindings.

Note that the list of AORs associated with a SIP-PBX is a matter of local provisioning at the SSP and the SIP-PBX. The mechanism defined in this document does not provide any means to detect or recover from provisioning mismatches (although the registration event package can be used as a standardized means for auditing such AORs; see Section 7.2.1).

5. Registering for Multiple Phone Numbers

5.1. SIP-PBX Behavior

To register for multiple AORs, the SIP-PBX sends a REGISTER request to the SSP. This REGISTER request varies from a typical REGISTER request in two important ways. First, it MUST contain an option tag of "gin" in both a "Require" header field and a "Proxy-Require" header field. (The option tag "gin" is an acronym for "generate implicit numbers".) Second, in at least one "Contact" header field, it MUST include a Contact URI that contains the URI parameter "bnc"

(which stands for "bulk number contact") and has no user portion (hence no "@" symbol). A URI with a "bnc" parameter MUST NOT contain a user portion. Except for the SIP URI "user" parameter, this URI MAY contain any other parameters that the SIP-PBX desires. These parameters will be echoed back by the SSP in any requests bound for the SIP-PBX.

Because of the constraints discussed in Section 2, the host portion of the Contact URI will generally contain an IP address, although nothing in this mechanism enforces or relies upon that fact. If the SIP-PBX operator chooses to maintain DNS entries that resolve to the IP address of his SIP-PBX via RFC 3263 resolution procedures, then this mechanism works just fine with domain names in the "Contact" header field.

The "bnc" URI parameter indicates that special interpretation of the Contact URI is necessary: instead of indicating the insertion of a single Contact URI into the location service, it indicates that multiple URIs (one for each associated AOR) should be inserted.

Any SIP-PBX implementing the registration mechanism defined in this document MUST also support the path mechanism defined by RFC 3327 [10], and MUST include a 'path' option tag in the "Supported" header field of the REGISTER request (which is a stronger requirement than imposed by the path mechanism itself). This behavior is necessary because proxies between the SIP-PBX and the registrar may need to insert "Path" header field values in the REGISTER request for this document's mechanism to function properly, and, per RFC 3327 [10], they can only do so if the User Agent Client (UAC) inserted the option tag in the "Supported" header field. In accordance with the procedures defined in RFC 3327 [10], the SIP-PBX is allowed to ignore the "Path" header fields returned in the REGISTER response.

5.2. Registrar Behavior

The registrar, upon receipt of a REGISTER request containing at least one "Contact" header field with a "bnc" parameter, will use the value in the "To" header field to identify the SIP-PBX for which registration is being requested. It then authenticates the SIP-PBX (e.g., using SIP digest authentication, mutual Transport Layer Security (TLS) [18], or some other authentication mechanism). After the SIP-PBX is authenticated, the registrar updates its location service with a unique AOR-to-Contact mapping for each of the AORs associated with the SIP-PBX. Semantically, each of these mappings will be treated as a unique row in the location service. The actual implementation may, of course, perform internal optimizations to reduce the amount of memory used to store such information.

For each of these unique rows, the AOR will be in the format that the SSP expects to receive from external parties (e.g., "sip:+12145550102@ssp.example.com"). The corresponding contact will be formed by adding to the REGISTER request's Contact URI a user portion containing the fully qualified, E.164-formatted number (including the preceding "+" symbol) and removing the "bnc" parameter. Aside from the initial "+" symbol, this E.164-formatted number MUST consist exclusively of digits from 0 through 9 and explicitly MUST NOT contain any visual separator symbols (e.g., "-", ".", "(", or ")"). For example, if the "Contact" header field contains the URI <sip:198.51.100.3:5060;bnc>, then the contact value associated with the aforementioned AOR will be <sip:+12145550102@198.51.100.3:5060>.

Although the SSP treats this registration as a number of discrete rows for the purpose of re-targeting incoming requests, the renewal, expiration, and removal of these rows is bound to the registered contact. In particular, this means that REGISTER requests that attempt to de-register a single AOR that has been implicitly registered MUST NOT remove that AOR from the bulk registration. In this circumstance, the registrar simply acts as if the UA attempted to unregister a contact that wasn't actually registered (e.g., return the list of presently registered contacts in a success response). A further implication of this property is that an individual extension that is implicitly registered may also be explicitly registered using a normal, non-bulk registration (subject to SSP policy). If such a registration exists, it is refreshed independently of the bulk registration and is not removed when the bulk registration is removed.

A registrar that receives a REGISTER request containing a Contact URI with both a "bnc" parameter and a user portion MUST NOT send a 200-class (Success) response. If no other error is applicable, the registrar can use a 400 (Bad Request) response to indicate this error condition.

> Note that the preceding paragraph is talking about the user portion of a URI:
>
> sip:+12145550100@example.com
> ^^^^^^^^^^^^

A registrar compliant with this document MUST support the path mechanism defined in RFC 3327 [10]. The rationale for support of this mechanism is given in Section 5.1.

Aside from the "bnc" parameter, all URI parameters present on the Contact URI in the REGISTER request MUST be copied to the contact value stored in the location service.

If the SSP servers perform processing based on User Agent Capabilities (as defined in RFC 3840 [13]), they will treat any feature tags present on a "Contact" header field with a "bnc" parameter in its URI as applicable to all of the resulting AOR-to-Contact mappings. Similarly, any option tags present on the REGISTER request that indicate special handling for any subsequent requests are also applicable to all of the AOR-to-Contact mappings.

5.3. SIP URI "user" Parameter Handling

This document does not modify the behavior specified in RFC 3261 [3] for inclusion of the "user" parameter on Request URIs. However, to avoid any ambiguity in handling at the SIP-PBX, the following normative behavior is imposed on its interactions with the SSP.

When a SIP-PBX registers with an SSP using a Contact URI containing a "bnc" parameter, that Contact URI MUST NOT include a "user" parameter. A registrar that receives a REGISTER request containing a Contact URI with both a "bnc" parameter and a "user" parameter MUST NOT send a 200-class (success) response. If no other error is applicable, the registrar can use a 400 (Bad Request) response to indicate this error condition.

> Note that the preceding paragraph is talking about the "user" parameter of a URI:
>
> sip:+12145550100@example.com;user=phone
> ^^^^^^^^^^

When a SIP-PBX receives a request from an SSP, and the Request URI contains a user portion corresponding to an AOR registered using a Contact URI containing a "bnc" parameter, then the SIP-PBX MUST NOT reject the request (or otherwise cause the request to fail) due to the absence, presence, or value of a "user" parameter on the Request URI.

6. SSP Processing of Inbound Requests

In general, after processing the AOR-to-Contact mapping described in the preceding section, the SSP proxy/registrar (or equivalent entity) performs traditional proxy/registrar behavior, based on the mapping. For any inbound SIP requests whose AOR indicates an E.164 number assigned to one of the SSP's customers, this will generally involve setting the target set to the registered contacts associated with

that AOR and performing request forwarding as described in Section 16.6 of RFC 3261 [3]. An SSP using the mechanism defined in this document MUST perform such processing for inbound INVITE requests and SUBSCRIBE requests to the "reg" event package (see Section 7.2.2) and SHOULD perform such processing for all other method types, including unrecognized SIP methods.

7. Interaction with Other Mechanisms

The following sections describe the means by which this mechanism interacts with relevant REGISTER-related extensions currently defined by the IETF.

7.1. Globally Routable User Agent URIs (GRUU)

To enable advanced services to work with UAs behind a SIP-PBX, it is important that the GRUU mechanism defined by RFC 5627 [20] work correctly with the mechanism defined by this document -- that is, that user agents served by the SIP-PBX can acquire and use GRUUs for their own use.

7.1.1. Public GRUUs

Support of public GRUUs is OPTIONAL in SSPs and SIP-PBXes.

When a SIP-PBX registers a Bulk Number Contact (a contact with a "bnc" parameter), and also invokes GRUU procedures for that contact during registration, then the SSP will assign a public GRUU to the SIP-PBX in the normal fashion. Because the URI being registered contains a "bnc" parameter, the GRUU will also contain a "bnc" parameter. In particular, this means that the GRUU will not contain a user portion.

When a UA registers a contact with the SIP-PBX using GRUU procedures, the SIP-PBX provides to the UA a public GRUU formed by adding an "sg" parameter to the GRUU parameter it received from the SSP. This "sg" parameter contains a disambiguation token that the SIP-PBX can use to route inbound requests to the proper UA.

So, for example, when the SIP-PBX registers with the following "Contact" header field:

```
Contact: <sip:198.51.100.3;bnc>;
  +sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
```

the SSP may choose to respond with a "Contact" header field that looks like this:

```
<allOneLine>
Contact: <sip:198.51.100.3;bnc>;
pub-gruu="sip:ssp.example.com;bnc;gr=urn:
uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6";
+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
;expires=7200
</allOneLine>
```

When its own UAs register using GRUU procedures, the SIP-PBX can then add whatever device identifier it feels appropriate in an "sg" parameter and present this value to its own UAs. For example, assume the UA associated with the AOR "+12145550102" sent the following "Contact" header field in its REGISTER request:

```
Contact: <sip:line-1@10.20.1.17>;
  +sip.instance="<urn:uuid:d0e2f290-104b-11df-8a39-0800200c9a66>"
```

The SIP-PBX will add an "sg" parameter to the pub-gruu it received from the SSP with a token that uniquely identifies the device (possibly the URN itself; possibly some other identifier), insert a user portion containing the fully qualified E.164 number associated with the UA, and return the result to the UA as its public GRUU. The resulting "Contact" header field sent from the SIP-PBX to the registering UA would look something like this:

```
<allOneLine>
Contact: <sip:line-1@10.20.1.17>;
pub-gruu="sip:+12145550102@ssp.example.com;gr=urn:
uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6;sg=00:05:03:5e:70:a6";
+sip.instance="<urn:uuid:d0e2f290-104b-11df-8a39-0800200c9a66>"
;expires=3600
</allOneLine>
```

When an incoming request arrives at the SSP for a GRUU corresponding to a bulk number contact ("bnc"), the SSP performs slightly different processing for the GRUU than it would for a URI without a "bnc" parameter. When the GRUU is re-targeted to the registered bulk number contact, the SSP MUST copy the "sg" parameter from the GRUU to the new target. The SIP-PBX can then use this "sg" parameter to determine to which user agent the request should be routed. For example, the first line of an INVITE request that has been re-targeted to the SIP-PBX for the UA shown above would look like this:

```
INVITE sip:+12145550102@198.51.100.3;sg=00:05:03:5e:70:a6 SIP/2.0
```

7.1.2. Temporary GRUUs

In order to provide support for privacy, the SSP SHOULD implement the temporary GRUU mechanism described in this section. Reasons for not doing so would include systems with an alternative privacy mechanism that maintains the integrity of public GRUUs (i.e., if public GRUUs are anonymized, then the anonymizer function would need to be capable of providing -- as the anonymized URI -- a globally routable URI that routes back only to the target identified by the original public GRUU).

Temporary GRUUs are used to provide anonymity for the party creating and sharing the GRUU. Being able to correlate two temporary GRUUs as having originated from behind the same SIP-PBX violates this principle of anonymity. Consequently, rather than relying upon a single, invariant identifier for the SIP-PBX in its UA's temporary GRUUs, we define a mechanism whereby the SSP provides the SIP-PBX with sufficient information for the SIP-PBX to mint unique temporary GRUUs. These GRUUs have the property that the SSP can correlate them to the proper SIP-PBX, but no other party can do so. To achieve this goal, we use a slight modification of the procedure described in Appendix A.2 of RFC 5627 [20].

The SIP-PBX needs to be able to construct a temp-gruu in a way that the SSP can decode. In order to ensure that the SSP can decode GRUUs, we need to standardize the algorithm for creation of temp-gruus at the SIP-PBX. This allows the SSP to reverse the algorithm in order to identify the registration entry that corresponds to the GRUU.

It is equally important that no party other than the SSP be capable of decoding a temporary GRUU, including other SIP-PBXes serviced by the SSP. To achieve this property, an SSP that supports temporary GRUUs MUST create and store an asymmetric key pair: {K_e1,K_e2}. K_e1 is kept secret by the SSP, while K_e2 is shared with the SIP-PBXes via provisioning.

All base64 encoding discussed in the following sections MUST use the character set and encoding defined in Section 4 of RFC 4648 [8], except that any trailing "=" characters are discarded on encoding and added as necessary to decode.

The following sections make use of the term "HMAC-SHA256-80" to describe a particular Hashed Message Authentication Code (HMAC) algorithm. In this document, HMAC-SHA256-80 is defined as the application of the SHA-256 [24] secure hashing algorithm, truncating the results to 80 bits by discarding the trailing (least-significant) bits.

7.1.2.1. Generation of "temp-gruu-cookie" by the SSP

An SSP that supports temporary GRUUs MUST include a "temp-gruu-cookie" parameter on all "Contact" header fields containing a "bnc" parameter in a 200-class REGISTER response. This "temp-gruu-cookie" MUST have the following properties:

1. It can be used by the SSP to uniquely identify the registration to which it corresponds.

2. It is encoded using base64. This allows the SIP-PBX to decode it into as compact a form as possible for use in its calculations.

3. It is of a fixed length. This allows for its extraction once the SIP-PBX has concatenated a distinguisher onto it.

4. The temp-gruu-cookie MUST NOT be forgeable by any party. In other words, the SSP needs to be able to examine the cookie and validate that it was generated by the SSP.

5. The temp-gruu-cookie MUST be invariant during the course of a registration, including any refreshes to that registration. This property is important, as it allows the SIP-PBX to examine the temp-gruu-cookie to determine whether the temp-gruus it has issued to its UAs are still valid.

The above properties can be met using the following algorithm, which is non-normative. Implementors may chose to implement any algorithm of their choosing for generation of the temp-gruu-cookie, as long as it fulfills the five properties listed above.

The registrar maintains a counter, I. This counter is 48 bits long and initialized to zero. This counter is persistently stored, using a back-end database or similar technique. When the registrar creates the first temporary GRUU for a particular SIP-PBX and instance ID (as defined by [20]), the registrar notes the current value of the counter, I_i, and increments the counter in the database. The registrar then maps I_i to the contact and instance ID using the database, a persistent hash-map, or similar technology. If the registration expires such that there are no longer any contacts with that particular instance ID bound to the GRUU, the registrar removes the mapping. Similarly, if the temporary GRUUs are invalidated due to a change in Call-ID, the registrar removes the current mapping from I_i to the AOR and instance ID, notes the current value of the counter I_j, and stores a mapping from I_j to the contact containing a "bnc" parameter and instance ID. Based on these rules, the hash-map

will contain a single mapping for each contact containing a "bnc" parameter and instance ID for which there is a currently valid registration.

The registrar maintains a symmetric key SK_a, which is regenerated every time the counter rolls over or is reset. When the counter rolls over or is reset, the registrar remembers the old value of SK_a for a while. To generate a temp-gruu-cookie, the registrar computes:

```
SA = HMAC(SK_a, I_i)
temp-gruu-cookie = base64enc(I_i || SA)
```

where || denotes concatenation. "HMAC" represents any suitably strong HMAC algorithm; see RFC 2104 [1] for a discussion of HMAC algorithms. One suitable HMAC algorithm for this purpose is HMAC-SHA256-80.

7.1.2.2. Generation of temp-gruu by the SIP-PBX

According to Section 3.2 of RFC 5627 [20], every registration refresh generates a new temp-gruu that is valid for as long as the contact remains registered. This property is both critical for the privacy properties of temp-gruu and is expected by UAs that implement the temp-gruu procedures. Nothing in this document should be construed as changing this fundamental temp-gruu property in any way. SIP-PBXes that implement temporary GRUUs MUST generate a new temp-gruu according to the procedures in this section for every registration or registration refresh from GRUU-supporting UAs attached to the SIP-PBX.

Similarly, if the registration that a SIP-PBX has with its SSP expires or is terminated, then the temp-gruu cookie it maintains with the SSP will change. This change will invalidate all the temp-gruus the SIP-PBX has issued to its UAs. If the SIP-PBX tracks this information (e.g., to include <temp-gruu> elements in registration event bodies, as described in RFC 5628 [9]), it can determine that previously issued temp-gruus are invalid by observing a change in the temp-gruu-cookie provided to it by the SSP.

A SIP-PBX that issues temporary GRUUs to its UAs MUST maintain an HMAC key: PK_a. This value is used to validate that incoming GRUUs were generated by the SIP-PBX.

To generate a new temporary GRUU for use by its own UAs, the SIP-PBX MUST generate a random distinguisher value: D. The length of this value is up to implementors, but it MUST be long enough to prevent collisions among all the temporary GRUUs issued by the SIP-PBX. A size of 80 bits or longer is RECOMMENDED. See RFC 4086 [16] for further considerations on the generation of random numbers in a security context. After generating the distinguisher D, the SIP-PBX MUST calculate:

```
M    = base64dec(SSP-cookie) || D
E    = RSA-Encrypt(K_e2, M)
PA   = HMAC(PK_a, E)

Temp-Gruu-userpart = "tgruu." || base64(E) || "." || base64(PA)
```

where || denotes concatenation. "HMAC" represents any suitably strong HMAC algorithm; see RFC 2104 [1] for a discussion of HMAC algorithms. One suitable HMAC algorithm for this purpose is HMAC-SHA256-80.

Finally, the SIP-PBX adds a "gr" parameter to the temporary GRUU that can be used to uniquely identify the UA registration record to which the GRUU corresponds. The means of generation of the "gr" parameter are left to the implementor, as long as they satisfy the properties of a GRUU as described in RFC 5627 [20].

> One valid approach for generation of the "gr" parameter is calculation of "E" and "A" as described in Appendix A.2 of RFC 5627 [20] and forming the "gr" parameter as:
>
> ```
> gr = base64enc(E) || base64enc(A)
> ```

Using this procedure may result in a temporary GRUU returned to the registering UA by the SIP-PBX that looks similar to this:

```
<allOneLine>
Contact: <sip:line-1@10.20.1.17>
;temp-gruu="sip:tgruu.MQyaRiLEd78RtaWkcP7N8Q.5qVbsasdo2pkKw@
ssp.example.com;gr=YZGSCjKD42ccxO08pA7HwAM4XNDIlMSL0HlA"
;+sip.instance="<urn:uuid:d0e2f290-104b-11df-8a39-0800200c9a66>"
;expires=3600
</allOneLine>
```

7.1.2.3. Decoding of temp-gruu by the SSP

When the SSP proxy receives a request in which the user part begins with "tgruu.", it extracts the remaining portion and splits it at the "." character into E' and PA'. It discards PA'. It then computes E by performing a base64 decode of E'. Next, it computes:

```
M = RSA-Decrypt(K_e1, E)
```

The SSP proxy extracts the fixed-length temp-gruu-cookie information from the beginning of this M and discards the remainder (which will be the distinguisher added by the SIP-PBX). It then validates this temp-gruu-cookie. If valid, it uses it to locate the corresponding SIP-PBX registration record and routes the message appropriately.

> If the non-normative, exemplary algorithm described in Section 7.1.2.1 is used to generate the temp-gruu-cookie, then this identification is performed by splitting the temp-gruu-cookie information into its 48-bit counter I and 80-bit HMAC. It validates that the HMAC matches the counter I and then uses counter I to locate the SIP-PBX registration record in its map. If the counter has rolled over or reset, this computation is performed with the current and previous SK_a.

7.1.2.4. Decoding of temp-gruu by the SIP-PBX

When the SIP-PBX receives a request in which the user part begins with "tgruu.", it extracts the remaining portion and splits it at the "." character into E' and PA'. It then computes E and PA by performing a base64 decode of E' and PA', respectively. Next, it computes:

```
PAc = HMAC(PK_a, E)
```

where HMAC is the HMAC algorithm used for the steps in Section 7.1.2.2. If this computed value for PAc does not match the value of PA extracted from the GRUU, then the GRUU is rejected as invalid.

The SIP-PBX then uses the value of the "gr" parameter to locate the UA registration to which the GRUU corresponds, and routes the message accordingly.

7.2. Registration Event Package

Neither the SSP nor the SIP-PBX is required to support the registration event package defined by RFC 3680 [12]. However, if they do support the registration event package, they MUST conform to the behavior described in this section and its subsections.

As this mechanism inherently deals with REGISTER transaction behavior, it is imperative to consider its impact on the registration event package defined by RFC 3680 [12]. In practice, there will be two main use cases for subscribing to registration data: learning about the overall registration state for the SIP-PBX and learning about the registration state for a single SIP-PBX AOR.

7.2.1. SIP-PBX Aggregate Registration State

If the SIP-PBX (or another interested and authorized party) wishes to monitor or audit the registration state for all of the AORs currently registered to that SIP-PBX, it can subscribe to the SIP registration event package at the SIP-PBX's main URI -- that is, the URI used in the "To" header field of the REGISTER request.

The NOTIFY messages for such a subscription will contain a body that contains one record for each AOR associated with the SIP-PBX. The AORs will be in the format expected to be received by the SSP (e.g., "sip:+12145550105@ssp.example.com"), and the contacts will correspond to the mapped contact created by the registration (e.g., "sip:+12145550105@98.51.100.3").

In particular, the "bnc" parameter is forbidden from appearing in the body of a reg-event NOTIFY request unless the subscriber has indicated knowledge of the semantics of the "bnc" parameter. The means for indicating this support are out of scope of this document.

Because the SSP does not necessarily know which GRUUs have been issued by the SIP-PBX to its associated UAs, these records will not generally contain the <temp-gruu> or <pub-gruu> elements defined in RFC 5628 [9]. This information can be learned, if necessary, by subscribing to the individual AOR registration state, as described in Section 7.2.2.

7.2.2. Individual AOR Registration State

As described in Section 6, the SSP will generally re-target all requests addressed to an AOR owned by a SIP-PBX to that SIP-PBX according to the mapping established at registration time. Although policy at the SSP may override this generally expected behavior, proper behavior of the registration event package requires that all

"reg" event SUBSCRIBE requests are processed by the SIP-PBX. As a consequence, the requirements on an SSP for processing registration event package SUBSCRIBE requests are not left to policy.

If the SSP receives a SUBSCRIBE request for the registration event package with a Request URI that indicates an AOR registered via the "Bulk Number Contact" mechanism defined in this document, then the SSP MUST proxy that SUBSCRIBE to the SIP-PBX in the same way that it would proxy an INVITE bound for that AOR, unless the SSP has and can maintain a copy of complete, accurate, and up-to-date information from the SIP-PBX (e.g., through an active back-end subscription).

If the Request URI in a SUBSCRIBE request for the registration event package indicates a contact that is registered by more than one SIP-PBX, then the SSP proxy will fork the SUBSCRIBE request to all the applicable SIP-PBXes. Similarly, if the Request URI corresponds to a contact that is both implicitly registered by a SIP-PBX and explicitly registered directly with the SSP proxy, then the SSP proxy will semantically fork the SUBSCRIBE request to the applicable SIP-PBX or SIP-PBXes and to the registrar function (which will respond with registration data corresponding to the explicit registrations at the SSP). The forking in both of these cases can be avoided if the SSP has and can maintain a copy of up-to-date information from the PBXes.

Section 4.9 of RFC 3680 [12] indicates that "a subscriber MUST NOT create multiple dialogs as a result of a single [registration event] subscription request". Consequently, subscribers who are not aware of the extension described by this document will accept only one dialog in response to such requests. In the case described in the preceding paragraph, this behavior will result in such clients receiving accurate but incomplete information about the registration state of an AOR. As an explicit change to the normative behavior of RFC 3680, this document stipulates that subscribers to the registration event package MAY create multiple dialogs as the result of a single subscription request. This will allow subscribers to create a complete view of an AOR's registration state.

Defining the behavior as described above is important, since the reg-event subscriber is interested in finding out about the comprehensive list of devices associated with the AOR. Only the SIP-PBX will have authoritative access to this information. For example, if the user has registered multiple UAs with differing capabilities, the SSP will not know about the devices or their capabilities. By contrast, the SIP-PBX will.

If the SIP-PBX is not registered with the SSP when a registration event subscription for a contact that would be implicitly registered if the SIP-PBX were registered is received, then the SSP SHOULD accept the subscription and indicate that the user is not currently registered. Once the associated SIP-PBX is registered, the SSP SHOULD use the subscription migration mechanism defined in RFC 3265 [5] to migrate the subscription to the SIP-PBX.

When a SIP-PBX receives a registration event subscription addressed to an AOR that has been registered using the bulk registration mechanism described in this document, then each resulting registration information document SHOULD contain an 'aor' attribute in its <registration/> element that corresponds to the AOR at the SSP.

> For example, consider a SIP-PBX that has registered with an SSP that has a domain of "ssp.example.com". The SIP-PBX used a Contact URI of "sip:198.51.100.3:5060;bnc". After such registration is complete, a registration event subscription arriving at the SSP with a Request URI of "sip:+12145550102@ssp.example.com" will be re-targeted to the SIP-PBX, with a Request URI of "sip:+12145550102@198.51.100.3:5060". The resulting registration document created by the SIP-PBX would contain a <registration/> element with an "aor" attribute of "sip:+12145550102@ssp.example.com".
>
> This behavior ensures that subscribers external to the system (and unaware of GIN (generate implicit numbers) procedures) will be able to find the relevant information in the registration document (since they will be looking for the publicly visible AOR, not the address used for sending information from the SSP to the SIP-PBX).

A SIP-PBX that supports both GRUU procedures and the registration event packages SHOULD implement the extension defined in RFC 5628 [9].

7.3. Client-Initiated (Outbound) Connections

RFC 5626 [19] defines a mechanism that allows UAs to establish long-lived TCP connections or UDP associations with a proxy in a way that allows bidirectional traffic between the proxy and the UA. This behavior is particularly important in the presence of NATs, and whenever TLS [18] security is required. Neither the SSP nor the SIP-PBX is required to support client-initiated connections.

Generally, the outbound mechanism works with the solution defined in this document, without any modifications. Implementors should note that the instance ID used between the SIP-PBX and the SSP's registrar

identifies the SIP-PBX itself, and not any of the UAs registered with the SIP-PBX. As a consequence, any attempts to use caller preferences (defined in RFC 3841 [14]) to target a specific instance are likely to fail. This shouldn't be an issue, as the preferred mechanism for targeting specific instances of a user agent is GRUU (see Section 7.1).

7.4. Non-Adjacent Contact Registration (Path) and Service-Route Discovery

RFC 3327 [10] defines a means by which a registrar and its associated proxy can be informed of a route that is to be used between the proxy and the registered user agent. The scope of the route created by a "Path" header field is contact specific; if an AOR has multiple contacts associated with it, the routes associated with each contact may be different from each other. Support for non-adjacent contact registration is required in all SSPs and SIP-PBXes implementing the multiple-AOR-registration protocol described in this document.

At registration time, any proxies between the user agent and the registrar may add themselves to the "Path" header field. By doing so, they request that any requests destined to the user agent as a result of the associated registration include them as part of the Route towards the user agent. Although the path mechanism does deliver the final path value to the registering UA, UAs typically ignore the value of the path.

To provide similar functionality in the opposite direction -- that is, to establish a route for requests sent by a registering UA -- RFC 3608 [11] defines a means by which a UA can be informed of a route that is to be used by the UA to route all outbound requests associated with the AOR used in the registration. This information is scoped to the AOR within the UA, and is not specific to the contact (or contacts) in the REGISTER request. Support of service route discovery is OPTIONAL in SSPs and SIP-PBXes.

The registrar unilaterally generates the values of the service route using whatever local policy it wishes to apply. Although it is common to use the "Path" and/or "Route" header field information in the request in composing the service route, registrar behavior is not constrained in any way that requires it to do so.

In considering the interaction between these mechanisms and the registration of multiple AORs in a single request, implementors of proxies, registrars, and intermediaries must keep in mind the following issues, which stem from the fact that GIN effectively registers multiple AORs and multiple contacts.

First, all location service records that result from expanding a single Contact URI containing a "bnc" parameter will necessarily share a single path. Proxies will be unable to make policy decisions on a contact-by-contact basis regarding whether to include themselves in the path. Second, and similarly, all AORs on the SIP-PBX that are registered with a common REGISTER request will be forced to share a common service route.

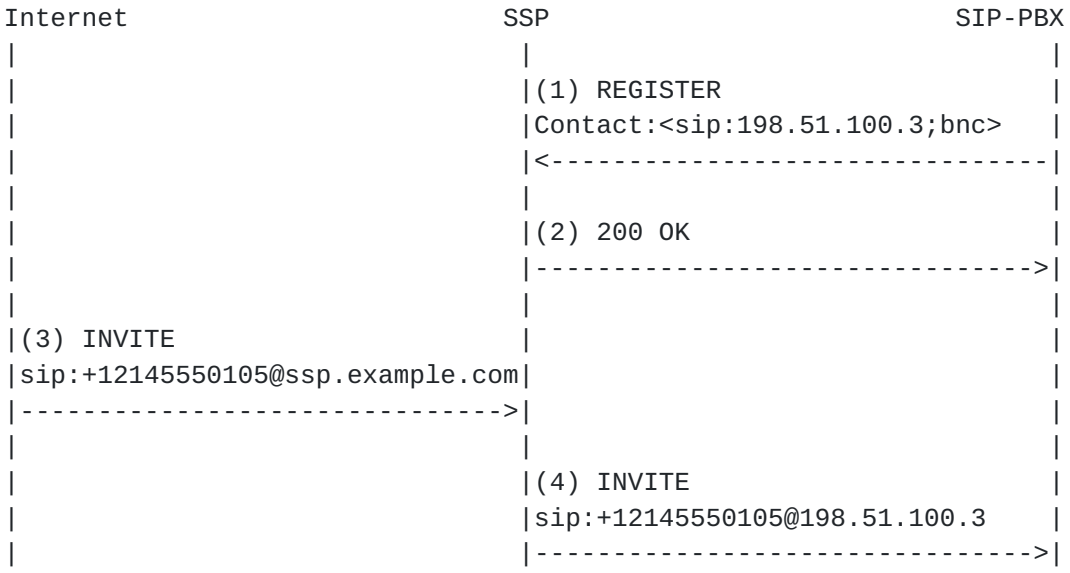
One interesting technique that the path and service route mechanisms enable is the inclusion of a token or cookie in the user portion of the service route or path entries. This token or cookie may convey information to proxies about the identity, capabilities, and/or policies associated with the user. Since this information will be shared among several AORs and several contacts when multiple AOR registration is employed, care should be taken to ensure that doing so is acceptable for all AORs and all contacts registered in a single REGISTER request.

8. Examples

Note that the following examples elide any steps related to authentication. This is done for the sake of clarity. Actual deployments will need to provide a level of authentication appropriate to their system.

8.1. Usage Scenario: Basic Registration

This example shows the message flows for a basic bulk REGISTER transaction, followed by an INVITE addressed to one of the registered UAs. Example messages are shown after the sequence diagram.



(1) The SIP-PBX registers with the SSP for a range of AORs.

```
REGISTER sip:ssp.example.com SIP/2.0
Via: SIP/2.0/UDP 198.51.100.3:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
To: <sip:pbx@ssp.example.com>
From: <sip:pbx@ssp.example.com>;tag=a23589
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Proxy-Require: gin
Require: gin
Supported: path
Contact: <sip:198.51.100.3:5060;bnc>
Expires: 7200
Content-Length: 0
```

(3) The SSP receives a request for an AOR assigned to the SIP-PBX.

```
INVITE sip:+12145550105@ssp.example.com SIP/2.0
Via: SIP/2.0/UDP foo.example;branch=z9hG4bKa0bc7a0131f0ad
Max-Forwards: 69
To: <sip:2145550105@some-other-place.example.net>
From: <sip:gsmith@example.org>;tag=456248
Call-ID: f7aecbfc374d557baf72d6352e1fbcd4
CSeq: 24762 INVITE
Contact: <sip:line-1@192.0.2.178:2081>
Content-Type: application/sdp
Content-Length: ...

<sdp body here>
```

(4) The SSP re-targets the incoming request according to the information received from the SIP-PBX at registration time.

```
INVITE sip:+12145550105@198.51.100.3 SIP/2.0
Via: SIP/2.0/UDP ssp.example.com;branch=z9hG4bKa45cd5c52a6dd50
Via: SIP/2.0/UDP foo.example;branch=z9hG4bKa0bc7a0131f0ad
Max-Forwards: 68
To: <sip:2145550105@some-other-place.example.net>
From: <sip:gsmith@example.org>;tag=456248
Call-ID: f7aecbfc374d557baf72d6352e1fbcd4
CSeq: 24762 INVITE
Contact: <sip:line-1@192.0.2.178:2081>
Content-Type: application/sdp
Content-Length: ...

<sdp body here>
```

8.2. Usage Scenario: Using Path to Control Request URI

This example shows a bulk REGISTER transaction with the SSP making use of the "Path" header field extension [10]. This allows the SSP to designate a domain on the incoming Request URI that does not necessarily resolve to the SIP-PBX when the SSP applies RFC 3263 procedures to it.

(1) The SIP-PBX registers with the SSP for a range of AORs.
    It includes the form of the URI it expects to receive in the
    Request URI in its "Contact" header field, and it includes
    information that routes to the SIP-PBX in the "Path" header
    field.

```
REGISTER sip:ssp.example.com SIP/2.0
Via: SIP/2.0/UDP 198.51.100.3:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
To: <sip:pbx@ssp.example.com>
From: <sip:pbx@ssp.example.com>;tag=a23589
Call-ID: 326983936836068@998sdasdh09
CSeq: 1826 REGISTER
Proxy-Require: gin
Require: gin
Supported: path
Path: <sip:pbx@198.51.100.3:5060;lr>
Contact: <sip:pbx.example;bnc>
Expires: 7200
Content-Length: 0
```

(3) The SSP receives a request for an AOR assigned
    to the SIP-PBX.

```
INVITE sip:+12145550105@ssp.example.com SIP/2.0
Via: SIP/2.0/UDP foo.example;branch=z9hG4bKa0bc7a0131f0ad
Max-Forwards: 69
To: <sip:2145550105@some-other-place.example.net>
From: <sip:gsmith@example.org>;tag=456248
Call-ID: 7ca24b9679ffe9aff87036a105e30d9b
CSeq: 24762 INVITE
Contact: <sip:line-1@192.0.2.178:2081>
Content-Type: application/sdp
Content-Length: ...

<sdp body here>
```

(4) The SSP re-targets the incoming request according to the information received from the SIP-PBX at registration time. Per the normal processing associated with "Path", it will insert the "Path" value indicated by the SIP-PBX at registration time in a "Route" header field, and set the Request URI to the registered contact.

```
INVITE sip:+12145550105@pbx.example SIP/2.0
Via: SIP/2.0/UDP ssp.example.com;branch=z9hG4bKa45cd5c52a6dd50
Via: SIP/2.0/UDP foo.example;branch=z9hG4bKa0bc7a0131f0ad
Route: <sip:pbx@198.51.100.3:5060;lr>
Max-Forwards: 68
To: <sip:2145550105@some-other-place.example.net>
From: <sip:gsmith@example.org>;tag=456248
Call-ID: 7ca24b9679ffe9aff87036a105e30d9b
CSeq: 24762 INVITE
Contact: <sip:line-1@192.0.2.178:2081>
Content-Type: application/sdp
Content-Length: ...

<sdp body here>
```

9. IANA Considerations

This document registers a new SIP option tag to indicate support for the mechanism it defines, two new SIP URI parameters, and a "Contact" header field parameter. The process governing registration of these protocol elements is outlined in RFC 5727 [21].

9.1. New SIP Option Tag

This section defines a new SIP option tag per the guidelines in Section 27.1 of RFC 3261 [3].

Name: gin

Description: This option tag is used to identify the extension that provides registration for Multiple Phone Numbers in SIP. When present in a "Require" or "Proxy-Require" header field of a REGISTER request, it indicates that support for this extension is required of registrars and proxies, respectively, that are a party to the registration transaction.

Reference: RFC 6140

9.2. New SIP URI Parameters

This specification defines two new SIP URI parameters, as per the registry created by RFC 3969 [7].

9.2.1. 'bnc' SIP URI Parameter

Parameter Name: bnc

Predefined Values: No (no values are allowed)

Reference: RFC 6140

9.2.2. 'sg' SIP URI Parameter

Parameter Name: sg

Predefined Values: No

Reference: RFC 6140

9.3. New SIP Header Field Parameter

This section defines a new SIP header field parameter per the registry created by RFC 3968 [6].

Header field: Contact

Parameter name: temp-gruu-cookie

Predefined values: No

Reference: RFC 6140

10. Security Considerations

The change proposed for the mechanism described in this document takes the unprecedented step of extending the previously defined REGISTER method to apply to more than one AOR. In general, this kind of change has the potential to cause problems at intermediaries -- such as proxies -- that are party to the REGISTER transaction. In particular, such intermediaries may attempt to apply policy to the user indicated in the "To" header field (i.e., the SIP-PBX's identity), without any knowledge of the multiple AORs that are being implicitly registered.

The mechanism defined by this document solves this issue by adding an option tag to a "Proxy-Require" header field in such REGISTER requests. Proxies that are unaware of this mechanism will not process the requests, preventing them from misapplying policy. Proxies that process requests with this option tag are clearly aware of the nature of the REGISTER request and can make reasonable policy decisions.

As noted in Section 7.4, intermediaries need to take care if they use a policy token in the path and service route mechanisms, as doing so will cause them to apply the same policy to all users serviced by the same SIP-PBX. This may frequently be the correct behavior, but circumstances can arise in which differentiation of user policy is required.

Section 7.4 also notes that these techniques use a token or cookie in the "Path" and/or "Service-Route" header values, and that this value will be shared among all AORs associated with a single registration. Because this information will be visible to user agents under certain conditions, proxy designers using this mechanism in conjunction with the techniques described in this document need to take care that doing so does not leak sensitive information.

One of the key properties of the outbound client connection mechanism discussed in Section 7.3 is the assurance that a single connection is associated with a single user and cannot be hijacked by other users. With the mechanism defined in this document, such connections necessarily become shared between users. However, the only entity in a position to hijack calls as a consequence is the SIP-PBX itself. Because the SIP-PBX acts as a registrar for all the potentially affected users, it already has the ability to redirect any such communications as it sees fit. In other words, the SIP-PBX must be trusted to handle calls in an appropriate fashion, and the use of the outbound connection mechanism introduces no additional vulnerabilities.

The ability to learn the identity and registration state of every user on the PBX (as described in Section 7.2.1) is invaluable for diagnostic and administrative purposes. For example, this allows the SIP-PBX to determine whether all its extensions are properly registered with the SSP. However, this information can also be highly sensitive, as many organizations may not wish to make their entire list of phone numbers available to external entities. Consequently, SSP servers are advised to use explicit (i.e., white-list) and configurable policies regarding who can access this information, with very conservative defaults (e.g., an empty access list or an access list consisting only of the PBX itself).

The procedure for the generation of temporary GRUUs requires the use of an HMAC to detect any tampering that external entities may attempt to perform on the contents of a temporary GRUU. The mention of HMAC-SHA256-80 in Section 7.1.2 is intended solely as an example of a suitable HMAC algorithm. Since all HMACs used in this document are generated and consumed by the same entity, the choice of an actual HMAC algorithm is entirely up to an implementation, provided that the cryptographic properties are sufficient to prevent third parties from spoofing GRUU-related information.

The procedure for the generation of temporary GRUUs also requires the use of RSA keys. The selection of the proper key length for such keys requires careful analysis, taking into consideration the current and foreseeable speed of processing for the period of time during which GRUUs must remain anonymous, as well as emerging cryptographic analysis methods. The most recent guidance from RSA Laboratories [25] suggests a key length of 2048 bits for data that needs protection through the year 2030, and a length of 3072 bits thereafter.

Similarly, implementors are warned to take precautionary measures to prevent unauthorized disclosure of the private key used in GRUU generation. Any such disclosure would result in the ability to correlate temporary GRUUs to each other and, potentially, to their associated PBXes.

Further, the use of RSA decryption when processing GRUUs received from arbitrary parties can be exploited by Denial-of-Service (DoS) attackers to amplify the impact of an attack: because of the presence of a cryptographic operation in the processing of such messages, the CPU load may be marginally higher when the attacker uses (valid or invalid) temporary GRUUs in the messages employed by such an attack. Normal DoS mitigation techniques, such as rate-limiting processing of received messages, should help to reduce the impact of this issue as well.

Finally, good security practices should be followed regarding the duration an RSA key is used. For implementors, this means that systems MUST include an easy way to update the public key provided to the SIP-PBX. To avoid immediately invalidating all currently issued temporary GRUUs, the SSP servers SHOULD keep the retired RSA key around for a grace period before discarding it. If decryption fails based on the new RSA key, then the SSP server can attempt to use the retired key instead. By contrast, the SIP-PBXes MUST discard the retired public key immediately and exclusively use the new public key.

11. Acknowledgements

This document represents the hard work of many people in the IETF MARTINI working group and the IETF RAI community as a whole. Particular thanks are owed to John Elwell for his requirements analysis of the mechanism described in this document, to Dean Willis for his analysis of the interaction between this mechanism and the "Path" and "Service-Route" extensions, to Cullen Jennings for his analysis of the interaction between this mechanism and the SIP Outbound extension, and to Richard Barnes for his detailed security analysis of the GRUU construction algorithm. Thanks to Eric Rescorla, whose text in the appendix of RFC 5627 was lifted directly to provide substantial portions of Section 7.1.2. Finally, thanks to Bernard Aboba, Francois Audet, Brian Carpenter, John Elwell, David Hancock, Ted Hardie, Martien Huysmans, Cullen Jennings, Alan Johnston, Hadriel Kaplan, Paul Kyzivat, and Radia Perlman for their careful reviews of and constructive feedback on this document.

12. References

12.1. Normative References

[1] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

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

[3] 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.

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

[5] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[6] Camarillo, G., "The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", BCP 98, RFC 3968, December 2004.

[7] Camarillo, G., "The Internet Assigned Number Authority (IANA) Uniform Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", BCP 99, RFC 3969, December 2004.

[8] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[9] Kyzivat, P., "Registration Event Package Extension for Session Initiation Protocol (SIP) Globally Routable User Agent URIs (GRUUs)", RFC 5628, October 2009.

12.2. Informative References

[10] Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts", RFC 3327, December 2002.

[11] Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration", RFC 3608, October 2003.

[12] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Registrations", RFC 3680, March 2004.

[13] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[14] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, August 2004.

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

[16] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[17] Sparks, R., Hawrylyshen, A., Johnston, A., Rosenberg, J., and H. Schulzrinne, "Session Initiation Protocol (SIP) Torture Test Messages", RFC 4475, May 2006.

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

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

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

[21] Peterson, J., Jennings, C., and R. Sparks, "Change Process for the Session Initiation Protocol (SIP) and the Real-time Applications and Infrastructure Area", BCP 67, RFC 5727, March 2010.

[22] Elwell, J. and H. Kaplan, "Requirements for Multiple Address of Record (AOR) Reachability Information in the Session Initiation Protocol (SIP)", RFC 5947, September 2010.

[23] Kaplan, H., "GIN with Literal AORs for SIP in SSPs (GLASS)", Work in Progress, November 2010.

[24] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-3, October 2008, <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>.

[25] Kaliski, B., "TWIRL and RSA Key Size", May 2003.

Appendix A. Requirements Analysis

The document "Requirements for Multiple Address of Record (AOR) Reachability Information in the Session Initiation Protocol (SIP)" [22] contains a list of requirements and desired properties for a mechanism to register multiple AORs with a single SIP transaction. This section evaluates those requirements against the mechanism described in this document.

REQ1 - The mechanism MUST allow a SIP-PBX to enter into a trunking arrangement with an SSP whereby the two parties have agreed on a set of telephone numbers assigned to the SIP-PBX.

The requirement is satisfied.

REQ2 - The mechanism MUST allow a set of assigned telephone numbers to comprise E.164 numbers, which can be in contiguous ranges, discrete, or in any combination of the two.

The requirement is satisfied. The Direct Inward Dialing (DID) numbers associated with a registration are established by bilateral agreement between the SSP and the SIP-PBX; they are not part of the mechanism described in this document.

REQ3 - The mechanism MUST allow a SIP-PBX to register reachability information with its SSP, in order to enable the SSP to route to the SIP-PBX inbound requests targeted at assigned telephone numbers.

The requirement is satisfied.

REQ4 - The mechanism MUST allow UAs attached to a SIP-PBX to register with the SIP-PBX for AORs based on assigned telephone numbers, in order to receive requests targeted at those telephone numbers, without needing to involve the SSP in the registration process.

The requirement is satisfied; in the presumed architecture, SIP-PBX UAs register with the SIP-PBX and require no interaction with the SSP.

REQ5 - The mechanism MUST allow a SIP-PBX to handle requests originating at its own UAs and targeted at its assigned telephone numbers, without routing those requests to the SSP.

The requirement is satisfied; SIP-PBXes may recognize their own DID numbers and GRUUs, and perform on-SIP-PBX routing without sending the requests to the SSP.

REQ6 - The mechanism MUST allow a SIP-PBX to receive requests to its assigned telephone numbers originating outside the SIP-PBX and arriving via the SSP, so that the SIP-PBX can route those requests onwards to its UAs, as it would for internal requests to those telephone numbers.

The requirement is satisfied

REQ7 - The mechanism MUST provide a means whereby a SIP-PBX knows which of its assigned telephone numbers an inbound request from its SSP is targeted at.

The requirement is satisfied. For ordinary calls and calls using public GRUUs, the DID number is indicated in the user portion of the Request URI. For calls using Temp GRUUs constructed with the mechanism described in Section 7.1.2, the "gr" parameter provides a correlation token the SIP-PBX can use to identify to which UA the call should be routed.

REQ8 - The mechanism MUST provide a means of avoiding problems due to one side using the mechanism and the other side not.

The requirement is satisfied through the 'gin' option tag and the 'bnc' Contact URI parameter.

REQ9 - The mechanism MUST observe SIP backwards compatibility principles.

The requirement is satisfied through the 'gin' option tag.

REQ10 - The mechanism MUST work in the presence of a sequence of intermediate SIP entities on the SIP-PBX-to-SSP interface (i.e., between the SIP-PBX and the SSP's domain proxy), where those intermediate SIP entities indicated during registration a need to be on the path of inbound requests to the SIP-PBX.

The requirement is satisfied through the use of the path mechanism defined in RFC 3327 [10]

REQ11 - The mechanism MUST work when a SIP-PBX obtains its IP address dynamically.

The requirement is satisfied by allowing the SIP-PBX to use an IP address in the Bulk Number Contact URI contained in a REGISTER "Contact" header field.

REQ12 - The mechanism MUST work without requiring the SIP-PBX to have a domain name or the ability to publish its domain name in the DNS.

The requirement is satisfied by allowing the SIP-PBX to use an IP address in the Bulk Number Contact URI contained in a REGISTER "Contact" header field.

REQ13 - For a given SIP-PBX and its SSP, there MUST be no impact on other domains, which are expected to be able to use normal RFC 3263 procedures to route requests, including requests needing to be routed via the SSP in order to reach the SIP-PBX.

The requirement is satisfied by allowing the domain name in the Request URI used by external entities to resolve to the SSP's servers via normal RFC 3263 resolution procedures.

REQ14 - The mechanism MUST be able to operate over a transport that provides end-to-end integrity protection and confidentiality between the SIP-PBX and the SSP, e.g., using TLS as specified in [3].

The requirement is satisfied; nothing in the proposed mechanism prevents the use of TLS between the SSP and the SIP-PBX.

REQ15 - The mechanism MUST support authentication of the SIP-PBX by the SSP and vice versa, e.g., using SIP digest authentication plus TLS server authentication as specified in [3].

The requirement is satisfied; SIP-PBXes may employ either SIP digest authentication or mutually authenticated TLS for authentication purposes.

REQ16 - The mechanism MUST allow the SIP-PBX to provide its UAs with public or temporary Globally Routable UA URIs (GRUUs) [20].

The requirement is satisfied via the mechanisms detailed in Section 7.1.

REQ17 - The mechanism MUST work over any existing transport specified for SIP, including UDP.

The requirement is satisfied to the extent that UDP can be used for REGISTER requests in general. The application of certain extensions and/or network topologies may exceed UDP MTU sizes, but such issues arise both with and without the mechanism described in this document. This document does not exacerbate such issues.

REQ18 - Documentation MUST give guidance or warnings about how authorization policies may be affected by the mechanism, to address the problems described in Section 3.3 [of RFC 5947].

These issues are addressed at length in Section 10, as well as summarized in Section 7.4.

REQ19 - The mechanism MUST be extensible to allow a set of assigned telephone numbers to comprise local numbers as specified in RFC 3966 [15], which can be in contiguous ranges, discrete, or in any combination of the two.

Assignment of telephone numbers to a registration is performed by the SSP's registrar, which is not precluded from assigning local numbers in any combination it desires.

REQ20 - The mechanism MUST be extensible to allow a set of arbitrarily assigned SIP URI's as specified in RFC 3261 [3], as opposed to just telephone numbers, without requiring a complete change of mechanism as compared to that used for telephone numbers.

The mechanism is extensible in such a fashion, as demonstrated by the document "GIN with Literal AoRs for SIP in SSPs (GLASS)" [23].

DES1 - The mechanism SHOULD allow an SSP to exploit its mechanisms for providing SIP service to normal UAs in order to provide a SIP trunking service to SIP-PBXes.

The desired property is satisfied; the routing mechanism described in this document is identical to the routing performed for singly registered AORs.

DES2 - The mechanism SHOULD scale to SIP-PBXes of several thousand assigned telephone numbers.

The desired property is satisfied; nothing in this document precludes DID number pools of arbitrary size.

DES3 - The mechanism SHOULD scale to support several thousand SIP-PBX's on a single SSP.

The desired property is satisfied; nothing in this document precludes an arbitrary number of SIP-PBXes from attaching to a single SSP.

Author's Address

Adam Roach
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

EMail: adam@nostrum.com