SIP
Internet-Draft
Expires: July 6, 2004

J. Rosenberg
dynamicsoft

January 6, 2004

Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)

draft-ietf-sip-gruu-00

Status of this Memo

Copyright Notice

Abstract

Several applications of the Session Initiation Protocol (SIP) require a user agent (UA) to construct and distribute a URI which can be used by anyone on the Internet to route a call to that specific UA instance. A URI which routes to a specific UA instance is called a Globally Routable UA URI (GRUU). This document describes an extension to SIP for obtaining a GRUU from a server, and for communicating a GRUU to a peer within a dialog.

Table of Contents

1. Introduction

Several applications of the Session Initiation Protocol (SIP) [1] require a user agent (UA) to construct and distribute a URI which can be used by anyone on the Internet to route a call to that specific UA instance. An example of such an application is call transfer, based on the REFER method [4]. Another application is the usage of endpoint-hosted conferences within the conferencing framework [8]. We call these URIs Globally Routable UA URIs (GRUU). This specification provides a mechanism for obtaining and using GRUUs.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [2] and indicate requirement levels for compliant implementations.

3. Defining a GRUU

A GRUU is a SIP URI which has a specific set of characteristics:

Global: It can be used by any UAC connected to the Internet. In that regard, it is like an address-of-record (AOR) for a user. The address-of-record for a user, sip:joe@example.com, is meant to be used by anyone to call that user. The same is true for a GRUU.

Temporally Scoped: It may be temporally scoped. In that regard, its not like an AOR for a user. The general assumption is that an AOR for a user is valid so long as the user resides within that domain (of course, policies can be imposed to limit its validity, but that is not the default case). However, a GRUU has a limited lifetime by default. It can never be valid for longer than the duration of the registration of the UA to which it is bound. For example, if my PC registers to the SIP network, a GRUU for my PC is only valid as long as my PC is registered. If the PC unregisters, the GRUU is invalid; calls to it would result in a 404. If the PC comes back, the GRUU may or may not be valid once more. Furthermore, it will frequently be the case that the GRUU has a lifetime shorter than the duration of the registration.

Instance Routing: It routes to a specific UA instance, and never forks. In that regard, it is unlike an address-of-record. When a call is made to a normal AOR which represents a user, routing logic is applied in proxies to deliver the call to one or more UAs. That logic can result in a different routing decision based on the time-of-day, or the identity of the caller. However, when a call is made to a GRUU, the routing logic is much more static. It

has to cause the call to be delivered to a very specific UA instance. That UA instance has to be the same UA instance throughout the lifetime of the GRUU. This does not mean that a GRUU represents a fundamentally different type of URI; it only means that the logic a proxy applies to a GRUU is going to generally be simpler than that it applies to a normal AOR.

4. Use Cases

We have encountered several use cases for a GRUU.

4.1 REFER

Consider a blind transfer application [12]. User A is talking to user B. A wants to transfer the call to user C. So, it sends a REFER to user C. That REFER looks like, in part:

```
REFER sip:C@example.com SIP/2.0
From: sip:A@example.com;tag=99asd
To: sip:C@example.com
Refer-To: (URI that identifiers B's UA)
```

The Refer-To header needs to contain a URI that can be used by C to place a call to B. However, this call needs to route to the specific UA which B is using to talk to A. If it didn't, the transfer service would not execute. This URI is provided to A by B. Because B doesn't know who A will transfer the call to, the URI has to be usable by anyone. Therefore, it is a GRUU.

4.2 Conferencing

A similar need arises in conferencing [8]. In that framework, a conference is described by a URI which identifies the focus of the conference. The focus is a SIP UA at the center of a conference. Each conference participant has a dialog with the focus. One case described in the framework is where a user A has made a call to B. They then put B on hold, and call C. Now, A has two separate dialogs for two separate calls - one to B, and one to C. A would like to conference them. One model is that A morphs itself into a focus. It sends a re-INVITE on each existing dialog, and provides both B and C with an updated URI that now holds the conference URI. It also has a callee capabilities [10] parameter which indicates that this URI is a conference URI. A proceeds to mix the media streams from B and C. This is called an ad-hoc conference.

At this point, normal conferencing features can be applied. That means that B can send another user, D, the conference URI, perhaps in

an email. D can send an INVITE to that URI, and join the conference. For this to work, the conference URI used by A in its re-INVITE has to be usable by anyone, and it has to route to the specific UA instance of A that is acting as the focus. If it didn't, basic conferencing features would fail. Therefore, it is a GRUU.

4.3 Presence

In a SIP-based presence [13] system, the presence agent (PA) generates notifications about the state of a user. This state is represented with the Presence Information Document Format (PIDF) [11]. In a PIDF document, a user is represented by a series of tuples, each of which identifies the devices that the user has and provides information about them. Each tuple also has a contact URI, which is a SIP URI representing that device. A watcher can make a call to that URI, with the expectation that the call is routed to the device whose presence is represented in the tuple.

The URI in the presence document therefore has to route to the specific UA instance whose presence was reported. Furthermore, since the presence document could be used by anyone who subscribes to the user, the URI has to be usable by anyone. As a result, it is a GRUU.

It is interesting to note that, in this case, the GRUU needs to be constructed by a presence agent. This may be a server in the network, or may be on an end-device, such as a PC.

5. Overview of Operation

This section is tutorial in nature, and does not specify any normative behavior.

This extension allows a UA to obtain a GRUU, and to use a GRUU. These two mechanisms are separate, in that a UA can obtain a GRUU in any way it likes, and use the mechanisms in this specification to use them. Similarly, a UA can obtain a GRUU but never use it.

A UA can obtain a GRUU by generating a normal REGISTER request, as specified in RFC 3261 [1]. This request contains a Supported header field with the value "gruu", indicating to the registrar that the UA supports this extension. If the domain that the user is registering against also supports GRUU, the REGISTER responses will contain the "gruu" parameter in each Contact header field. This parameter contains a GRUU which the domain guarantees will route to that specific Contact URI. That GRUU is guaranteed to remain valid for the duration of the registration.

Since the GRUU is a URI like any other, it can be handed out by a UA

by placing it in any header field which can contain a GRUU. A UA will normally place the GRUU into the Contact header field of dialog creating requests and responses it generates. However, it is important for the UA receiving the message to know whether the Contact URI is a GRUU or not. To make this determination, the UA looks for the presence of the Supported header field in the request or response. If it is present with a value of "gruu", it means that the Contact URI is a GRUU.

When a UA uses a GRUU, it has the option of adding the "grid" URI parameter to the GRUU. This parameter is opaque to the proxy server handling the domain. However, when the server maps the GRUU to the corresponding Contact URI, the server will copy the grid parameter into the Contact URI. As a result, when the UA receives the request, the Request URI will contain the grid parameter it placed in the corresponding GRUU.

6. User Agent Behavior

User agent behavior is divided into two separate parts - REGISTER processing, and GRUU usage.

6.1 REGISTER Processing

When a UA wishes to obtain a GRUU within the domain of its AOR, when it generates a REGISTER request (initial or refresh), it MUST include the Supported header field in the request. The value of that header field MUST include "gruu" as one of the option tags. This alerts the registrar for the domain that the UA supports the GRUU mechanism. Besides the presence of this option tag in the Supported header field, the REGISTER request is constructed identically to the case where this extension was not understood. Specifically, the Contact URI in the REGISTER request SHOULD NOT contain the gruu Contact header field parameter. Any such parameters are ignored by the registrar, as the UA cannot propose a GRUU for usage with the Contact URI.

If a UA wishes to guarantee that the request is not processed unless the domain supports and uses this extension, it MAY include a Require header field in the request with a value that contains the "gruu" option tag.

If the response is a 2xx, each Contact header field may contain a "gruu" parameter. This parameter contains a SIP URI that represents a GRUU corresponding to that Contact URI. Any requests sent to the GRUU URI will be routed by the domain to the Contact URI. The GRUU will not change in subsequent 2xx responses to REGISTER as long as the UA does not let the registration expire. However, if the UA waits until

the last moment to refresh its registration, it may cause a race condition where the registration expires while the registration is in transit. The resulting 200 OK might then contain a different GRUU. Since "last moment" is ill defined, it is RECOMMENDED that a UA be prepared to handle a change in the GRUU during a registration. Handling a change depends on the way in which it has been used. In the case where it is included in the Contact URI of a dialog initiating request or response, the UA would update the Contact URI with a target refresh request.

6.2 Using the GRUU

A UA first obtains a GRUU using the procedures of Section 6.1.

A UA can use the GRUU in the same way it would use any other SIP URI. However, a UA compliant to this specification MUST use a GRUU when populating the Contact header field of dialog-creating requests and responses. This includes the INVITE request and its 2xx response, the SUBSCRIBE [3] request, its 2xx response, and the NOTIFY request, and the REFER [4] request and its 2xx response. Similarly, in those requests and responses where the GRUU is used in the Contact header field, the UA MUST include a Supported header field that contains the option tag "gruu". However, it is not necessary for a UA to know whether or not its peer in the dialog uses a GRUU before inserting one into the Contact header field.

When placing a GRUU into the Contact header field of a request or response, a UA MAY add the "grid" URI parameter to the GRUU. This parameter MAY take on any value permitted by the grammar for the parameter. Note that there are no limitations on the size of this parameter. When a UA sends a request to the GRUU, the proxy for the domain that owns the GRUU will translate the GRUU in the Request-URI, replacing it with the corresponding Contact URI. However, it will retain the "grid" parameter when this translation is performed. As a result, when the UA receives the request, the Request-URI will contain the "grid" created by the UA. This allows the UA to effectively manufacture an infinite supply of GRUU, each of which differs by the value of the "grid" parameter. When a UA receives a request that was sent to the GRUU, it will be able to tell which GRUU was invoked by the "grid" parameter.

An implication of this behavior is that all mid-dialog requests will be routed through intermediate proxies. There will never be direct, UA to UA signaling. It is anticipated that this limitation will be addressed in future specifications.

Once a UA knows that the Contact URI provided by its peer is a GRUU, it can use it in any application or SIP extension which requires a

globally routable URI to operate. One such example is assisted call transfer.

7. Registrar Behavior

A registrar compliant to this specification is responsible for the creation and maintenance of GRUUs, and for providing those GRUU's to a UA in response to a REGISTER request.

7.1 Creation and Maintenance of GRUUs

There is a one-to-one mapping between a Contact URI for a particular AOR, and a GRUU. As a result, if two Contact/AOR pairs are different, the GRUU for each MUST be different. If two GRUUs are different, the Contact/AOR pair for those MUST be different. It is important to understand that this uniqueness is over the Contact/AOR pair, not just the Contact. For example, if a user registered the Contact sip:ua@pc.example.com to the AOR sip:user@example.com, and also registered the same Contact - sip:ua@pc.example.com to a second AOR, say sip:boss@example.com, each of those Contacts would have a different GRUU, since they belong to different AORs.

A registrar MAY create a GRUU for a particular AOR/Contact pair at any time. Of course, if a UA requests a GRUU in a registration, and the registrar has not yet created one, it will need to do so in order to respond to the registration request. However, the registrar can create the GRUU in advance of any request from a UA.

This specification does not mandate a particular mechanism for construction of the GRUU. However, the GRUU MUST exhibit the following properties:

- The domain part of the URI is an IP address present on the public Internet, or, if it is a host name, exists in the global DNS and corresponds to an IP address present on the public Internet.

- When a request is sent to this URI, it routes to a proxy server in the same domain as that of the registrar.

- A proxy server in the domain can determine that the URI is a GRUU.

- When a proxy server in this domain translates this URI, the result is equal to the Contact URI corresponding to the GRUU.

- It MUST NOT be possible, based on inspection of the URI, to determine the associated Contact URI or Address of Record.

With these rules, it is possible, though not required, to construct a

GRUU without requiring the maintenance of any additional state. To do that, the URI would be constructed in the following fashion:

```
user-part = "GRUU" + BASE64(E(K, (salt + Contact URI + AOR)))
```

Where E(K,X) represents a suitable encryption function (such as AES with 128 bit keys) with key K applied to data block X, and the "+" operator implies concatenation. Salt represents a random string that prevents a client from obtaining pairs of known plaintext and ciphertext. A good choice would be at least 128 bits of randomness in the salt.

The benefit of this mechanism is that a server need not store additional information on mapping a GRUU to its corresponding Contact URI. The user part of the GRUU can itself contain the Contact URI. Encryption is needed to prevent attacks whereby the server is sent requests with faked GRUU, causing the server to direct requests to any named URI. Even with encryption, the proxy should validate the user part after decryption. In particular, the AOR should be one managed by the proxy in that domain. Should a UA send a request with a fake GRUU, the proxy would decrypt and then discard it because there would be no URI or an invalid URI inside.

Once created, the registrar MUST maintain that GRUU for the duration over which that Contact remains registered to that AOR at the registrar. Furthermore, the registrar MUST NOT change the gruu during that duration. This is true even if the Contact is refreshed from a rebooted or different UA (known by a change in the Call-ID of the REGISTER request). After the Contact expires, the registrar ceases to maintain the binding. The registrar is under no obligation to use the same GRUU should that Contact be re-registered at a later time. Of course, it MAY create the same GRUU if it likes, but this would be an implementation choice.

The implication of these rules is that a registrar is responsible for reliable storage of the GRUU for the duration of a registration.

7.2 Providing GRUUs to User Agents

When a registrar compliant to this specification receives a REGISTER request, it checks for the presence of the Require header field in the request. If present, and if it contains the "gruu" option tag, the registrar MUST follow the procedures in the next paragraph for inclusion of the "gruu" parameter in a 2xx response to REGISTER. If not present, but a Supported header field was present with the "gruu" option tag, the registrar SHOULD follow the procedures in the next paragraph for inclusion of the "gruu" parameter in a 2xx response to REGISTER. If the Supported header field was not present, or it if was

present but did not contain the value "gruu", the registrar SHOULD NOT follow the procedures of the next paragraph for inclusion of the "gruu" parameter in a 2xx response to REGISTER.

If the register request contained any "gruu" Contact header field parameters, these MUST be ignored by the registrar. A UA cannot suggest or otherwise provide a GRUU to the registrar.

A GRUU is provided to a UA by including it in the "gruu" Contact header field parameter for a particular Contact URI. The value of this parameter is a quoted string containing the URI that is the GRUU for the associated Contact URI/AOR pair. If the server does not currently have a GRUU associated with the Contact URI, either because the Contact URI is being newly registered, or because it is being refreshed, but the registrar has not yet computed a GRUU for that Contact, one is created according to the procedures of Section 7.1. Otherwise, if a GRUU already exists for that AOR/Contact pair, the GRUU associated with that pair MUST be placed into the "gruu" Contact header field parameter of the REGISTER response.

Inclusion of a GRUU in the "gruu" Contact header field parameter of a REGISTER response is separate from the computation and storage of the GRUU. It is possible that the registrar has computed a GRUU for a contact at the request of one UA, but a different UA that queries for the current set of registrations doesn't understand GRUU. In that case, the REGISTER response sent to that second UA would not contain the "gruu" Contact header field parameter, even though the UA has a GRUU for that Contact.

8. Proxy Behavior

When a proxy server receives a request, and the proxy owns the domain in the Request URI, and the proxy is supposed to access a Location Service in order to compute request targets (as specified in Section 16.5 of RFC 3261 [1]), the proxy MUST check if the Request URI is a GRUU created by that domain.

If the URI is a GRUU, the proxy MUST determine if the Contact URI associated with the GRUU is still registered to the AOR it was registered to when the GRUU was constructed. If that AOR no longer has any registered contacts, or if it does have registered contacts, but none of them equal the Contact URI associated with the GRUU, the proxy MUST generate a 404 (Not Found) response to the request.

Otherwise, the proxy MUST populate the target set with a single URI. This URI MUST be equal to the Contact URI associated with that GRUU. Furthermore, if the GRUU contained a "grid" URI parameter, the URI in the target set MUST also contain the same parameter with the same

value.

A proxy MAY apply other processing to the request, such as execution of called party features. In particular, it is RECOMMENDED that non-routing called party features, such as call logging and screening, that are associated with the AOR are also applied to requests for the GRUU.

In many cases, a proxy will record-route an initial INVITE request, and the user agents will insert a GRUU into the Contact header field. When this happens, a mid-dialog request will arrive at the proxy with a Route header field that was inserted by the proxy, and a Request-URI that represents a GRUU. Proxies follow normal processing in this case; they will strip the Route header field, and then process the Request URI as described above.

The procedures of RFC 3261 are then followed to proxy the request. The request SHOULD NOT be redirected in this case. In many instances, a GRUU is used by a UA in order to assist in the traversal of NATs, and a redirection may prevent such a case from working.

9. Grammar

This specification defines a new Contact header field parameter, gruu, and a new URI parameter, grid.

```
contact-params    =  c-p-q / c-p-expires / c-p-gruu
                      / contact-extension
c-p-gruu          =  "gruu" EQUAL SWS DQUOTE SIP-URI DQUOTE
uri-parameter     =  transport-param / user-param / method-param
                     / ttl-param / maddr-param / lr-param / grid-param
                     / other-param
grid-param        = "grid=" pvalue
```

10. Requirements

This specification was created in order to meet the following requirements:

REQ 1: When a UA invokes a GRUU, it MUST cause the request to be routed to the specific UA instance to which the GRUU refers.

REQ 2: It MUST be possible for a GRUU to be invoked from anywhere on the Internet, and still cause the request to be routed appropriately. That is, a GRUU MUST NOT be restricted to use within a specific addressing realm.

REQ 3: It MUST be possible for a GRUU to be constructed without requiring the network to store additional state.

REQ 4: It MUST be possible for a UA to obtain a multiplicity of GRUUs, each one of which routes to that UA instance. This is needed to support ad-hoc conferencing, for example, where a a UA instance needs a different URI for each conference it is hosting.

REQ 5: When a UA receives a request sent to a GRUU, it MUST be possible for the UA to know the GRUU which was used to invoke the request. This is necessary as a consequence of requirement 4.

REQ 6: It MUST be possible for a UA to add opaque content to a GRUU, which is not interpreted or altered by the network, and used only by the UA instance to whom the GRUU refers. This provides a basic cookie type of functionality, allowing a UA to build a GRUU with state embedded within it.

REQ 7: It MUST be possible for a proxy to execute services and features on behalf of a UA instace represented by a GRUU. As an example, if a user has call blocking features, a proxy may want to apply those call blocking features to calls made to the GRUU in addition to calls made to the user's AOR.

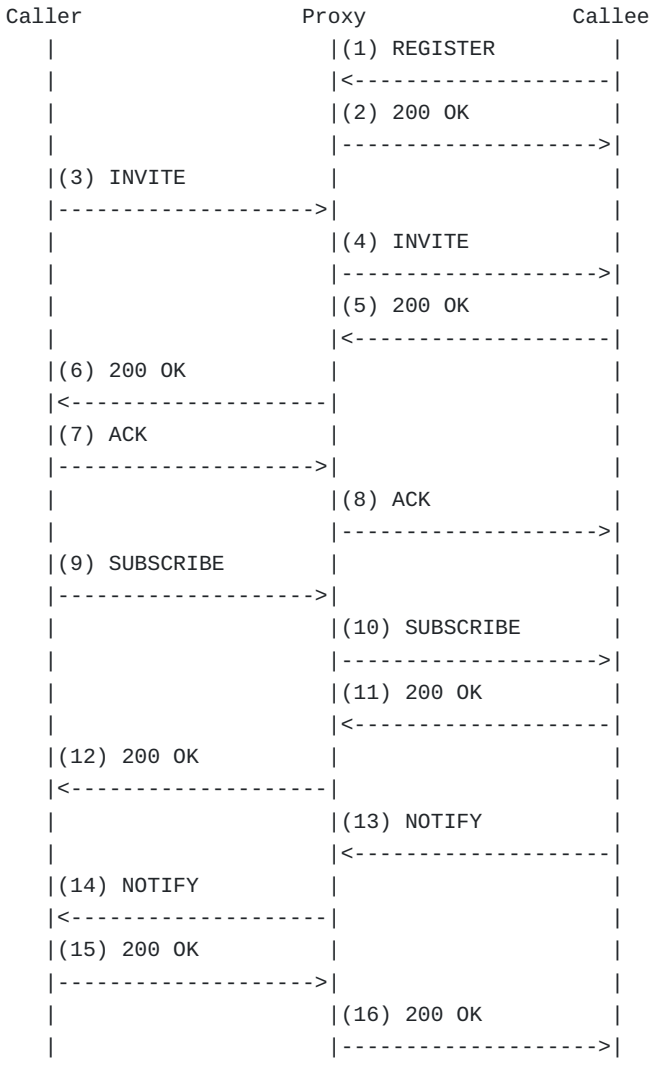
REQ 8: It MUST be possible for a UA in a dialog to inform its peer of its GRUU, and for the peer to know that the URI represents a GRUU. This is needed for the conferencing and dialog reuse applications of GRUUs, where the URIs are transferred within a dialog.

REQ 9: When transferring a GRUU per requirement 8, it MUST be possible for the UA receiving the GRUU to be assured of its integrity and authenticity.

REQ 10: It MUST be possible for a server, authoritative for a domain, to construct a GRUU which routes to a UA instace bound to an AOR in that domain. In other words, the proxy can construct a GRUU too. This is needed for the presence application.

11. Examples

The following call flow shows a basic registration and call setup, followed by a subscription directed to the GRUU.



The Callee supports the GRUU extension. As such, its REGISTER (1) looks like:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP client.example.com;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=a73kszlfl
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: 1j9FpLxk3uxtm8tn@client.example.com
CSeq: 1 REGISTER
Contact: <sip:callee@client.example.com>
Content-Length: 0
```

The REGISTER response would look like:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.example.com;branch=z9hG4bKnashds7
From: Callee <sip:callee@example.com>;tag=a73kszlfl
To: Callee <sip:callee@example.com> ;tag=b88sn
Call-ID: 1j9FpLxk3uxtm8tn@client.example.com
CSeq: 1 REGISTER
Contact: <sip:callee@client.example.com>;gruu="sip:hha9s8d=-999a@example.com"
Content-Length: 0
```

Note how the Contact header field in the REGISTER response contains the gruu parameter with the URI sip:hha9s8d=-999a@example.com. This represents a GRUU associated with the Contact URI sip:callee@client.example.com.

The INVITE from the caller is a normal SIP INVITE. The 200 OK generated by the callee, however, now contains a GRUU in the Contact header field. The UA has also chosen to include a grid URI parameter into the GRUU.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4bKnaa8
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK99a
From: Caller <sip:caller@example.com>;tag=n88ah
To: Callee <sip:callee@example.com> ;tag=a0z8
Call-ID: 1j9FpLxk3uxtma7@host.example.com
CSeq: 1 INVITE
Supported: gruu
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:hha9s8d=-999a@example.com;grid=99a>
Content-Length: --
Content-Type: application/sdp

[SDP Not shown]
```

At some point later in the call, the caller decides to subscribe to the dialog event package [9] at that specific UA. To do that, it generates a SUBSCRIBE request (message 9), but directs it towards the GRUU contained in the Contact header field.

```
SUBSCRIBE sip:hha9s8d=-999a@example.com;grid=99a SIP/2.0
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:caller@example.com>;tag=kkaz-
To: Callee <sip:callee@example.com>
Call-ID: faif9a@host.example.com
CSeq: 2 SUBSCRIBE
Supported: gruu
Event: dialog
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:bad998asd8asd0000a0@example.com>
Content-Length: 0
```

In this example, the caller itself supports the GRUU extension, and is using its own GRUU to populate the Contact header field of the SUBSCRIBE.

This request is routed to the proxy, which proceeds to perform a location lookup on the request URI. It is translated into the Contact URI bound to that GRUU, and then proxied there (message 10). Note how the grid parameter is maintained.

```
SUBSCRIBE sip:callee@client.example.com;grid=99a SIP/2.0
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4bK9555
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:caller@example.com>;tag=kkaz-
To: Callee <sip:callee@example.com>
Call-ID: faif9a@host.example.com
CSeq: 2 SUBSCRIBE
Supported: gruu
Event: dialog
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:bad998asd8asd0000a0@example.com>
Content-Length: 0
```

12. Security Considerations

Since GRUUs do not reveal information about the identity of the associated address-of-record or Contact URI, they provide routability without identity. However, GRUUs do not provide a complete or reliable solution for privacy. In particular, since the GRUU does not change during the lifetime of a registration, an attacker could correlate two calls as coming from the same source, which in and of itself reveals information about the caller. Furthermore, GRUUs do not address other aspects of privacy, such as the addresses used for media transport. For a discussion of how privacy services are provided in SIP, see RFC 3323 [7].

It is important for a UA to be assured of the integrity of a GRUU when it is given one in a REGISTER response. If the GRUU is tampered with by an attacker, the result could be denial of service to the UA. As a result, it is RECOMMENDED that a UA use the SIPS URI scheme when registering.

13. IANA Considerations

This specification defines a new Contact header field parameter and URI parameter.

13.1 Header Field Parameter

This specification defines a new header field parameter, as per the registry created by [5]. The required information is as follows:

Header field in which the parameter can appear: Contact

Name of the Parameter gruu

RFC Reference RFC XXXX [[NOTE TO IANA: Please replace XXXX with the RFC number of this specification.]]

13.2 URI Parameter

This specification defines a new SIP URI parameter, as per the registry created by [6].

Name of the Parameter grid

RFC Reference RFC XXXX [[NOTE TO IANA: Please replace XXXX with the RFC number of this specification.]]

14. Acknowledgements

The author would like to thank Rohan Mahy, Paul Kyzivat, Alan Johnston, and Cullen Jennings for their contributions to this work.

Normative References

[1] 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.

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

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

[4] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[5] Camarillo, G., "The Internet Assigned Number Authority Header Field Parameter Registry for the Session Initiation Protocol", draft-ietf-sip-parameter-registry-01 (work in progress), November 2003.

[6] Camarillo, G., "The Internet Assigned Number Authority Universal Resource Identifier Parameter Registry for the Session Initiation Protocol", draft-ietf-sip-uri-parameter-reg-01 (work in progress), November 2003.

Informative References

[7] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", RFC 3323, November 2002.

[8] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-01 (work in progress), October 2003.

[9] Rosenberg, J. and H. Schulzrinne, "An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP)", draft-ietf-sipping-dialog-package-03 (work in progress), October 2003.

[10] Rosenberg, J., "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", draft-ietf-sip-callee-caps-02 (work in progress), December 2003.

[11] Sugano, H. and S. Fujimoto, "Presence Information Data Format (PIDF)", draft-ietf-impp-cpim-pidf-08 (work in progress), May 2003.

[12] Sparks, R. and A. Johnston, "Session Initiation Protocol Call Control - Transfer", draft-ietf-sipping-cc-transfer-01 (work in progress), February 2003.

[13] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", draft-ietf-simple-presence-10 (work in progress), January 2003.

Author's Address

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.