SIP J. Rosenberg
Internet-Draft Cisco Systems
Expires: April 23, 2006 October 20, 2005

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

draft-ietf-sip-gruu-06

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 that can be used by anyone on the Internet to route a call to that specific UA instance. A URI that 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

The Session Initiation Protocol, RFC 3261 [1], is used to establish and maintain a dialog between a pair of user agents in order to manage a communications session. Messages within the dialog are sent from one user agent to another using a series of proxy hops called the route set. They are eventually delivered to the remote target (the user agent on the other side of the dialog). This remote target is identified by a SIP URI obtained from the value of the Contact header field in INVITE requests and responses.

RFC 3261 mandates that a user agent populate the Contact header field in INVITE requests and responses with a URI that is global (meaning that it can be used from any element connected to the Internet) and that routes to the user agent which inserted it. RFC 3261 also mandates that this URI be valid for requests sent outside of the dialog in which the Contact URI was inserted.

In practice, these requirements have proven very difficult to meet. Few endpoints have a hostname that is is present in DNS. Many endpoints have an IP address that is private because the client is behind a NAT. Techniques like the Simple Traversal of UDP Through NAT (STUN) [15] can be used to obtain IP addresses on the public Internet. However, many firewalls will prohibit incoming SIP requests from reaching a client unless they first pass through a proxy sitting in the DMZ of the network. Thus, URIs using STUN-obtained IP addresses often do not work.

Because of these difficulties, most clients have actually been inserting URIs into the Contact header field of requests and responses with the form sip:<IP-address>. These have the property of routing to the client, but they are generally only reachable from the proxy to which the user is directly connected. This limitation does not prevent SIP calls to an Address-of-Record (AOR) from proceeding because the user's proxy can usually reach these private addresses, and the proxy itself is generally reachable over the public network. However, this issue has impacted the ability of several other SIP mechanisms and applications to work properly.

An example of such an application is call transfer [25], based on the REFER method [7]. Another application is the usage of endpoint-hosted conferences within the conferencing framework [17]. Both of these mechanisms require that the endpoint be able to construct a URI that not only routes to that user agent, but is usable by entities anywhere on the Internet as a target for new SIP requests.

This specification formally defines a type of URI called a Globally Routable User Agent URI (GRUU) which has the properties of routing to

the UA and being reachable from anywhere. Furthermore, it defines a new mechanism by which a client can obtain a GRUU from its SIP provider, allowing it to use that URI in the Contact header fields of its dialog-forming requests and responses. Because the GRUU is provided by the user's SIP provider, the GRUU properties can be guaranteed by the provider. As a result, the various applications which require the GRUU property, including transfer, presence, and conferencing, can work reliably.

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 [5] and indicate requirement levels for compliant implementations.

This specification defines the following additional terms:

contact: The term "contact", when used in all lowercase, refers to a URI that is bound to an AOR or GRUU by means of a registration. A contact is usually a SIP URI, and is bound to the AOR and GRUU through a REGISTER request by appearing as the value of the Contact header field.

remote target: The term "remote target" refers to a URI that a user agent uses to identify itself for receipt of both mid-dialog and out-of-dialog requests. A remote target is established by placing a URI in the Contact header field of a dialog-forming request or response.

Contact header field: The term "Contact header field", with a capitalized C, refers to the header field which can appear in REGISTER requests and responses, redirects, or in dialog-creating requests and responses. Depending on the semantics, the Contact header field sometimes conveys a contact, and sometimes conveys a remote target.

3. Defining a GRUU

URIs have properties, which are granted to the URI based on the policies of the domain that owns the URI. Those properties are not visible by inspection of the URI. In this context, the domain that owns the URI is the one indicated in the host part of the SIP URI. Some of the properties that a domain can confer upon a URI are:

The AOR property: A URI has the Address of Record (AOR) property if a domain will allow it to appear in the To header field of REGISTER request.

The alias property: A URI is an alias if its treatment by the domain is identical to another URI.

The service treatment property: A URI has the service treatment property if the domain will apply applications, features, and services to calls made by, or made to, that URI, possibly based on associating that URI with a user that has "subscribed" to various features.

The anonymous property: A URI has the anonymous property when it is not possible, by inspection of the URI, to discern the user with whom the URI is associated.

The identity property: A URI is considered an identity when the domain will authorize it as a valid value in the >From header field of a request, such that an authentication service will sign a request with that URI [20].

This specification focuses on a property, called the Globally Routable User Agent URI (GRUU) property. A URI possesses this property when the following is true:

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

Routes to a Single Instance: A request sent to that URI will be routed to a specific UA instance. In that regard, it is unlike the address-of-record property. When a request is sent to a URI with the AOR property, routing logic is applied in proxies to deliver the request 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 request is made to a URI with the GRUU property, the routing logic is dictated by the GRUU property. The request has to be delivered to a very specific UA instance. That UA instance has to be the same UA instance for all requests sent to that URI.

Long Lived: The URI with the GRUU property persists for relatively long periods of time, ideally being valid for the duration of existence of the AOR itself. This property cannot be completely guaranteed, but providers are supposed to do their best to make

sure that a GRUU remains viable indefinitely.

A URI can have any combination of these properties. It is the responsibility of the domain which mints the URI to determine what properties are conferred upon that URI. This specification imposes requirements on a domain that mints a URI with the GRUU property.

For convenience, a URI that possesses the GRUU property is also referred to as a GRUU.

4. Use Cases

There are several use cases where the GRUU properties are truly needed in order for a SIP application to operate.

4.1 REFER

Consider a blind transfer application [25]. User A is talking to user B. User A wants to transfer the call to user C. So, user A 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 identifies B's UA)
```

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

4.2 Conferencing

A similar need arises in conferencing [17]. In that framework, a conference is described by a URI that identifies the focus of the conference. The focus is a SIP UA that acts as the signaling hub for the 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 user B. User A puts user B on hold, and calls user C. Now, user A has two separate dialogs for two separate calls -- one to user B, and one to user C. User A would like to conference them. To do this, user A's user agent morphs itself into a focus. It sends a re-INVITE or UPDATE [4] on both dialogs, and provides user B and user C with an updated remote target that now holds the conference URI. The URI in the Contact header field also has a callee capabilities

[11] parameter which indicates that this URI is a conference URI. User A proceeds to mix the media streams received from user B and user C. This is called an ad-hoc conference.

At this point, normal conferencing features can be applied. That means that user B can send another user, user D, the conference URI, perhaps in an email. User D can send an INVITE to that URI, and join the conference. For this to work, the conference URI used by user A in its re-INVITE or UPDATE has to be usable by anyone, and it has to route to the specific UA instance of user A that is acting as the focus. If it doesn't, basic conferencing features will fail. Therefore, this URI has to be a GRUU.

4.3 Presence

In a SIP-based presence [27] system, the Presence Agent (PA) generates notifications about the state of a user. This state is represented with the Presence Information Document Format (PIDF) [24]. In a PIDF document, a user is represented by a series of tuples, each of which describes the services that the user has. Each tuple also has a URI in the <contact> element, which is a SIP URI representing that service. A watcher can make a call to that URI, with the expectation that the call is routed to the service whose presence is represented in the tuple.

In some cases, the service represented by a tuple may exist on only a single user agent associated with a user. In such a case, the URI in the presence document has to route to that specific UA instance. 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 has to be a GRUU.

It is interesting to note that the GRUU may need to be constructed by a presence agent, depending on how the presence document is computed by the server.

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 it. This specification defines two mechanisms for obtaining a GRUU -- through registrations and through administrative operation. Only the former requires protocol operations.

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. The UA includes a "+sip.instance" Contact header field parameter of each contact for which a GRUU is desired. This Contact parameter contains a globally unique ID that identifies the UA instance. 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 UA instance. The GRUU is associated with the UA instance. Should the client change its contact, but indicate that it represents the same instance ID, the server would provide the same GRUU. Furthermore, if the registration for the contact expires, and the UA registers the contact at a later time with the same instance identifier, the server would provide the same GRUU.

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 URI. A UA will place the GRUU into the Contact header field of dialog creating requests and responses it generates. RFC 3261 mandates that the Contact header field have the GRUU property, and this specification provides a reliable way for a UA to obtain one. In other words, clients use the GRUU as a remote target. However, since the remote target used by clients to date has typically not had the GRUU properties, implementations have adapted their behaviors (oftentimes in proprietary ways) to compensate. To facilitate a transition away from these behaviors, it is necessary for a UA receiving the message to know whether the remote target 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 remote target is a GRUU.

A domain can construct a GRUU in any way it chooses. However, it is sometimes desirable to construct GRUUs so that any entity that receives a GRUU can determine the AOR for the subscriber associated with the UA instance. To facilitate that, the GRUU can be constructed such that it is identical to the subscriber's AOR, but includes the "opaque" URI parameter. The "opaque" URI parameter provides a general facility to construct a URI (such as a GRUU or a voicemail inbox for a user) that is related to an AOR, so that any element can extract the AOR from the constructed URI by removing the "opaque" parameter. For example:

```
AOR:   sip:alice@example.com
GRUU:  sip:alice@example.com;opaque="kjh29x97us97d"
```

When a proxy in the domain constructs the GRUU, it would set the

value of the "opaque" URI parameter such that it includes the instance ID. As such, when that proxy receives a request sent to the GRUU, it can extract the AOR and instance ID, both of which are needed to process the request.

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 contact bound to it, the server will add the "grid" parameter into the registered contact, and use the result in the Request URI. As a result, when the UA receives the request, the Request URI will contain the "grid" parameter it placed in the corresponding GRUU.

The "grid" and "opaque" URI parameters play similar roles, but complement each other. The "opaque" parameter is added by the owner of the domain to correlate the GRUU to its instance ID, and to easily recognize that the URI has the GRUU property. The "grid" parameter is added by the UA instance so that, when a request is received by that instance, it can determine the context of the request.

6. Creation of a GRUU

A GRUU is a URI that is created and maintained by a server authoritative for the domain in which the GRUU resides. Independently of whether the GRUU is created as a result of a registration or some other means, a server maintains certain information associated with the GRUU. This information, and its relationship with the GRUU, is modeled in Figure 1.

```
         +-----------+                 +-----------+
         |           |    associated   |           |
         |           |1   with        n|           |
         |    AOR    |<----------------|   GRUU    |
         |           |                 |           |
         |           |                 |           |
         +-----------+                 +-----------+
               ^1               is    ^^     |n
               |             bound  //0..1   |
             is|                to//         |associated
          bound|                //           |with
             to|              //             |
               |            //               |
               |0..n      //                 V1
         +-----------+  //             +-----------+
         |           | / 0..n          |           |
         |           |                 |           |
         |  contact  |---------------->|  Instance |
         |           |1      has   0..1|     ID    |
         |           |                 |           |
         +-----------+                 +-----------+
```

Figure 1

The instance ID plays a key role in this specification. It is an identifier, represented with a URN, that uniquely identifies a SIP user agent amongst all other user agents associated with an AOR. For hardware-based user agents, the instance ID would typically be burned into the device at the factory, similar to the way a unique serial number is encoded into each device. For software-based user agents, each installation represents a unique instance. As such, the identifier could be generated on installation and then stored on disk for persistence.

A GRUU is associated, in a many-to-one fashion, with the combination of an instance ID and an AOR. This combination is referred to as an instance ID/AOR pair. For each GRUU, there is one instance ID/AOR pair, and for each instance ID/AOR pair, there can be one or more GRUU. More than one GRUU might be defined in order to have aliases or URI that are anonymous or have other URI properties. However, this specification doesn't define any way for the client to learn about or use more than a single GRUU for each instance ID/AOR pair. The instance ID/AOR pair serves to uniquely identify a user agent instance servicing a specific AOR. The AOR identifies a resource, such as a user or service within a domain, and the instance ID identifies a specific UA instance servicing requests for that resource.

It is important to understand that GRUU is associated with the instance ID/AOR pair, not just the instance ID. For example, let's say a user registered the contact sip:ua@pc.example.com to the AOR sip:user@example.com, and included a +sip.instance="urn:foo:1" parameter in the Contact header field. If the user also registered the contact sip:ua-112@pc.example.com with the same +sip.instance Contact header field parameter to a second AOR (say sip:boss@example.com), each of those UA instances would have a different GRUU because they belong to different AORs. That is the reason why a single instance ID can be associated with multiple GRUU; there would be one such association for each AOR. The same goes for the association of AOR to GRUU; there would be one such association for each instance ID.

The contacts that are bound to the GRUU are always the ones that have an instance ID associated with that GRUU. If none of the contacts bound to the AOR have the instance ID associated with the GRUU, then there are no contacts bound to the GRUU. If a contact should become registered to the AOR that has an instance ID equal to the one associated with the GRUU, that contact also becomes bound to the GRUU. If that contact should expire, it will no longer be bound to the AOR, and similarly, it will no longer be bound to the GRUU. The URI of the contact is irrelevant in determining whether it is bound to a particular GRUU; only the instance ID and AOR are important.

This specification does not mandate a particular mechanism for construction of the GRUU. Several example approaches are given in Appendix A. 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, the resolution procedures of RFC 3263 [2], once applied, result in an IP address on the public Internet.

- When a request is sent to the GRUU, it routes to a server that can make sure the request is delivered to the UA instance. For GRUUs created through registrations, this means that the GRUU has to route to a proxy server with access to registration data.

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

- For each GRUU, both the SIP and Session Initiation Protocol Secure (SIPS) versions MUST exist. The SIPS URI may not always work, particularly if the proxy cannot establish a secure connection to the client. However, the SIPS URI always exists.

Section 8.4 defines additional behaviors that a proxy must exhibit on receipt of a GRUU.

When a domain constructs a URI with the GRUU properties, it MAY confer other properties upon this URI as a matter of domain policy. A domain can elect to confer properties like identity, anonymity, and service treatment. There is nothing in this specification that can allow the recipient of the GRUU to determine which of these properties (besides the GRUU property itself) have been conferred to the URI.

The service treatment property merits further discussion. Typically, the services a proxy executes upon receipt of a request sent to a GRUU will be a subset of those executed when a request is sent to the AOR. For requests that are outside of a dialog, it is RECOMMENDED to apply screening types of functions, both automated (such as black and white list screening) and interactive (such as interactive voice response (IVR) applications that confer with the user to determine whether to accept a call). In many cases, the new request is related to an existing dialog, and may be an attempt to join it (using the Join header field [30]) or replace it (using the Replaces header field [31]). In such cases, the UA will typically make its own authorization decisions, allowing the reuqest if the sender can prove it knows the dialog identifiers [19]. In such cases, bypassing screening services might make sense, but it needs to be carefully considered by network designers, as it depends on the specific type of screening service.

However, forwarding services, such as call forwarding, SHOULD NOT be provided for requests sent to a GRUU. The intent of the GRUU is to target a specific UA instance, and this is incompatible with forwarding operations.

Mid-dialog requests will also be sent to GRUUs, as they are included as the remote-target in dialog-forming requests and responses. However, in those cases, a proxy SHOULD only apply services that are meaningful for mid-dialog requests, generally speaking. This excludes screening functions, as well as forwarding ones.

The "opaque" URI parameter, defined in Section 9, provides a means for a domain to construct a GRUU such that the AOR associated with the GRUU is readily extractable from the GRUU. Unless the GRUU is meant to also possess the anonymity property, it is RECOMMENDED that GRUUs be constructed using this parameter.

Because the GRUU is associated with both the instance ID and AOR, for any particular AOR there can be a potentially infinite number of GRUUs, and potentially more than one for each instance ID. However, the instance IDs are only known to the network when an instance actually registers with one. As a result, it is RECOMMENDED that a GRUU be created at the time a contact with an instance ID is first

registered to an AOR (even if that registration indicates that the registering UA doesn't even support GRUUs), until the time that the AOR is no longer valid in the domain. In this context, the GRUU exists if the domain, upon receiving a request for that GRUU, recognizes it as a GRUU, can determine the AOR and instance ID associated with it, and translate the GRUU to a contact if there is one with that instance ID currently registered. This property of the GRUU (existing from the time the first registration until removal of the AOR) can be difficult to achieve through software failures and power outages within a network, and for this reason, providing the property is at RECOMMENDED strength, and not MUST.

7. Obtaining a GRUU

A GRUU can be obtained in many ways. This document defines two -- through registrations and through administrative operation.

7.1 Through Registrations

When a GRUU is associated with a user agent that comes and goes, and registers itself to the network to bind a contact to an AOR, a GRUU is provided to the user agent through SIP REGISTER messages.

7.1.1 User Agent Behavior

7.1.1.1 Generating a REGISTER Request

When a UA compliant to this specification 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.

Furthermore, for each contact for which the UA desires to obtain a GRUU, the UA MUST include a "sip.instance" media feature tag as a UA characteristic [11]. The instance ID MUST identify the UA that is performing the registration. As described in [11], this media feature tag will be encoded in the Contact header field as the "+sip.instance" Contact header field parameter. The value of this parameter MUST be a URN [10]. Usage of a URN is a MUST since it provides a persistent and unique name for the UA instance, allowing it to obtain the same GRUU over time. It also provides an easy way to guarantee uniquess within the AOR. However, this specification does not require a long-lived and persistent instance identifier to properly function, and in some cases, there may be cause to use an identifier with weaker temporal persistence.

[11] defines equality rules for callee capabilities parameters, and

according to that specification, the "sip.instance" media feature tag will be compared by case-sensitive string comparison. This means that the URN will be encapsulated by angle brackets "<" and ">" when it is placed within the quoted string value of the +sip.instance Contact header field parameter. The case-sensitive matching rules apply only to the generic usages defined in RFC 3840 [11] and in the caller preferences specification [23]. When the instance ID is used in this specification, it is effectively "extracted" from the value in the "sip.instance" media feature tag. Thus, equality comparisons are performed using the rules for URN equality that are specific to the scheme in the URN. If the element performing the comparisons does not understand the URN scheme, it performs the comparisons using the lexical equality rules defined in RFC 2141 [10]. Lexical equality may result in two URNs being considered unequal when they are actually equal. In this specific usage of URNs, the only element which provides the URN is the SIP UA instance identified by that URN. As a result, the UA instance SHOULD provide lexically equivalent URNs in each registration it generates. This is likely to be normal behavior in any case; clients are not likely to modify the value of the instance ID so that it remains functionally equivalent yet lexigraphically different to previous registrations.

A registering UAC SHOULD use a UUID URN [28] in its "+sip.instance" Contact header field parameter. The UUID URN allows for non-centralized computation of a URN based on time, unique names (such as a MAC address) or a random number generator. However, if a different URN scheme is used, the URN MUST be selected such that the instance can be certain that no other instance registering against the same AOR would choose the same URN value. An example of a URN that would not meet the requirements of this specification is the national bibliographic number [16]. Since there is no clear relationship between a SIP UA instance and a URN in this namespace, there is no way a selection of a value can be performed that guarantees that another UA instance doesn't choose the same value.

If a UA instance is registering against multiple AORs, it is RECOMMENDED that a UA instance provide a different contact URI for each AOR. This is needed for the UA to determine which GRUU to use as the remote target in responses to incoming dialog-forming requests, as discussed in Section 8.1.

Besides the procedures discussed above, the REGISTER request is constructed as it is in the case where this extension was not understood. Specifically, the contact in the REGISTER request SHOULD NOT contain the gruu Contact header field parameter, and the contact URI itself SHOULD NOT contain the "grid" parameter defined below. Any such parameters are ignored by the registrar, as the UA cannot propose a GRUU for usage with the contact.

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.

7.1.1.2 Processing the REGISTER Response

If the response is a 2xx, each Contact header field that contained the "+sip.instance" Contact header field parameter may also contain a "gruu" parameter. This parameter contains a SIP or SIPS URI that represents a GRUU corresponding to the UA instance that registered the contact. The URI will be a SIP URI if the To header field in the REGISTER request contained a SIP URI, else (if the To header field in the REGISTER request contained a SIPS URI) it will be a SIPS URI. Any requests sent to the GRUU URI will be routed by the domain to a contact with that instance ID. Normally, the GRUU will not change in subsequent 2xx responses to REGISTER. Indeed, even if the UA lets the contact expire, when it re-registers it at any later time, the registrar will normally provide the same GRUU for the same address-of-record and instance ID. However, as discussed above, this property cannot be completely guaranteed, as network failures may make it impossible to provide an identifier that persists for all time. As a result, a UA MUST be prepared to receive a different GRUU for the same instance ID/AOR pair in a subsequent registration response.

A non-2xx response to the REGISTER request has no impact on any existing GRUU previously provided to the UA. Specifically, if a previously successful REGISTER request provided the UA with a GRUU, a subsequent failed request does not remove, delete, or otherwise invalidate the GRUU.

7.1.2 Registrar Behavior

A registrar MAY create a GRUU for a particular instance ID/AOR 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.

A registrar MUST create both the SIP and SIPS versions of the GRUU, such that if the GRUU exists, both URI exist.

7.1.2.1 Processing a REGISTER Request

A REGISTER request might contain a Require header field; this indicates that the registration has to understand this extension in order to process the request.

As the registrar is processing the contacts in the REGISTER request according to the procedures of step 7 in Section 10.3 of RFC 3261, the registrar checks whether each Contact header field in the REGISTER message contains a "+sip.instance" header field parameter. If present, the contact is processed further. If the registrar had not yet created a GRUU for that instance ID/AOR pair, it MUST do so at this time according to the procedures of Section 6. If the contact contained a "gruu" Contact header field parameter, it MUST be ignored by the registrar. A UA cannot suggest or otherwise provide a GRUU to the registrar.

Registration processing then continues as defined in RFC 3261. If, after that processing, that contact is bound to the AOR, it also becomes bound to the GRUU associated with that instance ID/AOR pair. If, after that processing, the contact was not bound to the AOR (due, for example, to an expiration of zero), the contact is not bound to the GRUU either. The registrar MUST store the instance ID along with the contact.

When generating the 200 (OK) response to the REGISTER request, the procedures of step 8 of Section 10.3 of RFC 3261 are followed. Furthermore, for each Contact header field value placed in the response, if the registrar has stored an instance ID associated with that contact, that instance ID is returned as a Contact header field parameter. If the REGISTER request contained a Supported header field that included the "gruu" option tag, the server MUST add a "gruu" Contact header field parameter to that Contact header field. The value of the gruu parameter is a quoted string containing the URI that is the GRUU for the associated instance ID/AOR pair. If the To header field in the REGISTER request contains a SIP URI, the SIP version of the GRUU is returned. If the To header field in the REGISTER request contains a SIPS URI, the SIPS version of the GRUU is returned.

If the REGISTER response contains a gruu Contact header field parameter in any of the contacts, the REGISTER response MUST contain a Require header field with the value "gruu". This is because the client needs to extract its GRUU from the REGISTER response, and utilize a GRUU as the remote target of dialog-initiating requests and responses.

Note that handling of a REGISTER request containing a Contact header field with value "*" and an expiration of 0 still retains the meaning defined in RFC 3261 -- all contacts, not just those with a specific instance ID, are deleted. This removes the binding of each contact to the AOR and the binding of each contact to a GRUU.

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 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.

7.1.2.2 Timing Out a Registration

When a registered contact expires, its binding to the AOR is removed as usual. In addition, its binding to the GRUU is removed at the same time.

7.2 Through Administrative Operation

Administrative creation of GRUUs is useful when a UA instance is a network server that is always available, and therefore doesn't register to the network. Examples of such servers are voicemail servers, application servers, and gateways.

There are no protocol operations required to administratively create a GRUU. The proxy serving the domain is configured with the GRUU, and with the contact to which it should be translated. It is not strictly necessary to also configure the instance ID and AOR, since the translation can be done directly. However, they serve as useful tools for determining to which resource and UA instance the GRUU is supposed to map.

In addition to configuring the GRUU and its associated contact in the proxy serving the domain, the GRUU will also need to be configured into the UA instance associated with the GRUU.

It is also reasonable to model certain network servers as logically containing both a proxy and a UA instance. The proxy receives the request from the network, and passes it internally to the UA instance. In such a case, the GRUU routes directly to the server, and there is no need for a translation of the GRUU to a contact. The server itself would construct its own GRUU.

8. Using the GRUU

8.1 Sending a Message Containing a GRUU

A UA first obtains a GRUU using the procedures of Section 7, or by other means outside the scope of this specification.

A UA can use the GRUU in the same way it would use any other SIP or

SIPS URI. However, a UA compliant to this specification MUST use a GRUU when populating the Contact header field of dialog-creating requests and responses. In other words, a UA compliant to this specification MUST use its GRUU as its remote target. This includes the INVITE request, its 2xx response, the SUBSCRIBE [6] request, its 2xx response, the NOTIFY request, and the REFER [7] request, and its 2xx response.

If the UA instance has obtained multiple GRUUs for different AORs as a result of a registration, it MUST use one corresponding to the AOR used to send or receive the request. For sending a request, this means that the GRUU corresponds to the AOR present in the From header field. Furthermore, this means that the credentials used for authentication of the request correspond to the ones associated with that AOR. When receiving a request, the GRUU in the response corresponds to the AOR to which the original request was targeted. However, that AOR will be rewritten by the proxy to correspond to the UA's registered contact. It is for this reason that different contacts are needed for each AOR that an instance registers against. In this way, when an incoming request arrives, the Request URI can be examined. It will be equal to a registered contact. That contact can be used to map directly to the AOR, and from there, the correct GRUU can be selected.

In those requests and responses where a GRUU is used as the remote target, 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 its peer in the dialog supports this specification before the UA uses a GRUU as a remote target.

When using a GRUU as a remote target, a UA MAY add the "grid" URI parameter to the GRUU. This parameter MAY take any value permitted by the grammar for the 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 contact bound to that GRUU. However, the proxy 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 looking at 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 unless the UA is co-resident with the proxy (which is the case for administratively constructed GRUUs). It is

anticipated that this limitation will be addressed in future specifications.

Once a UA knows that the remote target provided by its peer is a GRUU, it can use the GRUU in any application or SIP extension which requires a globally routable URI to operate. One such example is assisted call transfer.

8.2 Sending a Message to a GRUU

There is no new behavior associated with sending a request to a GRUU. A GRUU is a URI like any other. When a UA receives a request or response, it can know that the remote target is a GRUU if the request or response had a Supported header field that included the value "gruu". The UA can take the GRUU, send a request to it, and then be sure that the request is delivered to the UA instance which sent the request or response.

If the GRUU contains the "opaque" URI parameter, a UA can obtain the AOR for the user by stripping the parameter. The resulting URI is the AOR. If the GRUU does not have the "opaque" URI parameter, there is no mechanism defined for determining the AOR from the GRUU. Extraction of the AOR from the GRUU is useful for call logs and other accounting functions where it is desirable to know the user to whom the request was directed.

Because the instance ID is a callee capabilities parameter, a UA might be tempted to send a request to the AOR of a user, and include an Accept-Contact header field [23] that indicates a preference for routing the request to a UA with a specific instance ID. Although this would appear to have the same effect as sending a request to the GRUU, it does not. The caller preferences expressed in the Accept-Contact header field are just preferences. Its efficacy depends on a UA constructing an Accept-Contact header field that interacts with domain-processing logic for an AOR, to cause a request to route to a particular instance. Given the variability in routing logic in a domain (for example, time-based routing to only selected contacts), this doesn't work for many domain-routing policies. However, this specification does not forbid a client from attempting such a request, as there may be cases where the desired operation truly is a preferential routing request.

8.3 Receiving a Request Sent to a GRUU

When a User Agent Server (UAS) receives a request sent to its GRUU, the incoming request URI will be equal to the contact that was registered (through REGISTER or some other action) by that UA instance. If the user agent had previously handed out its GRUU with

a "grid" parameter, the incoming request URI may contain that parameter. This indicates to the UAS that the request is being received as a result of a request sent by the UAC to that GRUU/grid combination. This specification makes no normative statements about when to use a "grid" parameter, or what to do when receiving a request made to a GRUU/grid combination. Generally, any differing behaviors are a matter of local policy.

It is important to note that, when a user agent receives a request, and the request URI does not have a "grid" parameter, the user agent cannot tell by inspection of the Request URI whether the request was sent to the AOR or to the GRUU. The To header field might be different, but, due to forwarding, cannot be depended on as a differentiator. As such, the UAS will process such requests identically. If a user agent needs to differentiate its behavior based on these cases, it will need to use a "grid" parameter.

8.4 Proxy Behavior

Proxy behavior is fully defined in Section 16 of RFC 3261 [1]. GRUU processing impacts that processing in two places -- request targeting and record routing.

8.4.1 Request Targeting

When a proxy server receives a request, owns the domain in the Request URI, and 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 examines the Request URI. If the Request URI is an AOR against which there are multiple registered contacts with the same instance ID parameter, the proxy MUST populate the target set so that there is never more than one contact at a time with a given instance ID. It is RECOMMENDED that the most recently registered contact be used. It is expected that standards track extensions will be developed that provide additional criteria for selecting which contact to use [18].

> NOTE: The contact that is the most recently registered is the one that has been bound to the AOR for the shortest period of time. This corresponds to the minimum value for the "duration-registered" attribute from the registration event package [29]. It is important to note that a refresh of the contact in a REGISTER message does not reset the duration it has been registered to zero. For example, if a softphone is started at 9 am when a user logs into their computer, and the softphone refreshes its registration every hour, by 12:30 pm the contact has been registered for three and a half hours.

If the request URI is within the domain of the proxy, and the URI has been constructed by the domain such that the proxy is able to determine that it has the form of a GRUU for an AOR that is unknown within the domain, the proxy rejects the request with a 404 (Not Found). If the request URI is within the domain of the proxy, and the URI has been constructed by the domain such that the proxy is able to determine that it has the form of a GRUU for an AOR that is known within the domain, but the instance ID is unknown, the proxy SHOULD generate a 480 (Temporarily Unavailable).

If the GRUU does exist, handling of the GRUU proceeds as specified in RFC 3261 Section 16. For GRUUs, the abstract location service described in Section 16.5 is utilized, producing a set of zero or more contacts, each of which is associated with the same instance ID. If there is more than one contact bound to the GRUU, the proxy MUST populate the target set such that there is never more than one contact at a time. It is RECOMMENDED that the most recently registered contact be used. This produces zero or one contacts. The server MUST copy the "grid" parameter from the Request URI (if present) into the new target URI obtained from the registered contact. If the grid was already present in the contact bound to the GRUU, it is overwritten in this process. If no contacts were bound to the GRUU, the lookup of the GRUU in the abstract location service will result in zero target URIs, eventually causing the proxy to reject the request with a 480 (Temorarily Unavailable) response.

If the contact was registered using a Path header field [3], then that Path is used to construct the route set for reaching the contact through the GRUU, as well as through the AOR, using the procedures specified in RFC 3327 [3]. However, support for GRUU at a registrar does not require support for RFC 3327.

A proxy MAY apply other processing to the request, such as execution of called party features, as discussed in Section 6.

A request sent to a GRUU SHOULD NOT be redirected. In many instances, a GRUU is used by a UA in order to assist in the traversal of NATs and firewalls, and a redirection may prevent such a case from working.

8.4.2 Record Routing

As described above, a user agent uses its GRUU as a remote target. This has an impact on the path taken by subsequent mid-dialog requests. Depending on the desires of the proxies involved, this may impact record route processing.

Two cases can be considered. The first is shown in Figure 2. In

this case, there is a single proxy in the user's domain. An incoming INVITE request arrives for the users AOR (message 1) and is forwarded to the user agent at its registered contact C1 (message 2). The proxy inserts a Record-Route header field into the proxied request, with a value of R1. The user agent generates a 200 OK to the request, using its GRUU G1 as the remote target.

```
     (1) + (2): initial INVITE
     (2) + (3): mid-dialog request




    (1)  +-----------+       (2)      +-----------+
  ------>|   Proxy   |--------------->|   User    |
         |           |                |   Agent   |
    (3)  |           |       (4)      |           |
  ------>|R1         |--------------->|G1         |
         |           |                |           |
         +-----------+                +-----------+
```

Figure 2

When a mid-dialog request shows up destined for the user agent (message 3), it will arrive at the proxy in the following form:

```
INVITE G1
Route: R1
```

Since the top Route header field value identifies the proxy, the proxy removes it. As there are no more Route header field values, the proxy processes the request URI. However, the request URI is a GRUU, and is therefore a domain under the control of the proxy. The proxy will need to perform the processing of Section 8.4.1, which will result in the translation of the GRUU into the contact C1, followed by transmission of the request to the user agent (message 4).

This sequence of processing in the proxy is somewhat unusual, in that mid-dialog requests (that is, requests with a Route header field that a proxy inserted as a result of a Record-Route operation) do not normally cause a proxy to invoke a location service to process the request URI. Because this processing is unusual, it is explicitly called out here.

The previous case assumed that there was a single proxy in the

domain. In more complicated cases, there can be two or more proxies within a domain on the initial request path. This is shown in Figure 3. In this figure, there is a home proxy to which requests targeted to the AOR are sent. The home proxy executes the abstract location service and runs user features. The edge proxy acts as the outbound proxy for users, performs authentication, manages TCP/ Transport Layer Security (TLS) connections to the client, and does other functions associated with the transition from the provider proxy network to the client. This specific division of responsibilities between home and edge proxy is just for the purposes of illustration; the discussion applies to a disaggregation of proxy logic into any number of proxies. In such a configuration, registrations from the user agent would pass through the edge proxy, which would insert a Path header field [3] for itself.

```
    (1) + (2) + (3): initial INVITE
    (4) through (9): mid-dialog request




   (1) +-----------+    (2) +-----------+     (3) +-----------+
  ---->|           |------->|           |-------->|           |
   (4) |           |    (5) |           |         |           |
  ---->|H1   Home  |------->|E1   Edge  |         |     User  |
   (7) |     Proxy |    (8) |     Proxy |     (9) |     Agent |
   +-->|G1         |------->|           |-------->|           |
   |   +-----------+        +-----------+         +-----------+
   |                              |
   |                              |
   +------------------------------+
                  (6)
```

Figure 3

When an incoming request arrives for the AOR (message 1), the home proxy would look it up, discover the registered contact and Path, and then send the request to the edge proxy as a result of the Route header field inserted with the Path value. The home proxy record routes with the URI H1. The edge proxy would forward the request to the request URI (which points to the client), and insert a Record-Route header field value with the URI E1 (message 2). This request is accepted by the user agent, which inserts its GRUU G1 as the remote target.

When the peer in the dialog sends a mid-dialog request, it will have

the following form:

```
INVITE G1
Route: H1, E1
```

This request will arrive at the home proxy (due to H1 in the Route header field) (message 4). The home proxy will forward it to the edge proxy (due to E1 in the Route header field) (message 5). The edge proxy, seeing no more Route header field values, sends the request to the Request URI. The Request URI will contain a GRUU, causing the edge proxy to route the request back around to the home proxy (message 6). The home proxy performs the GRUU processing of Section 8.4.1, causing the request to be forwarded to the edge proxy a second time (this time, as a result of a Route header field value obtained from the Path header in the registration) (message 7), and then delivered to the client (message 8).

Although this flow works, it is highly inefficient because it causes each mid-dialog request to spiral route. This behavior is not desirable. To prevent it, the following procedures SHOULD be followed. When a client generates a REGISTER request, this request passes through the edge proxy on its way to the home proxy. The REGISTER request will contain the AOR of the user (in the To header field). If the client is registering an instance, there will be one or more Contact header field values with the +sip.instance Contact header field parameter, and a Supported header field that includes the value "gruu". Normally there would be just one Contact header field value with an instance ID, but it is possible for there to be more than one. However, all Contact header field values are required to identify the instance performing the registration.

The proxy can decide to insert itself on the Path for a particular AOR on a case-by-case basis. However, if it does so for one registration, it SHOULD do so for all registrations for the same AOR that contain an instance ID in the contact. The value of the Path header field inserted by the proxy SHOULD be constructed so that it indicates whether the proxy inserted itself on the Path for this AOR.

When a request arrives from the home proxy towards the client, the proxy inspects the Route header field. This header field will contain the URI the edge proxy had placed into the Path. If the value indicates that the edge proxy had put itself on the Path for the registration from this client, there is no need for the proxy to retain its record-route in the response. The proxy MAY remove its record-route value from the 200 OK response in this case. If the value indicates that the proxy had not put itself on the Path, it would retain the Record-Route in the response.

Similarly, if a request arrives from the client towards the home proxy, the edge proxy would look at the identity of the sender of the request. If the proxy knows that it is placing itself on the Path for registrations from that AOR, and the request contains a Supported header field with the value "gruu", the edge proxy would insert a Record-Route into the request, and then remove it in the response. Similarly, if the identity of the sender of the request is one for which the client has not put itself on the Path, the edge proxy would keep its Record-Route in the response.

Removing its Record-Route value from the response will result in the route set being seen differently by the caller and callee; the callee (which is the user agent in the figure) will have a route set entry for its edge proxy, while the caller will not. The caller will have a route set entry for its edge proxy, while the callee will not.

In such a case, a mid-dialog request that arrives at the home proxy will be of the form:

```
INVITE G1
Route: H1
```

This does the "right thing" and causes the request to be routed from the home proxy to the edge proxy to the client, without the additional spiral.

9. The opaque SIP URI Parameter

This specification defines a new SIP URI parameter, "opaque". The "opaque" URI parameter is used to construct a URI (called the derived URI) that is related to another URI (called the base URI, frequently an AOR) in some way. In this specification, the parameter is used to construct the GRUU (the derived URI) from the AOR (the base URI). However, there are many other applications outside of GRUU. It can be used, for example, to construct a URI for a voicemail inbox (the derived URI) from a subscriber's AOR (the base URI), or the URI for a video service advertised via presence [26] (the derived URI) from the subscriber's AOR (the base URI).

To construct a derived URI, the owner of the domain adds the "opaque" URI parameter to the base URI, resulting in the derived URI. In fact, these are the only semantics associated with the "opaque" URI parameter: a URI containing the parameter MUST be related to another URI, obtained by stripping the "opaque" URI parameter. Because the "opaque" URI parameter implies a relationship, any element (including those outside the domain that owns the URI) that receives a URI with the "opaque" URI parameter will know definitively that it is a

derived URI, and can strip it to obtain the base URI.

The value of the "opaque" URI parameter is not relevant to anyone except for the owner of the domain. It typically contains information needed by the owner of the domain to correctly process a request targeted to that URI according to the desired semantics of the URI. As such, the parameter is a form of cookie. In the case of a GRUU, the "opaque" URI parameter contains enough information for the owner of the domain to determine the instance ID. Since the structure of its value is not subject to standardization, it can only be interpreted by the same proxy or cluster of proxies that created the derived URI. For this reason, a proxy or cluster of proxies MUST NOT create a derived URI unless a request sent to the base URI (and consequently the derived URI) will be routed back to that same proxy or cluster of proxies without any upstream proxies requiring interpretation of the "opaque" URI parameter. Simply put, a request sent to a derived URI has to get back to the same proxy farm that created the derived URI.

The presence of the "opaque" URI parameter in a URI implies a relationship between that URI and its base URI. However, the nature of that relationship cannot be determined from inspection of the URI alone. In some cases, there may be no way to know the relationship outside of the domain that constructed the URI. In other cases, the nature of the relationship is determined from the context in which the URI was obtained. When used to construct a GRUU, it means that the URI formed by stripping the "opaque" parameter corresponds to the AOR associated with the GRUU. The recipient of a GRUU cannot determine that it is a GRUU by direct examination of the URI. However, the recipient may know if it received the GRUU in the Contact header field of a SIP request or in a response that contained a Supported header field with the option tag "gruu". If it knows its a GRUU because it received it in such a SIP message, and the GRUU contains the "opaque" parameter, the UA knows it can obtain the AOR by removing the "opaque" parameter.

10. Grammar

This specification defines two new Contact header field parameters, gruu and +sip.instance, and two new URI parameters, "grid" and "opaque". The grammar for string-value is obtained from [11], and the grammar for uric is defined in RFC 3986 [9].

```
   contact-params  =  c-p-q / c-p-expires / c-p-gruu / cp-instance
                       / contact-extension
   c-p-gruu        =  "gruu" EQUAL DQUOTE (SIP-URI / SIPS-URI) DQUOTE
   cp-instance     =  "+sip.instance" EQUAL LDQUOT "<"
                      instance-val ">" RDQUOT
   uri-parameter   =  transport-param / user-param / method-param
                      / ttl-param / maddr-param / lr-param / grid-param
                      / opaque-param / other-param
   grid-param      = "grid=" pvalue        ; defined in RFC3261
   opaque-param    = "opaque=" pvalue      ; defined in RFC3261
   instance-val    = *uric ; defined in RFC 2396
```

11. 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 that each route to that UA instance. For example, this is needed to support ad-hoc conferencing where 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 that was used to invoke the request. This is necessary as a consequence of REQ 4.

REQ 6: It MUST be possible for a UA to add opaque content to a GRUU. This content is not interpreted or altered by the network, and is 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 the state embedded.

REQ 7: It MUST be possible for a proxy to execute services and features on behalf of a UA instance 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 REQ 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 that is authoritative for a domain to construct a GRUU which routes to a UA instance bound to an AOR in that domain. In other words, the proxy can construct a GRUU, too. This is needed for the presence application.

12. Example Call Flow

The following call flow shows a basic registration and call setup, followed by a subscription directed to the GRUU. It then shows a failure of the callee, followed by a re-registration. The conventions of [22] are used to describe representation of long message lines.

```
          Caller                 Proxy                Callee
             |                     |(1) REGISTER         |
             |                     |<--------------------|
             |                     |(2) 200 OK           |
             |                     |-------------------->|
             |(3) INVITE           |                     |
             |-------------------->|                     |
             |                     |(4) INVITE           |
             |                     |-------------------->|
             |                     |(5) 200 OK           |
             |                     |<--------------------|
             |(6) 200 OK           |                     |
             |<--------------------|                     |
             |(7) ACK              |                     |
             |-------------------->|                     |
             |                     |(8) ACK              |
             |                     |-------------------->|
             |(9) SUBSCRIBE        |                     |
             |-------------------->|                     |
             |                     |(10) SUBSCRIBE       |
             |                     |-------------------->|
             |                     |(11) 200 OK          |
             |                     |<--------------------|
             |(12) 200 OK          |                     |
             |<--------------------|                     |
             |                     |(13) NOTIFY          |
             |                     |<--------------------|
             |(14) NOTIFY          |                     |
             |<--------------------|                     |
             |(15) 200 OK          |                     |
             |-------------------->|                     |
             |                     |(16) 200 OK          |
             |                     |-------------------->|
             |                     |                     |Crashes,
             |                     |(17) REGISTER        | Reboots
             |                     |<--------------------|
             |                     |(18) 200 OK          |
             |                     |-------------------->|
```

Figure 4

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 192.0.2.1;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=a73kszlfl
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: 1j9FpLxk3uxtm8tn@192.0.2.1
CSeq: 1 REGISTER
Contact: <sip:callee@192.0.2.1>
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
Content-Length: 0
```

The REGISTER response would look like:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.1;branch=z9hG4bKnashds7
From: Callee <sip:callee@example.com>;tag=a73kszlfl
To: Callee <sip:callee@example.com> ;tag=b88sn
Require: gruu
Call-ID: 1j9FpLxk3uxtm8tn@192.0.2.1
CSeq: 1 REGISTER
<allOneLine>
Contact: <sip:callee@192.0.2.1>
 ;gruu="sip:callee@example.com;
opaque=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
;expires=3600
</allOneLine>
Content-Length: 0
```

Note how the Contact header field in the REGISTER response contains the gruu parameter with the URI sip:callee@ example.com;opaque=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6. This represents a GRUU that translates to the contact sip:callee@192.0.2.1.

The INVITE from the caller is a normal SIP INVITE. However, the 200 OK generated by the callee (message 5) now contains a GRUU as the remote target. 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
<allOneLine>
Contact:
<sip:callee@example.com
;opaque=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6;grid=99a>
</allOneLine>
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 [21] at that specific UA. To do that, it generates a SUBSCRIBE request (message 9), but directs it towards the remote target, which is a GRUU:

```
<allOneLine>
SUBSCRIBE sip:callee@example.com;opaque=urn:uuid:f8
1d4fae-7dec-11d0-a765-00a0c91e6bf6;grid=99a
 SIP/2.0
</allOneLine>
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:caller@example.com>;tag=kkaz-
<allOneLine>
To: sip:callee@example.com;opaque=urn:uuid:f8
1d4fae-7dec-11d0-a765-00a0c91e6bf6;grid=99a
</allOneLine>
Call-ID: faif9a@host.example.com
CSeq: 2 SUBSCRIBE
Supported: gruu
Event: dialog
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:caller@example.com;opaque=hdg7777ad7aflzig8sf7>
Content-Length: 0
```

In this example, the caller itself supports the GRUU extension, and is using its own GRUU to populate its remote target.

This request is routed to the proxy, which proceeds to perform a

location lookup on the request URI. It is translated into the contact for that instance, and then proxied to that contact. Note how the "grid" parameter is maintained.

```
SUBSCRIBE sip:callee@192.0.2.1;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-
<allOneLine>
To: sip:callee@example.com;opaque=urn:uuid:f8
1d4fae-7dec-11d0-a765-00a0c91e6bf6;grid=99a
</allOneLine>
Call-ID: faif9a@host.example.com
CSeq: 2 SUBSCRIBE
Supported: gruu
Event: dialog
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:caller@example.com;opaque=hdg7777ad7aflzig8sf7>
Content-Length: 0
```

At some point after message 16 is received, the callee's machine crashes and recovers. It obtains a new IP address, 192.0.2.2. Unaware that it had previously had an active registration, it creates a new one (message 17 below). Notice how the instance ID remains the same, as it persists across reboot cycles:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbba
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=ha8d777f0
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: hf8asxzff8s7f@192.0.2.2
CSeq: 1 REGISTER
Contact: <sip:callee@192.0.2.2>
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
Content-Length: 0
```

The registrar notices that a different contact, sip:callee@192.0.2.1, is already associated with the same instance ID. It registers the new one too and returns both in the REGISTER response. Both have the same GRUU. However, only this new contact (the most recently registered one) will be used by the proxy for population in the target set. The registrar then generates the following response:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbba
From: Callee <sip:callee@example.com>;tag=ha8d777f0
To: Callee <sip:callee@example.com>;tag=99f8f7
Require: gruu
Call-ID: hf8asxzff8s7f@192.0.2.2
CSeq: 1 REGISTER
<allOneLine>
Contact: <sip:callee@192.0.2.2>
;gruu="sip:callee@example.com;opaque=urn:
uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
;expires=3600
</allOneLine>
Contact: <sip:callee@192.0.2.1>
;gruu="sip:callee@example.com;opaque=urn:
uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
;expires=400
</allOneLine>
Content-Length: 0
```

13. Security Considerations

It is important for a UA to be assured of the integrity of a GRUU given 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 in the Request-URI when registering. Proxies and registrars MUST support the sips URI and MUST support TLS. Note that this does not represent a change from the requirements in RFC 3261.

The example GRUU construction algorithm in Appendix A.1 makes no attempt to create a GRUU that hides the AOR and instance ID associated with the GRUU. In general, determination of the AOR associated with a GRUU is considered a good property, since it allows for easy tracking of the target of a particular call. Learning the instance ID provides little benefit to an attacker. To register or otherwise impact registrations for the user, an attacker would need to obtain the credentials for the user. Knowing the instance ID is insufficient.

The example GRUU construction algorithm in Appendix A.1 makes no attempt to create a GRUU that prevents users from guessing a GRUU based on knowledge of the AOR and instance ID. A user that is able to do that will be able to direct a new request at a particular instance. However, this specification recommends that service

treatment (in particular, screening features) be given to requests that are sent to a GRUU. That treatment will make sure that the GRUU does not provide a back door for attackers to contact a user that has tried to block the attacker.

GRUUs do not provide a 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 [14].

14. IANA Considerations

This specification defines a new Contact header field parameter, two SIP URI parameters, a media feature tag, and a SIP option tag.

14.1 Header Field Parameter

This specification defines a new header field parameter, as per the registry created by [12]. 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.]]

14.2 URI Parameters

This specification defines two new SIP URI parameters, as per the registry created by [13].

Name of the Parameter: grid

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

Name of the Parameter: opaque

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

14.3 Media Feature Tag

This section registers a new media feature tag, per the procedures defined in RFC 2506 [8]. The tag is placed into the sip tree, which is defined in [11].

Media feature tag name: sip.instance

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: This feature tag contains a string containing a URN that indicates a unique identifier associated with the UA instance registering the Contact.

Values appropriate for use with this feature tag: String.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application, for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Routing a call to a specific device.

Related standards or documents: RFC XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

Security Considerations: This media feature tag can be used in ways which affect application behaviors. For example, the SIP caller preferences extension [23] allows for call routing decisions to be based on the values of these parameters. Therefore, if an attacker can modify the values of this tag, they may be able to affect the behavior of applications. As a result, applications which utilize this media feature tag SHOULD provide a means for ensuring its integrity. Similarly, this feature tag should only be trusted as valid when it comes from the user or user agent described by the tag. As a result, protocols for conveying this feature tag SHOULD provide a mechanism for guaranteeing authenticity.

14.4 SIP Option Tag

This specification registers a new SIP option tag, as per the guidelines in Section 27.1 of RFC 3261.

Name: gruu

Description: This option tag is used to identify the Globally Routable User Agent URI (GRUU) extension. When used in a Supported header, it indicates that a User Agent understands the extension, and has included a GRUU in the Contact header field of its dialog-initiating requests and responses. When used in a Require header field of a REGISTER request, it indicates that the registrar should assign a GRUU to the Contact URI.

15. Acknowledgements

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

16. References

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

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

[4] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

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

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

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

[8] Holtman, K., Mutz, A., and T. Hardie, "Media Feature Tag Registration Procedure", BCP 31, RFC 2506, March 1999.

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

January 2005.

[10] Moats, R., "URN Syntax", RFC 2141, May 1997.

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

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

[13] 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.

16.2 Informative References

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

[15] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[16] Hakala, J., "Using National Bibliography Numbers as Uniform Resource Names", RFC 3188, October 2001.

[17] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-05 (work in progress), May 2005.

[18] Jennings, C. and R. Mahy, "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)", draft-ietf-sip-outbound-00 (work in progress), July 2005.

[19] Rosenberg, J., "Request Authorization through Dialog Identification in the Session Initiation Protocol (SIP)", draft-ietf-sip-target-dialog-01 (work in progress), July 2005.

[20] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-05 (work in progress), May 2005.

[21] Rosenberg, J., "An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP)",

draft-ietf-sipping-dialog-package-06 (work in progress), April 2005.

[22] Sparks, R., "Session Initiation Protocol Torture Test Messages", draft-ietf-sipping-torture-tests-07 (work in progress), May 2005.

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

[24] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

[25] Sparks, R., "Session Initiation Protocol Call Control - Transfer", draft-ietf-sipping-cc-transfer-05 (work in progress), July 2005.

[26] Rosenberg, J., "A Data Model for Presence", draft-ietf-simple-presence-data-model-04 (work in progress), August 2005.

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

[28] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

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

[30] Mahy, R. and D. Petrie, "The Session Initiation Protocol (SIP) "Join" Header", RFC 3911, October 2004.

[31] Mahy, R., Biggs, B., and R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header", RFC 3891, September 2004.

Author's Address

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

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

Appendix A. Example GRUU Construction Algorithms

The mechanism for constructing a GRUU is not subject to specification. This appendix provides two examples that can be used by a registar. Of course, others are permitted, as long as they meet the constraints defined for a GRUU.

A.1 Instance ID in "opaque" URI Parameter

The most basic approach for constructing a GRUU is to utilize the "opaque" URI parameter. The user and domain portions of the URI are equal to the AOR, and the "opaque" parameter is populated with the instance ID.

A.2 Encrypted Instance ID and AOR

In many cases, it will be desirable to construct the GRUU in such a way that it will not be possible, based on inspection of the URI, to determine the Contact URI that the GRUU translates to. It may also be desirable to construct it so that it will not be possible to determine the instance ID/AOR pair associated with the GRUU. Whether a GRUU should be constructed with this property is a local policy decision.

With these rules, it is possible 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 | " " | AOR | " " |
instance ID)))
```

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 signifies concatenation. The single space (" ") between components is used as a delimiter, so that the components can easily be extracted after decryption. 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.

This mechanism uses the user-part of the SIP URI to convey the encrypted AOR and instance ID. The user-part is used instead of the "opaque" URI parameter because it has the desired anonymity properties.

The benefit of this mechanism is that a server need not store additional information on mapping a GRUU to its corresponding contact. The user-part of the GRUU contains the instance ID and AOR. Assuming that the domain stores registrations in a database indexed by the AOR, the proxy processing the GRUU would look up the AOR, extract the currently registered contacts, and find the one that matches the instance ID encoded in the request URI. The contact whose instance ID is that instance ID is then used as the translated version of the GRUU. Encryption is needed to prevent attacks whereby the server is sent requests with fake GRUUs, 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 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.

While this approach has many benefits, it has the drawback of producing fairly long GRUUs.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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