

Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in
the Session Initiation Protocol (SIP)
draft-ietf-sip-gruu-02

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with [RFC 3668](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on December 31, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

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	3
2.	Terminology	3
3.	Defining a GRUU	3
4.	Use Cases	3
4.1	REFER	3
4.2	Conferencing	4
4.3	Presence	4
5.	Overview of Operation	5
6.	Creation of a GRUU	6
7.	Obtaining a GRUU	9
7.1	Through Registrations	9
7.1.1	User Agent Behavior	9
7.1.2	Registrar Behavior	11
7.2	Administratively	12
8.	Using the GRUU	13
8.1	Sending a Message Containing a GRUU	13
8.2	Sending a Message to a GRUU	14
8.3	Receiving a Request Sent to a GRUU	14
8.4	Proxy Behavior	15
9.	425 (Instance Conflict) Response Code	15
10.	Grammar	16
11.	Requirements	16
12.	Example Call Flow	17
13.	Security Considerations	23
14.	IANA Considerations	23
14.1	Header Field Parameter	23
14.2	Response Code	23
14.3	URI Parameter	24
14.4	Media Feature Tag	24
14.5	SIP Option Tag	25
15.	Acknowledgements	25
16.	References	25
16.1	Normative References	25
16.2	Informative References	26
	Author's Address	27
A.	Example GRUU Construction Algorithms	27
A.1	Encrypted Instance ID and AOR	27
A.2	Hashed Indices	28
	Intellectual Property and Copyright Statements	29

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 [[18](#)], based on the REFER method [[5](#)]. Another application is the usage of endpoint-hosted conferences within the conferencing framework [[14](#)]. 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](#) [[3](#)] and indicate requirement levels for compliant implementations.

3. Defining a GRUU

A GRUU is a SIP URI which has two 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 reach that user. The same is true for a GRUU.

Routes to a Single Instance: It routes to a specific UA instance, and never forks. In that regard, it is unlike an address-of-record. When a request is sent to a normal AOR which represents a user, 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 GRUU, the routing logic is dictated by the properties of a GRUU. 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 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 [[18](#)]. 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 which user B is using to talk to user A. If it didn't, the transfer service would 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 is a GRUU.

[4.2](#) Conferencing

A similar need arises in conferencing [[14](#)]. In that framework, a conference is described by a URI which 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 [[2](#)] on both dialogs, and provides user B and user C with an updated Contact URI that now holds the conference URI. The Contact URI also has a callee capabilities [[9](#)] 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 didn't, basic conferencing features would fail. Therefore, this URI is a GRUU.

[4.3](#) Presence

In a SIP-based presence [[19](#)] system, the Presence Agent (PA) generates notifications about the state of a user. This state is represented with the Presence Information Document Format (PIDF)

[17]. 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 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 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 is 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 them. Similarly, a UA can obtain a GRUU but never use it. This specification defines two mechanisms for obtaining a GRUU - through registrations, and through administrative operation. Only the former requires protocol operations.

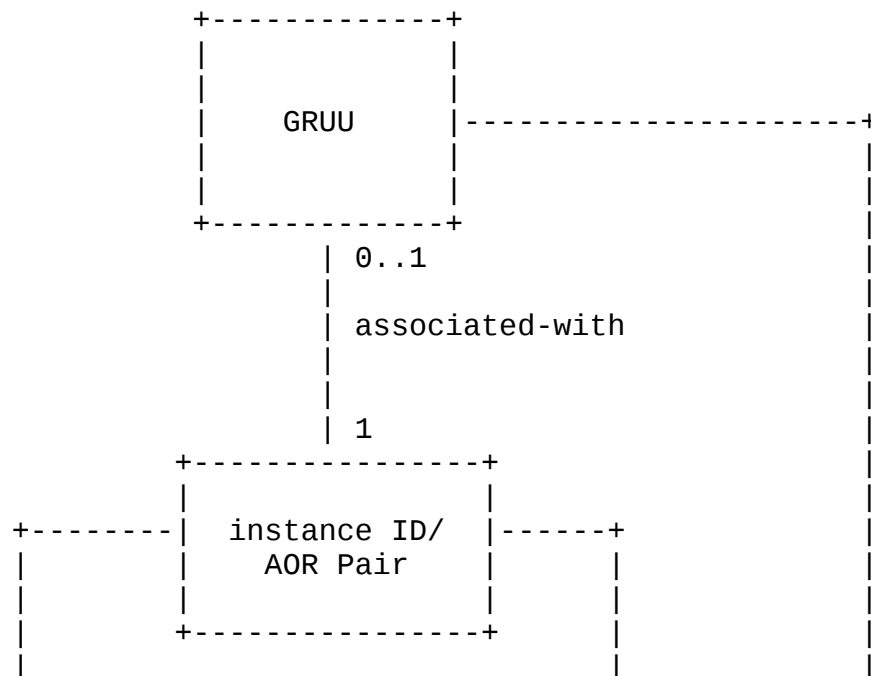
A UA can obtain a GRUU by generating a normal REGISTER request, as specified in [RFC 3261](#) [1]. This request contains a Supported header field with the value "gruu", indicating to the registrar that the UA supports this extension. The UA includes a "sip.instance" media feature tag in the Contact header field of each Contact for which a GRUU is desired. This media feature tag 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. That GRUU is guaranteed to remain valid for the duration of the registration. The GRUU is bound to the UA instance. Should the client change its Contact URI, 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 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. 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 MUST maintain certain information associated with the GRUU. This information, and its relationship with the GRUU, are modeled in Figure 2.



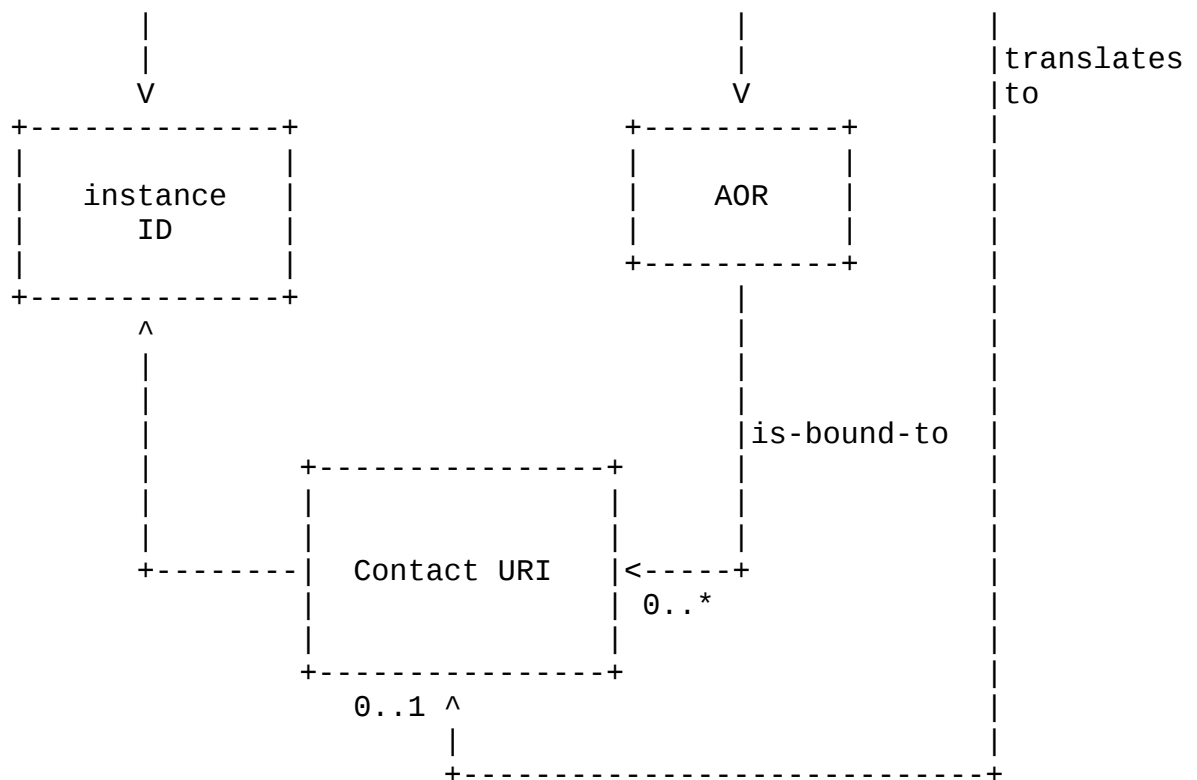


Figure 2

The instance ID plays a key role in this specification. It is an identifier, represented by a URI, that uniquely identifies a SIP user agent amongst all other user agents with a Contact URI bound to an Address of Record (AOR). The instance ID allows a domain to create a GRUU that maps to the same UA instance, even if the Contact URI of that instance changes. Furthermore, the instance ID allows a domain to enforce the restriction that a specific UA instance can only be registered once against an AOR. When elements compliant to this specification compare two instance IDs for equality, the comparison is done using the equality rules for the scheme associated with that URI.

A GRUU is associated, in a one-to-one fashion, with the combination of an Address of Record (AOR) and instance ID. The GRUU is said to be associated with the combination, and the combination is associated with the GRUU. This combination is referred to as an instance ID/AOR pair. The instance ID/AOR pair serve 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 this uniqueness is over the instance ID/AOR pair, not just the instance ID. For example, if a user registered the Contact `sip:ua@pc.example.com;+sip.instance="urn:foo:1"`, representing a device with instance ID `urn:foo:1`, to the AOR `sip:user@example.com`, and also registered the same Contact, representing the same instance ID - `sip:ua@pc.example.com;+sip.instance="urn:foo:1"` to a second AOR, say `sip:boss@example.com`, each of those UA instances would have a different GRUU, since they belong to different AORs.

A GRUU translates to zero or one Contact URIs. The Contact URI is a temporary URI that can be used to reach the instance ID/AOR pair. This URI can change due to changes in the IP address associated with the instance ID/AOR pair. If the instance ID associated with the GRUU is the instance ID of a Contact URI currently bound to the AOR associated with that GRUU, then the GRUU translates to that Contact URI. If, however, the instance ID associated with the GRUU is not an instance ID of a Contact URI currently bound to the AOR associated with the GRUU (possibly because there are no Contact URIs bound to the AOR), the GRUU maps to no Contact URI, and the GRUU is said to be invalid.

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:

- o 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.
- o When a request is sent to this URI, it routes to a proxy server in the same domain as that of the registrar.
- o A proxy server in the domain can determine that the URI is a GRUU.
- o When a proxy server in this domain receives a request sent to a URI that is a GRUU, that URI MUST be translated to the Contact URI currently bound to the AOR associated with that GRUU whose instance ID is the one associated with the GRUU.

Once an association from an instance ID/AOR to a GRUU is created, that mapping MUST remain in existence, and valid, as long as there exists any Contact bound to that AOR whose instance ID is that instance ID. If, through a de-registration or expiration, there is no longer any Contact bound to that AOR whose instance ID is that instance ID, the registrar MUST remove the mapping, and invalidate the GRUU. However, at any time in the future, should a Contact become bound to that same AOR, and that Contact is associated with the same instance ID, the domain SHOULD create the same GRUU that was previously associated with that instance ID/AOR pair. Indeed, this requirement would ideally be a MUST if it was achievable, but even with the stateless algorithm described above, key rotation or server

failures may cause the GRUU associated with an instance ID/AOR pair to change. The value of associating the GRUU with an instance ID/AOR pair, as opposed to a Contact URI/AOR pair, is that the association can transcend changes in IP address. As a result, domains SHOULD make every effort possible to maintain the association for as long as possible.

[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 therefore registers to the network to bind itself to an AOR, a GRUU is provided to the user agent through SIP REGISTER messages.

[7.1.1](#) User Agent Behavior

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 [\[9\]](#). As described in [\[9\]](#), 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 URI [\[7\]](#). [\[9\]](#) 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. Those equality rules apply only to the generic usages defined there and in the caller preferences specification [\[16\]](#). When the instance ID is used in this specification, it is effectively "extracted" from the value in the "sip.instance" media feature tag, and thus equality comparisons are performed using the rules for URI equality specific to the scheme in the URI.

It is RECOMMENDED that the URI be a Uniform Resource Name (URN) [\[8\]](#). This specification makes no normative recommendation on the specific URI or URN that is to be used. However, the URI MUST be selected such that the instance can be certain that no other instance registering against the same AOR would choose the same URI value. Usage of a URN is RECOMMENDED 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 uniqueness 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.

One URN that readily meets the requirements of this specification is the UUID URN [20], which allows for non-centralized computation of a URN based on time, unique names (such as a MAC address) or a random number generator. An example of a URN that would not meet the requirements of this specification is the national bibliographic number [13]. Since there is no clear relationship between an 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.

Besides the presence of the "gruu" option tag in the Supported header field and the "+sip.instance" Contact header field parameter, 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 that contained the "+sip.instance" Contact header field parameter may also contain a "gruu" parameter. This parameter contains a SIP URI that represents a GRUU corresponding to the UA instance that registered the contact. Any requests sent to the GRUU URI will be routed by the domain to the Contact URI currently bound to that instance ID. The GRUU will not normally 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, 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 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.

If the response to the REGISTER request was a 425, it means that one of the Contact URI in the REGISTER request contained an instance ID that was already associated with a different registered Contact. It is up to the client to resolve this conflict. The conflict normally arises when a client registers a Contact with its instance ID, crashes, and reboots. After reboot, it obtains a new IP address, and attempts to register a Contact for that address, containing the same instance ID. In such a case, the proper course of action is to remove the old registration. To do that, the client can send a REGISTER request with no Contacts. The 200 OK contains the list of currently registered Contacts, including their instance IDs. The client can find the existing contact that matches its instance ID, and then send a new REGISTER request. This request would include the old Contact, with the instance ID, and an expires value of 0. Then, the client can retry its failed registration.

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.

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 remainder of this section (that is, the procedures which result in the creation of new GRUUs for Contacts indicating an instance ID, and the listing of GRUUs in the REGISTER response). If not present, but a Supported header field was present with the "gruu" option tag, the registrar SHOULD follow the procedures in the remainder of this section. 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 in the remainder of this section.

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 additionally checks whether each contact contains a "+sip.instance" header field parameter. If it does, the registrar takes the value of that parameter as an instance ID. The registrar checks to see if there is any other contact bound to the same AOR with the same instance ID (recall that equality is computed using URI equality for the scheme in question). If there is, this is an error condition. Only a single Contact URI at a time can be registered for

each instance ID. As a result, the registrar MUST reject the request with a 425 (Instance Conflict) error response. This response code informs the client that its registration failed because the instance ID provided in the request is already registered to a different Contact. It is up to the client to decide how to proceed.

If there is no other contact bound to the same AOR with the same instance ID, the server allocates and/or creates a GRUU for that instance ID/AOR pair 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. In addition to storing the contact URI, the server MUST store the instance ID.

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 URI, the server MUST add a "gruu" Contact header field parameter. This parameter contains the instance ID for the user agent. The value of the gruu parameter is a quoted string containing the URI that is the GRUU for the associated instance ID/AOR pair.

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 ones with a specific instance ID, are deleted.

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.

[7.2](#) Administratively

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 URI it should be translated to. It is not strictly necessary to also configure the instance ID and AOR, since

the translation can be done directly. However, they serve as a useful tool for determining which resource and UA instance the GRUU is supposed to map to.

In addition to configuring the GRUU and its associated Contact URI in the proxy serving the domain, the GRUU will also need to be configured into the UA instance associated with the 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 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 [\[4\]](#) request, its 2xx response, the NOTIFY request, and the REFER [\[5\]](#) 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.

[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 will know that the Contact header field contained a GRUU if the request or response had a Supported header field that included the value "gruu". The UA can take the GRUU, and send a request to it, and then be sure that it is delivered to the UA instance which sent the request or response.

Since 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 [16] which 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, and do not work with the same reliability as GRUU. 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 UAS receives a request sent to its GRUU, the incoming request URI will be equal to the Contact URI 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 whether the request was sent to the AOR or to the GRUU. 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

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 there is still a Contact URI bound to AOR associated with the GRUU, whose instance ID is the instance ID associated with the GRUU. If that AOR no longer has any contacts bound to it, or if it does have contacts bound to it, but none of them have an instance ID equal to the instance ID associated with the GRUU, the proxy MUST generate a 480 (Temporarily Unavailable) response to the request. If, however, the proxy does not recognize the GRUU as one it had constructed previously for the domain, 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 that is translated from the 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 all GRUUs associated with that AOR.

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 firewalls, and a redirection may prevent such a case from working.

9. 425 (Instance Conflict) Response Code

This specification defines a new response code for SIP. The response

code is 425, and it has a default reason phrase of "Instance Conflict". This response code is valid only for REGISTER responses. It informs the UA that its registration failed because the instance ID provided in the request is already registered to a different Contact.

10. Grammar

This specification defines two new Contact header field parameters, gruu and +sip.instance, and a new URI parameter, grid. The grammar for string-value is obtained from [\[9\]](#), and the grammar for uric is defined in [RFC 2396](#) [\[7\]](#).

```
contact-params      = c-p-q / c-p-expires / c-p-gruu / cp-instance
                      / contact-extension
c-p-gruu            = "gruu" EQUAL DQUOTE SIP-URI DQUOTE
cp-instance         = "+sip.instance" EQUAL LDQUOTE instance-val RDQUOTE
uri-parameter       = transport-param / user-param / method-param
                      / ttl-param / maddr-param / lr-param / grid-param
                      / other-param
grid-param          = "grid=" 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, each one of which routes to that UA instance. This is needed to support ad-hoc conferencing, for example, 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 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 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.

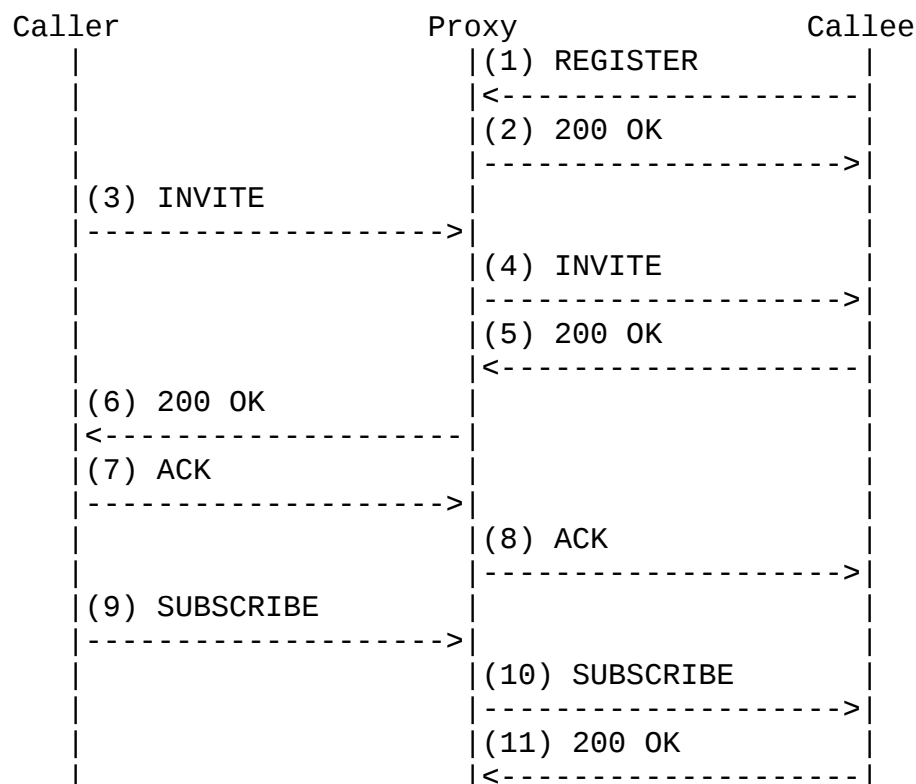
REQ 8: It MUST be possible for a UA in a dialog to inform its peer of its GRUU, and for the peer to know that the URI represents a GRUU. This is needed for the conferencing and dialog reuse applications of GRUUs, where the URIs are transferred within a dialog.

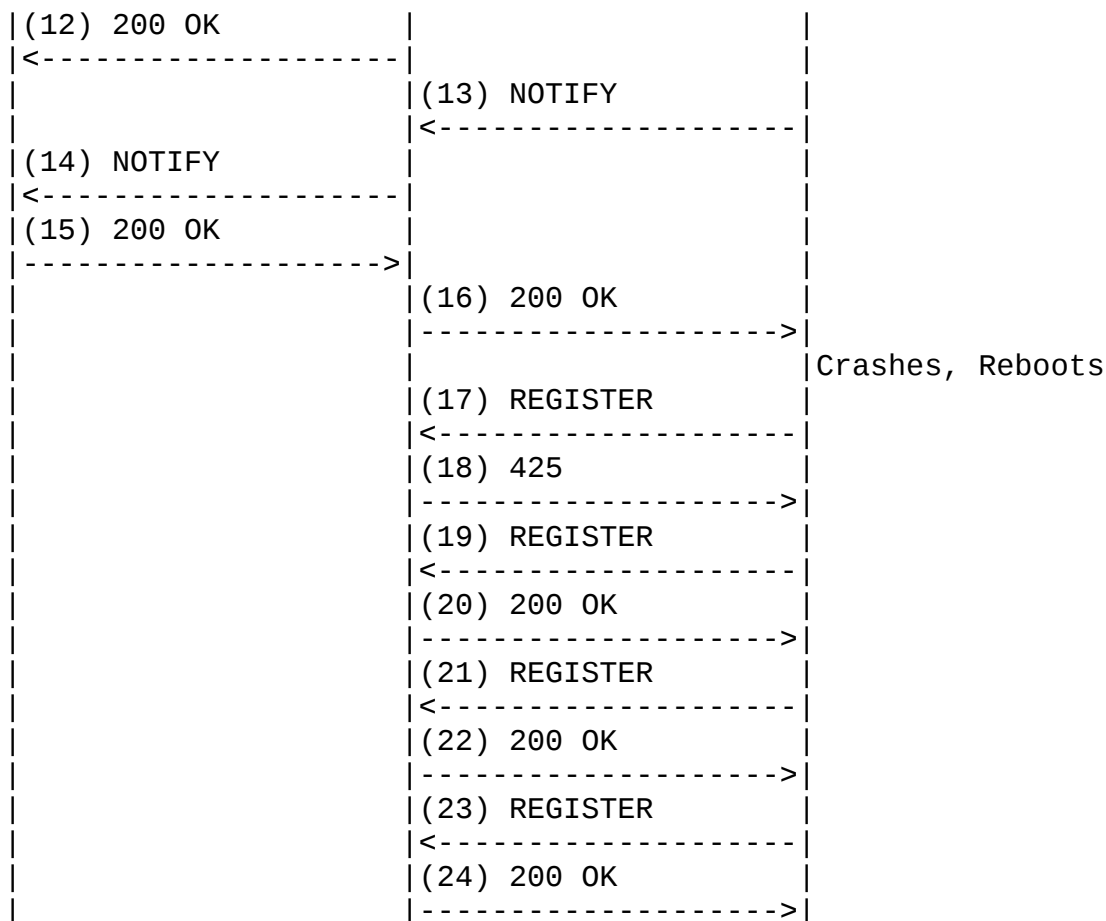
REQ 9: When transferring a GRUU per requirement 8, it MUST be possible for the UA receiving the GRUU to be assured of its integrity and authenticity.

REQ 10: It MUST be possible for a server, authoritative for a domain, to construct a GRUU which routes to a UA 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 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
Call-ID: 1j9FpLxk3uxtm8tn@192.0.2.1
CSeq: 1 REGISTER
Contact: <sip:callee@192.0.2.1>
        ;gruu="sip:hha9s8d=-999a@example.com"
        ;+sip.instance="urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
        ;expires=3600
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 that translates to the Contact URI sip:callee@192.0.2.1.

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 [\[15\]](#) 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 of that GRUU, and then proxied there (message 10 below). 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-
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
```

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. Thus, it rejects the request in message 18, below:

```
SIP/2.0 425 Instance Conflict
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=776554
Call-ID: hf8asxzff8s7f@192.0.2.2
CSeq: 1 REGISTER
```

Next, the client formulates a new REGISTER request, to query for the existing set of registrations (message 19, below):

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbb
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=ha8d777f1
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 2 REGISTER
```

This generates a 200 (OK) response (message 20, below) that includes the existing contact:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbb
From: Callee <sip:callee@example.com>;tag=ha8d777f1
To: Callee <sip:callee@example.com>;tag=8asd7d666
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 2 REGISTER
Contact: <sip:callee@192.0.2.1>
```

```
;gruu="sip:hha9s8d=-999a@example.com"  
;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"  
;expires=2000
```

The client realizes that a different IP address is registered with the same instance ID. Since the client knows that its instance ID is globally unique, it deletes that registration (message 21, below):

```
REGISTER sip:example.com SIP/2.0  
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbc  
Max-Forwards: 70  
From: Callee <sip:callee@example.com>;tag=ha8d777f2  
Supported: gruu  
To: Callee <sip:callee@example.com>  
Call-ID: hf8asxzff8s7g@192.0.2.2  
CSeq: 3 REGISTER  
Contact: <sip:callee@192.0.2.1>  
;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"  
;expires=0
```

This deletes the contact, as indicated by the lack of of the Contact header field in the resulting 200 OK (message 22, below):

```
SIP/2.0 200 OK  
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbc  
From: Callee <sip:callee@example.com>;tag=ha8d777f2  
To: Callee <sip:callee@example.com>;tag=7asdnj7d6f  
Call-ID: hf8asxzff8s7g@192.0.2.2  
CSeq: 3 REGISTER
```

Finally, the client can retry its original registration (message 23, below):

```
REGISTER sip:example.com SIP/2.0  
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbd  
Max-Forwards: 70  
From: Callee <sip:callee@example.com>;tag=ha8d777f3  
Supported: gruu  
To: Callee <sip:callee@example.com>  
Call-ID: hf8asxzff8s7g@192.0.2.2  
CSeq: 4 REGISTER  
Contact: <sip:callee@192.0.2.2>  
;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
```

This time, the registration succeeds, and the client is registered.

The response, message 24, is shown below:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbd
From: Callee <sip:callee@example.com>;tag=ha8d777f3
To: Callee <sip:callee@example.com>;tag=asd7salll
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 4 REGISTER
Contact: <sip:callee@192.0.2.2>
        ;+sip.instance="urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
        ;expires=3600
```

13. Security Considerations

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](#) [12].

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.

14. IANA Considerations

This specification defines a new Contact header field parameter, a new SIP response code, a SIP URI parameter, 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 [10]. 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 Response Code

This specification defines the new SIP response code, 425, per the

guidelines in [Section 27.4 of RFC 3261](#).

RFC Number: This specification, RFC XXXX [[NOTE to IANA: Please replace XXXX with the RFC number for this specification.]].

Response Code Number: 425

Default Reason Phrase: Instance Conflict

[14.3](#) URI Parameter

This specification defines a new SIP URI parameter, as per the registry created by [\[11\]](#).

Name of the Parameter grid

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

[14.4](#) Media Feature Tag

This section registers a new media feature tag, per the procedures defined in [RFC 2506](#) [\[6\]](#). The tag is placed into the sip tree, which is defined in [\[9\]](#).

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 URI, and ideally 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 [\[16\]](#) 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 of this, 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.

14.5 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., "The Session Initiation Protocol (SIP) UPDATE Method", [RFC 3311](#), October 2002.
- [3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [4] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", [RFC 3265](#), June 2002.
- [5] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", [RFC 3515](#), April 2003.
- [6] Holtman, K., Mutz, A. and T. Hardie, "Media Feature Tag Registration Procedure", [BCP 31](#), [RFC 2506](#), March 1999.
- [7] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", [RFC 2396](#), August 1998.
- [8] Moats, R., "URN Syntax", [RFC 2141](#), May 1997.
- [9] Rosenberg, J., "Indicating User Agent Capabilities in the

Session Initiation Protocol (SIP)",
[draft-ietf-sip-callee-caps-03](#) (work in progress), January 2004.

- [10] Camarillo, G., "The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", [draft-ietf-sip-parameter-registry-02](#) (work in progress), June 2004.
- [11] Camarillo, G., "The Internet Assigned Number Authority (IANA) Universal Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", [draft-ietf-sip-uri-parameter-reg-02](#) (work in progress), June 2004.

16.2 Informative References

- [12] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", [RFC 3323](#), November 2002.
- [13] Hakala, J., "Using National Bibliography Numbers as Uniform Resource Names", [RFC 3188](#), October 2001.
- [14] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol",
[draft-ietf-sipping-conferencing-framework-01](#) (work in progress), October 2003.
- [15] Rosenberg, J. and H. Schulzrinne, "An INVITE Initiated Dialog Event Package for the Session Initiation Protocol (SIP)",
[draft-ietf-sipping-dialog-package-04](#) (work in progress), February 2004.
- [16] Rosenberg, J., Schulzrinne, H. and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)",
[draft-ietf-sip-callerprefs-10](#) (work in progress), October 2003.
- [17] Sugano, H. and S. Fujimoto, "Presence Information Data Format (PIDF)", [draft-ietf-imp-pim-pidf-08](#) (work in progress), May 2003.
- [18] Sparks, R. and A. Johnston, "Session Initiation Protocol Call Control - Transfer", [draft-ietf-sipping-cc-transfer-02](#) (work in progress), February 2004.
- [19] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", [draft-ietf-simple-presence-10](#) (work in progress), January 2003.

- [20] Mealling, M., "A UUID URN Namespace",
[draft-mealling-uuid-urn-03](#) (work in progress), March 2004.

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>

[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 registrar. Others are, of course, permitted, as long as they meet the constraints defined for a GRUU.

[A.1](#) 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 or not 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 implies 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.

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 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 matching the instance ID encoded in the request URI. The Contact URI whose instance ID is that instance ID is then used as the translated version of the 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.

While this approach has many benefits, it has the drawback of producing fairly long GRUUs. The approach in the following section produces smaller results, at the cost of additional structures in the database.

[A.2](#) Hashed Indices

As an alternative approach, the server can construct the GRUU by computing a cryptographic hash of the AOR and instance ID, taking 64 bits of the result, and placing a string representation of those 64 bits into the user part of the URI.

When a GRUU is created through registration or administrative action, the server computes this hash and stores the hash in the database. This hash acts the primary key, with the columns of the table providing the instance ID, AOR and Contact. When the registration is deleted, the corresponding row from the table is removed. When a request arrives to a proxy, the user part of the URI is looked up in the database, and the Contact, AOR and instance ID can be extracted.

This approach produces GRUUs of relatively short length. However, it requires additional structures to be created and stored in a database that would be used by the registrar (at least, new structures are needed for efficient operation). However, it does not require the registrar to store anything for longer than the duration of the registration.

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

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