SIPPING
Internet-Draft
Expires: April 24, 2005

D. Petrie
Pingtel Corp.

October 24, 2004

A Framework for Session Initiation Protocol User Agent Profile Delivery
draft-ietf-sipping-config-framework-05.txt

Status of this Memo

Copyright Notice

Abstract

This document defines the application of a set of protocols for providing profile data to SIP user agents. The objective is to define a means for automatically providing profile data a user agent needs to be functional without user or administrative intervention. The framework for discovery, delivery, notification and updates of user agent profile data is defined here. As part of this framework a new SIP event package is defined here for the notification of profile changes. This framework is also intended to ease ongoing administration and upgrading of large scale deployments of SIP user

agents. The contents and format of the profile data to be defined is outside the scope of this document.

Table of Contents

1. Motivation

Today all SIP user agent implementers use proprietary means of delivering user, device, application and local network policy profiles to the user agent. The profile delivery framework defined in this document is intended to enable a first phase migration to a standard means of providing profiles to SIP user agents. It is expected that UA implementers will be able to use this framework as a means of delivering their existing proprietary data profiles (i.e. using their existing proprietary binary or text formats). This in itself is a tremendous advantage in that a SIP environment can use a single profile delivery server for profile data to user agents from multiple implementers. Follow-on standardization activities can:

1. define a standard profile content format framework (e.g. XML with namespaces [W3C.REC-xml-names11-20040204] or name-value pairs [RFC0822]).
2. specify the content (i.e. name the profile data parameters, xml schema, name spaces) of the data profiles.

One of the objectives of the framework described in this document is to provide a start up experience similar to that of users of an analog telephone. When you plug in an analog telephone it just works (assuming the line is live and the switch has been provisioned). There is no end user configuration required to make analog phone work, at least in a basic sense. So the objective here is to be able to take a new SIP user agent out of the box, plug it in or install the software and have it get its profiles without human intervention other than security measures. This is necessary for cost effective deployment of large numbers of user agents.

Another objective is to provide a scalable means for ongoing administration of profiles. Administrators and users are likely to want to make changes to profiles.

Additional requirements for the framework defined in this document are described in: [I-D.ietf-sipping-ua-prof-framewk-reqs], [I-D.sinnreich-sipdev-req]

2. Introduction

2.1 Requirements Terminology

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [RFC2119].

2.2 Profile Delivery Framework Terminology

profile - data set specific to a user, device, user's application or the local network.
device - software or hardware appliance containing one or more SIP user agent.
profile content server - The server that provides the content of the profiles using the protocol specified by the URI scheme.
notifier - As defined in [RFC3265] the SIP user agent server which processes SUBSCRIBE requests for events and sends NOTIFY requests with profile data or URI(s) that point to the data.
profile delivery server - The logical collection of the notifier and the server which provides the contents of the notification either directly in the NOTIFY requests or indirectly via profile URI(s).

2.3 Overview

The profile life cycle can be described by five functional steps. These steps are not necessarily discrete. However it is useful to describe these steps as logically distinct. These steps are named as follows:

Discovery - discover a profile delivery server
Enrollment - enroll with the profile delivery server
Profile Retrieval - retrieve profile data
Profile Change Notification - receive notification of profile changes
Profile Change Upload - upload profile data changes back to the profile delivery server

Discovery is the process by which a UA finds the address and port at which it enrolls with the profile delivery server. As there is no single discovery mechanism which will work in all network environments, a number of discovery mechanisms are defined with a prescribed order in which the UA tries them until one succeeds.

Enrollment is the process by which a UA makes itself known to the profile delivery server. In enrolling the UA provides identity information, requested profile type(s) and supported protocols for profile retrieval. It also subscribes to a mechanism for notification of profile changes. As a result of enrollment, the UA receives the data or the URI for each of the profiles that the profile delivery server is able to provide. Each profile type (set) requires a separate enrollment or SUBSCRIBE session. A profile type may represent one or more data sets (e.g. one profile data set for each of a user's applications).

Profile Retrieval is the process of retrieving the content for each of the profiles the UA requested.

Profile Change Notification is the process by which the profile delivery server notifies the UA that the content of one or more of the profiles has changed. If the content is provided indirectly the UA MAY retrieve the profile from the specified URI upon receipt of the change notification.

Profile Change Upload is the process by which a UA or other entity (e.g. OSS, corporate directory or configuration management server) pushes a change to the profile data back up to the profile delivery server.

This framework defines a new SIP event package [RFC3265] to solve enrollment and profile change notification steps. This event package defines everything but the mandatory content type. This makes this event package abstract until the content type is bound. The profile content type(s) will be defined outside the scope of this document. It is the author's belief that it would be a huge accomplishment if all SIP user agent used this framework for delivering their existing proprietary profiles. Even though this does not accomplish interoperability of profiles, it is a big first step in easing the administration of SIP user agents. The definition of standard profiles and data sets (see [I-D.petrie-sipping-profile-datasets]) will enable interoperability as a subsequent step.

The question arises as to why SIP should be used for the profile delivery framework. In this document SIP is used for only a small portion of the framework. Other existing protocols are more appropriate for transport of the profile contents (to and from the user agent) and are suggested in this document. The discovery step is simply a specified order and application of existing protocols. SIP is only needed for the enrollment and change notification functionality of the profile delivery framework. In many SIP environments (e.g. carrier/subscriber and multi-site enterprise) firewall, NAT and IP addressing issues make it difficult to get messages between the profile delivery server and the user agent requiring the profiles.

With SIP the users and devices already are assigned globally routable addresses. In addition the firewall and NAT problems are already presumably solved in the environments in which SIP user agents are to be used. Therefore SIP is the best solution for allowing the user agent to enroll with the profile delivery server, which may require traversal of multiple firewalls and NATs. For the same reason the notification of profile changes is best solved by SIP. It should be noted that this document is scoped to providing profiles for devices which contain one or more SIP user agents. This framework may be applied to non-SIP devices, however more general requirements for non-SIP devices are beyond the scope of this document.

The content delivery server may be either in the public network or accessible through a DMZ. The user agents requiring profiles may be behind firewalls and NATs and many protocols, such as HTTP, may be used for profile content retrieval without special consideration in the firewalls and NATs (e.g. an HTTP client on the UA can typically pull content from a server outside the NAT/firewall.).

A conscious separation of device, user, application and local network profiles is made in this document. This is useful to provide features such as hotelling as well as securing or restricting user agent functionality. By maintaining this separation, a user may walk up to someone else's user agent and direct that user agent to get the new user's profile data. In doing so the user agent can replace the previous user's profile data while still keeping the device's and the local network's profile data which may be necessary for core functionality and communication described in this document. The local network profiles are relevant to a visiting device which gets plugged in to a foreign network. The concept of the local network providing profile data is useful to provide hotelling (described above) as well as local policy data that may constrain the user or device behavior relative to the local network. For example media types and codecs may be constrained to reflect the network's capabilities.

The separation of these profiles also enables the separation of the management of the profiles. The user profile may be managed by a profile delivery server operated by the user's ISP. The device profile may be delivered from a profile delivery server operated by the user's employer. The application profile(s) may be delivered from the user's ASP. The local network profile may delivered by a WIFI hotspot service provider. Some interesting services and mobility applications are enabled with this separation of profiles.

A very high level data model is implied here with the separation of these four profile types. Each profile type instance requires a separate subscription to retrieve the profile. A loose hierarchy exists mostly for the purpose of boot strapping and discovery or formation of the profile URIs. No other meaning is implied by this hierarchy. However the profile format and data sets to be defined outside this document may define additional meaning to this hierarchy. In the boot strapping scenario, a device straight out of the box (software or hardware) does not know anything about it's user or local network. The one thing that is does know is it's instance id. So the hierarchy of the profiles exists as follows.

The instance id is used to form the user id part of the URI for subscribing to the device profile. The device profile may contain a default user AOR for that device. The default user AOR may then be

used to retrieve the user profile. Applications to be used on the device may be defined in the device and user profiles. The user's AOR is also used to retrieve any application profiles for that user. The local network profile is not referenced in any way from the device, user, application profiles. It is subscribed to and retrieved based upon a URI formed from the local network domain.

3. Profile Change Event Notification Package

This section defines a new SIP event package [RFC3265]. The purpose of this event package is to send to subscribers notification of content changes to the profile(s) of interest and to provide the location of the profile(s) via content indirection [I-D.ietf-sip-content-indirect-mech] or directly in the body of the NOTIFY. Frequently the profiles delivered to the user agent are much larger (e.g. several KB or even several MB) than the MTU of the network. These larger profiles will cause larger than normal SIP messages and consequently higher impact on the SIP servers and infrastructure. To avoid the higher impact and load on the SIP infrastructure, content indirection SHOULD be used if the profile is large enough to cause packet fragmentation over the transport protocol. The presence of the MIME type for content indirection [I-D.ietf-sip-content-indirect-mech] in the Accept header indicates that the user agent supports content indirection and that the profile delivery server SHOULD use content indirection. Similarly the content type for the differential notification of profile changes [I-D.ietf-simple-xcap-package] may be used in the Accept header to express support for receiving profile change deltas.

The MIME types or formats of profile to be delivered via this framework are to be defined in the documents that define the profile contents. These profile MIME types specified in the Accept header along with the profile types specified in the Event header parameter "profile-type" MAY be used to specify which profiles get delivered either directly or indirectly in the NOTIFY requests. As this event package does not specify the mandatory content type, this package is abstract. The profile definition documents will specify the mandatory content type to make a concrete event package.

3.1 Event Package Name

The name of this package is "sip-profile". This value appears in the Event header field present in SUBSCRIBE and NOTIFY requests for this package as defined in [RFC3265].

3.2 Event Package Parameters

This package defines the following new parameters for the event

header: "profile-type", "vendor", "model", "version", "effective-by", "document", "app-id", "network-user". The "effective-by" parameter is for use in NOTIFY requests only. The "effected-by" parameter is ignored if it appears in a SUBSCRIBE request. The others parameters are for use in the SUBSCRIBE request and are ignored if they appear in NOTIFY requests.

The "profile-type" parameter is used to indicate the token name of the profile type the user agent wishes to obtain data or URIs for and to be notified of subsequent changes. Using a token in this parameter allows the URI semantics for retrieving the profiles to be opaque to the subscribing user agent. All it needs to know is the token value for this parameter. This document defines four logical types of profiles and their token names. The contents or format of the profiles is outside the scope of this document.

The four types of profiles define here are "device", "user", "application" and "local". Specifying "device" type profile(s) indicates the desire for the profile data (URI when content indirection is used) and change notification of the contents of the profile(s) that are specific to the device or user agent. Specifying "user" type profile indicates the desire for the profile data (URI when content indirection is used) and change notification of the profile content for the user. Specifying "application" type profile indicates the desire for the profile data (URI when content indirection is used) and change notification of the profile content for the user's applications. Specifying "local" type profile indicates the desire for profiles data (URI when content indirection is used) specific to the local network. The device, user, application or local network is identified in the URI of the SUBSCRIBE request. The Accept header of the SUBSCRIBE request MUST include the MIME types for all profile content types for which the subscribing user agent wishes to retrieve profiles or receive change notifications.

```
Profile-type       =  "profile-type" HCOLON profile-value
profile-value      =  profile-types / token
profile-types      =  "device" / "user" / "application" / "local"
```

The "device", "user", "application" or "local" token in the profile-type parameter may represent a class or set of profile properties. As standards are defined for specific profile contents related to the user device or local network, it may be desirable to define additional tokens for the profile-type parameter. Also additional content types may be defined along with the profile formats that can be used in the Accept header of the SUBSCRIBE to filter or indicate what data sets of the profile are desired.

The rational for the separation of user, device, application and local network type profiles is provided in Section 2.3. It should be noted that any of the types may result in zero or more profiles or URIs being provided in the NOTIFY request. As discussed, a default user may be assigned to a device. The default user's AOR may in turn be used as the URI to SUBSCRIBE to the "user" and "application" profile types.

The data provided in the four types of profiles may overlap. As an example the codecs that a user prefers to use, the codecs that the device supports (and the enterprise or device owner wishes to use), the codecs that the local network can support (and the network operator wishes to allow) all may overlap in how they are specified in the three corresponding profiles. This policy of merging the constraints across the multiple profile types can only unambiguously be defined along with the profile format and syntax. This is out of scope for this document.

The "vendor", "model" and "version" parameter values are tokens specified by the implementer of the user agent. These parameters MUST be provided in the SUBSCRIBE request for all profile types. These parameters are useful to the profile delivery server to affect the profiles provided. In some scenarios it is desirable to provide different profiles based upon these parameters. For example feature property X in a profile may work differently on two versions of user agent. This gives the profile delivery server the ability to compensate for or take advantage of the differences.

```
Vendor       =  "vendor" HCOLON token / quoted-string
Model        =  "model" HCOLON token / quoted-string
Version      =  "version" HCOLON token / quoted-string
```

The "network-user" parameter MAY be used when subscribing for device and local network profiles. When the profile-type is "device" or "local" , the SUBSCRIBE URI addresses the device or local network profile delivery server. It by design cannot indicate the user's identity. The "network-user" parameter is used to indicate the user's AOR. The SUBSCRIBE server may authenticate the subscriber to verify this AOR. If the value of the "profile-type" parameter is not "device" or "local", the "network-user" parameter has no defined meaning and is ignored.

```
Network-User =  "network-user" HCOLON name-addr / addr-spec
```

When the profile-type is "device", the user agent MAY set the "network-user" parameter to the user's AOR. This is an indication to the profile delivery server to set or change the association of the default user with the device indicated in the SUBSCRIBE URI. If the

profile delivery server implements and allows this policy of setting the default user with a device, the user agent can utilize this mechanism to allow a user to login and make the user agent and user association stick.

In the case where the profile-type is "local", the user agent MAY set the "network-user" parameter. If the user has special privileges beyond that of an anonymous user in the local network, the "network-user" parameter identifies the user to the local network. The value of this parameter is the user's address of record.

The "effective-by" parameter in the Event header of the NOTIFY request specifies the maximum number of seconds before the user agent must attempt to make the new profile effective. A value of 0 (zero) indicates that the subscribing user agent must attempt to make the profiles effective immediately (despite possible service interruptions). This gives the profile delivery server the power to control when the profile is effective. This may be important to resolve an emergency problem or disable a user agent immediately. The "effective-by" parameter is ignored in all messages other than the NOTIFY request.

```
Effective-By =  "effective-by" HCOLON 1*DIGIT
```

The "document" parameter is used to specify a relative URI for a specific profile document that the user agent wishes to retrieve and to receive change notification. This is particularly useful for profile content like XCAP [I-D.ietf-simple-xcap] where there is a well defined URI schema and the user agent knows the specific content that it wants. The "document" parameter value syntax is a quoted string. For more details on the use of this package with XCAP see Section 4.6. The "document" parameter MAY be set in SUBSCRIBE requests. It is ignored in all other messages.

```
Document     =  "document" HCOLON quoted-string
```

The "app-id" parameter MAY be set when the "profile-type" parameter value is "application". The "app-id" indicates that the user agent wishes to retrieve the profile data or URI and change notification for the application profile data for the specific application indicated in the value of the "app-id" parameter. The "app-id" parameter value is a token. The "app-id" parameter has meaning only in SUBSCRIBE requests when the "profile-type" Event header parameter is set to "application". The "app-id" parameter is ignored in all other messages.

```
App-Id       =  "app-id" HCOLON token / quoted-string
```

```
SUBSCRIBE request Event header examples:
Event: sip-profile;profile-type=device;
            vendor=acme;model=Z100;version=1.2.3

Event: sip-profile;profile-type="user";document=
   "http://example.com/services/user-profiles/users/freds.xml";
            vendor=premier;model=trs8000;version=5.5

NOTIFY request Event header examples:
Event:sip-profile;effective-by=0

Event:sip-profile;effective-by=3600
```

3.3 SUBSCRIBE Bodies

This package defines no new use of the SUBSCRIBE request body. Future follow on documents may specify a filter-like mechanism using etags to minimize the delivery or notification of profiles where the user agent already has a current version.

3.4 Subscription Duration

As the presence (or lack of) a device or user agent it not very time critical to the functionality of the profile delivery server, it is recommended that default subscription duration be 86400 seconds (one day).

3.5 NOTIFY Bodies

The size of profile content is likely to be hundreds to several thousand bytes in size. Frequently even with very modest sized SDP bodies, SIP messages get fragmented causing problems for many user agents. For this reason if the Accept header of the SUBSCRIBE included the MIME type: message/external-body indicating support for content indirection the profile delivery server SHOULD use content indirection [I-D.ietf-sip-content-indirect-mech] in the NOTIFY body for providing the profiles.

When delivering profiles via content indirection the profile delivery server MUST include the Content-ID defined in [I-D.ietf-sip-content-indirect-mech] for each profile URI. This is to avoid unnecessary download of the profiles. Some user agents are not able to make a profile effective without rebooting or restarting. Rebooting is something to be avoided on a user agent performing services such as telephony. In this way the Content-ID allows the user agent to avoid unnecessary interruption of service as well. The

Content-Type MUST be specified for each URI.

> Initially user agent implementers may use a proprietary content type for the profiles retrieved from the URI(s). This is a good first step towards easing the management of user agents. Standard profile contents, content type and formats will need to be defined for true interoperability of profile delivery. The specification of the content is out of the scope of this document.

Likewise the URI scheme [RFC2396] used in the content indirection is outside the scope of this document. This document is agnostic to the URI schemes as the profile content may dictate what is required. It is expected that FTP [RFC0959], HTTP [RFC2616], HTTPS [RFC2818], LDAP [RFC3377], XCAP [I-D.ietf-simple-xcap] and other URI schemes are supported by this package and framework.

3.6 Notifier processing of SUBSCRIBE requests

The general rules for processing SUBSCRIBE requests [RFC3265] apply to this package. If content indirection is used for delivering the profiles, the notifier does not need to authenticate the subscription as the profile content is not transported in the SUBSCRIBE or NOTIFY transaction messages. With content indirection only URIs are transported in the NOTIFY request which may be secured using the techniques in Section 6. If content indirection is not used, SIPS with SIP authentication SHOULD be used.

The behavior of the profile delivery server is left to the implementer. The profile delivery server may be as simple as a SIP SUBSCRIBE UAS and NOTIFY UAC front end to a simple HTTP server delivering static files that are hand edited. At the other extreme the profile delivery server can be part of a configuration management system that integrates with a corporate directory and IT system or carrier operations support systems, where the profiles are automatically generated. The design of this framework intentionally provides the flexibility of implementation from simple/cheap to complex/expensive.

If the user or device is not known to the profile delivery server, the implementer MAY accept the subscription or reject it. It is recommended that the implementer accept the subscription. It is useful for the profile delivery server to maintain the subscription as an administrator may add the user or device to the system, defining the profile contents. This allows the profile delivery server to immediately send a NOTIFY request with the profile URIs. If the profile delivery server does not accept the subscription from an unknown user or device, the administer or user must manually provoke the user agent to reSUBSCRIBE. This may be difficult if the

user agent and administrator are at different locations.

When the Event header "profile-type" is "device" and the user agent has provided the user's AOR in the "network-user" parameter, the profile delivery server MAY set or change the default user associated with the device indicated in the SUBSCRIBE URI. This is an implementation or policy decision. The profile delivery server SHOULD authenticate the user for the SUBSCIRBE request before effecting the default user indicated in the "network-user" parameter.

3.7 Notifier generation of NOTIFY requests

As in [RFC3265], the profile delivery server MUST always send a NOTIFY request upon accepting a subscription. If the device or user is unknown to the profile delivery server and it chooses to accept the subscription, the implementer has two choices. A NOTIFY MAY be sent with no body or content indirection containing the profile URI(s). Alternatively a NOTIFY MAY be sent with a body or content indirection containing URI(s) pointing to a default data set. The data sets provided may allow for only limited functionality of the user agent (e.g. a phone user agent with data to enable calls to help desk and emergency services.). This is an implementation and business policy decision for the profile delivery server.

If the URI in the SUBSCIRBE request is a known identity and provisioned with the requested profile type (i.e. as specified in the profile-type parameter of the Event header), the profile delivery server SHOULD send a NOTIFY with profile data or content indirection (if the content type was included in the Accept header) containing the URI for the profile.

A user agent can provide hotelling by collecting a user's AOR and credentials needed to SUBSCRIBE and retrieve the user's profiles. Hotelling functionality is achieved by subscribing to the user's AOR and specifying the "user" profile type. This same mechanism can also be used to secure a user agent, requiring a non-mobile user to login to enable functionality beyond the default user's restricted functionality.

The profile delivery server may specify when the new profiles must be made effective by the user agent. The profile delivery server MAY specify a maximum time in seconds (zero or more), in the "effective-by" event header parameter, by which the user agent is required to make the new profiles effective for all dialogs.

3.8 Subscriber processing of NOTIFY requests

The user agent subscribing to this event package MUST adhere to the

NOTIFY request processing behavior specified in [RFC3265]. The user agent MUST attempt to make the profiles effective within the time in seconds given in the "effective-by" Event header parameter if present in the NOTIFY request (see Section 3.7). By default the user agent makes the profiles effective as soon as it thinks that it is non-obtrusive. Profile changes SHOULD affect behavior on all new dialogs which are created after the notification, but may not be able to affect existing dialogs. The user agent SHOULD use one of the techniques specified in Section 6 to securely retrieve the profiles.

3.9 Handling of forked requests

This event package allows the creation of only one dialog as a result of an initial SUBSCRIBE request. The techniques to achieve this are described in section 4.4.9 of [RFC3265].

3.10 Rate of notifications

It is anticipated that the rate of change for user and device profiles will be very infrequent (i.e. days or weeks apart). For this reason no throttling or minimum period between NOTIFY requests is specified for this package.

3.11 State Agents

State agents are not applicable to this event package.

3.12 Examples

Example SUBSCRIBE and NOTIFY request using content indirection:

```
SUBSCRIBE sip:ff00000036c5@acme.com SIP/2.0
Event: sip-profile;profile-type=device;vendor=acme;
                        model=Z100;version=1.2.3
From: sip:ff00000036c5@acme.com;tag=1234
To: sip:ff00000036c5@acme.com;tag=abcd
Call-ID: 3573853342923422@10.1.1.44
CSeq: 2131 SUBSCRIBE
Contact: sip:ff00000036c5@10.1.1.44
Via: SIP/2.0/TCP 10.1.1.41;
  branch=z9hG4bK6d6d35b6e2a203104d97211a3d18f57a
Accept: message/external-body, application/z100-device-profile
Content-Length: 0
```

```
NOTIFY sip:ff00000036c5@10.1.1.44 SIP/2.0
Event: sip-profile;effective-by=3600
From: sip:ff00000036c5@acme.com;tag=abcd
To: sip:ff00000036c5@acme.com;tag=1234
Call-ID: 3573853342923422@10.1.1.44
CSeq: 321 NOTIFY
Via: SIP/2.0/UDP 192.168.0.3;
  branch=z9hG4bK1e3effada91dc37fd5a0c95cbf6767d1
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=boundary42
Content-Length: ...

--boundary42
Content-Type: message/external-body;
    access-type="URL";
    expiration="Mon, 24 June 2002 09:00:00 GMT";
        URL="http://www.example.com/devices/ff00000036c5";
        size=1234

Content-Type: application/z100-device-profile
Content-ID: <39EHF78SA@example.com>

--boundary42--
```

3.13 Use of URIs to Retrieve State

The URI for the SUBSCRIBE request is formed differently depending upon which profile type the subscription is for. This allows the different profile types to be potentially managed by different profile delivery servers (perhaps even operated by different entities).

3.13.1 Device URIs

The URI for the "device" type profile (device URI) is base upon the identity of the device. The device URI MUST be unique over time and space for all devices and implementations. If an instance id is used as the user part of the device URI, it SHOULD remain the same for the lifetime of the user agent. The device URI is used to identify which profile is associated with a specific instance of a user agent.

> If the user agent were to change its device URI, the profile delivery server would lose its association between the profile and the device. This would also make it difficult for the profile delivery server to track user agents under profile management. The profile delivery server may decide to provide the same device profile to all devices of the same vendor, model and version. However this is a implementation choice on the profile delivery server. The subscribing device has no way of knowing the profile difference. As an example the device profile for similar devices may differ with properties such as the default user. This is how the bootstrapping mechanism works as described in Section 4.1.3.

The URI for the device type profile should use a unique identifier as the user portion of the URI. The host and port portion of the URI is set to that of the domain or address of the profile deliver server which manages that user agent. A means of discovering the host and port portion is discussed in Section 4.1. Like the call-id header value in SIP, consistency of the format across implementations is less important than the guarantee of uniqueness across all instances. There is a administration aspect of the unique identifier, that makes it desirable for the id to be obtainable or predictable prior to installation of the device (hard or soft). Also from a human factors perspective, ids that are easily distinguished and communicated will make the administrators job a little easier. Two approaches are suggested for constructing a unique identifier to be used in the user portion of the device URI.

> The MAC address of the device may be used if there will always be no more than one user agent using that MAC address over time (e.g. a dedicate telephone appliance). The MAC address may not be used if more than one user agent instance exists or use the same MAC address (e.g. multiple instances of a softphone may run on a general purpose computing device). The advantage of the MAC address is that many vendors put bar codes on the device with the actual MAC address on it. A bar code scanner is a convenient means of collecting the instance id for input and provisioning on the profile delivery server. If the MAC address is used, it is recommended that the MAC address is rendered in all lower case with no punctuation for consistency across implementations. For

example a device managed by sipuaconfig.example.com using its MAC address to form the device URI might look like: sip:00df1e004cd0@sipuaconfig.example.com.

For devices where there is no MAC address or the MAC address is not unique to an instance of a user agent (e.g. multiple softphones on a computer or a gateway with multiple logical user agents) it is recommended that a URN [RFC2141] is used as the user portion of the device URI. The approach to defining a user agent instance ID for GRUU [I-D.ietf-sip-gruu] should be considered. When constructing the instance id the implementer should also consider that a human may need to manual enter the instance id to provision the device in the profile delivery server (i.e. longer strings are more error prone in data entry). When the URN is used as the user part of URI, it MUST be URL escaped. The ":" is not a legal character (without being escaped) in the user part of a name-addr. For example the instance ID: urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 would be escaped to look as follows in a URI: sip:urn%3auuid%3af81d4fae-7dec-11d0-a765-00a0c91e6bf6@example.com. Soft user agents are likely to need to use this approach due to the multi-user nature of general purpose computers. The software installer program might generate the uuid as part of the install process so that it remains persistent for the installation. It may also be desirable that any upgrades of the software maintain the unique id. However these are all implementation choices.

3.13.2 User and Application URIs

The URI for the "user" and "application" type profiles is based upon the identity of the user. The user's address of record (AOR) is used as the URI in the SUBSCRIBE request. A new user agent or device may not know the user's AOR. The user's AOR may be obtained as part of a default user property in the device profile. Alternatively the user agent may prompt the user for an AOR and credentials to be used to authenticate the request. This can provide a login and/or hotelling feature on the user agent. The user agent may be pre-provisioned with the user's AOR or provided as information on a SIM or flash key. These are only examples not an exhaustive list of sources for the user AOR.

3.13.3 Local Network URIs

The URI for the "local" type profile is based upon the identity of the local network. When subscribing to the local network profile, the user part of the URI is "anonymous". The host and port part of the URI is the local network name/domain. The discovery of the local network name or domain is discussed in Section 4.1. The user agent

may provide the user's AOR as the value to the "network-user" event header parameter. This is useful if the user has privileges in the local network beyond those of the default user. The profile delivery server SHOULD authenticate the user before providing the profile if additional privileges are granted. Example URI: sip:ananymous@example.com

4. Profile Delivery Framework Details

The following describes how different functional steps of the profile delivery framework work. Also described here is how the event package defined in this document provides the enrollment and notification functions within the framework.

4.1 Discovery of Subscription URI

The discovery approach varies depending upon which profile type URI is to be discovered. The order of discovery is important in the boot strapping situation as user agent may not have any information provisioned. The local network profile should be discovered first as it may contain key information such as how to traverse a NAT/firewall to get to outside services (e.g. the user's profile delivery server). The device profile URI should be discovered next. The device profile may contain the default user's AOR or firmware/ software information that should be updated first before proceeding with the discovery process. The user and application profile subscription URIs should be discovered last. The URIs are formed differently for each of the profile types. This is to support the delegation of the profile management to potentially four different entities. However all four profile types may be provided by the same entity. As the user agent has no way of knowing whether the profiles are provide by one or more different profile delivery servers ahead of time, it must subscribe to all four profile types in separate SUBSCRIBE requests to get the profiles.

4.1.1 Discovery of Local Network URI

The "discovered" host for the "local" profile subscription URI is the local IP network domain for the user agent, either provisioned as part of the device's static network configuration or discovered via DHCP. The local network profile subscription URI should not be cached as the user agent may move from one local network to the other. The user agent should perform the local network discovery every time it starts up or network connectivity is regained.

For example: The user agent requested and received the local domain name via DHCP: loganairport.com. The local network URI would look like: sip:anonymous@loganairport.com. The user agent should send this request using the normal SIP locating mechanisms defined in [RFC3263]. The Event header would look like the following if the user agent decided to provide the user's AOR: sip:alice@example.com as Alice may have a prior arrangement with the local network operator giving her special policy privileges:

```
Event: sip-profile;profile-type=local;
   network-user=sip:alice@example.com
```

4.1.2 Discovery of Device URI

The discovery function is needed to bootstrap user agents to the point of knowing where to enroll with the profile delivery server. Section 3.13.1 describes how to form the device URI used to send the SUBSCRIBE request for enrollment. However the bootstrapping problem for the user agent (out of the box) is what to use for the host and port in the device URI. Due to the wide variation of environments in which the enrolling user agent may reside (e.g. behind residential router, enterprise LAN, WIFI hotspot, ISP, dialup modem) and the limited control that the administrator of the profile delivery server (e.g. enterprise, service provider) may have over that environment, no single discovery mechanism works everywhere.

Therefore a number of mechanisms should be tried in the specified order: SIP DHCP option [RFC3361], SIP DNS SRV [RFC3263], DNS A record and manual. The user agent may be pre-provisioned with the host and port (e.g. service providers may pre-provision a device before sending it to a subscriber, provide a SIM or flash key, etc.) in which case this discovery mechanism is not needed. Before performing the discovery steps, the user agent should provide a means to skip the discovery stage and manually enter the device URI host and port. In addition the user agent should allow the user to accept or reject the discovered host and port, in case an alternate to the discovered host and port are desired.

1. The first discovery mechanism that should be tried to construct the device SUBSCRIBE URI, as described in Section 3.13.1, is to use the host and port of the out bound proxy discovered by the SIP DHCP option as described in [RFC3361]. If the SIP DHCP option is not provided in the DHCP response; or no SIP response is received for the SUBSCRIBE request; or a SIP failure response other than for authorization is received for the SUBSCRIBE request to the sip-profile event, the next discovery mechanism should be tried.

For example: Consider a dedicated hardware device with a single user agent having the MAC address: abc123efg456. The user agent sends a DHCP request including the request for the DHCP option for SIP: 120 (see [RFC3361]). If the DHCP response includes an answer for option 120, then the DNS name or IP address included is used in the host part of the device URI. For this example let's assume: example.com. The device URI would look like: sip:abc123efg456@example.com. The user agent should send this request using the normal SIP locating mechanisms defined in [RFC3263]. If the response fails then, the next discovery mechanism is tried.

2. The local IP network domain for the user agent, either configured or discovered via DHCP, should be used with the technique in [RFC3263] to obtain a host and port to use in the SUBSCRIBE URI. If no SIP response or a SIP failure response other than for authorization is received for the SUBSCRIBE request to the sip-profile event, the next discovery mechanism should be tried.

   For example: The user agent requested and received the local domain name (option 15) in the DHCP response: boston.example.com. The device URI would look like: sip:abc123efg456@boston.example.com. The user agent should send this request using the normal SIP locating mechanisms defined in [RFC3263]. If the response fails then, the next discovery mechanism is tried.

3. The fully qualified host name constructed using the host name "sipuaconfig" and concatenated with the local IP network domain (as provided via DHCP or provisioned) should be tried next using the technique in [RFC3263] to obtain a host and port to use in the SUBSCRIBE URI. If no SIP response or a SIP failure response other than for authorization is received for the SUBSCRIBE request to the sip-profile event, the next discovery mechanism should be tried.

   For example: The user agent requested and received the local domain name via DHCP as in the above example: boston.example.com. The device URI would look like: sip:abc123efg456@sipuaconfig.boston.example.com. The user agent should send this request using the normal SIP locating mechanisms defined in [RFC3263]. If the response fails then, the next discovery mechanism is tried.

4. If all other discovery techniques fail, the user agent MUST provide a manual means for the user to enter the host and port used to construct the SUBSCRIBE URI.

Once a user agent has successfully discovered, enrolled and received a NOTIFY response with profile data or URI(s), the user agent should cache the device profile SUBSCRIBE URI to avoid having to rediscover the profile delivery server again in the future. Caching of the device URI is necessary when the user agent is likely to move to different local network domains as the local network may not be the provider for the device's profile. The user agent should not cache the device URI until it receives a NOTIFY with profile data or URI(s). The reason for this is that a profile delivery server may send 202 responses to SUBSCRIBE requests and NOTIFY responses to unknown user agent (see Section 3.6) with no URIs. Until the profile delivery server has sent a NOTIFY request with profile data or URI(s), it has not agreed to provide profiles.

To illustrate why the user agent should not cache the device profile SUBSCRIBE URI until profile data or URI(s) are provided in the NOTIFY, consider the following example: a user agent running on a laptop plugged into a visited LAN in which a foreign profile delivery server is discovered. The profile delivery server never provides profile URIs in the NOTIFY request as it is not provisioned to accept the user agent. The user then takes the laptop to their enterprise LAN. If the user agent cached the SUBSCRIBE URI from the visited LAN (which did not provide profiles), when subsequently placed in the enterprise LAN which is provisioned to provide profiles to the user agent, the user agent would not attempt to discover the profile delivery server.

4.1.3 Discovery of User and Application URI

The default user's AOR from the device profile (if provided) may then be used to subscribe to the "user" and "application" profiles. The user's AOR may be prepovisioned or provided via SIM or flash key, etc. Alternatively the user's AOR to be used for the "user" and "application" subscription URI, may be "discovered" manually by prompting the user. This "discovered" URI for the user and application profile subscription may be cached.

4.2 Enrollment with Profile Server

Enrollment is accomplished by subscribing to the event package described in Section 3. The enrollment process is useful to the profile delivery server as it makes the server aware of user agents to which it may deliver profiles (those user agents the profile delivery server is provisioned to provide profiles to; those present to which the server may provide profiles in the future; and those that the server can automatically provide default profiles). It is an implementation choice and business policy as to whether the profile delivery server provides profiles to user agents that it is

not explicitly provisioned to do so. However the profile delivery server SHOULD accept (with 2xx response) SUBSCRIBE requests from any user agent as explained in Section 3.5.

4.3 Notification of Profile Changes

The NOTIFY request in the sip-profile event package serves two purposes. First it provides the user agent with a means to obtain the profile data directly or via URI(s) for desired profiles without requiring the end user to manually enter them. It also provides the means for the profile delivery server to notify the user agent that the content of the profiles has changed and should be made effective. Optionally the differential changes may be obtained by including the content-type: "application/xcap-diff+xml" defined in [I-D.ietf-simple-xcap-package] in the Accept header of the SUBSCRIBE request.

4.4 Retrieval of Profile Data

The user agent retrieves its needed profile(s) directly or via the URI(s) provided in the NOTIFY request as specified in Section 3.5. The profile delivery server SHOULD secure the content of the profiles using one of the techniques described in Section 6. The user agent SHOULD make the new profiles effective in the timeframe described in Section 3.2.

The contents of the profiles SHOULD be cached by the user agent. This it to avoid the situation where the content delivery server is not available, leaving the user agent non-functional.

4.5 Upload of Profile Changes

The user agent or other service MAY push changes up to the profile delivery server using the technique appropriate to the profile's URL scheme (e.g. HTTP PUT method, FTP put command). The technique for pushing incremental or atomic changes MUST be described by the specific profile data framework. A means for pushing changes up into the profile delivery server for XCAP is defined in [I-D.ietf-simple-xcap].

4.6 Usage of XCAP with the Profile Package

This framework allows for the usage of several different protocols for the retrieval of profiles. One protocol which is suitable is XCAP [I-D.ietf-simple-xcap], which allows for HTTP URIs to represent XML documents, elements and attributes. XCAP defines a specific hierarchy for how documents are organized. As a result, it is necessary to discuss how that organization relates to the rough data

model presented here.

When a user or device enrolls with a SUBSCRIBE request, the request UIR will contain some kind of identifying information for that user or device. This identity is mapped to an XCAP User ID (XUID) based on an implementation specific mapping. The "profile-type" along with the "app-id" Event header parameters specify the specific XCAP application usage.

In particular, when the Event header parameter "profile-type" is "application", the "app-id" MAY be included to contain the XCAP Application Unique ID (AUID). When the "profile-type" is "application", but the "app-id" parameter is absent, this specifies that the user wishes to SUBSCRIBE to all documents for all application usages associated with the user in the request-uri. This provides a convenient way for a single subscription to be used to obtain all application data. The XCAP root is determined by a local mapping.

When the "profile-type" is "device", or "user" or "local", this maps to an AUID and document selector for representing device, user and local-network data, respectively. The mapping is a matter of local policy. This allows different providers to use different XCAP application usages and document schemas for representing these profiles, without having to configure the device with the specific AUID which is being used.

Furthermore, when the "document" attribute is present, it identifies a specific document that is being requested. If the "profile-type" is "application", the "app-id" MAY be present as well if the "document" relative path does not indicate the specific application profile. The "document" attribute then specifies a relative path reference. Its first path segment is either "global", specifying global data, or "user", specifying user data for the user in the request URI. The next path segment identifies the path in the global directory or the user's home directory. For "profile-type" "application", if "app-id" is not present the next path segment (i.e. after "global" or the user's home directory segment) MAY indicate the XCAP Application Unique ID (AUID) if the user agent wishes to subscribe to a specific application profile.

For example, consider a phone with an instance ID of urn:uuid:00000000-0000-0000-0000-0003968cf920. To obtain its device profile, it would generate a SUBSCRIBE that contain the following Request-Line and Event header:

```
SUBSCRIBE
 sip:urn%3auuid%3a00000000-0000-0000-0000-0003968cf920@example.com
 SIP/2.0
Event: sip-profile;profile-type=device
```

If the profile data is stored in an XCAP server, the server would map the "device" profile to an application usage and document selector based on local policy. If this mapping specifies the AUID "vendor2-device-data" and a document called "index" within the user directory, the corresponding HTTP URI for the document is:

```
http://xcap.example.com/root/vendor2-device-data/users/
urn%3auuid%3a00000000-0000-0000-0000-0003968cf920/index
```

and indeed, if a content indirection is returned in a NOTIFY, the URL would equal this.

That user profile might specify the user identity (as a SIP AOR) and their application-usages. From that, the device can enroll to learn about its application data. To learn about all of the data:

```
SUBSCRIBE sip:user-aor@example.com SIP/2.0
Event: sip-profile;profile-type=application
```

The server would map the request URI to an XUI (user-aor, for example) and the xcap root based on local policy. If there are two AUIDs, "resource-lists" [I-D.ietf-simple-xcap-list-usage] and "rls-services" [I-D.ietf-simple-xcap-list-usage], this would result in a subscription to all documents within:

```
http://xcap.example.com/root/rls-services/users/user-aor
http://xcap.example.com/root/resource-lists/users/user-aor
```

The user would not be subscribed to the global data for these two application usages, since that data is not important for users.

However, the user/device could be made aware that it needs to subscribe to a specific document. In that case, its subscribe would look like:

```
SUBSCRIBE sip:user-aor@example.com SIP/2.0
Event: sip-profile;profile-type=application;app-id=resource-lists
  ;document="global/index"
```

this would result in a subscription to the single global document for resource-lists.

In some cases, these subscriptions are to a multiplicity of documents. In that case, the notification format will need to be one which can indicate what document has changed. This includes content indirection, but also the xcap diff format [I-D.ietf-simple-xcap-package].

5. IANA Considerations

There are several IANA considerations associated with this specification.

5.1 SIP Event Package

This specification registers a new event package as defined in [RFC3265]. The following information required for this registration:

Package Name: sip-profile
Package or Template-Package: This is a package
Published Document: RFC XXXX (Note to RFC Editor: Please fill in XXXX with the RFC number of this specification).
Person to Contact: Daniel Petrie dpetrie AT pingtel.com
New event header parameters: profile-type, vendor, model, version, effective-by, document, app-id, network-user

6. Security Considerations

Profiles may contain sensitive data such as user credentials. The protection of this data depends upon how the data is delivered. If the data is delivered in the NOTIFY body, SIP authentication MUST be used for SUBSCRIPTION and SIPS and/or S/MIME MAY be use to encrypt the data. If the data is provided via content indirection, SIP authentication is not necessary for the SUBSCRIBE request. With content indirection the data is protected via the authentication, authorization and encryption mechanisms provided by the profile URL scheme. Use of the URL scheme security mechanisms via content indirection simplifies the security solution as the SIP event package does not need to authenticate, authorize or protect the contents of the SIP messages. Effectively the profile delivery server can safely provide profile URI(s) to anyone. The profile content is protected via the URL scheme transport mechanisms for authentication, authorization and encryption (e.g. via HTTPS). HTTPS provides two possible mechanisms for authentication: 1) the device may have a certificate that the profile deliver server can request in the TLS setup; or 2) the profile deliver server may use HTTP authentication [RFC2617] with the device or users credentials.

6.1 Symmetric Encryption of Profile Data

If the transport for the URL scheme used for content indirection does not provide authentication, authorization or encryption, a technique to provide this is to encrypt the profiles on the content delivery server using a symmetric encryption algorithm using a shared key. The encrypted profiles are delivered by the content delivery server via the URIs provided in the NOTIFY requests. Using this technique the profile delivery server does not need to provide authentication or authorization for the retrieval as the profiles are obscured. The user agent must obtain the username and password from the user or other out of band means to generate the key and decrypt the profiles.

7. Change History

Many thanks to those who contributed and commented on the many iterations of this document. Detailed input was provided by Jonathan Rosenberg from Dynamicsoft, Henning Schulzrinne from Columbia U., Cullen Jennings from Cisco, Rohan Mahy from Cisco, Rich Schaaf from Pingtel, Volker Hilt from Bell Labs, Hisham khartabil from Nokia, Henry Sinnreich from MCI, Martin Dolly from ATT Labs, and John Elwell from Siemens.

7.1 Changes from draft-ietf-sipping-config-framework-04.txt

Clarified usage of instance-id
Specify which event header parameters are mandatory or optional and in which messages.
Included complete list of event header parameters in parameter overview and IANA sections.
Removed TFTP reference as protocol for profile transport.
Added examples for discovery.
Added ABNF for all event header parameters.
Changed profile-name parameter back to profile-type. This was changed profile-name in 02 when the parameter could contain either a token or a path. Now that the path is contained in the separate parameter: "document", profile-type make more sense as the parameter name.
Fixed some statements that should have and should not have been normative.
Added the ability for the user agent to request that the default user associated with the device be set/changed using the "network-user" parameter.
A bunch of editorial nits and fixes.

7.2 Changes from draft-ietf-sipping-config-framework-03.txt

Incorporated changes to better support the requirements for the use

of this event package with XCAP and SIMPLE so that we can have one package (i.e. simple-xcap-package now defines a content type not a package). Added an additional profile type: "application". Added document and app-id Event header parameters in support of the application profile. Define a loose high level data model or relationship between the four profile types. Tried to edit and fix the confusing and ambiguous sections related to URI formation and discovery for the different profile types. Better describe the importance of uniqueness for the instance id which is used in the user part of the device URI.

7.3 Changes from draft-ietf-sipping-config-framework-02.txt

Added the concept of the local network as a source of profile data. There are now three separate logical sources for profile data: user, device and local network. Each of these requires a separate subscription to obtain.

7.4 Changes from draft-ietf-sipping-config-framework-01.txt

Changed the name of the profile-type event parameter to profile-name. Also allow the profile-name parameter to be either a token or an explicit URI.

Allow content indirection to be optional. Clarified the use of the Accept header to indicate how the profile is to be delivered.

Added some content to the Iana section.

7.5 Changes from draft-ietf-sipping-config-framework-00.txt

This version of the document was entirely restructured and re-written from the previous version as it had been micro edited too much.

All of the aspects of defining the event package are now organized in one section and is believed to be complete and up to date with [RFC3265].

The URI used to subscribe to the event package is now either the user or device address or record.

The user agent information (vendor, model, MAC and serial number) are now provided as event header parameters.

Added a mechanism to force profile changes to be make effective by the user agent in a specified maximum period of time.

Changed the name of the event package from sip-config to sip-profile

Three high level security approaches are now specified.

7.6 Changes from draft-petrie-sipping-config-framework-00.txt

Changed name to reflect SIPPING work group item

Synchronized with changes to SIP DHCP [RFC3361], SIP [RFC3261] and [RFC3263], SIP Events [RFC3265] and content indirection [I-D.ietf-sip-content-indirect-mech]

Moved the device identity parameters from the From field parameters to User-Agent header parameters.

Many thanks to Rich Schaaf of Pingtel, Cullen Jennings of Cisco and Adam Roach of Dyamicsoft for the great comments and input.

7.7 Changes from draft-petrie-sip-config-framework-01.txt

Changed the name as this belongs in the SIPPING work group.

Minor edits

7.8 Changes from draft-petrie-sip-config-framework-00.txt

Split the enrollment into a single SUBSCRIBE dialog for each profile. The 00 draft sent a single SUBSCRIBE listing all of the desired. These have been split so that each enrollment can be routed differently. As there is a concept of device specific and user specific profiles, these may also be managed on separate servers. For instance in a roaming situation the device might get its profile data from a local server which knows the LAN specific profile data. At the same time the user specific profiles might come from the user's home environment profile delivery server.

Removed the Config-Expires header as it is largely superfluous with the SUBSCRIBE Expires header.

Eliminated some of the complexity in the discovery mechanism.

Suggest caching information discovered about a profile delivery server to avoid an avalanche problem when a whole building full of devices powers up.

Added the User-Profile From header field parameter so that the device can request a user specific profile for a user that is different from the device's default user.

8 References

[I-D.ietf-simple-xcap]
Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-04 (work in progress), October 2004.

[I-D.ietf-simple-xcap-list-usage]
Rosenberg, J., "Extensible Markup Language (XML) Formats for Representing Resource Lists", draft-ietf-simple-xcap-list-usage-04 (work in progress), October 2004.

[I-D.ietf-simple-xcap-package]
Rosenberg, J., "An Extensible Markup Language (XML) Document Format for Indicating Changes in XML Configuration Access Protocol (XCAP) Resources", draft-ietf-simple-xcap-package-02 (work in progress), July 2004.

[I-D.ietf-sip-content-indirect-mech]
Burger, E., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", draft-ietf-sip-content-indirect-mech-05 (work in progress), October 2004.

[I-D.ietf-sip-gruu]
Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-02 (work in progress), July 2004.

[I-D.ietf-sipping-ua-prof-framewk-reqs]
Petrie, D. and C. Jennings, "Requirements for SIP User Agent Profile Delivery Framework", draft-ietf-sipping-ua-prof-framewk-reqs-00 (work in progress), March 2003.

[I-D.petrie-sipping-profile-datasets]
Petrie, D., "A Schema for Session Initiation Protocol User Agent Profile Data Sets", draft-petrie-sipping-profile-datasets-00 (work in progress), July 2004.

[I-D.sinnreich-sipdev-req]
Butcher, I., Lass, S., Petrie, D., Sinnreich, H. and C. Stredicke, "SIP Telephony Device Requirements and

Configuration", draft-sinnreich-sipdev-req-04 (work in progress), July 2004.

[RFC0822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

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

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2396] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

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

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

[RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[RFC3361] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers", RFC 3361, August 2002.

[RFC3377] Hodges, J. and R. Morgan, "Lightweight Directory Access Protocol (v3): Technical Specification", RFC 3377, September 2002.

[W3C.REC-xml-names11-20040204] Tobin, R., Hollander, D., Layman, A. and T. Bray, "Namespaces in XML 1.1", W3C REC REC-xml-names11-20040204, February 2004.

Author's Address

Daniel Petrie
Pingtel Corp.
400 W. Cummings Park
Suite 2200
Woburn, MA 01801
US

Phone: "Dan Petrie (+1 781 938 5306)"<sip:dpetrie AT pingtel.com>
EMail: dpetrie AT pingtel.com
URI: http://www.pingtel.com/

Appendix A. Acknowledgments

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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