SIPPING
Internet-Draft
Expires: September 7, 2006

D. Petrie
SIPez LLC.

Mar 6, 2006

A Framework for Session Initiation Protocol User Agent Profile Delivery

draft-ietf-sipping-config-framework-08.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. Introduction 4
2. Requirements Terminology 4
3. Profile Delivery Framework Terminology 5
4. Overview 5
5. Use Cases 7
5.1. Service Provider Use Case Scenario Bootstrapping with Digest Authentication 7
5.2. Service Provider Use Case Scenario Bootstrapping with Device Certificate 9
6. Data Model 9
7. Profile Change Event Notification Package 11
7.1. Event Package Name 11
7.2. Event Package Parameters 11
7.3. SUBSCRIBE Bodies 15
7.4. Subscription Duration 16
7.5. NOTIFY Bodies 16
7.6. Notifier processing of SUBSCRIBE requests 17
7.7. Notifier generation of NOTIFY requests 18
7.8. Subscriber processing of NOTIFY requests 19
7.9. Handling of forked requests 19
7.10. Rate of notifications 19
7.11. State Agents 19
7.12. Examples 19
7.13. Use of URIs to Retrieve State 20
7.13.1. Device URIs 21
7.13.2. User and Application URIs 22
7.13.3. Local Network URIs 23
8. Profile Delivery Framework Details 23
8.1. Discovery of Subscription URI 23
8.1.1. Discovery of Local Network URI 24
8.1.2. Discovery of Device URI 24
8.1.3. Discovery of User and Application URI 27
8.2. Enrollment with Profile Server 28
8.3. Notification of Profile Changes 28
8.4. Retrieval of Profile Data 28
8.5. Upload of Profile Changes 29
9. IANA Considerations 29

9.1. SIP Event Package 29
10. Security Considerations 29
10.1. Confidential Profile Content in NOTIFY Request 30
10.2. Confidential Profile Content via Content Indirection . . 31
10.3. Integrity protection for non-confidential profiles . . . 32

10.4. Initial Enrollment Using a Manufacturer's Certificate . . 32
11. Acknowledgements 34
12. Change History 34
12.1. Changes from draft-ietf-sipping-config-framework-07.txt 34
12.2. Changes from draft-ietf-sipping-config-framework-06.txt 34
12.3. Changes from draft-ietf-sipping-config-framework-05.txt 35
12.4. Changes from draft-ietf-sipping-config-framework-04.txt 35
12.5. Changes from draft-ietf-sipping-config-framework-03.txt 36
12.6. Changes from draft-ietf-sipping-config-framework-02.txt 36
12.7. Changes from draft-ietf-sipping-config-framework-01.txt 36
12.8. Changes from draft-ietf-sipping-config-framework-00.txt 36
12.9. Changes from draft-petrie-sipping-config-framework-00.txt 37
12.10. Changes from draft-petrie-sip-config-framework-01.txt . . 37
12.11. Changes from draft-petrie-sip-config-framework-00.txt . . 37
13. References 38
13.1. Normative References 38
13.2. Informative References 39
Author's Address 41
Intellectual Property and Copyright Statements 42

1. Introduction

Today all SIP (Session Initiation Protocol) [RFC3261] 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 (User Agent) 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. 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].

3. 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 agents.
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 URIs (Uniform Resource Identifiers) 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).
hotelling- when a user moves to a new user agent (i.e. that is not already provisioned to know the user's identity, credentials or profile data) and gives the user agent sufficient information to retrieve the user's profile(s). The user agent either permanently or temporarily makes the user's profiles effective on that user agent.
roaming- when the user agent moves to a different local network

4. 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. The means of discovery is described in Section 8.1.

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). Enrollment which is performed by the device by constructing and sending a SUBSCRIBE request to profile delivery server for the event package described in Section 7.

Profile Retrieval is the process of retrieving the content for each of the profiles the UA requested. The profiles are retrieved either directly or indirectly from the NOTIFY request body as describe in Section 7.5 and Section 8.4.

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 notification is provided by the NOTIFY request for the event package as described in Section 7.8 and Section 8.3.

Profile Change Upload is the process by which a UA or other entity (e.g. corporate directory or configuration management server) pushes a change to the profile data back up to the profile delivery server. This process is described in Section 8.5.

This framework defines a new SIP event package [RFC3265] to solve enrollment and profile change notification steps. The event package in Section 7 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 agents 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 (see Section 8.1). SIP is only needed for the enrollment (see

Section 8.2) and change notification functionality (see Section 8.3) of the profile delivery framework. In many SIP environments (e.g. carrier/subscriber and multi-site enterprise) firewall, NAT (Network Address Translation) 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. The local network profile (see Section 6, Section 7.13.3 and Section 8.1.1) provides the means to get firewall and NAT traversal mechanism information to the device. 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 private network. 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.).

5. Use Cases

The following use case are intended to help give an understanding of how the profile delivery framework can be used. These use cases are not intended to be exhaustive in demonstrating all the capabilities or ways the framework can be applied.

5.1. Service Provider Use Case Scenario Bootstrapping with Digest Authentication

The following describes a use case scenario for bootstrapping a new user agent, which has had no prior provisioned information, to the point of being functional with a SIP Service Provider's system. In this example scenario, the user has purchased a new SIP user agent. The user signs up for the service to obtain three pieces of information: a hostname, a user ID and a password. These three pieces of information may be one-time use, that become invalid after

the one use. This scenario assumes that no association or mapping

between the device and the user's account is created before the following steps:

1. The user plugs the device in to provide power and network connectivity the first time (or installs the software in the case of a software user agent). The device subscribes to the local network to get the local network profile. However as the device is plugged into a residential LAN or router, there is no profile delivery server for the local network profile (see Section 8.1.1 and Section 7.13.3). The device assumes symmetric SIP signalling as there is not local network profie which may have provided other firewall or NAT traversal mechanism information.
2. The device prompts the user for the hostname to subscribe to for the device profile. The hostname was provided by the service provider and is used as the host part of the SUBSCRIBE profile URI described in Section 7.13.1. Note: in a scenario where the system operator (e.g. enterprise) has control of the network, the hostname for the SUBSCRIBE can be discovered (see Section 8.1.2) to avoid the need for the user to enter the hostname.
3. The device creates a TLS connection for the SIP SUBSCRIBE request to the provided hostname. The device verifies the server's certificate. If the common name does not match the hostname or the certificate is not valid, the device warns the user and prompts whether to continue.
4. The profile delivery server receives the SUBSCRIBE request for the device profile and sends a NOTIFY with content indirection containing the HTTPS URI for the device profile (see Section 7.5).
5. The device receives the NOTIFY request with the device profile URI. The device prompts the user for the user ID and password provided by the service provider. The device does an HTTPS GET to retrieve the device profile (see Section 8.4 and Section 7.8). The profile delivery server challenges for Digest authentication. The device re-sends the HTTPS GET with Digest credentials using the user ID and password entered by the user. Note: for devices with only DTMF style input, the service provider may provide the host, user ID and password in octal format that can be entered requiring only digits.
6. The profile delivery server receives the HTTP GET request for the device profile along with the user ID and password for the specific user. At this point the profile delivery server has authenticated the user and can create an association between a specific device identified in the HTTPS URI and the user or user account (see Section 10.2). The profile delivery server provides the device profile which may contain the on-going SUBSCRIBE request URIs for the device, user and application profiles along with credentials for retrieving the profiles.

7. The device receives the device profile from the HTTPS response, re-SUBSCRIBEs using the device profile URI provided in the profile. The device profile also may contain URIs for the default user's user and application profile SUBSCRIBE request URIs for the SIP event package defined in Section 7. The device uses these URIs to retrieve user and application profiles in a similar way to the device profile. After retriving these profiles the device is fully functional in the service provider's SIP service.

5.2. Service Provider Use Case Scenario Bootstrapping with Device Certificate

The following describes another use case scenario where the device implementor provides a certificate for the device which authenticates the device ID. In this scenario, the user signs up for the SIP service with the service provider and provides the device ID (see Section 7.13.1 for more information on device ID) to the service provider prior to the following steps, so that the service provider has an association or mapping between the device ID and the user account ahead of time. The service provide gives the user a hostname to be entered on the device.

1. Step 1-3 occur the same as in the prior use case described in Section 5.1.
2. The device receives the NOTIFY request with the device profile URI. The device does an HTTPS GET to retrieve the device profile (see Section 8.4 and Section 7.8).
3. The profile delivery server requests the device certficate in the TLS connection used for the HTTPS GET. The device has a certificate that contains the MAC address used in the device ID. The device certificate is signed and provided by the implementor for the purpose of authenticating the device ID in the initial bootstrapping process only. The profile delivery server validates the device ID and returns the device profile using HTTPS.
4. The device receives the device profile in the HTTPS response. The process continues in a similar way to step 6 in the above use case. The device profile contains a more perminent device certificate and private key or Digest authentication credentials which are used for on-going device ID authentication.

6. Data Model

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 (described above) 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 roaming (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 (Application Service Provider). The local network profile may delivered by a WLAN (Wireless LAN) 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 bootstrapping 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 bootstrapping scenario, a device straight out of the box (software or hardware) does not know anything about its 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 local network profile is subscribed to and retrieved based upon a URI formed from the local network domain. The local network profile is subscribed to first as it may contain information on how to

communicate to the Internet or primary network from the local network (e.g. HTTP proxy, SIP firewall or NAT traversal information). The device instance id is used to form the user id part of the URI for subscribing to the device and local network profiles. The device profile may contain a default user AOR (Address of Record) 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.

7. 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-diff] may be used in the Accept header to express support for receiving profile change deltas.

The MIME types or formats of profiles 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.

7.1. Event Package Name

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

7.2. Event Package Parameters

This package defines the following new parameters for the event header: "profile-type", "vendor", "model", "version", "effective-by", and "network-user". The "effective-by" parameter is for use in NOTIFY requests only. The "effective-by" parameter is ignored if it appears in a SUBSCRIBE request. The other 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 defined here are "device", "user", "application" and "local-network". 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 that is 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-network" type profile indicates the desire for profile 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. A separate SUBSCRIBE dialog is used for each profile type. The profile type associated with the dialog can then

be used to infer which profile type changed and is contained in the NOTIFY or content indirection URI. 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. In the following ABNF, EQUAL and token are defined in [RFC3261].

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

The "device", "user", "application" or "local-network" 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 rationale for the separation of user, device, application and local network type profiles is provided in Section 4. 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, if defined in the device profile, 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 for merging the constraints across the multiple profile types can only unambiguously be defined in the context of the profile syntax and semantics. 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. The implementer SHOULD use their DNS domain name (e.g. example.com) as the value of the "vendor" parameter so that it is known to be unique. 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 the same user agent. This gives the profile delivery server the ability to compensate for or take advantage of the differences. In the following ABNF, EQUAL and quoted-string are defined in [RFC3261].

```
Vendor       =  "vendor" EQUAL quoted-string
Model        =  "model" EQUAL quoted-string
Version      =  "version" EQUAL quoted-string
```

The "network-user" parameter SHOULD be set when subscribing for device and local network profiles if the user's AOR is known. When the profile-type is "device" or "local-network", the SUBSCRIBE URI addresses the device or local network profile delivery server. As the SUBSCRIBE request URI for the "device" or "local-network" profile must contain the device identity, it cannot contain the user profile identifier. The "network-user" parameter is used to indicate the user profile resource identifier. The SUBSCRIBE server SHOULD authenticate the subscriber to verify the resource identifier in the "network-user" parameter if the profile provided is specific to the user (e.g. granting policies or privileges beyond those of an anonymous user). If the value of the "profile-type" parameter is not "device" or "local-network", the "network-user" parameter has no defined meaning and is ignored. If the "network-user" parameter is provided in the SUBSCRIBE request, it MUST be present in the NOTIFY

request as well. In the following ABNF, EQUAL, LDQUOT, RDQUOT and addr-spec are defined in [RFC3261].

```
Network-User =  "network-user" EQUAL LDQUOT addr-spec RDQUOT
```

The entity that is subscribing and getting the "device" and "local-network" profiles is the device. For this reason the From field should indicate the device's identity. These profiles types contain device specific information and it is the device's identity that gets authenticated for the "device" profile. Depending upon the local adminstration policy and segmentation of

services, the device identity and user profile identity association may not be know to the the configuration delivery server ahead of time. So because the From field and SUBSCRIBE request URI indicate the "device" profile resource identifier, the "network-user" parameter is needed to indicate the additional resource identifier for the user assocated with this device.

When the profile-type is "device", the user agent SHOULD set the "network-user" parameter to the "user" profile resource identifier is known. 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 permanent.

In the case where the profile-type is "local-network", the user agent SHOULD set the "network-user" parameter if the user's AOR is known. 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. The "effective-by" parameter MAY be provided in the NOTIFY request for any of the profile types. 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. In the following ABNF, EQUAL and DIGIT are defined in [RFC3261].

Effective-By = "effective-by" EQUAL 1*DIGIT

SUBSCRIBE request Event header examples:

```
Event: ua-profile;profile-type=device;
            vendor="vendor.example.com";model="Z100";version="1.2.3"
```

```
Event: ua-profile;profile-type="user";
 vendor="premier";model="trs8000";version="5.5"

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

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

The following table shows the use of Event header parameters in SUBSCRIBE requests for the four profile types:

profile-type	device	user	application	local-network
vendor	m	m	m	m
model	m	m	m	m
version	m	m	m	m
network-user	s			s
effective-by				

m - mandatory
s - SHOULD be provided
o - optional
Non-specified means that the parameter has no meaning and should be ignored.

The following table shows the use of Event header parameters in NOTIFY requests for the four profile types:

profile-type	device	user	application	local-network
vendor				
model				
version				
network-user				s
effective-by	o	o	o	o

7.3. SUBSCRIBE Bodies

This package defines no new use of the SUBSCRIBE request body. Future 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.

7.4. Subscription Duration

As the presence (or lack of) a device or user agent is not very time critical to the functionality of the profile delivery server, it is recommended that default subscription duration be 86400 seconds (one day). A one-time fetch of a profile can be accomplished by setting the Expires parameter to 0 as defined in [RFC3265] resulting in a single NOTIFY with no change notification.

7.5. NOTIFY Bodies

The size of profile content is likely to be hundreds to several thousand of bytes in size. 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 MIME header described 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. By examining the Content-ID, the user agent can recognize if it already has the indirected content, thus avoiding unnecessary interruption of service. The Content-Type MUST be specified for each URI. For minimal interoperability, the profile delivery server MUST support the "http:" and "https:" URI schemes for content indirection. Other URI schemes MAY also be provided in the content indirection. However the security considerations are define for content indirection using HTTP and HTTPS. Other protocols MAY be supported for content indirection, but are out of scope of this document.

> 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.

The URI scheme [RFC2396] used in content indirection may be dictated by the profile content that is required. It is expected that FTP [RFC0959], HTTP [RFC2616], HTTPS [RFC2818], LDAP [RFC3377], XCAP [I-D.ietf-simple-xcap] and other URI schemes could be used by this package and framework if the subscribing user agent and profile

delivery server both support the same scheme. The negotiation of the

URI scheme is described in the following sections.

7.6. 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 10. If content indirection is not used, the subscribe server SHOULD reject SUBSCRIBE requests from conections that are not over TLS and SHOULD challenge the SUBSCRIBE request with SIP Digest authentication. The subscriber MUST support the "http:" or "https:" URI scheme for content indirection. If the subscriber wishes to use a URI scheme other than "http:", the subscriber must use the "schemes" Contact header field parameter to indicate the URI scheme as defined in [I-D.ietf-sip-content-indirect-mech]. For example the subscriber may request that content indirection use the "ldaps:" URI scheme by including "ldaps" in the "scheme" Contact header parameter of the SUBSCRIBE request. If the subscriber does not specify the URI scheme, the notifier may use either "http:" or "https:".

> The profile generation 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 for unprovisioned users or devices as an administrator may add the

user or device to the system after the initial subscription, 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 re-subscribe. This may be difficult if the user agent and administrator are at different locations.

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.

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 SUBSCRIBE request before changing the default user associated with the device.

7.7. 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. for a user agent with telephony capabilities, 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 SUBSCRIBE request is a known identity and is 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 indirection mime type was included in the Accept header) containing the URI for the profile. To protect the integrety of the profile data or indirect content profile data URIs, the notifier SHOULD send the NOTIFY request on the same TLS connection as the SUBSCRIBE request came in on if TLS was used.

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.

7.8. 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 7.7). By default the user agent makes the profiles effective as soon as it thinks that it is non-obtrusive to do so (e.g. when there are no active calls). 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 10 to securely retrieve the profiles. If the subscriber included the MIME type message/external-body for content indirection in the SUBSCRIBE request Accept header, the subscriber MUST support the http: or https: URI schemes for content indirection. If the subscriber indicated alternative URI schemes for content indirection it MUST also indicate support for http: or https:. The subscriber should still be prepared to use http: or https: as the profile delivery server may not support the alternative URI schemes.

7.9. 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].

7.10. 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.

7.11. State Agents

State agents are not applicable to this event package.

7.12. Examples

Example SUBSCRIBE and NOTIFY request using content indirection:


```
SUBSCRIBE sip:MAC%3aFF00000036C5@acme.example.com SIP/2.0
Event: ua-profile;profile-type=device;vendor="vendor.example.com";
                          model="Z100";version="1.2.3"
From: sip:MAC%3aFF00000036C5@acme.example.com;tag=1234
To: sip:MAC%3aFF00000036C5@acme.example.com;tag=abcd
Call-ID: 3573853342923422@10.1.1.44
CSeq: 2131 SUBSCRIBE
Contact: sip:MAC%3aFF00000036C5@10.1.1.44
Via: SIP/2.0/TCP 10.1.1.41;
  branch=z9hG4bK6d6d35b6e2a203104d97211a3d18f57a
Accept: message/external-body, application/x-z100-device-profile
Content-Length: 0
```

```
NOTIFY sip:MAC%3aFF00000036C5@10.1.1.44 SIP/2.0
Event: ua-profile;effective-by=3600
From: sip:MAC%3aFF00000036C5@acme.example.com;tag=abcd
To: sip:MAC%3aFF00000036C5@acme.example.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/x-z100-device-profile
Content-ID: <39EHF78SA@example.com>

--boundary42--
```

7.13. 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). The To and From field will typically contain the same URI as is used in the original SUBSCIRBE request URI.

7.13.1. Device URIs

The URI for the "device" type profile (device URI) is based upon the identity of the device. The device URI MUST be unique across 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 changed its device URI, the profile delivery server would not know the 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 of the profile delivery server. The subscribing device has no way of knowing whether the profiles for each device are different. For this reason the device must always use a unique id in the device SUBSCRIBE request URI. 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 8.1.3.

The URI for the device type profile MUST 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 delivery server which manages that user agent. A means of discovering the host and port portion is discussed in Section 8.1. There is an 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. The MAC address or UUID SHOULD be used for constructing a unique identifier to be used in the user portion of the device URI.

If the identifier is a MAC address, it MUST be formatted as the characters "MAC:" followed by a 12 digit hexadecimal representation of the MAC address. The address can not include ":", whitespace, or other formatting.

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 dedicated telephone appliance). The MAC address may not be used if more than one user agent instance exists using 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 upper case with no punctuation for consistency across implementations. A prefix of "MAC:" should be added to the MAC address to form a

proper URN [RFC2141]. For example a device managed by sipuaconfig.example.com using its MAC address to form the device URI might look like: sip:MAC%3a00DF1E004CD0@sipuaconfig.example.com.

```
UHEX  =  DIGIT / %x41-46 ;uppercase A-F
MAC   =  %x4d.41.43 ; MAC in caps
mac-ident = MAC ":" 12UHEX
```

When the MAC address is not used in the device URI, UUID SHOULD be used.

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 UUID [RFC4122] is used as the user portion of the device URI. The same approach to defining a user agent instance ID as [I-D.ietf-sip-outbound] should be used. When constructing the instance id the implementer should also consider that a human may need to manually enter the instance id to provision the device in the profile delivery server (e.g. 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 addr-spec [RFC4122]. For example the instance ID: urn:uuid:f81d4fae-7ced-11d0-a765-00a0c91e6bf6 would be escaped to look as follows in a URI: sip:urn%3auuid%3af81d4fae-7ced-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.

7.13.2. User and Application URIs

The URI for the "user" and "application" type profiles is based upon the identity of the user. It is an administration policy on how user profile identities are assigned. Typically 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.

7.13.3. Local Network URIs

The URI for the "local-network" type profile is based upon the identity of the local network. When subscribing to the local network profile, the user part of the URI SHOULD be the same device ID used as the user part of the device profile SUBSCRIBE request URI defined in Section 7.13.1. 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 8.1. 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. When "network-user" is provided the profile delivery server SHOULD authenticate the user before providing the profile if additional privileges are granted. Example URI: sip:MAC%3a00DF1E004CD0@example.com

> The local network profile SUBSCRIBE request URI uses the device ID in the user part of the local network request URI so that every device in the network has a unique and constant request URI. Even though every device may get the same or similar local network profiles, the uniqueness of the URI provides an important capability. Having unique URIs allows the management of the local network to track user agents present in the network and consequently also manage resources such as bandwidth and port allocation.

8. 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.

8.1. 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 bootstrapping situation as the 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.

8.1.1. Discovery of Local Network URI

The "discovered" host for the "local-network" 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 [RFC2131](option 15 [RFC2132]). The local network profile subscription URI SHOULD not be remembered if the user agent moves from one local network to another other. The user agent should perform the local network discovery to construct the network profile subscription request URI every time it starts up or network connectivity is regained.

> For example: The user agent requested and received the local domain name via DHCP: airport.example.net. If the device ID is: MAC:00DF1E004CD0, the local network profile SUBSCRIBE request URI would look like: sip:MAC%3a00DF1E004CD0@airport.example.net. 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 sip:alice@example.com as the user's AOR. (Alice may have a prior arrangement with the local network operator giving her special privileges.):

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

8.1.2. 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 7.13.1 describes how to form the user part of the device profile SUBSCRIBE request URI used 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, WLAN 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 request URI, as described in Section 7.13.1, is to use the host and port of the outbound proxy discovered by the SIP DHCP option 120 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 ua-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: abc123efd456. 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:MAC%3aABC123EFD456@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 option 15, 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 ua-profile event, the next discovery mechanism

should be tried.

For example: The user agent requested and received the local domain name (option 15 [RFC2132]) in the DHCP response: boston.example.com. The device URI would look like: sip:MAC%3aABC123EFD456@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 by concatenating "sipuaconfig" and the local IP network domain (as provided via DHCP option 15 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 ua-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:MAC%3aABC123EFD456@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, a manual means for the

user to enter the host and port used to construct the SUBSCRIBE request URI MUST be provided by the user agent.

Two approaches to the manual discovery process are suggested. In the first approach using SIP, the user agent provides a means for entering the subscription host and port information for the request URI along with the user id and password to be used for authentication of the SUBSCRIBE request. With this approach the user agent begins with the enrollment process followed by the change notification and profile retrieve steps.

An alternative to the manual discovery using SIP, is to start with the retrieve process. The user agent provides a means of entering a HTTPS URI along with the user id and password to be used for authentication of the retrieval of the profile. The retrieved device profile may contain the properties for the SUBSCRIBE request URI and credentials to enroll and get change notification of profile changes. This approach bootstraps the process in a different step in the cycle, but uses the same profile framework. When the device starts with retrieval of the profile via HTTPS (instead of a SIP SUBSCRIBE

to the event package), the device MUST provide the Event header in the HTTPS request using the same format as described for the SUBSCRIBE request (see Section 7.2) . The Event header is necessary to determine which profile is requested as well as for providing specific information about the device.

Once a user agent has successfully discovered, enrolled and received a NOTIFY response with profile data or URI(s), the user agent should cache (i.e. store persistantly) the device profile SUBSCRIBE request URI (rather than reconstructing it as described in the discovery process every time the device is restarted) 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 7.6) with no profile data or 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.

8.1.3. Discovery of User and Application URI

The user's AOR may be preprovisioned or provided via SIM or flash key, etc. The device profile may define a default user and AOR. If provided in the device profile and a preprovisioned user AOR is not provided, the default user's AOR is used to subscribe to the "user" and "application" profiles. If not provided through the above two approaches, the AOR to be used for the "user" and "application" subscription URI, is "discovered" manually by prompting the user. The URI obtained in the discovery steps described above for the "user" and "application" profile subscriptions is stored persistantly in the device until explicitly reset or updated by the user or profile.

8.2. Enrollment with Profile Server

Enrollment is accomplished by subscribing to the event package described in Section 7. 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 7.5.

8.3. Notification of Profile Changes

The NOTIFY request in the ua-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 notification by including the content-type: "application/xcap-diff+xml" defined in [I-D.ietf-simple-xcap-diff] in the Accept header of the SUBSCRIBE request.

8.4. 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 7.5. The profile delivery server SHOULD secure the content of the profiles using one of the techniques described in Section 10. The user agent SHOULD make the new profiles effective in the timeframe described in Section 7.2.

The contents of the profiles SHOULD be cached (i.e. stored persistently) by the user agent. The cache should be used if the user agent is unable to successfully SUBSCRIBE or receive the NOTIFY providing the most recent profile. The cached profile should be replaced each time a profile is received in a NOTIFY or retrieved via content indirection. This it to avoid the situation where the content delivery server being not available, leaves the user agent non-functional. The user agent should verify that it has the latest profile content using the "hash" parameter defined in [I-D.ietf-sip-content-indirect-mech].

8.5. 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 URI 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].

9. IANA Considerations

There are several IANA considerations associated with this specification.

9.1. SIP Event Package

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

> Package Name: ua-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 dan.ietf AT SIPez DOT com
> New event header parameters: profile-type, vendor, model, version, effective-by, network-user

10. Security Considerations

Profiles may contain sensitive data such as user credentials and personal information. The protection of this data depends upon how the data is delivered. Some profiles may be safe to deliver without the need to protect the content. For example in some environments the local network profile may contain the list of codecs that are acceptable for use in the network and information on NAT traversal such as a STUN server to use. As the information in this example local network profile does not contain passwords or sensitive information it may be acceptable to provide it without authentication or confidentiality (encryption). We refer to these as non-confidential profiles. Non-confidential profiles require message integrity and profile server authentication, as described in Section 10.3. However any profiles that contain personal information, passwords or credentials (confidential profiles) require mutual authentication, confidentiality, and message integrity, and must follow the guidance provided in the next two subsections.

Profile specifications that define schemas MUST identify if they contain confidential data to indicate which of the security approaches describer here should be used.

The profile data is delivered in either the NOTIFY request or via the URI scheme indicated in the content indirection in the NOTIFY request. The security approach is different for these two delivery mechanisms.

Subscribers implementing this specification MUST implement either HTTP or HTTPS. Subscribers also MUST implement the hash verification scheme described in SIP content indirection [I-D.ietf-sip-content-indirect-mech]. SIP profile delivery servers MUST implement both HTTP and HTTPS, and SHOULD implement a SIP Authentication Service as described in the SIP Identity mechanism [I-D.ietf-sip-identity]. All SIP entities are already required to implement SIP Digest authentication [RFC3261].

10.1. Confidential Profile Content in NOTIFY Request

When the profile data is delivered directly in the NOTIFY request, the SUBSCRIBE request MUST be authenticated using the SIP Digest authentication mechanism. As the profile content is delivered in the resulting NOTIFY request to the subscription, authenticating the SUBSCRIBE is the only way to prevent unauthorized access to the profile data. To provide message integrity and confidentiality over the profile data, a direct TLS connection MUST be established for the SUBSCRIBE request. The device SHOULD authenticate the server via the TLS connection, which also provides a means of verifying that a direct TLS connection was used (e.g. The device may prompt the user to verify the Common Name in the server's certificate.). The server may challenge the device for its certificate, when establishing the TLS connection, to obtain the public key to use to S/MIME encode the NOTIFY request body containing the profile data. Because the device verified that it has a direct TLS connection by verifying the server's certificate and the server verified the identity of the device using Digest Authentication, the server can assume the certficate provided by the device is authenticated. The use of S/MIME in the NOTIFY request does not relieve the need to authenticate the SUBSCRIBE request using SIP Digest authentication. In this scenario S/MIME only provides message integrity and confidentiality of the content of the profile. If S/MIME is not used for the profile data in the NOTIFY request, the notifier MUST use the same direct TLS connection established by the device for the SUBSCRIBE request to send the notification. In this scenario the use of a user-specific ID and secret for Digest Authentication can be used to establish an association between the user ID and the device ID provided in the device profile SUBSCRIBE request.

10.2. Confidential Profile Content via Content Indirection

When the profile data is delivered via content indirection, authentication, integrity, confidentiality are all provided in the profile indirection retrieval scheme. When content indirection is used, the SUBSCRIBE request does not need to be authenticated. There is a TLS certificate approach and a Digest Authentication approach which may be used to provide the required security. The profile delivery server MUST support both of these methods. The device MUST support the Digest Authentication method to provide minimal interoperablity.

For the TLS certificate approach, the device requests the profile using HTTPS. To provide authentication, the server challenges the device for its certificate. The server obtains the user part of the SIP URI in the Subject Alternative Name field of the device's certificate. The user part of the SIP URI in the device's certificate is used as the device ID to authenticate if the device is authorized to retieive the specified profile. The device certificates chain of authorities MUST also be verified. This approach for providing security requires that the device ID and associated user are provisioned to the content indirection retreival.

For the Digest Authentication approach, HTTPS SHOULD be used to provide confidentiality of the profile data. HTTP Digest Authentication [RFC2617] MUST be used to authenticate and authorize the device to retrieve the profile. The shared secret used in the Digest Authentication is provided through out of band means to the device or user of the device. The same credentials used for SIP Digest authentication (e.g. authenication of SIP SUBSCRIBE and REGISTER requests) are used in the HTTPS request. Other URI schemes may be used, but are not defined in this document. A non-replayable authentication mechanism such as Digest authentication MUST be used for the content indirection URI scheme which provides the profile data (e.g. LDAP, HTTP and HTTPS all support Digest authentication). URI schemes which provide no authentication or only clear-text authentication SHOULD NOT be used for profile delivery as they are vulnerable to replay attacks (e.g. TFTP does not provide authentication).

Without a suitable authentication mechanism, the content indirection profile delivery URI scheme is susceptible to replay attacks. Even if the profile is symmetrically encrypted, if it can be retrieved through a replay attack, the encrypted profile

can be used for offline attacks to crack the encryption key.

The profile delivery scheme MUST use channel security such as TLS (e.g. HTTPS) to protect the content from being snooped in transport to the user agent. Mutual authentication using the client and server

certificates MAY be used to verify the authenticity of the user or device identity and the profile delivery server identity. The user agent SHOULD provide a mechanism for the user to approve the SUBSCRIBE server identity or provision the acceptable server identity through out of band means.

10.3. Integrity protection for non-confidential profiles

Even for non-confidential profiles, the subscriber MUST verify the authenticity of the profile delivery server, and MUST verify that the integrity of the profile data and content indirection URI, if one is provided. To meet these requirements in the SIP messaging the NOTIFY request MUST use a SIP Identity header [I-D.ietf-sip-identity], or S/MIME. If content is provided via redirection, the content indirection "hash" parameter MUST be included unless the profile data is delivered via a protocol which natively provides authentication and message integrity, such as HTTP or LDAP protected by TLS. The content retrieved via the content indirection URI MUST be integrity checked using the "hash" parameter.

For example, Alice subscribes to the local domain profile for paris.example.com. She receives a NOTIFY request which uses content indirection, including a "hash" parameter. Alice uses the Identity header from the NOTIFY to verify that the request came from paris.example.com and that the body was not modified. Then she fetches the content at the provided URI and verifies that the hash she calculates from the profile matches the hash provided in the SIP signaling.

10.4. Initial Enrollment Using a Manufacturer's Certificate

A UA with a manufacturer certificate can use this certificate for initial enrollment into the configuration framework. In order to safely deploy this scenario, the profile delivery server MUST maintain a list of enrolled devices and a separate list of devices which it expects to enroll.

When the device sends a subscription request to the notifier, the notifier extracts the device-id from the userpart of the Request URI and checks if the device is expected to enroll. If the device is expected, the notifier provides an https: URL to the subscriber and uses the SIP Identity mechanism to protect the integrity of this URL. This URL MUST contain enough information that the profile content server can correlate a request to this URL with the device-id that was in the subscription.

The subscriber then establishes a TLS connection to the profile content server and performs ordinary authentication of the server

certificate. During the TLS handshake, the profile content server requests the certificate of the subscriber. The subscriber provides its device certificate. Typically this certificate is created by the manufacturer of the device. If no client certificate is provided, the profile content server SHOULD return a 403 Forbidden response.

Next the profile content server checks the client certificate according to the following steps:

1. The client certificate MUST be valid, and MUST be rooted in a certificate authority that the administrator of the profile content server trusts to assert a valid "enrollment identity", for example a MAC address, serial number, or device-id.
2. The profile content server MUST verify that the device-id provided in the https: URL corresponds to the subject or subjectAltName of the client certificate, in an implementation specific way. For example, the profile content server could extract the MAC address from the device-id and the certificate and compare them. How device certificates are arranged is not standardized at the time of this writing, and is outside the scope of this document.
3. The profile content server SHOULD verify that the issuer of the certificate is expected and authorized to assert an enrollment identity for this type of device. In other words, the profile content server should not allow acme.example to assert an enrollment identity for a device manufactured by rival company widgets.example.
4. The profile content server MUST verify that the device referred to by the device-id is not already enrolled.
5. The profile content server MUST verify that the device referred

to by the device-id is expected to enroll at the current time. Typically, an administrator would configure a time-window of hours or days during which a new device can enroll.

If the profile content server successfully performs all these steps, it provides an initial device profile to the subscriber in the body of the HTTP response. This initial device profile MUST contain new credentials (for example, credentials for Digest authentication) that the subscriber can use for subsequent authentication. Integrity and confidentiality of the new profile is provided since the response is sent over a TLS channel. If one of the verification steps above fails, the profile content server sends a 403 Forbidden response.

Entities other than the profile content server do not accept manufacturer device certificates to secure ordinary communications, such as SIP TLS or SIP S/MIME.

11. Acknowledgements

Many thanks to those who contributed and commented on the many iterations of this document. Detailed input was provided by Jonathan Rosenberg from Cisco, Henning Schulzrinne from Columbia University, Cullen Jennings from Cisco, Rohan Mahy from Plantronics, Rich Schaaf from Pingtel, Volker Hilt from Bell Labs, Adam Roach of Estacado Systems, Hisham Khartabil from Telio, Henry Sinnreich from MCI, Martin Dolly from AT&T Labs, John Elwell from Siemens, Elliot Eichen and Robert Liao from Verizon, Dale Worley from Pingtel, Francois Audet from Nortel.

12. Change History

[[RFC Editor: Please remove this entire section upon publication as an RFC.]]

12.1. Changes from draft-ietf-sipping-config-framework-07.txt

Made XCAP informative reference. Removed "document" and "auid" event header parameters, and Usage of XCAP section to be put in

separate supplementary draft.
Fixed ABNF for network-user to be addr-spec only (not name-addr) and to be quoted as well.
Syncronized with XCAP path terminology. Removed XCAP path definition as it is already defined in XCAP.
User agent instance ID is now defined in output (not GRUU).
Clarified the rational for the network-user parameter.
Added text to suggest URIs for To and From fields.
Clearified use of network-user parameter.
Allow the use of the auid and document parameters per request by the OMA.

12.2. Changes from draft-ietf-sipping-config-framework-06.txt

Restructured the introduction and overview section to be more consistent with other Internet-Drafts.
Added additional clarifcation for the Digest Authentication and Certificate based authentication cases in the security section.
Added two use case scenarios with cross referencing to better illustrate how the framework works. Added better cross referencing in the overview section to help readers find where concepts and functionality is defined in the document.
Clarified the section on the use of XCAP. Changed the Event parameter "App-Id" to "auid". Made "auid" mutually exclusive to "document". "auid" is now only used with XCAP.

Local network subscription URI changed to <device-id>@<local-network> (was anonymous@<local-network>). Having a different request URI for each device allows the network management to track user agents and potentially manage bandwidth, port allocation, etc.
Changed event package name from sip-profile to ua-profile per discussion on the list and last IETF meeting.
Changed "local" profile type token to "local-network" per discussion on the list and last IETF meeting.
Simplified "Vendor", "Model", "Version" event header parameters to allow only quoted string values (previously allowed token as well).
Clarified use of the term cache.
Added references for ABNF constructs.
Numerous editorial changes. Thanks Dale!

12.3. Changes from draft-ietf-sipping-config-framework-05.txt

Made HTTP and HTTPS profile transport schemes mandatory in the profile delivery server. The subscribing device must implement HTTP or HTTPS as the profile transport scheme.
Rewrote the security considerations section.
Divided references into Normative and Informative.
Minor edits throughout.

12.4. 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 to 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.

12.5. 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-diff 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.

12.6. 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.

12.7. 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.

12.8. 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 ua-profile

Three high level security approaches are now specified.

12.9. 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 Estacado Systems for the great comments and input.

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

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

Minor edits

12.11. 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.

13. References

13.1. Normative References

[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-identity]
Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-06 (work in progress), October 2005.

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

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

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

[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.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

13.2. Informative References

[I-D.ietf-simple-xcap]
Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-08 (work in progress), October 2005.

[I-D.ietf-simple-xcap-diff]
Rosenberg, J., "An Extensible Markup Language (XML) Document Format for Indicating A Change in XML Configuration Access Protocol (XCAP) Resources", draft-ietf-simple-xcap-diff-02 (work in progress), October 2005.

[I-D.ietf-simple-xcap-list-usage]
Rosenberg, J., "Extensible Markup Language (XML) Formats for Representing Resource Lists", draft-ietf-simple-xcap-list-usage-05 (work in progress), February 2005.

[I-D.ietf-sip-outbound]
Jennings, C. and R. Mahy, "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)", draft-ietf-sip-outbound-01 (work in progress), October 2005.

[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 and Guidelines for Defining Session Initiation Protocol User Agent Profile Data Sets", draft-petrie-sipping-profile-datasets-03 (work in

progress), October 2005.

[I-D.sinnreich-sipdev-req]
Sinnreich, H., "SIP Telephony Device Requirements and Configuration", draft-sinnreich-sipdev-req-08 (work in progress), October 2005.

[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.

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

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

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

[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
SIPez LLC.
34 Robbins Rd
Arlington, MA 02476
US

Phone: "+1 617 273 4000
Email: dan.ietf AT SIPez DOT com
URI: http://www.SIPez.com/

Intellectual Property Statement

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

Copyright Statement

Acknowledgment

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