SIPPING   D. Petrie
Internet-Draft   SIPez LLC.
Intended status: Standards Track   S. Channabasappa, Ed.
Expires: September 2, 2007   CableLabs
                                                             March 2007

A Framework for Session Initiation Protocol User Agent Profile Delivery

draft-ietf-sipping-config-framework-10

Status of this Memo

Copyright Notice

Abstract

This document defines a framework to enable configuration of Session Initiation Protocol (SIP) User Agents in SIP deployments. The framework provides a means to deliver profile data that User Agents need to be functional, automatically and with minimal (preferably none) User and Administrative intervention. The framework describes how SIP User Agents can discover sources, request profiles and receive notifications related to profile modifications. As part of this framework, a new SIP event package is defined for notification of profile changes. The framework provides for multiple data retrieval options, without requiring or defining retrieval protocols. The framework does not include specification of the profile data within its scope.

Table of Contents

1. Introduction

SIP User Agents require configuration data to function properly. Examples include network, Client and user specific information. Ideally, this configuration process should be automatic and require minimal or no user intervention.

Many deployments of SIP User Agents require dynamic configuration and cannot rely on pre-configuration. This framework provides a standard means of providing dynamic configuration which simplifies deployments containing SIP User Agents from multiple vendors.

This framework also addresses modifications to profiles and the corresponding change notifications to the SIP User Agents using a new event package. However, the framework does not define the content or format of the actual profile data, leaving that to future standardization activities.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In addition, this document introduces and utilizes the following terms:

Client: software or hardware entity containing one or more SIP user agents.

Device: the terms 'Client' and 'Device' are used interchangeably within this framework.

Service Provider: a logical entity providing one or more services. This can refer to private enterprises or public entities.

Profile: configuration data set specific to an entity (for example, user, device, local network or other).

Profile Type: a particular category of Profile data (for example, User, Device, Local Network or other).

Profile Delivery Server (PDS): the source of a Profile, it is the logical collection of the Profile Notification Component (PNC) and the Profile Content Component(PCC).

Profile Notification Component (PNC): the logical component of a Profile Delivery Server that is responsible for enrolling Clients and providing profile notifications.

Profile Content Component (PCC): the logical component of a Profile Delivery Server that is responsible for storing, providing and accepting profile content.

Profile Discovery: discovery of a Profile Delivery Server (PDS) by the Client.

Profile Enrollment: process of enrolling with one or more Profile Delivery Server(s) by a Client.

Profile Notification: notification of a requested or changed profile by the PDS.

Profile Retrieval: retrieval of Profile data from a PDS by a Client.

Profile Change Upload: upload of profile data changes to one or more PDSs by authorized entities such as a Client

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.

Instance ID: text identifier globally unique across all Clients.

3. Overview

This section provides an overview of the configuration framework. It introduces the reference model and explains key concepts such as the Profile Life Cycle and the Profile types. The framework is presented in Section 5.

3.1. Reference Model

The design of the framework was the result of a careful analysis to identify the configuration needs of a wide range of SIP deployments. As such, the reference model provides for a great deal of flexibility, while breaking down the interactions to their basic forms which can be reused in many different scenarios.

In its simplest form, the reference model for the framework defines the interactions between the Profile Delivery Server(PDS) and the Client. The Client is a SIP UA which needs the profile data to effectively function in the network. The PDS is responsible for responding to Client requests and providing the profile data. The set of interactions between these entities is referred to as the Profile Life Cycle. This reference model is illustrated in the diagram below.

```
                                       +-------------------------+
    +----------+       Interactions       | Profile Delivery Server |
    | Client   |<===========================>|  +---+           +---+    |
    | (SIP UA)|    (Profile Life Cycle)    |  |PNC|           |PCC|    |
    +----------+                            |  +---+           +---+    |
                                       +-------------------------+

                              PNC = Profile Notification Component
                              PCC = Profile Content Component
```

Framework Reference Model

The PDS is subdivided into two logical components:

- o Profile Notification Component (PNC), responsible for enrolling Clients in Profile event subscriptions and providing Profile change notifications;
- o Profile Content Component (PCC), responsible for storing, providing access to, and accepting updates related to profile content.

SIP deployments vary considerably. To be effective, the configuration framework needs to consider a comprehensive set of scenarios that is representative of most deployments. The figure below provides a system level view of the device, user and Service Provider relationships that may be involved.

```
        --------
      /          \
     |   Service  |
     |  Provider  |            - > Provides 'Client'(e.g. allowed Users)
      \     Y    /                 & 'User'(such as Services) profile
        --------
            |         -----
            |       / Local \
            |      | Network |
            |      | Provider| - > Provides 'Local Network' profile
            |       \   Z   /      data (e.g. STUN Server Address)
            |         -----
            |         /
            |        /
      ===============
     ( Local Network )
      ===============
            |
            |
            |
        ---------
       | Client X|            - > Needs the 'Client' profile (from Y)
        ---------                  & 'local network' profile (from Z)
         /     \
        /       \
      ------   ------
     |User A| |User B|        - > Users need 'User' profile (from Y)
      ------   ------
```

Framework System Level Model

Based on the system level model, the following considerations are relevant.

Client connectivity:

- Clients can connect either directly to a Service Provider or via other local networks (for example, home network, Public Wi-Fi Hotspots, enterprise managed LAN, etc.);
- Local networks through which Clients connect may wish to provide their own configuration information particular to that specific network (for example, STUN server information, local Proxy, etc.) which is independent of the Service Provider (who provides services) or the particular User.

Service provider relationships:

- The local network provider (the network the Client connects to) and the Service Provider (that hosts the actual voice or other services) can often be different entities, with no administrative or business relationship to each other;
- There may be multiple different Service Providers involved, one for each service type a User subscribes to (telephony service, instant messaging, etc); this Framework does not specify explicit behavior in such a scenario, but it does not prohibit its usage either
- Each User accessing services via a Client may subscribe to different sets of services, from different Service Providers;

User-Client relationship:

- The relationship between Clients and Users can be many-to-many (for example, a particular UA instance may allow for many Users to obtain subscription services through it, and individual Users may have access to multiple different UA devices);
- Each User may have different preferences for use of services, and presentation of those services in the Client user interface;
- Each User may have different personal information applicable to use of the Client device, either as related to particular services, or independent of them.

The observations above show a need for a clear distinction between different Profile Types, based on the source and purpose of the configuration data contained, and a need for these profiles to be manageable by different PDSs. Accordingly, the framework identifies the following minimal Profile Types.

Local-Network Profile: refers to profile data as provided by the Local Network to which a Client is directly connected;

Device Profile: refers to profile data provided by the Service Provider or other entity which is specific to the particular Client;

User Profile: refers to profile data provided by the Service Provider or other entity which is specific to the particular User.

The definition of additional Profile Types and their usage is allowed, but is outside the scope of this document.

The remainder of this section provides more information on the two vital components of the framework: Profile Life Cycle and Profile Types.

3.2. Profile Life Cycle

Automated Profile delivery to Clients requires proactive behavior on the part of a Client. It also requires one or more PDSs which provide the profile data. Profile Delivery is usually initiated when the Client discovers PDSs and requests profile data. The profile data can be modified by the Client (for example, by a User) and subsequently uploaded to the PDS. Alternatively, the profile data can be modified by an authorized entity such as an administrative or user interface and the Client is notified through an event notification.

The specific functional steps involved in these interactions, collectively termed Profile Life Cycle, are as follows:

Profile Discovery: The process by which a Client finds PDS(s) capable of providing the Profiles it requires. This Framework defines multiple Profile Types which may be served by one or more PDSs.

Profile Enrollment: The process by which a Client makes itself known to a PDS. While enrolling, the Client provides identity information and requested Profile Type(s) for profile retrieval. It also subscribes for notification of profile changes. As a result of enrollment, the Client receives profile information (contents or content indirection information). Each Profile Type

requires a separate enrollment or SUBSCRIBE session.

Profile Notification: The process by which the PDS notifies the Client that either requested Profile contents are available, or the content of one or more of the Profiles has changed. If the content is provided indirectly, the Client may retrieve the profile from the specified URI upon receipt of the change notification.

Profile Retrieval: The process of retrieving the content for each of the Profiles requested by the Client.

Profile Change Upload: The process by which a Client or other entity (for example, configuration management server) pushes a change to Profile data to the PDS.

3.3. Data Model and Profile Types

As outlined previously, this framework defines three specific Profile Types. Additional extended profiles may also be defined. The Profile Types specified in this framework are:

Local Network Profile: Contains configuration data related to the Local Network to which a Client is directly connected, as required for the Client to operate effectively in that network. It is expected to be provided by a PDS in the Local Network (or proxied in some way).

Device Profile: Contains configuration data related to a specific Client, required for operation in the Service Provider's environment. It is expected to be provided by the Service Provider responsible for configuring the Client.

User Profile: Contains configuration data related to the specific User, as required to reflect that User's preferences and the particular services subscribed to. It is expected to be provided by Service Provider(s) responsible for maintaining the User's configuration data.

To function effectively, the Client should obtain all of the

necessary Profiles. Since each profile may potentially be served by a different source and the Client has no way of ascertaining that in advance, the framework requires the Client to discover the PDS sources independently and request the corresponding Profiles from each individually.

4. Use Cases

This section provides a small - non-comprehensive - set of representative use cases to further illustrate how this Framework can be utilized in SIP deployments.

For Security Considerations please refer to Section 9.

4.1. Client with different Data and SIP Service Providers

Description: Consider a user who obtains data (broadband) and SIP Services from two different Service Providers. For example, a user obtaining SIP services from a SIP Service Provider, via data connectivity provided through a WiFi hotspot or hotel network.

The following assumptions apply:

- For the sake of simplicity, the Client is assumed to be pre-configured with a) the domain name of the SIP Service Provider, b) the ability to generate a Client identifier (such as, based on MAC address) that can be used to request the device profile, and b) a user identity which can be used to request the user profile
- The Client is pre-configured to request local-network, Client and user profiles - in that order - to obtain information related to the local-network, itself and the pre-configured user
- The profile data provided upon request are based on data models that are comprehenisble by the Client, i.e. the Client understands the data models used for the creation of the profile data

The following diagram illustrates this use case and highlights the communications relevant to the framework specified in this document.

```
                    +-----------------+  +----------------------+
    +--------+      |  Data Service   |  | SIP Service Provider |
    | Client |      |     Provider    |  |                      |
    |(SIP UA)|      |                 |  |  SIP     PDS    PDS  |
    +--------+      |  DHCP     PDS   |  | PROXY (Client) (User)|
                    +-----------------+  +----------------------+
         |               |       |           |       |      |
     (A) |<==== DHCP ===>|       |           |       |      |
         |                       |           |       |      |
         |                       |           |       |      |
         |   SUBSCRIBE/NOTIFY    |           |       |      |
     (B) |<=== local-network ===>|           |       |      |
         |        profile
         |
         |   <<Profile Retrieval>>
         |
         |          SUBSCRIBE/NOTIFY         |       |      |
     (C) |<========= device profile ========>|<=====>|      |
         |                                   |       |      |
         |   <<Profile Retrieval>>
         |
         |
         |          SUBSCRIBE/NOTIFY         |              |
     (D) |<========== user profile  ========>|<============>|
         |                                   |              |
         |   <<Profile Retrieval>>
         |
```

The following is an explanation of the interactions in the diagram.

(A) Upon initialization, the Client obtains IP parameters (IP address, DNS) using DHCP (as an example)

(B) The Client proceeds to request the 'local-network' Profile Type. The PDS in the local network responds, allowing the Client to retrieve the local-network profile

(C) The Client then proceeds to request the 'device' Profile Type using the pre-configured SIP Service Provider's domain name. This request is received by a SIP Proxy in the SIP Service Provider's network. The request is then proxied to a relevant PDS within its network. The PDS responds to the request and provides profile retrieval information. The Client retrieves the Device Profile (this can contain information such as enabling or disabling usage, based on the subscription status)

(D) The Client then proceeds to request the 'User' Profile Type for the pre-configured User. This message is proxied to the same or different PDS (diagram assumes the latter) which responds with the profile retrieval information. The Client retrieves the User profile (this can contain information such as service profiles to be retrieved, based on the subscription). The Client then starts providing services.

4.2. Clients supporting multiple users from different Service Providers

Description: Consider a single Client (for example, Kiosk at an airport) that allows for multiple users to obtain services from a list of pre-configured SIP Service Providers.

The following assumptions apply:

- o The Client is provided and managed by SIP Service Provider A. It is not pre-configured with any User Identities, but offers an interactive User Interface to enter Service Provider and User information
- o SIP Service Provider A provides the local network connectivity, 'local-network' and 'device' profiles for the Client. The Service Provider also provides 'user' profiles for existing subscribers
- o SIP Service Provider B provides SIP services and has pre-existing agreements with SIP Service Provider A. This Service Provider also provides 'user' profiles for existing subscribers

The following diagram illustrates the use case and highlights the communications relevant to the framework specified in this document.

```
     User User
       A   B        +----------------------+  +----------------------+
    +--------+      | SIP Service Provider |  | SIP Service Provider |
    | Client |      |          A           |  |          B           |
    |(SIP UA)|      |                      |  |                      |
    +--------+      | DHCP    PROXY   PDS  |  |  PROXY        PDS    |
                    +----------------------+  +----------------------+
         |              |        |      |          |           |
     (A) |<====DHCP====>|        |      |          |           |
         |                       |      |          |           |
         |                       |      |          |           |
         |  SUBSCRIBE/NOTIFY     |      |          |           |
     (B) |<local-network profile>|<====>|          |           |
         |
         |   <<Profile Retrieval>>
         |
         |
         |  SUBSCRIBE/NOTIFY     |      |          |           |
     (C) |<== device profile ==> |<====>|          |           |
         |
         |   <<Profile Retrieval>>
         |
                      .
                      .
                      .
             [[User A attempts services]]


         |     SUBSCRIBE/NOTIFY  |      |          |           |
     (D) |<= user profile (A) => |<====>|          |           |
         |                       |      |          |           |
         |
         |   <<Profile Retrieval>>
                              .
                      .
                      .
                      .
             [[User B attempts services]]

         |
         |             SUBSCRIBE/NOTIFY            |           |
     (E) |<=========== user profile (B) ==========>|<=========>|
         |                                         |           |
         |   <<Profile Retrieval>>
         |
```

The following is an explanation of the interactions in the diagram.

(A) Upon initialization, the Client obtains IP parameters (IP address, DNS) using DHCP

(B) Once local IP connectivity is established and the SIP stack initialized, the Client proceeds to request the 'local-network' Profile Type. It receives a response from the PDS in Service Provider A's network (the local network). The Client retrieves the profile (this may contain useful information such as firewall port restrictions, available bandwidth etc)

(C) The Client then proceeds to request the 'device' Profile Type. It receives a response containing the profile retrieval from the PDS in Service Provider A's network. The Client retrieves the data provided in the Client Profile (this may provide data regarding Users such as the list of SIP Service Providers the Client can communicate with). The Client initializes the User interface for services.

(D) User A with a pre-existing subscription with Service Provider A attempts communication via the User Interface. This results in a prompt - and responses - for identification and authentication. The Client uses the provided information and communicates with Service Provider A. Once authenticated and authorized, it proceeds to request the 'User' Profile Type. The PDS responds with the profile retrieval information. The Client provides services to User A.

(E) At a different point in time, User B with a pre-existing subscription with Service Provider B attempts communication via the User Interface. This results in a prompt - and responses - for identification and authentication. Since Service Provider B is in the list of approved Service Provider, the Client uses the provided information and communicates with Service Provider B. Once authenticated and authorized, it proceeds to request the 'User' Profile Type. The PDS responds with the profile retrieval information. The Client provides services to User B.

It is to be noted that this Client may allow for exclusive or simultaneous access to both Users.

5. Profile Delivery Framework

This section details the framework requirements. The Profile Life Cycle (introduced in Section 3), is examined in further detail, with requirements that apply to the Client and the PDS. Unless explicitly enhanced or indicated by an implementing specification, the Client and the PDS MUST follow the Profile Life Cycle requirements stated in this section for all supported Profile Types.

A high-level representation of the framework is shown in the

following state diagram. Each of the specified Profile Types is retrieved individually, in the specified order (see below), until all needed Profiles have been received. For each retrieved Profile, the Client then awaits any Change Notifications

```
                          ---------------
                         /    Client     \
                         \ Initialization/
                          ---------------
                                |
                                | Completes IP initialization;
                                | Initializes SIP stack
                                |
                                V
                          --------------  YES    ---------------
               ________\ / All profiles?\_____\ | Await Change  |
              |        / \ retrieved?   /     / | Notifications |
              |           --------------         ---------------
              |                 |
              |                 | NO; attempt
              |                 | Profile Request
              |                 | in specified order
              |                 |
              |                 V
              |            ------------
               ___________/  Profile   \
                          \ Life Cycle /
                           ------------
```

Framework state diagram

The Profile Life Cycle, within each Profile Type, is illustrated further as in the state diagram below.

```
                           -------------   All methods   --------
                ________\ /   Profile   \ ............\ / Error  \
               |        / \  Discovery  /   exhausted / \Handling/
               |           -------------                 --------
               |                 |
               |                 |
               |   Try           | Send request
               | alternate       | for Profile
               | method(s)       | Enrollment
               |                 |
               |                 |
               |                 V
               | FAILURE   ------------
               |__________/   Profile   \
               ^          \  Enrollment /
               ^           ------------
               |                 |
               | FAILURE         |
               |                 |
               |                 |
               |                 V
               |  Timeout  -------------
                _________ /   Profile   \
                          \ Notification/
                           -------------
                                 |
                                 |
                                 |SUCCESS
                                 |
                                 V
                              -------    Failure    ---------
                            / Profile \ _________\ /  Error  \
                            \Retrieval/          / \ Handling/
                              -------               ---------
                                 .
                                 .     If allowed
                                 . by Profile Retrieval
            ------               .     Framework
           (Client)  --          V
            ------      \  -------------
                          /Profile Change\
                          \    Upload    /
          ----------    /  -------------
         {Authorized}--
         {  Entity  }
          ----------
```

The Profile Life Cycle is initiated when the Client starts the 'Profile Discovery' process for a particular Profile Type. Discovery leads to transmission of a request for 'Profile Enrollment'. Successful enrollment leads to 'Profile Notification'. Successful initial notification results in 'Profile Retrieval' (either as data within the notification or using content indirection). 'Profile Change Upload' can be initiated by any authorized entity (examples include Clients and administrative interfaces).

'Profile Discovery' and 'Profile Enrollment' are closely coupled. Failure to enroll (for example, no response is received for the SUBSCRIBE) results in alternate 'Profile Discovery' methods until success is achieved or all the methods are exhausted (resulting in error handling). Simiarly, the initial 'Profile Notification' is closely coupled to enrollment. Failure to receive the initial notification also results in alternate discovery methods.

'Profile Retrieval' is accomplished using the contents of the Profile Notification. This can either contain the profile data or a content indirection method to achieve it.

The Profile Life Cycle is the same for all the Profile Types, but there are different requirements in each step based on the Profile Types. This framework defines three Profile Types and an order that MUST be followed by the Client in requesting them (when it retrieves two or more of the defined Profile Types), as follows:

- local-network
- device
- user

The sub-sections that follow specify the Profile Life Cycle details, with specific requirements based on each Profile Type.

5.1. Profile Discovery

The first step to obtaining a profile is PDS Discovery. This is accomplished by creating a profile subscription using the Event Package described in Section 6, and preparing for Profile Enrollment.

Each Profile Type requires its own subscription and based on the entity requesting it, presents certain unique requirements (for example, the Client identifier is provided for the Device Profile Type where as the User identifier is provided for the User Profile Type). Further, the Profile Types are aimed at different PDSs and hence are identified differently (for example, the local-network is identified by the local domain name where as the Service Provider is

identified based on the Service Provider's domain name). Some of this information can be obtained in multiple ways (such as local domain information that can be configured statically or dynamically) and the Client may have to try different information sources to obtain the required information (for example, dynamic configuration can override statically configured information). Based on these considerations, the framework defines different rules for obtaining and presenting the information for each Profile Type. Additionally, when more than one information source is possible for the information, it is presented as well. This is highlighted in the following sub-sections.

5.1.1. SIP SUBSCRIBE for the Local-Network Profile Type

Before attempting to create a SIP SUBSCRIBE requesting the Local-Network Profile, the Client MUST have established local network connectivity. It MUST also have knowledge of the local network domain either via static configuration or dynamic discovery (using DHCP [RFC2131], option 15 [RFC2132]). The following requirements apply:

- the user part of the Request URI MUST NOT be provided. The host and port part of the Request URI MUST be set to the local network domain
- the user part of the "From" field MUST be the Identifier that the Client will use to request the 'device' Profile Type
- the host and port part of the "From" field MUST be set to the local network domain
- a user AOR, if known to the Client MUST be provided in the "network-user" event header parameter, unless privacy requirements prohibit its use (this is useful if the user has privileges in the local network beyond those of the default user)

For example: If the Client requested and received the local domain name via DHCP to be: airport.example.net, then the Local-Network Profile SUBSCRIBE Request URI would look like:

```
sip:airport.example.net
```

The Event header would look like the following if the Client 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"
```

The Local-Network Profile SUBSCRIBE Request URI does not have a user part so that the URI is distinct between the "local" and "device" URIs when the domain is the same for the two. This provides a means of routing to the appropriate PDS in domains where they are distinct servers. The From field 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 From field. Even though every client may get the same (or similar) Local-Network Profile, the uniqueness of the From field provides an important capability. Having unique From fields 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.

For example: If the Client requested and received the local domain name via DHCP to be: airport.example.net and the device ID is: MAC: 00DF1E004CD0, the From field would contain:

sip:MAC%3a00DF1E004CD0@airport.example.net

5.1.2. SIP SUBSCRIBE for the Device Profile Type

The Device Profile Type allows the Service Provider managing a Client to provide Client-specific configuration information. To enable this, the Request URI needs to identify the Client and the PDS domain within which it is recognizable. Accordingly, this Framework presents the following requirements for the formation of a Subscription Request URI to request the "device" Profile Type

- the user portion of the Request URI MUST be set to a unique Client Identifier
- the host and port portion of the Request URI MUST be set to the PDS domain

The following sub-sections explain identification of - and the requirements related to - the Client Identifier and the PDS domain discovery.

5.1.2.1. Client Identifier

The Client profile could be specific to each Client in a SIP deployment (for example, vendor/model) or shared across Client types (for example, based on services and service tiers). Further, the same Client might be provided different configuration profiles based on deployment models. Client Identifiers play a significant role in ensuring delivery of the correct profile and hence need to be unique within a PDS domain to support the various deployment models.

This Framework requires that Client Identifiers MUST be unique and persistent over the lifetime of a Client. Client Identifier representations auto-generated by Clients SHOULD be based on MAC address or UUID ([RFC4122]) based representations. A Client may use alternate Client identifiers (for example, SIP URIs) obtained via pre-configuration or dynamic configuration (for example, Client profile).

If a MAC address is used, the following requirements apply:

- o the Client identifier MUST be formatted as the characters "MAC:" followed by a twelve digit hexadecimal upper case representation of the MAC address to form a proper URN ([RFC2141]). The MAC address representation MUST NOT include visual separators such as colons and whitespaces. The representation is denoted using the following ABNF syntax

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

- o the MAC address MUST only be used to represent a single Client. It MUST NOT be used if more than one Client can potentially use the same MAC Address (for example, multiple software Clients on a single platform). In such cases, the UUID representation SHOULD be used

If a UUID is used, the following requirements MUST apply:

- o the same approach to defining a user agent Instance ID as [RFC4122] MUST be used
- o when the URN is used as the user part of the URI, it MUST be URL escaped

  The colon (":") is not a legal character (without being escaped) in the user part of an addr-spec ([RFC4122]).
  For example the instance ID:
  urn:uuid:f81d4fae-7ced-11d0-a765-00a0c91e6bf6@example.com
  would be escaped to look as follows in a URI:
  sip:urn%3auuid%3af81d4fae-7ced-11d0-a765-00a0c91e6bf6@
  example.com

  The ABNF for the UUID representation is provided in [RFC4122]

5.1.2.2. PDS Domain Discovery

A Client needs to identify the PDS domain to form the host and port part of the Request URI. Ideally, this information should be obtained via a single method. However, support for various deployment models implies multiple Client environments (for example,

residential routers, enterprise LANs, WLAN hotspots, dialup modem etc) and presents hurdles to specifying a single method (for example, if a Client is always in the SIP Service Provider's network one could use DHCP). To accommodate multiple deployment scenarios, the framework specified in this document presents multiple approaches.

Clients MUST follow the procedures specified below in the order presented, unless exceptions are made by Client manufacturers or Service Providers who may provide an option for the user to choose the order (to suit specific deployment models, for example).

1. Service Provider pre-configuration

   The Client MAY be pre-configured with information that can be utilized to identify the host and port of the Request URI. The information can be provided - as examples - when the Client is manufactured, by using Service Provider entities (flash card, SIM card) or via a Service Provider specific method (for example, information or methods that lead to self subscription). If the Client is specified to utilize this approach, it MUST attempt to do so before trying other methods. The details of how this is accomplished are beyond the scope of this document.

2. IP Configuration

   If pre-configuration is not an option, or not available, IP configuration MUST be utilized to try and obtain information that can help with identification of the host and port for the Request URI. The framework defines the following methods within this procedure to accomplish this. Clients MUST follow the methods defined, in the order specified, i.e. if the first option cannot be accomplished or results in a failure, then next method is tried. Failure of a specific method is indicated when the Client cannot successfully complete Profile Enrollment.

   2a. DHCP option 120:

      Clients that support DHCP MUST attempt to obtain the host and port of the outbound proxy during the DHCP process, using the SIP DHCP option 120 [RFC3361] and use these as the host and port part of the request URI.

      For example, a MAC based Client Identifier with a DHCP option 120 indicating example.com, the Request URI would be constructed as sip:MAC%3aABC123EFD456@example.com

2b. Local IP Network Domain:

- Clients that support DHCP MUST attempt to obtain the local IP network domain during the DHCP process, using DHCP option 15 and use these as the host and port part of the request URI using the technique specificed in [RFC3263]

  + For example, a MAC based Client Identifier with a DHCP option 15 indicating local.example.com, the Request URI would be constructed as sip:MAC%3aABC123EFD456@local.example.com

- If the local IP network domain is available (previous method), but the usage of the local IP Network domain results in a failure, the Client MUST use the local IP network domain, prefixing it using the label "sipuaconfig."

  + For example, a MAC based Client Identifier with a DHCP option 15 indicating local.example.com, the Request URI would be constructed as sip:MAC%3aABC123EFD456@sipuaconfig.local.example.com

3. Manual

If pre-configuration and IP Configuration are not options or result in failures, the Client SHOULD provide a means for the user to present information that may help with the retrieval process. Exceptions to this requirement MAY include Clients with no user interface appropriate for such entry.

This framework provides the following alternatives which can be considered individually or together, in any order.

Service Provider PDS information: The user SHOULD be allowed to present the host and port information which can help with the creation of the Subscription URI to locate a PDS capable of providing the profile.

Service Provider Configuration Server information The user MAY be allowed to present information pertaining to a configuration server that provides the Device Profile, not using a PDS as defined in this framework. This framework specifies one such possible process in Section 5.6.1.

5.1.3. SIP SUBSCRIBE for the User Profile Type

The User Profile allows the responsible SIP Service Provider to provide user-specific configuration. This is based on the User's Identity that is usually known in the network (for example, associated with a subscription). Similar to the profiles provided to Clients, the content and propagation of User Profiles may partake differently, based on deployment scenarios (for example, users belonging to the same subscription might - or might not - be provided the same profile). However, each User is uniquely identified in a SIP Service Provider's network using an Address Of Record (AOR). Clients implementing this framework MUST use the User's AOR to populate the Request URI.

A Client MAY obtain the User's AOR using various methods such as pre-configuration, via the Device Profile or dynamically via a User Interface.

5.1.4. Caching of SIP Subscription URIs

Creation of Subscription URIs is vital for successful Profile Enrollment, required for Profile Notification and ultimately Profile Retrieval. Further - unlike the User Profile - Local-Network and Device Profiles are expected to be requested based on discovered information (for example, domain name discovered via DHCP). These Profile Types have different goals and hence, caching of the Subscription URI should be carefully considered.

The Local-Network Profile Type is aimed at obtaining information from the local network. The local network can change across Client initializations (for example, User moves the Client from a home network to a workplace LAN). Thus, the Client SHOULD NOT remember local-network profile subscription URIs across initializations. The Client SHOULD re-create the Subscription URI every time it moves to a new network or gets re-initialized. Exceptions may be cases where the Client can unambiguously determine changes to the local network.

The Device Profile Type is aimed at obtaining information from the SIP Service Provider managing the Client. Once established, the Service Provider does not change often (an example of an exception would be the re-use of Clients across Service Providers). However, if the discovery process is used, the Client can only be sure of having reached the Service Provider upon successful Profile Enrollment and Profile Notification. Thus, the Client SHOULD cache the Subscription URI for the Device Profile. When cached, the Client should use the cached Subscription URI upon a reset. Exceptions include cases where the Client identifier has changed (for example, new network card with a new MAC address), Service Provider

information has changed (for example, user initiates change) or the Client cannot obtain its profile using the Subscription URI.

> Clients SHOULD NOT cache the Subscription URI for the Device Profile Type until successful Profile Notification. The reason for this is that a PDS may send 202 responses to SUBSCRIBE requests and NOTIFY responses to unknown Clients (see Section 6.6) with no profile data or URIs. Thus, successful Profile Notification is the only sure way to know that the Subscription URI is valid.

5.2. Profile Enrollment

Clients implementing the framework specified in this document are required to perform Profile Enrollment prior to Profile Retrieval (the only exception is noted in Section 5.6.1). Enrollment for a specific profile happens once the specific Subscription URI is formed and is accomplished using the Event Package specified.

Thus, a Client requesting a Profile Type specified in this document - and is successful in forming a Subscription URI - MUST enroll using the event package defined, and as specified, in this framework (see Section 6) . The following requirements apply:

- the Client MUST cater to the Event Package requirements specified in Section 6.2 (for example, indicate the Profile Type being requested in the profile-type parameter)
- the Client MUST use the Subscription URI pertaining to the Profile Type being requested, as specified in Section 5.1

The SIP infrastructure receiving such requests is expected to relay and process profile enrollment requests. When a Profile Enrollment request is received by a PDS, it SHOULD accept and respond to any profile requests. Exceptions are when Service Provider policy prevents such a response (for example, requesting entity is unknown).

Successful Profile Enrollment involves the following

- Acceptance of the SUBSCRIBE request by a PDS (indicated via a 200 response)
- Receipt of an initial Profile Notification within the timeouts as specified in [RFC3265]

A Client SHOULD follow suitable BackOff and Retry mechanisms if a successful Profile Enrollment does not happen within the expected period.

5.3. Profile Notification

Successful Profile Enrollment leads to Profile Notification. This serves two purposes a) initial profile content following successful Profile Enrollment and b) notification to the Client of modifications to profile content. Failure to receive the initial NOTIFY following a successful enrollment MUST be treated the same as a failed enrollment. Whenever a profile is changed, the PDS MUST NOTIFY all Clients currently subscribed to the changed profile.

For NOTIFY content please refer to Section 6.5.

5.4. Profile Retrieval

Upon successful Profile Enrollment and Profile Notification, the Client can retrieve the documents pertaining to the requested profile directly or via the URI(s) provided in the NOTIFY request as specified in Section 6.5.

The following requirements hold good:

- the PDS SHOULD secure the content of the profiles using one of the techniques described in Section 9
- the Client MUST make the new profiles effective within the specified timeframe, as described in Section 6.2
- if content indirection is used, the Client SHOULD verify that it has the latest profile content using the "hash" parameter defined in [RFC4483]
- the Client SHOULD cache (i.e. store persistently) the contents of retrieved profiles, until overridden by subsequent Profile Notifications (this avoids situations where a PDS is unavailable, leaving the Client without required configuration)

5.5. Profile Change Upload

Configuration Profiles can change over time. This can be initiated by various entities (for example, via the Client, back-office components, end-user web interfaces into configuration servers, etc) and for various reasons (such as, change in user preferences, modifications to services, enterprise-imposed common features or restrictions). This framework allows for such changes to be communicated to the PDS, using the term Profile Change Upload.

Any changes to a Profile as a result of Profile Change Upload MUST result in a Profile Notification to all enrolled clients for that Profile, if any.

Definition of specific mechanisms for Profile Change Upload are out of scope of this document.

5.6. Additional Considerations

This section provides a special case for retrieval of the Device Profile and highlights considerations and requirements on external entities such as Profile Data Frameworks.

5.6.1. Manual retrieval of the Device Profile

At a minimum, a Client requires the Device Profile to be able to function effectively. However, the methods specified in this document many fail to provide a Client with a profile. To illustrate with an example, consider the case of a Client that finds itself behind a local network which does not provide information about DNS servers in the network (for example, misconfigured home network). In such cases, it would be beneficial to employ an alternative means to obtain the profile information (for example, resolvable DNS Servers could be part of the Client profile). While this specification recommends that such a method be made available, it also specifies one such option using HTTP that is described in this sub-section. Clients expected to encounter scenarios where Client profile retrieval can be hindered may employ the specified - or any alternative - process.

The method being described involves the Client to utilize a HTTPS URI (and any required credentials) based on either pre-configuration or manual entry by the User (in cases where such an interface is possible). This can lead to the retrieval of the Device Profile which may contain the properties for the SUBSCRIBE Request URI and credentials for Profile Enrollment and Profile Notification. This approach bootstraps the process in a different step in the cycle, but uses the same framework.

Further, this document defines a new HTTP request header "Event". The syntax of the HTTP Event header is the same as the SIP Event header defined in this document. Similar to the SIP Event header the purpose of the HTTP Event header is to define the content of the state information to be retrieved. In particular, the state information is the Device, User or Local-Network Profile for the Client. The SIP Event header parameters for this event package ("profile-type", "vendor", "model", "version") are also mandatory for the HTTP Event header as they are used to provide information as to what profile type is requested along with information about the device which may impact the contents of the profile.When the Client starts with retrieval of the profile via HTTPS (instead of a SIP SUBSCRIBE to the event package), the device MUST provide the Event

header defined.

5.6.2. Client Types

The examples in this framework tend to associate Clients with entities that are accessible to end-users. However, this is not necessarily the only type of Client that can utilize the specified Framework. Clients can be entities such as User Interfaces (that allow for Client Configuration), entities in the network that do not directly communicate with any Users (for example, Service Provider deployed gateways) or elements in the Service Provider's network (for example, SIP servers).

5.6.3. Profile Data

This framework does not specify the contents for any Profile Type. Follow-on standardization activities can address profile contents. However, it makes the following assumptions and recommendations:

- When the Client receives multiple profiles, the contents of each Profile Type will only contain data relevant to the entity it represents. As an example, consider a Client that obtains all the defined profiles. Information pertaining to the local network is contained in the 'local-network' profile and not the'user' profile. This does not preclude relevant data about a different entity from being included in a Profile Type, for example, the 'device' Profile Type may contain information about the Users allowed to access services via the Client. A profile may also contain starting information to obtain subsequent Profiles
- Data overlap SHOULD be avoided across Profile Types, unless necessary. If data overlap is present, prioritization of the data is left to data definitions. As an example, the Device Profile may contain the list of codecs to be used by the Client and the User Profile (for a User on the Client) may contain the codecs preferred by the User. Thus, the same data (usable codecs) is present in two profiles. However, the data definitions may indicate that to function effectively, any codec chosen for communication needs to be present in both the profiles.

5.6.4. Profile Data Frameworks

This framework specified in this document does not address profile data representation, storage or retrieval protocols. It assumes that the PDS has a PCC based on existing or other Profile Data Frameworks, for example, XCAP ([I-D.ietf-simple-xcap]).

While it does not impose vast constraints on any such framework, it

does allow for the propagation of profile content to PDS (specifically the PCC). Thus, Profile Data or Retrieval frameworks used in conjunction with this framework MAY consider techniques for propagating incremental, atomic changes to the PDS. For example, a means for propagating changes to a PDS is defined in XCAP ([I-D.ietf-simple-xcap]).

5.6.5. Additional Profile Types

This document specifies three profile types: local-network, device and user. However, there may be use cases for additional profile types. For example, Profile Types for application specific profile data. Definition of such additional Profile Types is not prohibited, but considered out of scope for this document.

6. Event Package Definition

The framework specified in this document proposes and specifies a new SIP Event Package as allowed by [RFC3265]. The purpose is to allow for Clients to subscribe to specific Profile Types with PDSs and for the PDSs to notify the Clients with - or pointers to - profile data.

The requirements specified in [RFC3265] apply to this package. The following sub-sections specify the Event Package description and the associated requirements. The framework requirements are defined in Section 5.

6.1. 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].

6.2. 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 following rules apply:

o All the new parameters, with the exception of the "effective-by" parameter MUST only be used in SUBSCRIBE requests and ignored if they appear in NOTIFY requests

- The "effective-by" parameter is for use in NOTIFY requests only and MUST be ignored if it appears in SUBSCRIBE requests

The semantics of these new parameters are specified in the following sub-sections.

6.2.1. profile-type

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. This document defines three logical types of profiles and their token names. They are as follows:

local-network Specifying "local-network" type profile indicates the desire for profile data (URI when content indirection is used) specific to the local network.

device 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.

user 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.

The "profile-type" is identified is identified in the Event header parameter: profile-type. 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 syntax definition using ABNF, EQUAL and token are defined in [RFC3261]. It is to be noted that additional Profile Types may be defined in subsequent documents.

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

The "device", "user" or "local-network" token in the profile-type parameter may represent a class or set of profile properties. Follow-on standards defining specific profile contents may find it 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.

6.2.2. vendor, model and version

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 (for example, example.com) as the value of the "vendor" parameter so that it is known to be unique. These parameters are useful to the PDS 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 PDS the ability to compensate for or take advantage of the differences. In the following ABNF defining the syntax, EQUAL and quoted-string are defined in [RFC3261].

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

6.2.3. network-user

The "network-user" parameter MUST be set when subscribing for "local-network" profiles if it is known, unless the Client is provisioned to preserve privacy within the local network. This allows the Client to indicate a user who may have special privileges in the local network that impact the contents of the "local-network" profile. It MAY also be provided in a subscription for a "device" profile. In such cases the Client is requesting the PDS to recognize the indicated user as the default user for itself.

The Notifier SHOULD authenticate the subscriber to verify the resource identifier in the "network-user" parameter, if the profile provided is specific to the user (for example, granting policies or privileges beyond those of a default 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
```

6.2.4. effective-by parameter

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 PDS 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
```

6.2.5. Summary of event parameters

The following are example Event headers which may occur in SUBSCRIBE requests. These examples are not intended to be complete SUBSCRIBE requests.

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

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

The following are example Event headers which may occur in NOTIFY requests. These example headers are not intended to be complete SUBSCRIBE requests.

```
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 three Profile Types:

profile-type	device	user	local-network
vendor	m	m	m
model	m	m	m
version	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 three Profile Types:

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

6.3. SUBSCRIBE Bodies

This package defines no use of the SUBSCRIBE request body. If present, it MUST be ignored.

Future enhancements to the framework may specify a use for the SUBSCRIBE request body (for example,, mechanisms using etags to minimize Profile Notifications to Clients with current profile versions).

6.4. Subscription Duration

The duration of a subscription is specific to SIP deployments and no specific recommendation is made by this Event Package. If absent, a value of 86400 seconds is RECOMMENDED since the presence (or absence) of a Client subscription is not time critical to the regular functioning of the PDS.

It is to be noted that a one-time fetch of a profile can be accomplished by setting the 'Expires' parameter to a value of Zero, as specified in [RFC3265].

6.5. NOTIFY Bodies

The framework specifying the Event Package allows for the NOTIFY body to contain the profile data or a pointer to the profile data using content direction. The framework does not define any profile data and delegates specification of utilized MIME types Profile Data Frameworks. For profile data delivered via content indirection, the following apply:

- the Content-ID MIME header, as described in [RFC4483] MUST be used for each Profile document URI
- at a minimum, the "http:" and "https:" URI schemes MUST be supported; other URI schemas MAY be supported based on the Profile Data Frameworks (examples include FTP [RFC0959], TFTP [RFC3617],HTTP [RFC2616], HTTPS [RFC2818], LDAP [RFC3377], XCAP [I-D.ietf-simple-xcap], XCAP-DIFF [I-D.ietf-simple-xcap-diff])

The NOTIFY body SHOULD include a MIME type specified in the 'Accept' header of the SUBSCRIBE. Further, if the Accept header of the SUBSCRIBE included the MIME type message/external-body (indicating support for content indirection) the content indirection SHOULD be used in the NOTIFY body for providing the profiles. If none are specified, the Profile Data frameworks are responsible for, and MUST specify, the MIME type to be assumed.

6.6. Notifier Processing of SUBSCRIBE Requests

A successful SUBSCRIBE request results in a NOTIFY with either profile contents or a pointer to it (via Content Indirection). If the NOTIFY is expected to contain profile contents or the Notifier is unsure, the SUBSCRIBE SHOULD be either authenticated or transmitted over an integrity protected SIP communication channels. Exceptions to authenticating such SUBSCRIBEs include cases where the identity of the Subscriber is unknown and the Notifier is configured to accept such requests.

The Notifier MAY also authenticate SUBSCRIBE messages even if the NOTIFY is expected to only contain a pointer to profile data. Securing data sent via Content Indirection is covered in Section 9.

If the Profile Type indicated in the "profile-type" Event header parameter is unavailable or the Notifier is configured not to provide it, the Notifier SHOULD return a 404 response to the SUBSCRIBE request. If the specific user or Client is unknown, the Notifier MAY either accept or reject the subscription.

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 Client indicated in the Subscription request. However, the Notifier SHOULD authenticate the user indicated before making such a change.

6.7. Notifier Generation of NOTIFY Requests

As specified in [RFC3265], the Notifier MUST always send a NOTIFY request upon accepting a subscription. If the Client or User is unknown and the Notifier choose to accept the subscription, the Notifier MAY either respond with profile data (for example, default profile data) or provide no profile information (i.e. no body or content indirection).

If the URI in the SUBSCRIBE request is a known identity and the requested profile information is available (i.e. as specified in the profile-type parameter of the Event header), the Notifier SHOULD send a NOTIFY with profile data. Profile data MAY be sent as profile contents or via Content Indirection (if the content indirection MIME type was included in the Accept header). To allow for Content Indirection, the Subscriber MUST support the "http:" or "https:" URI schemas. If the Subscriber wishes to support alternative URI schemas it MUST be indicated in the "schemes" Contact header field parameter as defined in [RFC4483]. If the subscriber does not specify the URI scheme, the Notifier may use either "http:" or "https:".

The Notifier MAY specify when the new profiles must be made effective by the Subscriber by specifying a maximum time in seconds (zero or more) in the "effective-by" event header parameter.

If the SUBSCRIBE was received over an integrity protected SIP communications channel, the Notifier SHOULD send the NOTIFY over the same channel.

6.8. Subscriber Processing of NOTIFY Requests

A Subscriber to this event package MUST adhere to the NOTIFY request processing behavior specified in [RFC3265]. If the Notifier indicated an effective time (using the "effective-by" Event Header parameter), it SHOULD attempt to make the profiles effective within the specified time. Exceptions include deployments that prohibit such behavior in certain cases (for example, emergency sessions are in progress). When profile data cannot be applied within the recommended timeframe and this affects Client behavior, any actions to be taken SHOULD be defined by the profile data definitions. By default, the Subscriber is RECOMMENDED to make the profiles effective as soon as possible.

The Subscriber MUST always support "http:" or "https:" and be prepared to accept NOTIFY messages with those URI schemas.The subscriber MUST also be prepared to receive a NOTIFY request with no body. The subscriber MUST NOT reject the NOTIFY request with no body. The subscription dialog MUST NOT be terminated by a NOTIFY with no body.

6.9. Handling of Forked Requests

This Event package allows the creation of only one dialog as a result of an initial SUBSCRIBE request as described in section 4.4.9 of [RFC3265]. It does not support the creation of multiple subscriptions using forked SUBSCRIBE requests.

6.10. Rate of Notifications

The rate of notifications for the profiles in this framework is deployment specific, but expected to be infrequent. Hence, the Event Package specification does not specify a throttling or minimum period between NOTIFY requests

6.11. State Agents

State agents are not applicable to this Event Package.

7. Examples

This section provides examples along with sample SIP message bodies relevant to this framework. Both the examples are derived from a snapshot of Section 4.1, specifically the request for the Device Profile. The examples are purely informative and in case of conflicts with the framework or protocols used for illustration, the latter should be deemed normative.

7.1. Example 1: Client requesting profile

This example illustrates the detailed message flows between the Client and the SIP Service Provider's network for requesting and retrieving the profile (the flow uses the Device Profile as an example).

The following are assumed for this example:

- Client is assumed to have established local network connectivity; NAT and Firewall considerations are assumed to have been addressed by the SIP Service Provider
- examples are a snapshot only and do not illustrate all the interactions between the Client and the Service Provider's network (and none between the entities in the SIP Service Provider's network)
- All SIP communication with the SIP Service Provider happens via a SIP Proxy
- HTTP is assumed to be the Profile Data method used (any suitable alternative can be used as well)
- TLS is assumed to be the protocol for securing the Profile Retrieval (any other suitable protocol can be employed); authentication and security requirements are not addressed

The flow diagram and an explanation of the messages follow.

```
                                      +----------------------+
    +--------+                        | SIP Service Provider |
    | Client |                        |                      |
    |(SIP UA)|                        |  SIP     PDS   HTTP  |
    +--------+                        | PROXY         Server |
                                      |                      |
                                      +----------------------+
         |                                |       |      |
         |                                |       |      |
         |          SUBSCRIBE             |       |      |
   (SReq)|--------device profile--------->|       |      |
         |                                |------>|      |
         |                                |200 OK |      |
         |            200 OK              |<------|      |
   (SRes)|<-------------------------------|       |      |
         |                                |       |      |
         |                                | NOTIFY|      |
         |    NOTIFY (Content Indirection)|<------|      |
   (NTFY)|<-------------------------------|       |      |
         |            200 OK              |       |      |
   (NRes)|------------------------------->|200 OK |      |
         |                                |------>|      |
         |                                               |
         |                                               |
         |                                               |
         |<<<<<<<<<<<<<  TLS establishment  >>>>>>>>>>>>>|
         |                                               |
         |                HTTP Request                   |
   (XReq)|---------------------------------------------->|
         |                                               |
         |                HTTP Response                  |
   (XRes)|<----------------------------------------------|
         |                                               |
```

(SReq)

the Client transmits a request for the 'device' profile using the SIP SUBSCRIBE utilizing the Event Package specified in this framework.

*   Note: Some of the header fields (for example, Event, via) are continued on a separate line due to format constraints of this document

```
SUBSCRIBE sip:MAC%3a000000000000@sip.example.net SIP/2.0
Event: ua-profile;profile-type=device;vendor="vendor.example.net";
 model="Z100";version="1.2.3";network-user="sip:user@sip.example.net"
From: sip:MAC%3A000000000000@sip.example.net;tag=1234
To: sip:MAC%3A000000000000@sip.example.net
Call-ID: 3573853342923422@10.1.1.44
CSeq: 2131 SUBSCRIBE
Contact: sip:MAC%3A000000000000@sip.example.net
Via: SIP/2.0/TCP 10.1.1.41;
  branch=z9hG4bK6d6d35b6e2a203104d97211a3d18f57a
Accept: message/external-body, application/x-z100-device-profile
Content-Length: 0
```

(SRes)

the SUBSCRIBE request is received by a SIP Proxy in the Service Provider's network which transmits it to the PDS. The PDS accepts the response and responds with a 200 OK

- *   Note: The Client and the SIP proxy may have established a secure communications channel (for example, TLS)

(NTFY)

subsequently, the PDS transmits a SIP NOTIFY message indicating the profile location

- * Note: Some of the fields (for example, content-type) are continued on a separate line due to format constraints of this document

```
NOTIFY sip:MAC%3A000000000000@10.1.1.44 SIP/2.0
Event: ua-profile;effective-by=3600
From: sip:MAC%3A000000000000@sip.example.net;tag=abca
To: sip:MAC%3A000000000000@sip.example.net;tag=1231
Call-ID: 3573853342923422@10.1.1.44
CSeq: 322 NOTIFY
Via: SIP/2.0/UDP 192.168.0.3;
  branch=z9hG4bK1e3effada91dc37fd5a0c95cbf6767d0
MIME-Version: 1.0
Content-Type: message/external-body; access-type="URL";
              expiration="Mon, 01 Jan 2010 09:00:00 UTC";
              URL="http://sip.example.net/z100-000000000000.html";
              size=9999
              hash=10AB568E91245681AC1B

Content-Type: application/x-z100-device-profile
Content-ID: <39EHF78SA@sip.example.net>
.
.
.
```

(NRes)

Client accepts the NOTIFY message and responds with a 200 OK

(XReq)

once the necessary secure communications channel is established, the Client sends an HTTP request to the HTTP server indicated in the NOTIFY

(XRes)

the HTTP server responds to the request via a HTTP response containing the profile contents

7.2. Example 2: Client obtaining change notification

The following example illustrates the case where a User (X) is simultaneously accessing services via two different Clients (for example, Multimedia Soft Clients on a PC and PDA) and has access to a User Interface (UI) that allows for changes to the User profile.

The following are assumed for this example:

o The Clients (A & B) obtain the necessary profiles from the same SIP Service Provider

o The SIP Service Provider also provides a User Interface (UI) that allows the User to change preferences that impact the User profile

The flow diagram and an explanation of the messages follow.

o Note: The example only shows retrieval of User X's profile, but it may request and retrieve other profiles (for example, local-network, Client).

```
               -----           -----
              |User |_________| UI* | * = User Interface
              |  X  |         |     |
               -----           -----
             /       \
            /         \
           /           \              +----------------------+
    +--------+      +--------+        | SIP Service Provider |
    | Client |      | Client |        |                      |
    |    A   |      |    B   |        |  SIP     PDS   HTTP  |
    +--------+      +--------+        | PROXY         Server |
                                      +----------------------+
         |                                |       |      |
         |                                |       |      |
   (A-EX)|<=Enrolls for User X's profile=>|<=====>|      |
         |                                |       |      |
         |                                               |
   (A-RX)|<===Retrieves User X's profile================>|
         |                                               |
         |               |                |       |      |
         |               |  Enrolls for   |       |      |
         |         (B-EX)|<== User X's ==>|<=====>|      |
         |               |    profile     |       |      |
         |               |                |       |      |
         |               |                               |
         |         (B-RX)|<= Retrieves User X's profile=>|
         |                                               |
         |                       |                       |
         |                 (HPut)|---------------------->|
         |                       |                       |
         |                 (HRes)|<----------------------|
         |                                               |
         |                                |       |      |
         |                                | NOTIFY|      |
         |            NOTIFY              |<------|      |
```

```
   (A-NT)|<-------------------------------|       |       |
         |            200 OK              |       |       |
   (A-RS)|------------------------------->|200 OK |       |
         |                                |------>|       |
         |                                                |
         |               |                | NOTIFY|       |
         |               |    NOTIFY      |<------|       |
         |         (B-NT)|<---------------|       |       |
         |               |    200 OK      |       |       |
         |         (B-RS)|--------------->|200 OK |       |
         |               |                |------>|       |
         |                                                |
         |                                                |
   (A-RX)|<===Retrieves User X's profile================>|
         |                                                |
         |               |                                |
         |               |                                |
         |         (B-RX)|<= Retrieves User X's profile=>|
         |               |                                |
```

(A-EX) Client A discovers, enrolls and obtains notification related to User X's profile

(A-RX) Client A retrieves User X's profile

(B-EX) Client B discovers, enrolls and obtains notification related to User X's profile

(B-RX) Client B retrieves User X's profile

(HPut) Changes affected by the User via the User Interface (UI) are uploaded to the HTTP Server

- Note: The UI itself can act as a Client and subscribe to User X's profile. This is not the case in the example shown.

(HRes) Changes are accepted by the HTTP server

(A-NT) PDS transmits a NOTIFY message to Client A indicating the changed profile. A sample message is shown below:

> Note: Some of the fields (for example, Via) are continued on a separate line due to format constraints of this document

```
   NOTIFY sip:userX@10.1.1.44 SIP/2.0
   Event: ua-profile;effective-by=3600
   From: sip:userX@sip.example.net;tag=abcd
   To: sip:userX@sip.example.net.net;tag=1234
   Call-ID: 3573853342923422@10.1.1.44
   CSeq: 322 NOTIFY
   Via: SIP/2.0/UDP 192.168.0.3;
     branch=z9hG4bK1e3effada91dc37fd5a0c95cbf6767d1
   MIME-Version: 1.0
   Content-Type: message/external-body; access-type="URL";
                 expiration="Mon, 01 Jan 2010 09:00:00 UTC";
                 URL="http://www.example.com/user-x-profile.html";
                 size=9999
                 hash=123456789AAABBBCCCDD
   .
   .
   .
```

(A-RS) Client A accepts the NOTIFY and sends a 200 OK

(B-NT) PDS transmits a NOTIFY message to Client B indicating the changed profile. A sample message is shown below:

Note: Some of the fields (for example, Via) are continued on a separate line due to format constraints of this document

```
   NOTIFY sip:userX@10.1.1.43 SIP/2.0
   Event: ua-profile;effective-by=3600
   From: sip:userX@sip.example.net;tag=abce
   To: sip:userX@sip.example.net.net;tag=1235
   Call-ID: 3573853342923422@10.1.1.43
   CSeq: 322 NOTIFY
   Via: SIP/2.0/UDP 192.168.0.3;
     branch=z9hG4bK1e3effada91dc37fd5a0c95cbf6767d2
   MIME-Version: 1.0
   Content-Type: message/external-body; access-type="URL";
                 expiration="Mon, 01 Jan 2010 09:00:00 UTC";
                 URL="http://www.example.com/user-x-profile.html";
                 size=9999
                 hash=123456789AAABBBCCCDD
   .
   .
   .
```

(B-RS) Client B accepts the NOTIFY and sends a 200 OK
(A-RX) Client A retrieves the updated profile pertaining to User X
(B-RX) Client B retrieves the updated profile pertaining to User X

8. IANA Considerations

There are two IANA considerations associated with this document, SIP Event Package and HTTP header. These are outlined in this section.

8.1. 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).
Persons to Contact: Daniel Petrie dan.ietf AT SIPez DOT com, sumanth@cablelabs.com
New event header parameters: profile-type, vendor, model, version, effective-by, network-user (the profile-type parameter has predefined values. The new event header parameters do not)

The following table illustrates the additions to the IANA SIP Header Field Parameters and Parameter Values: (Note to RFC Editor: Please fill in XXXX with the RFC number of this specification)

Header Field	Parameter Name	Predefined Values	Reference
Event	profile-type	Yes	[RFCXXXX]
Event	vendor	No	[RFCXXXX]
Event	model	No	[RFCXXXX]
Event	version	No	[RFCXXXX]
Event	effective-by	No	[RFCXXXX]
Event	network-user	No	[RFCXXXX]

8.2. New HTTP Event Header

This document defines a new permanent HTTP request header field: Event.

Header field name: Event
Applicable protocol: http
Status: standard
Author/Change controller: IETF

Specification document(s): [RFCXXXX] (Note to RFC Editor: Please fill in XXXX with the RFC number of this specification).

9. Security Considerations

The framework specified in this document allows Service Providers to propagate profile data to Clients. This is accomplished by requiring deployed Clients to implement the framework. The framework (explained in Section 5) specifies a Profile Life Cycle that allows Clients to request and obtain profile data. The Profile Life Cycle is enabled using an Event Package (defined in Section 6) as per [RFC3265]. Thus, the primary components requiring security considerations are: Event Package, Profile Life Cycle and Profile Data. The considerations, requirements and recommendations are presented in the following sub-sections.

9.1. Event Package

The Event Package usage MUST adhere to the security considerations and requirements (access control, Notifier privacy mechanism, Denial-of-Service attacks, replay attacks, and Man-in-the Middle attacks) specified in Section 5 of [RFC3265]. Specifically for the Event Package defined in this framework, this sub-section hightlights additional considerations and security requirements.

The Notifier MUST authenticate any SUBSCRIBE request with a known identity. It MUST NOT accept any SUBSCRIBE requests that fail an authentication challenge. Refer to [I-D.ietf-sip-identity] and [RFC3261] for RECOMMENDED SIP authentication methods.

Unless configured otherwise, the Notifier SHOULD NOT respond to SUBSCRIBEs without an identity that can be authenticated. Exceptions include deployments catering to unknown Clients (for example, for self-subscription) or for troubleshooting (for example, credentials misplaced by a user). Refer to Section 9.3 for Profile Data considerations in such cases.

The Notifier MUST transmit NOTIFY messages with sensitive profile data over an authenticated, integrity protected channel. Refer to Section 9.3 for information on profile data classification. It SHOULD transmit Content Indirection information (without profile data) over an integrity-protected channel, unless configured otherwise (for example, if the Service Provider is catering to unknown Clients). For data provided via content indirection, Subscribers MUST implement the hash verification scheme described in [RFC4483].

Subscribers with the ability to authenticate a PDS (for example, Service Provider Certificates, mutual shared secrets) MUST employ such mechanisms prior to retrieving data. This framework RECOMMENDS that Service Providers consider providing this ability to deployed Clients.

9.2. Profile Life Cycle

Profile Discovery involves various protocols such as DHCP and DNS that may provide unauthenticated information. Thus, successful Profile Enrollment and subsequent Profile Notification with an authenticated PDS (for example, via mutual authentication) are required to prevent threats such as impersonation or Denial of Service. Given the nature of these mechanisms and to prevent service disruption due to such threats, the specification recommends caching of retrieved profiles (see Section 5.4) by the Clients. It also provides for multiple Profile Discovery mechanisms (based on Profile Types) which can minimally aid in thwarting security threats from individual mechanisms (for example, impersonated DNS).

The specification strongly RECOMMENDS that solutions implementing the Framework provide the Clients with the ability to recognize, mutually authenticate and establish integrity protected SIP communication channels (for example, mutual TLS using certificates). Clients without such an ability SHOULD report changes to sensitive profile data (refer to Profile Data) using suitable mechanisms (for example, management reporting). Further, Clients with access to credentials (even if obtained via a User Interface) MUST respond to authentication challenges.

Profile Enrollment and Profile Notification are done via the Event Package definition and the security requirements have been presented in Section 9.1. Profile Retrieval and Profile Change Upload are accomplished using Profile Data Frameworks and are addressed in Section 9.3.

9.3. Profile Data

Profile data provided using any of the Profile Types is expected to happen via suitable Profile Data Framework (such as XCAP) or suitable protocol (such as HTTP). Data defined using such frameworks may be sensitive (for example, user credentials) or non-sensitive (for example, list of DNS servers).

If a profile contains sensitive data, it MUST be provided over a mutual-authenticated, integrity protected channel. Even if the data is non-sensitive, it SHOULD still be provided over a secure channel. Exceptions include cases where deployments cater to unknown Clients

or for troubleshooting.

For profile data delivered within the framework (i.e. data is provided in the NOTIFY), the requirements specified in Section 9.1.

When the profile data is delivered via content indirection, authentication, integrity, confidentiality MUST be provided by the Profile Data Frameworks containing the retrieval mechanisms. Further, a non-replayable authentication mechanism (for example, Digest authentication) MUST be used.

10. Acknowledgements

Many thanks to those who contributed and commented on the many iterations of this document. Detailed comments were provided by the following individuals: 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, Roni Even from Polycom, Jason Fischl from Counterpath, Josh Littlefield from Cisco, Nhut Nguyen from Samsung.

The editor would like to extend a special thanks to the experts who contributed to the restructuring and revisions as proposed by the SIPPING WG, specifically Keith Drage from Lucent (restructuring proposal), Peter Blatherwick from Mitel (who also contributed to the Overview and Introduction sections), Josh Littlefield from Cisco (examples and diagram suggestions), Alvin Jiang of Engin, Martin Dolly from AT&T, and Jason Fischl from Counterpath. Additionally, sincere appreciation is extended to the chairs (Mary Barnes from Nortel and Gonzalo Camarillo from Ericsson) and the Area Directors (Cullen Jennings from Cisco and Jon Peterson and Cisco) for facilitating discussions, and for reviews and contributions.

11. Open Items

[[Editor's note: This is being used a place holder only and will be removed once the items listed are addressed]]

The following comments are considered to be open (i.e. not addressed) in this version of the I-D

- Replace 'Service Provider' with a term better representative of its definition
- Analyze potential unformity in the formation of the Subscription URI across Profile Types. If not, provide a bried explanation of the analysis
- Analyze the current SHOULD v/s MUST requirements for the Profile Framework to obtain consensus and facilitate interoperability
- Present an analysis of the Local Network Profile discovery methods in DNS-less environments
- Check on potentially referencing RFC4122 instead of OUTBOUND
- Security Considerations requires further review

12. Change History

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

12.1. Changes from draft-ietf-sipping-config-framework-09.txt

Following the ad-hoc SIPPING WG discussions at IETF#67 and as per the email from Gonzalo Camarillo dated 12/07/2006, Sumanth was appointed as the new editor. This sub-section highlights the changes made by the editor (as per expert recommendations from the SIPPING WG folks interested in this effort) and the author.

Changes incorporated by the editor:

- Document was restructured based on a) Keith's recommendations in the email dated 11/09/2006 and responses (Peter, Sumanth, Josh) b) subsequent discussions by the ad-hoc group consisting of the editor, the author, expert contributors (Peter Blatherwick, Josh Littlefield, Alvin Jiang, Jason Fischl, Martin Dolly, Cullen Jennings) and the co-chairs . Further changes follow.
- Use cases were made high-level with detailed examples added later on
- Several sections were modified as part of the restructuring (for example, Overview, Introduction, Framework Requirements, Security Sections)
- General editorial updates were made

Changes incorporated by the author:

- Incorporated numerous edits and corrections from CableLabs review.
- Used better ascii art picture of overview from Josh Littlefield

- o Fixed the normative text for network-user so that it is now consistant: MAY provide for device profile, MUST provide for local-network profile.

12.2. Changes from draft-ietf-sipping-config-framework-08.txt

The Request URI for profile-type=localnet now SHOULD not have a user part to make routing easier. The From field SHOULD now contain the device id so that device tracking can still be done.
Described the concept of profile-type as a filter and added normative text requiring 404 for profile types not provided.
Moved "application" profile type to draft-ietf-sipping-xcap-config-01. The "application" value for the profile-type parameter will also be used as a requirement that XCAP be supported.
Fixed text on certificate validation.
Added new HTTP header: Event to IANA section and clean up the IANA section.
Added diagram for Service Provider use case schenario.
Added clarification for HTTP Event header.
Added clarification of subscriber handling of NOTIFY with no body.

12.3. 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.
Synchronized 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.
Clarified use of network-user parameter.
Allow the use of the auid and document parameters per request by the OMA.

12.4. 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 clarification 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!

12.5. 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.

12.6. 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.

12.7. 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.

12.8. 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.

12.9. 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.

12.10. 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.

12.11. 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 [RFC4483]

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.

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

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

Minor edits

12.13. 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 nomadic 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-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.

[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.

[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.

[RFC4483] Burger, E., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", RFC 4483,

May 2006.

13.2. Informative References

[I-D.ietf-simple-xcap]
Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-12 (work in progress), October 2006.

[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-04 (work in progress), October 2006.

[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.

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

[RFC3617] Lear, E., "Uniform Resource Identifier (URI) Scheme and Applicability Statement for the Trivial File Transfer Protocol (TFTP)", RFC 3617, October 2003.

Authors' Addresses

Daniel Petrie
SIPez LLC.
34 Robbins Rd
Arlington, MA 02476
USA

Email: dan.ietf AT SIPez DOT com
URI: http://www.SIPez.com/

Sumanth Channabasappa (Editor)
CableLabs
858 Coal Creek Circle
Louisville, Co 80027
USA

Email: sumanth@cablelabs.com
URI: http://www.cablelabs.com/

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).