SIPPING	M. Dolly
Internet-Draft	AT&T
Intended status: Standards Track	D. Petrie
Expires: September 10, 2009	SIPez LLC
	D. Worley (Editor)
	Nortel Networks
	March 09, 2009

TOC

A Schema and Guidelines for Defining Session Initiation Protocol User Agent Profile Datasets
draft-ietf-sipping-profile-datasets-03.txt

Status of this Memo

Copyright Notice

Abstract

This document defines the requirements and a format for SIP user agent profile data. An overall schema is specified for the definition of

profile datasets. The schema also provides for expressing constraints for how multiple sources of profile data are to be combined. This document provides a guide to considerations, policies and syntax for defining datasets to be included in profile data.

Table of Contents

1. Introduction
2. Terminology
3. Overview
4. Design Considerations
    - 4.1. Requirement Descriptions
        - 4.1.1. Implementer Extensibility
        - 4.1.2. Flexible Capabilities
        - 4.1.3. Access Control
        - 4.1.4. Data Constraints and Range Definition
        - 4.1.5. Support of User, Device, Local Network Sources
        - 4.1.6. The Ability to Specify Policy
        - 4.1.7. XML
5. Overall Dataset Schema
    - 5.1. Data Primitives
    - 5.2. Use of Namespaces
    - 5.3. The 'propertySet' Element
    - 5.4. The Abstract 'setting_container' Element
    - 5.5. The Abstract 'setting' Element
        - 5.5.1. The 'visibility' Attribute
        - 5.5.2. The 'policy' Attributes
        - 5.5.3. The 'excludedPolicy' Attributes
        - 5.5.4. The 'direction' Attribute
        - 5.5.5. The 'q' Attribute
    - 5.6. The 'profileUri' Element
    - 5.7. The 'profileCredential' Element
        - 5.7.1. realm Element
        - 5.7.2. authUser Element
        - 5.7.3. a1Digest Element
        - 5.7.4. password Element
    - 5.8. The 'profileContactUri' Element
    - 5.9. The 'profileInfo' Element
    - 5.10. Example Profile Dataset
    - 5.11. Merging Property Sets
        - 5.11.1. Single Numeric Value Merging Algorithm
        - 5.11.2. Multiple Enumerated Value Merging Algorithm
        - 5.11.3. Closest Value First Merging Algorithm
    - 5.12. Common Types
6. Defining Data Sets
    - 6.1. Namespace
    - 6.2. Property Definitions

6.3. Merging Data Sets
7. Candidate Data Sets
8. Security Considerations
9. IANA Considerations
9.1. Content-type registration for 'application/uaprofile+xml'
10. Contributors
11. Acknowledgments
12. References
12.1. Normative References
12.2. Informative References
Appendix A. Relax NG SIP UA Profile Schema
Appendix B. Use Cases
B.1. Outbound Proxy Setting
B.2. Codec Settings
B.2.1. Codec Setting Not Set
B.2.2. Codec Set in Device Profile
B.2.3. Set in Device and User Profiles
B.2.4. Set in Device and Local Profiles
B.2.5. Set in Device, User and Local Profiles
B.2.6. Example Derived Requirements
§ Authors' Addresses

---

1. Introduction

TOC

The SIP User Agent Profile Delivery Framework [I-D.ietf-sipping-config-framework] (Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery," February 2010.) and the Framework for SIP Session Policies [I-D.ietf-sip-session-policy-framework] (Hilt, V., Camarillo, G., and J. Rosenberg, "A Framework for Session Initiation Protocol (SIP) Session Policies," February 2010.) specify how SIP user agents locate and retrieve profile data specific to the user, the device, and the local network. It is important for SIP User Agents to be able to obtain and use these multiple sources of profile data in order to support a wide range of applications without undue complexity.

While these frameworks define the mechanisms for transmitting profile data, they do not define a format for the actual profile data. This document defines the requirements, the default/mandatory to support content type for [I-D.ietf-sipping-config-framework] (Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery," February 2010.) , a high level schema for, and guide to how these datasets can be defined. The goal is to enable any SIP user agent to obtain profile data and be functional in a new environment independent of the implementation or model of user agent. The nature of having profile data from three potential sources requires the

definition of policies on how to apply the data in an interoperable way across implementations which may have widely varying capabilities. The ultimate objective of the framework described here, together with the SIP User Agent Profile delivery framework, is to a start up experience requiring minimal user intervention. One should be able to take a new SIP user agent out of the box, plug it in or install the software and have it get its profiles without human intervention other than security measures. This is necessary for cost effective deployment of large numbers of user agents.

---

2. Terminology

TOC

"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] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).
This document uses the terms "profile" and "device" as defined in [I-D.ietf-sipping-config-framework] (Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery," February 2010.):

> **profile -** a set of configuration data intended to configure a specific device or devices and obtained from a specific source (e.g., user, device, local network or other). Has a concrete representation as an XML document.
>
> **profile type -** a particular category of Profile data in regard to its source (e.g., user, device, local network or other).

In addition, this document defines the following terms:

> **profile schema -** a definition of a set of possible profiles that are seen as alternative configuration data for a set of UAs. Has a specific XML namespace and a concrete representation in XML Schema Language and/or in Relax NG schema language.
>
> **profile meta -** schema: the schema of the XML namespace "urn:ietf:params:xml:ns:uaprof", from which are derived the various profile schemas
>
> **user profile -** the profile that applies to a specific user. The user profile is that set of profile data the user wants to associate with a given device (e.g. ringtones used when someone calls them, the user's shortcuts) and relate to the user's identity

device profile -
data profile that applies to a specific device. This is the data that is bound to the device itself independent of the user that is bound to the device. It relates to specific capabilities of the device and/or preferences of the owner of the device.

local network profile - data that applies to the user agent in the context of the local network. This is best illustrated by roaming applications; a new device appears in the local network (or a device appears in a new network, depending on the point of view). The local network profile includes settings and perhaps policies that allow the user agent to function in the local network (e.g. how to traverse NAT or firewall, bandwidth constraints).

merging - the operation of resolving overlapping settings from multiple profiles. Overlap occurs when the same property occurs in multiple profiles (e.g. device, user, local network).

working profile - the set of property values utilized in a SIP User Agent; logically constructed by merging the profiles from the relevant sources

property - a named configurable characteristic of a user agent; a named datum within a profile schema. A given property has a well-defined range of possible values. A given property may be defined to have a range of values, allow for simultaneous use of many values (as in a list of allowed possibilities), or a set of related values that collectively form a single profile information item.

dataset - a collection of properties.

setting - the binding of a specific value or set of values to a given property.

Thus, a profile schema defines a dataset, and a profile is a set of settings that conforms to a particular profile schema.

---

3. Overview

TOC

In this document requirements are specified for containing and expressing profile data for use on SIP user agents. Though much of this can be considered independent of SIP there is one primary requirement that is not well satisfied through more generic profile data mechanisms. SIP User Agent set up requires the agent to merge settings, which may overlap, from potentially three different sources (see

[I-D.ietf-sipping-config-framework] (Channabasappa, S., “A Framework for Session Initiation Protocol User Agent Profile Delivery,” February 2010.)); each source must not only be able to provide profile information, but also express policies regarding how the profile settings may be combined with that from other sources.
A schema and syntax is defined in this document to specify properties that may be aggregated to construct profiles. The general design philosophy is that many small datasets provide flexibility to the implementer to support the aggregated set that best matches the capability of the user agent. The actual properties are not defined in this document and will be the subject of derived drafts. However, some examples are provided in Appendix B (Use Cases) to illustrate the proposed mechanisms and to validate the requirements.
This document defines a set of considerations, syntax and policies that must be specified when defining datasets. These are to help authors of dataset specifications to define datasets that will work in the overall schema defined in this document. The actual specification of these datasets is outside the scope of this document.

---

4. Design Considerations

TOC

The following section defines some of the design considerations that were taken into account when defining the schema, syntax and policies for generating and applying profile data.

4.1. Requirement Descriptions

TOC

4.1.1. Implementer Extensibility

TOC

It does not serve user agent administrators to have to require a coordinated and orchestrated upgrade of every user agent and corresponding profile delivery servers for a new capability to be supported. Datasets MUST be extensible without breaking the user agents that support that dataset. This may require the user agents to ignore parts of the extended dataset that it does not support. It may also be possible to tag the extensions with minimum version numbers to facilitate the user agents decision making.

4.1.2. Flexible Capabilities

TOC

Since user agents vary greatly in their capabilities, it MUST be possible for the implementer to tailor the datasets to the capabilities of the user agent device. This implies that the profile is built up of a series of small datasets based upon the capabilities of the user agent. The user agents MAY ignore datasets for capabilities they do not support. This allows the profile delivery server to be agnostic of device capabilities. It is however the implementer's choice to customize the delivered profile to the device capabilities.

4.1.3. Access Control

TOC

There are likely to be properties in various profile datasets that the Operators and Administrators do not want the users to sometimes not be able to modify or even see. It MUST be possible to disallow the user from modifying a property. It MUST be possible to obfuscate the user from seeing a property or its setting. This access control information SHOULD be optional for a given property.This is supported by the admin value of the visibility attribute.

4.1.4. Data Constraints and Range Definition

TOC

Property values are likely to have an allowed set of values under most circumstances rather than completely unconstrained in their values. It MUST be possible for the schema to specify constraints on a property value, viz, as a range or as a set of discrete values. These constraints SHOULD be optional to the dataset and SHOULD be expressible independent of the property itself.

4.1.5. Support of User, Device, Local Network Sources

TOC

[I-D.ietf-sipping-config-framework] (Channabasappa, S., “A Framework for Session Initiation Protocol User Agent Profile Delivery,” February 2010.) specifies a mechanism where the user agent retrieves profile data from as many as three different sources. The separation of the user profile facilitates a hotelling capability and the ability to easily re-assign a user to a different device. The separation of the local network profile facilitates properties specific to operating in the local network in a roaming scenario (e.g. outbound proxy or NAT

traversal properties). The device profile facilitates device capability based properties as well as a means for the device owner or manager to impose policy.
While increasing the complexity of the user agent in that it must aggregate and consolidate separate profiles into one working profile, constraining the properties of the various profiles to be mutually exclusive, or constraining even the merging rules would severely restrict functionality.
Profile merging rules are associated with individual datasets or even associated with individual properties inside a dataset. A profile MUST have a merging algorithm defined. An individual property inside MAY contain a merging rule, in which case this merging rule is specific to the property. If however, there is no merging rule associated with a property, but the profile dataset in its entirety has a merging rule, this merging rule MUST be applied to each of the properties that form part of the profile.
A few of the more commonly used merging algorithms are defined in this document. Most settings are likely to use the common set defined in this document. However authors of profile datasets may define new algorithms for merging dataset properties (see Section 5.11 (Merging Property Sets) and Section 6.3 (Merging Data Sets)).

4.1.6. The Ability to Specify Policy

TOC

Local Network Operators may wish to impose policy on users and devices on their network such as constraining codecs, media streams, outbound proxy or emergency services. It MUST be possible to impose policy in any of the profile sources that constrains, overwrites or modifies properties provided in datasets from other sources.

4.1.7. XML

TOC

XML is perhaps not really a requirement, but a solution base upon requirements. However it is hard to ignore the desire to utilize readily available tools to manage and manipulate profile data such as XSLT, XPATH and XCAP. The requirement that should be considered when defining the schema and syntax is that many user agents have limited resources for supporting advanced XML operation. The simplest XML construct possible should be used, that support the required functionality. It is not a requirement that user agents validate the profile XML document. This relieves the requirement that the Relax NG schema defined in this and other datasets documents be enforced on the user agent. The Relax NG schema should not be used to strictly validate profile XML documents. Unknown elements and attributes should be

ignored to allow extensions to be supported. Strict enforcement of the Relax NG schema would make it very difficult to deploy new user agents without lock step upgrades of the profile delivery server. Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols [RFC3470] (Hollenbeck, S., Rose, M., and L. Masinter, “Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols,” January 2003.) provides useful information in this regard.

---

5. Overall Dataset Schema

TOC

Notifiers and Subscribers of the event package defined in [I-D.ietf-sipping-config-framework] (Channabasappa, S., “A Framework for Session Initiation Protocol User Agent Profile Delivery,” February 2010.) SHOULD support the content-type: application/ uaprofile+xml. The Notifier SHOULD indicate all of the dataset schemas that is supports by listing all of the MIME types for the supported datasets in the SUBSCRIBE request header: Accept. This document defines an Relax NG Schema for that content-type with the namespace: urn:ietf:params:xml:ns:uaprof, for SIP Profile Datasets that provides:

- a base element type "setting" from which all settings in other schema definitions inherit (this allows other definitions to specify the content models for ways of combining settings; it is analogous to a C++ virtual base class).

- Attributes to the "setting" element that define constraints and other properties used to impose policy that apply to the element value. These attributes are inherited by elements that are derived from the abstract settings element.

- A root element for all property sets (the outermost container).

The full text of the schema is in Appendix A (Relax NG SIP UA Profile Schema) ; the following describes the usage of the schema in defining properties and combining them to construct the working profile of a User Agent.

5.1. Data Primitives

TOC

Each property in a profile data set is defined using XML Schema Datatypes [W3C.REC-xmlschema-2] (Biron, P. and A. Malhotra, “XML Schema Part 2: Datatypes,” May 2001.) and Relax NG Schema. A property is modeled by an XML element derived from the "setting" element in the SIP Profile Data Set Schema. The element content is the setting value.

Properties consisting of one single value can be expressed using a single XML element. Properties that may consist of multiple values require the use of container elements. A container element is defined for such a property. This container can contain multiple XML elements, which each defines a possible value for that property (see examples in Section 5.5.2 (The 'policy' Attributes)).
When constructing a property set, the creator of a profile may not be able to know all of the capabilities of the User Agent that will receive that property set. The creator of profile constraints or policies should be aware that a user agent may ignore properties that are unsupported or do not apply to its capabilities.

5.2. Use of Namespaces

TOC

XML namespaces [W3C.REC-xml-names-19990114] (Hollander, D., Bray, T., and A. Layman, “Namespaces in XML,” January 1999.) provide a means to uniquely identify the elements and datatypes defined in a data set. It is therefore RECOMMENDED that each data set specifies its own namespace. The namespace URIs SHOULD be URNs (Moats, R., “URN Syntax,” May 1997.) [RFC2141] , using the namespace identifier 'ietf' defined by [RFC2648] (Moats, R., “A URN Namespace for IETF Documents,” August 1999.) and extended by [RFC3688] (Mealling, M., “The IETF XML Registry,” January 2004.) . The core schema defined in this document defines the namespace: "urn:ietf:params:xml:ns:uaprof". Profile datasets that extend this schema SHOULD define a new namespace by appending a ":" and a unique name to the "urn:ietf:params:xml:ns:uaprof" namespace. These namespaces MUST be registered with IANA.

5.3. The 'propertySet' Element

TOC

The root element of a property set is "propertySet"; it is the container that is provided to the user agent. The elements contained within a propertySet contain the specific properties which are to be applied to the user agent. The properties may be simple types with a single value, complex types or container elements with a list of properties.

TOC

5.4. The Abstract 'setting_container' Element

The "setting_container" element is the abstract element in which a list of properties which allow multiple values may be contained. Elements derived from the "setting_container" element may contain zero or more elements derived from the "setting" element. The "setting_container" element has an "excludedPolicy" attribute.

5.5. The Abstract 'setting' Element

TOC

The setting element is the abstract element from which all profile properties or settings shall inherit.
The setting element has a number of attributes that provide functionalities, which are generally useful for many properties. These attributes are inherited by properties that are derived from the settings element. This enables the re-use of common functionalities and ensures a common syntax for these elements across different data sets. The following functionalities are provided by attributes of the settings element:

*Property Access Control: 'visibility' attribute

*Policies: 'policy' attribute

Additional attributes are defined in the schema that may used in elements derived from "setting". By default these attributes cannot be set. These attribute must be explicitly added to elements derived from the "setting" element:

*Unidirectional Properties: 'direction' attribute

*Preferences: 'q' attribute

5.5.1. The 'visibility' Attribute

TOC

The attribute "visibility" is defined on the "setting" element to specify whether or not the user agent is permitted to display the property value to the user. This is used to hide setting values that the profile administrator may not want the user to see or know. The "visibility" attribute has two possible values:

*user: Specifies that display of the property value is not restricted to the user. This is the default value of the attribute if it is not specified.

*admin: Specifies that the user agent SHOULD NOT display the property value. Display of the property value may be allowed using special administrative interfaces, but is not appropriate to the ordinary user.

5.5.2. The 'policy' Attributes

TOC

The setting element has an optional 'policy' attribute. The policy attribute is used to define the constraining properties of an element. It defines how the element value is used by an endpoint (e.g. whether it can or can not be used in a session). The following values are defined for the 'policy' attribute:

*allow: the value contained in the element is allowed and SHOULD be used in sessions. This is the default value that is used if the 'policy' attribute is omitted.

*disallow: the value contained in the element is forbidden and SHOULD NOT be used in sessions.

The policy attribute can be omitted if the default policy 'allow' applies.
The policy attribute is used only inside containers.

5.5.3. The 'excludedPolicy' Attributes

TOC

The "setting_container" element has an optional 'excludedPolicy' attribute. This attribute specifies the default policy for all values that are not in the container. Elements that are present in the container have their own 'policy' attribute, which defines the policy for that element. The following values are defined for the 'excludedPolicy' attribute:

*allow: values not listed in the container are allowed and MAY be used in sessions. This is the default value that is used if the 'excludedPolicy' attribute is omitted.

*disallow: values not listed in the container are forbidden and MUST NOT be used in sessions.

The excludedPolicy attribute can be omitted if the default policy 'allow' applies. The following example shows a policy that allows the media type audio and disallows all other media types in sessions (effectively, this construct requires the use of audio):

```
<media-types excludedPolicy="disallow">
  <media-type policy="allow">audio</media-type>
</media-types>
```

5.5.4. The 'direction' Attribute

TOC

Some properties are unidirectional and only apply to messages or data streams transmitted into one direction. For example, a property for media streams can be restricted to outgoing media streams only. Unidirectional properties can be expressed by adding a 'direction' attribute to the respective element.
The 'direction' attribute can have the following values:

*recvonly: the property only applies to incoming messages/streams.

*sendonly: the property only applies to outgoing messages/streams.

*sendrecv: the property applies to messages/streams in both directions. This is the default value that is used if the 'direction' attribute is omitted.

5.5.5. The 'q' Attribute

TOC

It should be possible to express a preference for a certain value, if multiple values are allowed within a property. For example, it should be possible to express that the codecs G.711 and G.729 are allowed, but G.711 is preferred. Preferences can be expressed by adding a 'q' attribute to a property element. Elements derived from the "setting" element for which multiple occurrences and values are allowed SHOULD have a "q" attribute if the order is significant. Typically these elements are contained in an element derived from the "setting_container" element. The 'q' attribute is only meaningful if

the 'policy' attribute set to 'allowed'. It must be ignored in all other cases.
An element with a higher 'q' value is preferred over one with a lower 'q' value. 'q' attribute values range from 0 to 1. The default value is 0.5.

5.6. The 'profileUri' Element

TOC

The <profileUri> element contains the URI of this profile on the profile server. The value contained in the profileUri element may be different than the URI subscribe to when retrieving this profile. When the user agent retrieves a profile where the profileUri is different than the subscribe to URI, the user agent SHOULD unsubscribe to the current URI and then subscribe to the new URI.
The <profileUri> element is optional and MAY occur only once inside a <propertySet> element. The profileUri element is specific to the local-network, device or user profile in which it occurs. It has no meaning outside of the profile in which it occurs and SHOULD NOT be merged.

5.7. The 'profileCredential' Element

TOC

The <profileCredential> element contains the digest authentication information that SHOULD be used for authentication for the profile subscription via SIP or profile retrieval via HTTP, HTTPS, etc. The profileCredential element is optional and MAY occur only once inside a propertySet element. The profileCredential element is specific to the local-network, device or user profile in which it occurs. It has no meaning outside of the profile in which it occurs and SHOULD NOT be merged. The profileCredential element MUST contain exactly one of each of the elements: realm, authUser and one of either a1Digest or password.

5.7.1. realm Element

TOC

The realm element contains the string that defines the realm to which this credential pertains. The value of the realm element is the same as the realm parameter in the [RFC2617] (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," June 1999.) headers: WWW-Authenticate, Authorization and the SIP (Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson,

J., Sparks, R., Handley, M., and E. Schooler, “SIP: Session Initiation Protocol,” June 2002.) [RFC3261] headers: Proxy-Authenticate and Proxy-Authorization. If a match of the realm value is found, the user agent uses the values in the authUser and a1Digest elements contained in the profileCredential element. Exactly one realm element MUST be contained in a profileCredential element. A wildcard of "*" MAY be used as the realm value in which case the user agent MUST calculate the A1 DIGEST for the realm given in the authentication challenge. If the wildcard is given for the realm, the clear text form of the password contained in the password element MUST also be used.

5.7.2. authUser Element

TOC

The authUser element contains the string value of the "username" parameter which SHOULD be used in Authorization and Proxy-Authorization request headers when retrying a request that was challenged for authentication. Exactly one authUser element SHOULD be contained in a profileCredential element.

5.7.3. a1Digest Element

TOC

The a1Digest element contains a string with the value of the A1 digest of the username, realm and password as defined in [RFC2617] (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, “HTTP Authentication: Basic and Digest Access Authentication,” June 1999.) . At most one a1Digest element MUST be contained in a profileCredential element. The a1Digest element MUST NOT exist in a profileCredential element containing a password element. The username and realm used to construct the value of the a1Digest element MUST match the values of the realm and authUser elements contained in the profileCredential element with the a1Digest element.

5.7.4. password Element

TOC

The password element contains the clear text password for use with DIGEST Authentication (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, “HTTP Authentication: Basic and Digest Access Authentication,” June 1999.) [RFC2617] . At most one password MUST be contained in a profileCredential element. The password element MUST NOT exist in a profileCredential element containing a a1Digest element. The user agent

uses this password along with the realm and authUser elements to calculate the A1 digest used for DIGEST Authentication.

5.8. The 'profileContactUri' Element

TOC

The <profileContactUri> element contains a contact URI (e.g. a SIP, HTTP URI or email address) under which the issuer of this profile can be reached. The contact element may, for example, contain the address of a support hotline.
The <profileContactUri> element is optional and MAY occur multiple times inside a <propertySet> element. Multiple instances of the profileContactUri element allow multiple URI schemes to be provided for contact information. The user agent MAY use the URI contained profile-contact-info element which has a URI scheme that the user agent supports and can make work to provide support help for the profile. The user agent MAY provide the URIs to the user to contact the creator of the profile through other communication channels. The profileContactUri element is specific to the local-network, device or user profile in which it occurs. It has no meaning outside of the profile in which it occurs and SHOULD NOT be merged.

5.9. The 'profileInfo' Element

TOC

The <profileInfo> element provides a short textual description of the property that should be intelligible to the human user. This element may, for example, contain information about the nature of this profile, such as "Access Network Profile". The text in the <profileInfo> element is in particular be helpful when a user needs to decide whether or not to use a newly downloaded profile or when problems with a profile (e.g. a policy conflict) occur. A user agent MAY display this information in these cases.
The <profileInfo> element is optional and MAY occur only once inside a <propertySet> element. The profileInfo element is specific to the local-network, device or user profile in which it occurs. It has not meaning outside of the profile in which it occurs and SHOULD NOT be merged.

5.10. Example Profile Dataset

TOC

The following XML example shows a SIP Profile Dataset with example extension setting elements: ddd, foo, bar, boo, containerElement and

setting container elements: myContainer, myContainer1, myContainer2 and container3.

```
<?xml version="1.0" encoding="UTF-8"?>
<propertySet xmlns="urn:ietf:params:xml:ns:uaprof">
    <profileUri>sip:a1b2c3d4e5f6@example.com</profileUri>
    <profileCredential>
         <realm>example.com</realm>
         <authUser>fred</authUser>
         <a1Digest>b6b577fd12aa7e1df8d60735ef56fc2e</a1Digest>
         <!-- <password>123</password> -->
    </profileCredential>
    <profileContactUri>tel:+16175551212</profileContactUri>
    <profileContactUri>sip:411@example.com</profileContactUri>
    <profileContactUri>
        http:example.com/sipProfile.html
    </profileContactUri>
    <profileInfo>
        This is an example profile from example.com
    </profileInfo>
    <ddd xmlns="blatz" policy="allow">fff</ddd>
    <foo xmlns="blatz" visibility="user" policy="allow"
         direction="sendrecv" q="0">
    </foo>
    <bar xmlns="blatz" visibility="admin" policy=""
         direction="sendonly" q="0.1000">
    </bar>
    <myContainer xmlns="blatz" excludedPolicy="disallow">
        <containerElement q="0.1">aaa</containerElement>
        <containerElement>bbb</containerElement>
        <containerElement q="0.8">ccc</containerElement>
    </myContainer>
    <boo xmlns="newns" q="1">ggg</boo>
    <myContainer1 xmlns="blatz" excludedPolicy="allow">
        <myContainer2 xmlns="newns" excludedPolicy="allow">
        </myContainer2>
    </myContainer1>
    <container3 xmlns="ns3">
        <containerElement q="0.1">111</containerElement>
        <containerElement>222</containerElement>
        <containerElement q="0.8">333</containerElement>
    </container3>
</propertySet>
```

5.11. Merging Property Sets

TOC

A UA may receive property sets from multiple sources, which need to be merged into a single combined document the UA can work with. Properties that have a single value (e.g. the maximum bandwidth allowed) require that a common value is determined for this property during the merging process. The merging rules for determining this value need to be defined individually for each element in the schema definition. Properties that allow multiple values (i.e. property containers) need to be merged by combining the values from the different data sets. The following sections describe recommended common merging algorithms. A data set definition may refer to these algorithms.

5.11.1. Single Numeric Value Merging Algorithm

TOC

A general merging rule for elements with numeric values is to select the largest or the smallest value. For example, a merging rule for a <max-bandwidth> element would be to select the smallest value from the values that are in the competing data sets.

5.11.2. Multiple Enumerated Value Merging Algorithm

TOC

Multiple values in property containers are merged by combining the values from each of the competing data sets. This is accomplished by copying the elements from each property container into the merged container. Elements with identical values are only copied once. The 'policy' attribute of two elements with the same value is adjusted during the merging process according to Table 1. If an element exists only in one property container, then the default policy of the other container (i.e. the excludedPolicy) is used when accessing Table 1. For example, if an element is disallowed in one data set and the element is not contained in the other data set but the default policy is allowed for that data set, then the values disallowed and allowed are used to access Table 1. Consequently, the element will be disallowed in the merged data set. Finally, the excludedPolicy attributes of the containers are also merged using Table 1. In addition to these merging rules, each schema may define specific merging rules for each property container.

```
set 1 \ set 2 |  allow   | disallow
--------------+----------+----------
allow         |  allow   | disallow
disallow      | disallow | disallow
```

Table 1: merging policies.

The following example illustrates the merging process for two data sets. All elements are merged into one container and the policy attributes are adjusted according to Table 1. The merged container has the default policy disallow, which is determined using Table 1. The entry for PCMA in the merged data set is redundant since it has the default policy.

```
Data set 1:
<codecs excludedPolicy='allow'>
  <codec policy='disallow'>PCMA</codec>
</codecs>

Data set 2:
<codecs excludedPolicy='disallow'>
  <codec policy='allow'>PCMA</codec>
  <codec policy='allow'>G729</codec>
</codecs>

Merged data set:
<codecs excludedPolicy='disallow'>
  <codec policy='disallow'>PCMA</codec>
  <codec policy='allow'>G729</codec>
</codecs>
```

Some constellations of policy attributes result in an illegal merged data set. They constitute a conflict that can not be resolved automatically. For example, two data sets may define two non-overlapping sets of allowed audio codecs and both disallow all other codes. The resulting merged set of codecs would be empty, which is illegal according to the schema definition of the codecs element. If the use of these properties is enforced by both networks, the UA may experience difficulties or may not be able to set up a session at all. The combined property set MUST again be valid and well-formed according to the schema definitions. A conflict occurs if the combined property set is not a well-formed document after the merging process is completed.

5.11.3. Closest Value First Merging Algorithm

TOC

Some properties require that the values from different data sets are ordered based on the origin of the data set during the merging process. Property values that come from a domain close to the user agent take precedence over values that were in a data set delivered by a remote domain. This order can be used, for example, to select the property value from the closest domain. In many cases, this is the local domain of the user agent. For example, the URI of an outbound proxy could be merged this way. This order can also be used to generate an ordered list of property values during the merging process. For example, multiple values for media intermediaries can be ordered so that the closest media intermediary is traversed before the second closest intermediary and so on.

This merging algorithm requires that the source of a data set is considered.

If property sets are delivered through the configuration framework [I-D.ietf-sipping-config-framework] (Channabasappa, S., “A Framework for Session Initiation Protocol User Agent Profile Delivery,” February 2010.) , the value received through a subscription using the "local-network" profile-type takes precedence over values received through other profile-type subscriptions, followed by device and then user profile-types.

The session-specific policy mechanism [I-D.ietf-sip-session-policy-framework] (Hilt, V., Camarillo, G., and J. Rosenberg, “A Framework for Session Initiation Protocol (SIP) Session Policies,” February 2010.) provides an order among policy servers. This order is based on the order, in which a SIP message traverses the network, starting with the closest domain. This order can directly be used to order property values as described above.

5.12. Common Types

TOC

The schema also defines a set of common types that are used in defining data sets (e.g. DataIpPort). [Need to document common types.]

---

6. Defining Data Sets

TOC

This section covers several issues that should be take into consideration when specifying new data set schemas. This is intended to be a guide to authors writing specifications defining a new data set schema or extensions to existing ones.

6.1. Namespace

TOC

It is RECOMMENDED that a data set defines a new XML namespace (Hollander, D., Bray, T., and A. Layman, “Namespaces in XML,” January 1999.) [W3C.REC-xml-names-19990114] to scope all of the properties that are defined in the name space.

6.2. Property Definitions

TOC

The properties defined in a data set schema may be simple (i.e. having a single value) or they may be complex (i.e. a container with multiple values). Each property in the data set SHOULD inherit from the "setting" element. Complex properties and all of their child elements each should inherit from "setting" as well.
A data set specification should contain a section which defines the meaning of all of the properties contained in the data set. The objective is to define the property such that implementers have a clear definition and semantics to interpret properties in a consistent way. User agents not only need to use the same profile content, they need to apply the properties in a consistent way to achieve true interoperability.
The following information should be defined for each property in a data set:

*description: describe the meaning and application of the property.

*cardinality: define how many instances of this property element may occur in a data set (e.g. zero, one or many) as well as its relationship to any other properties in this or other data sets.

*default value: define the default value of this property if it is not set. Describe if the default is different if the property is present and not set vs. completely absent from the data set. Define if the default varies in relation to another property.

6.3. Merging Data Sets

TOC

User agents may receive data sets from multiple sources. They need to merge these data sets in order to create an overall data set they can work with. Collisions on data sets may occur if multiple sources

provide different values for the same properties. These collisions need to be resolved during the merging process.
A data set schema MUST define rules for merging data sets from different sources for each property that is defined. Recommendations for merging data sets are discussed in Section 5.11 (Merging Property Sets) . A data set schema must define if and how these recommendations apply and MAY define alternative merging rules for specific settings. A data set schema must also identify combinations of properties that constitute a conflict that can't resolved. It may provide additional guidelines for the behavior of a user agent in these cases.

---

7. Candidate Data Sets

TOC

The following sections name some of the candidate data sets that are or may be defined. These data sets can be aggregated to form profiles appropriate to the capabilities of a user agent implementation.

*SIP Protocol Data Set: the lowest common denominator set of properties common to all SIP user agents of any capability. A schema covering the elements of this data set can be found in [I-D.petrie-sipping-sip-dataset] (Channabasappa, S. and S. Ganesan, “The Core Session Initiation Protocol User Agent Protocol Data Set,” November 2007.) .

*Media Data Set: this data set contains media related policies. A schema covering the elements of this data set can be found in [I-D.ietf-sipping-media-policy-dataset] (Hilt, V., Worley, D., Camarillo, G., and J. Rosenberg, “A User Agent Profile Data Set for Media Policy,” March 2010.) .

*Identity Data Set: AORs and lines (see [I-D.petrie-sipping-identity-dataset] (Petrie, D., Channabasappa, S., and S. Ganesan, “The Session Initiation Protocol User Agent Identity Profile Data Set,” November 2007.))

*HTTP Protocol Data Set: server settings. Proxy for clients.

*NAT Traversal Data Set: settings for STUN, TURN etc.

*SIP Digit Maps Data Set: [I-D.petrie-sipping-voip-features-dataset] (Petrie, D., Channabasappa, S., and S. Ganesan, “The Session Initiation Protocol User Agent VoIP Features Data Set,” November 2007.)

*VoIP Features: [I-D.petrie-sipping-voip-features-dataset] (Petrie, D., Channabasappa, S., and S. Ganesan, “The Session

Initiation Protocol User Agent VoIP Features Data Set," November 2007.)

---

8. Security Considerations

TOC

Security is mostly a delivery problem. The delivery framework SHOULD provide a secure means of delivering the profile data as it may contain sensitive data that would be undesirable if it were stolen or sniffed. Storage of the profile on the profile delivery server and user agent is an implementation problem. The profile delivery server and the user agent SHOULD provide protection that prevents unauthorized access of the profile data. The profile delivery server and the user agent SHOULD enforce the access control policies defined in the profile data sets if present.

> The point of the access control construct on the data set is to provide some security policy on the visibility and ability to change sensitive properties.

Some transport mechanisms for delivery of the profile data do not provide a secure means of delivery. In addition some user agents may not have the resources to support the secure mechanism used for delivery (e.g. TLS).

---

9. IANA Considerations

TOC

XML name space registration: urn:ietf:params:xml:ns:uaprof

9.1. Content-type registration for 'application/uaprofile+xml'

TOC

> **To: ietf-types@iana.org**
> **Subject: Registration of MIME media type application/uaprofile+xml**
> **MIME media type name: application**
> **MIME subtype name: uaprofile+xml**
> **Required parameters: (none)**
> **Optional parameters: charset**
> **Encoding considerations:**
>
> Indicates the character encoding of enclosed XML. Default is UTF-8.
>
> Uses XML, which can employ 8-bit characters, depending on the character encoding used. See RFC 3023 [RFC3023]

(Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.), section 3.2.

Security considerations: This content type is designed to carry SIP user agent profile data, which may be considered private information. Appropriate precautions should be adopted to limit disclosure of this information.

Interoperability considerations: This content type provides a common format for exchange of SIP user agent profile information.

Published specification: RFC XXXX (Note to RFC Editor: Please fill in XXXX with the RFC number of this specification)

Applications which use this media type: SIP user agents and profile delivery servers.

Additional information: Magic number(s): File extension(s): Macintosh File Type Code(s):

Person & email address to contact for further information: Sam Ganesan EMail: sam.ganesan@motorola.com com

Intended usage: LIMITED USE

Author/Change controller: This specification is a proposed work item of the IETF SIPPING working group, with mailing list address: sipping@ietf.org

Other information: This media type is a specialization of application/xml [RFC3023] (Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.), and many of the considerations described there also apply to application/ uaprof+xml.

---

10. Contributors

TOC

Sumanth Channabasappa
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
US

Email: sumanth@cablelabs.com
Sam Ganesan
Motorola

80 Central Street
Boxborough, MA 01719
US

Email: sam.ganesan@motorola.com
Volker Hilt
Bell Labs/Alcatel-Lucent
791 Holmdel-Keyport Rd
Holmdel, NJ 07733
US

Email: volkerh@bell-labs.com

---

11. Acknowledgments

TOC

The WG version of the document is based on an individual draft authored by Dan Petrie, Scott Lawrence, Martin Dolly and Volker Hilt. It has been reviewed by many members of the SIPPING WG. In particular, we thank Henning Schulzrinne, Henry Sinnreich, Christian Stredicke for feedback on early versions of the document. We also thank Mary Barnes for her reviews and assistance with the current version of the document.

---

12. References

TOC

12.1. Normative References

TOC

[I-D.ietf-sip-session-policy-framework]	Hilt, V., Camarillo, G., and J. Rosenberg, “A Framework for Session Initiation Protocol (SIP) Session Policies,” draft-ietf-sip-session-policy-framework-07 (work in progress), February 2010 (TXT).
[I-D.ietf-sipping-config-framework]	Channabasappa, S., “A Framework for Session Initiation Protocol User Agent Profile Delivery,” draft-ietf-sipping-config-framework-17 (work in progress), February 2010 (TXT).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2617]	

	Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, June 1999 (TXT, HTML, XML).
[RFC3023]	Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types," RFC 3023, January 2001 (TXT).
[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 (TXT).
[RFC3688]	Mealling, M., "The IETF XML Registry," BCP 81, RFC 3688, January 2004 (TXT).
[W3C.REC-xml-names-19990114]	Hollander, D., Bray, T., and A. Layman, "Namespaces in XML," W3C REC REC-xml-names-19990114, January 1999.
[W3C.REC-xmlschema-1]	Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures," W3C REC-xmlschema-1, May 2001.
[W3C.REC-xmlschema-2]	Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes," W3C REC-xmlschema-2, May 2001.

12.2. Informative References

TOC

[I-D.ietf-sipping-media-policy-dataset]	Hilt, V., Worley, D., Camarillo, G., and J. Rosenberg, "A User Agent Profile Data Set for Media Policy," draft-ietf-sipping-media-policy-dataset-09 (work in progress), March 2010 (TXT).
[I-D.petrie-sipping-identity-dataset]	Petrie, D., Channabasappa, S., and S. Ganesan, "The Session Initiation Protocol User Agent Identity Profile Data Set," draft-petrie-sipping-identity-dataset-02 (work in progress), November 2007 (TXT).
[I-D.petrie-sipping-sip-dataset]	Channabasappa, S. and S. Ganesan, "The Core Session Initiation Protocol User Agent Protocol Data Set," draft-petrie-sipping-sip-dataset-03 (work in progress), November 2007 (TXT).
[I-D.petrie-sipping-voip-features-dataset]	Petrie, D., Channabasappa, S., and S. Ganesan, "The Session Initiation Protocol User Agent VoIP Features Data Set," draft-petrie-sipping-voip-features-dataset-02 (work in progress), November 2007 (TXT).
[RFC2141]	Moats, R., "URN Syntax," RFC 2141, May 1997 (TXT, HTML, XML).
[RFC2648]	Moats, R., "A URN Namespace for IETF Documents," RFC 2648, August 1999 (TXT).
[RFC3470]	

	Hollenbeck, S., Rose, M., and L. Masinter, “Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols,” BCP 70, RFC 3470, January 2003 (TXT, HTML, XML).

TOC

Appendix A. Relax NG SIP UA Profile Schema

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
 ns="urn:ietf:params:xml:ns:uaprof"
 datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
    <start>
       <element name="propertySet">
           <optional>
              <ref name="ElementProfileUri"/>
           </optional>
           <optional>
              <ref name="ElementProfileCredential"/>
           </optional>
           <zeroOrMore>
              <element name="profileContactUri">
                  <!-- who to contact for help with this profile -->
                  <data type="anyURI"/>
              </element>
           </zeroOrMore>
           <optional>
               <element name="profileInfo">
                   <text/>
               </element>
           </optional>
           <zeroOrMore>
               <choice>
                   <ref name="PropertySetExtension"/>
                   <ref name="ElementGenericSetting"/>
                   <ref name="ElementGenericSettingContainer"/>
               </choice>
           </zeroOrMore>
       </element>
    </start>
    <!-- example setting with all setting attributes -->
    <!-- <define name="ElementFoo">
             <element name="foo">
                 <ref name="SettingAttributes"/>
                 <text/>
              </element>
         </define>
     -->
    <define name="ElementProfileUri">
        <!-- URI to subscribe to for this profile -->
        <element name="profileUri">
            <ref name="DataSipUri"/>
        </element>
    </define>
    <define name="ElementProfileCredential">
```

```
    <!-- credentials for subscribing or getting profile -->
    <element name="profileCredential">
        <ref name="DataCredential"/>
    </element>
</define>
<define name="PropertySetExtension">
    <!-- place to add new settings in other namespaces -->
    <empty/>
</define>
<define name="ElementGenericSettingContainer">
    <element>
        <anyName>
            <except>
                <nsName ns="urn:ietf:params:xml:ns:uaprof"/>
                <nsName ns=""/>
            </except>
        </anyName>
        <ref name="SettingContainerAttributes"/>
        <zeroOrMore>
            <ref name="AttributeGeneric"/>
        </zeroOrMore>
      <!-- container can have containers or settings not both -->
        <choice>
            <zeroOrMore>
                <ref name="ElementGenericSetting"/>
            </zeroOrMore>
            <zeroOrMore>
                <ref name="ElementGenericSettingContainer"/>
            </zeroOrMore>
        </choice>
    </element>
</define>
<define name="ElementGenericSetting">
    <element>
        <anyName>
            <except>
                <nsName ns="urn:ietf:params:xml:ns:uaprof"/>
                <nsName ns=""/>
            </except>
        </anyName>
        <ref name="SettingAttributes"/>
        <zeroOrMore>
            <ref name="AttributeGeneric"/>
        </zeroOrMore>
        <zeroOrMore>
            <choice>
                <text/>
                <ref name="ElementGenericSetting"/>
            </choice>
```

```
        </zeroOrMore>
    </element>
</define>
<define name="AttributeGeneric">
    <attribute>
        <anyName>
            <except>
                <nsName ns="urn:ietf:params:xml:ns:uaprof"/>
                <nsName ns=""/>
            </except>
        </anyName>
    </attribute>
</define>
<define name="DataCredential">
    <element name="realm">
        <text/>
    </element>
    <element name="authUser">
        <text/>
    </element>
    <choice>
        <element name="a1Digest">
            <data type="string">
                <param name="pattern">[0-9,a-f]{32,32}</param>
            </data>
        </element>
        <element name="password">
            <text/>
        </element>
    </choice>
</define>
<define name="DataSipUri">
    <choice>
        <data type="anyURI">
            <param name="pattern">sip:.*</param>
        </data>
        <data type="anyURI">
            <param name="pattern">sips:.*</param>
        </data>
    </choice>
</define>
<define name="DataSipNameAddr">
    <choice>
        <data type="anyURI"><!-- need to tighten this up -->
            <param name="pattern">".*"?&lt;?sip:.*</param>
        </data>
        <data type="anyURI">
            <param name="pattern">".*"?&lt;?sips:.*</param>
        </data>
```

```
    </choice>
</define>
<define name="SettingContainerAttributes">
    <optional>
        <attribute name="excludedPolicy">
            <ref name="DataPolicies"/>
        </attribute>
    </optional>
</define>
<define name="SettingAttributes">
    <interleave>
       <optional>
           <ref name="AttributePolicy"/>
       </optional>
       <optional>
           <ref name="AttributeVisibility"/>
       </optional>
       <optional>
           <ref name="AttributeDirection"/>
       </optional>
       <optional>
           <ref name="AttributeQ"/>
       </optional>
    </interleave>
</define>
<define name="AttributePolicy">
   <attribute name="policy">
       <ref name="DataPolicies"/>
   </attribute>
</define>
<define name="DataPolicies">
    <choice>
        <value></value><!-- default of allow -->
        <value>allow</value>
        <value>disallow</value>
    </choice>
</define>
<define name="AttributeVisibility">
    <attribute name="visibility">
       <choice>
           <value></value><!-- default of user -->
           <value>user</value>
           <value>admin</value>
       </choice>
    </attribute>
</define>
<define name="AttributeDirection">
    <attribute name="direction">
       <choice>
```

```
                <value></value><!-- default of sendrecv -->
                <value>sendrecv</value>
                <value>sendonly</value>
                <value>recvonly</value>
            </choice>
        </attribute>
    </define>
    <define name="AttributeQ">
        <attribute name="q">
            <data type="float">
                <!-- default of 0.5 -->
                <param name="minInclusive">0</param>
                <param name="maxInclusive">1</param>
            </data>
        </attribute>
    </define>
    <define name="DataIpPort">
        <data type="integer">
            <param name="minInclusive">1</param>
            <param name="maxInclusive">65535</param>
        </data>
    </define>
    <define name="DataIpTransport">
        <choice>
            <value></value><!-- default of UDP -->
            <value>UDP</value>
            <value>TCP</value>
            <value>TLS</value>
            <value>DTLS</value>
            <value>SCTP</value>
        </choice>
    </define>
</grammar>
```

Appendix B. Use Cases

TOC

In the following use case scenarios the device profile is provided by the device owner/manager. The owner/manager may be a service provider, an enterprise or a user administering the device setup. The user is assumed to be the end user operating the user agent. In the scenarios that the user profile is provided, the user profile contains user specific properties that the end user has set directly or indirectly through an administration process. The local network profiles represent the suggested policy behavior that the local network operator would

like user agents to adhere to [I-D.ietf-sip-session-policy-framework] (Hilt, V., Camarillo, G., and J. Rosenberg, “A Framework for Session Initiation Protocol (SIP) Session Policies,” February 2010.) . From a security perspective, the local network operator cannot trust the user agent to follow the local network profile policy. The local network operator must use a means external to the user agent to enforce these policies. The local network profile is intended to express to the user agent, the policies that the user agent should follow if the user agent wants to function properly in the local network.
Two different use cases are developed and discussed below. Similar use cases can be developed for individual datasets. For example, analysis of Transport protocol settings for SIP can be carried out in exactly the same fashion as the codec use case described below and a set of derived requirements to drive the schema for the associated datset can be arrived at.

B.1. Outbound Proxy Setting

TOC

In the case of the outbound proxy, it is unlikely that the user would want to influence the outbound proxy for SIP signaling. The device owner/manager or the local network operator are likely to want to set the outbound proxy property. The device profile may define an outbound proxy so that the device owner/manager can monitor all signaling. The local network operator also defines an outbound proxy because the proxy allows the SIP signaling to get through a NAT or firewall.
Two possible solutions to this problem are listed.

- Define a policy where the local network profile overrides the device profile. In this approach the local network profile wins.

- Aggregate the outbound proxies. In this scenario SIP messages would be sent with a pre-populated route set that had two hops. First the outbound proxy set in the local network profile, then the outbound proxy set in the device profile.

The aggregation approach is closest to solving the requirements to the use case above. By aggregating the two outbound proxies, the local network provided outbound proxy allows the signaling to get out of the local network and the device profile provided outbound proxy is able to monitor all SIP signaling from the user agent.

TOC

B.2. Codec Settings

Use cases for the codec properties are likely one of the more complicated sets of properties with respect to merging and constraining across more than one profile. There are reasonable scenarios where requirements can be rationalized that the device, user and local network profiles may each wish to express preferences and constraints on permitted codecs. Without getting into details or syntax of the codec properties, it is assumed that codec properties will need to express a codec definition and a preference order. This is the order that these codecs will be put in SDP for codec negotiation purposes. The following scenarios illustrate some possible combinations of sources of codec properties from the device, user and local network profiles.

B.2.1. Codec Setting Not Set

TOC

In the scenario where a device has no profiles or the profiles contain no codec properties, the device will enable a default set and preference order of codecs, which could be a subset of the codecs the device is capable of supporting. The default set and preference order of codecs is a implemention specific.

B.2.2. Codec Set in Device Profile

TOC

This scenario assumes that the device profile is the only source of codec properties.
The codecs in the device profile may differ from the set of codecs supported by the device, due to administrative constraints on codec usage.
In this scenario the device profile data will dictate the ordered list of codecs to be applied. The use agent will ignore codec types included in the profile that the user agent does not support.

B.2.3. Set in Device and User Profiles

TOC

This scenario covers the case where both the device profile and the user profile provide an ordered list of codecs. The user may prefer a higher quality codec to be used, if available. Thereby the user profile data may provide an ordered list of codecs to be applied. The device profile also specifies a list of codecs and a default preference order.

The merging of the data sources is as follows:

*The ordering of the codecs will be determined from the user profile data, which overrides the codec preference ordering from the device profile data.

*The set of codecs that may be applied, are the codecs listed in the user data constrained by the list of codecs from the device profile data.

The case in which none of the codecs in the resulting merged profile datasets are supported by the device, the profile data constitutes a misconfiguration between device and user profiles. It may not be possible to successfully establish a session in this case. It is suggested that the user agent provide feedback to the user indicating the misconfiguration. The user agent may also attempt to function in the network by ignoring one or more of the profile constraints.

B.2.4. Set in Device and Local Profiles

TOC

In this scenario both the local network profile and the device profile each provide an ordered set of codecs. Both the local network operator and the device provider may feel the need to constrain and order the set of codecs used. This scenario is very much alike to the previous scenario and may be resolved using a similar method. However, it likely that the local network codec preferences will override and constrain the device profile, given the caveat that in the circumstances where the resulting ordered set of codecs is an empty set. In this case there is a misconfiguration/incompatibility between the device profile and the local network profile with regard to the codecs, which may render the device non functional. The user agent may attempt to function in the network by ignoring one or more of the profile constraints.

B.2.5. Set in Device, User and Local Profiles

TOC

In this scenario all profiles namely, device, user and local network profiles, provide an ordered set of codecs as preferences. For example, these may be the result of device capabilities, user's preference for higher quality media and the network providers desire to constrain

bandwidth usage and or enforce uniformity of codec usage. The data sources could be merged as follows:

*The ordering of the codecs will be determined from the user profile data, which overrides the ordering from the device profile data.

*The set of codecs that may be used are the codecs listed in the device profile data, constrained by the list of codecs from the user profile data and further constrained by the list of codecs from the local network profile data.

A resulting null set of codecs would imply a misconfiguration and may prevent the device from functioning under these circumstances. The user agent may also attempt to function in the network by ignoring one or more of the profile constraints.

B.2.6. Example Derived Requirements

TOC

An example set of derived requirements for the codec definition is presented here. These requirements in turn would drive the profile definition for codec usage.

1. The list of codecs in the device profile data that get applied is the subset of the codecs supported by the device. Codecs listed in profiles that are not supported by the device are ignored.

2. The device profile data will have a default ordered list of codecs, which implies a preference order to be used in the sdp offer.

3. The user profile data may provide an ordered list of user preferred codecs. The ordering of the codecs in the user profile data will override the ordering of the codecs in the device profile data. The user list of codecs may further constrain the list of codecs to be used.

4. The local network profile data may provide a list of codecs supported. This list will further constrain the list of codecs that may be offered.

5. The application profile data containing codec data will be ignored.

6. The profiles need the ability to express codecs that may be used and codecs that should not be used.

Authors' Addresses

TOC

	Martin Dolly
	AT&T
	200 Laurel Ave.
	Middletown, NJ
	US
Phone:	
Email:	mdolly@att.com
URI:	
	Daniel Petrie
	SIPez LLC
	34 Robbins Rd.
	Arlington, MA 02476
	US
Phone:	+1 617 273 4000
Email:	dan.ietf AT SIPez DOT com
URI:	http://www.SIPez.com/
	Dale R. Worley
	Nortel Networks Corp.
	600 Technology Park Dr.
	Billerica, MA 01821
	US
Phone:	+1 978 288 5505
Email:	dworley@nortel.com
URI:	http://www.nortel.com