SIMPLE
Internet-Draft
Expires: October 29, 2004

J. Rosenberg
dynamicsoft

April 30, 2004

Presence Authorization Rules
draft-ietf-simple-presence-rules-00

Status of this Memo

Copyright Notice

Abstract

Authorization is a key function in presence systems. Authorization policies, also known as authorization rules, specify what presence

information can be given to which watchers, and when. This specification defines an Extensible Markup Language (XML) document format for expressing presence authorization rules. Such a document can be manipulated by clients using the XML Configuration Access Protocol (XCAP), although other techniques are permitted.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Structure of Permission Statements 5
3.1 Conditions 5
3.1.1 Identity 5
3.1.2 Anonymous 5
3.2 Actions 6
3.2.1 Subscription Handling 6
3.3 Transformations 6
3.3.1 Inclusion Set 7
3.3.2 Provide Contact URI 8
3.3.3 Provide Activity 8
3.3.4 Provide Tuples 8
3.3.5 Provide Class 8
3.3.6 Provide Contact Type 8
3.3.7 Idle Detail 9
3.3.8 Provide Idle 9
3.3.9 Provide PlaceType 9
3.3.10 Provide Privacy 9
3.3.11 Provide Relationship 10
3.3.12 Provide Sphere 10
3.3.13 Provide Unknown Status 10
4. Example Document 12
5. XML Schema 13
6. Schema Extensibility 15

7. XCAP Usage 16
7.1 Application Unique ID 16
7.2 Structure of Permission Statements 16
7.3 Additional Constraints 16
7.4 Naming Conventions 16
7.5 Authorization Policies 16
7.6 XML Schema 16
8. Security Considerations 17
9. IANA Considerations 18
9.1 XCAP Application Usage ID 18
9.2 URN Sub-Namespace Registration 18
9.3 XML Schema Registrations 19
Normative References 20
Informative References 21
Author's Address 21
Intellectual Property and Copyright Statements 22

1. Introduction

The Session Initiation Protocol (SIP) for Instant Messaging and Presence (SIMPLE) specifications allow a user, called a watcher, to subscribe to another user, called a presentity [13], in order to learn their presence information [16]. This subscription is handed by a presence agent. In order to process the subscription, the presence agent must make a determination about whether the subscription is authorized. This authorization decision includes whether or not to accept the subscription, but also includes decisions about when the watcher should receive notifications, and when it does receive them, what the content of those notifications should be.

Typically, the authorization decision will be a combination of the authorization policies of the service provider, combined with the authorization policies of the presentity. In order for the PA to

compute the final authorization decision, it needs access to the presentity's authorization policies.

[10] specifies a framework for representing such authorization policies, and is applicable to systems such as geo-location and presence. In that framework, an authorization document is a sequence of rule elements. Each rule element contains a conditions element, an actions element, and a transformations element. The conditions element specifies under what conditions the rule is to be applied to a subscription request. The actions element tells the server what actions to take against the request. The transformations element indicates how the presence data is to be manipulated before being presented to that watcher. [10] identifies a small number of specific conditions, actions and permissions common to presence and location services, and leaves it to other specifications, such as this one, to fill in usage specific details.

These documents can be manipulated by clients using several means. One such mechanism is the XML Configuration Access Protocol (XCAP) [2]. This specification defines the details necessary for using XCAP to manage presence authorization documents.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY",

and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant implementations.

3. Structure of Permission Statements

A permission statement is an XML document, formatted according to the schema defined in [10]. As described in [10], this document is composed of three parts - conditions, actions, and transformations. Each action or transformation, which is also called an attribute, has the property of being a positive grant of information to the watcher. As a result, there is a well-defined mechanism for combining actions and transformations obtained from several sources. This mechanism is privacy safe, since the lack of any action or transformation can only result in less information being presented to a watcher.

This section defines the new conditions, actions and transformations defined by this specification.

3.1 Conditions

3.1.1 Identity

Although the "identity" element is defined in [10], that specification indicates that the interpretation of the "uri" element depends on the specific protocol in use and its authentication mechanisms. This sub-section defines that interpretation for systems based on [16] [[NOTE: "uri" is a bad choice of name for this element, since its not a URI. That will be corrected in a subsequent revision of the common policy document.]]

For requests that are authenticated using SIP [9] digest authentication [8], the user part of the URI is matched against the username attribute in the Authorization request header field. The domain part of the URI is matched against the realm attribute in the Authorization request header field.

For requests that are authenticated using [17], the username and domain part of the URI are matched against the user and host parts of the SIP URI in the P-Asserted-Identity header field.

For any other authentication mechanism in SIP which might be

identified in other specifications, a similar pattern would be followed.

3.1.2 Anonymous

The "anonymous" element, which is a boolean type, indicates whether or not the request was authenticated using the "anonymous" username defined in RFC 3261. It allows for the presentity to specify policies based on whether or not the requestor was anonymous.

3.2 Actions

3.2.1 Subscription Handling

The "sub-handling" element defines the action that the server is to take in the processing of this subscription. It is an enumerated integer type. The defined values are:

block: This action tells the server to reject the subscription. It has the value of zero, and it represents the default value. No value of the sub-handling element can ever be lower than this. Strictly speaking, it is not necessary to every include an explicit block action, since the default in the absence of any action will be block. However, it is included for completeness.

confirm: This action tells the server to place the subscription in the "pending" state, and await input from the presentity to determine how to proceed. It has a value of one.

polite-block: This implies that the subscription is accepted, but inaccurate presence data is provided to the watcher. The specific mechanism for generating inaccurate presence data is at the

discretion of the implementation. Providing a single tuple [3] with a basic status of closed represents one reasonable choice. This action has a value of two.

allow: This implies that the subscription is accepted, and accurate information, within the constraints of the transformations specified by the rule, is supplied. This action has a value of three.

NOTE WELL: Placing a value of block for this element does not guarantee that a subscription is denied! If any matching rule has any other value for this element, the subscription will receive treatment based on the maximum of those other values. This is based on the combining rules defined in [10].

3.3 Transformations

Each transformation defined here defines the visibility a watcher is granted to a particular component of the presence document. Many of these transformations are set types, as defined in [10]. In particular, they are a specific kind of set which is defined here, called an inclusion set.

3.3.1 Inclusion Set

An inclusion set is used to identify a set of tuples in which a particular presence attribute will appear. Each element in the set is itself a presence attribute and value. If a particular presence attribute and value exists in the set, it means that any tuple in the document that has that particular presence attribute with that particular value will belong to the inclusion set.

As an example, consider the following inclusion set:

- placetype=home

- class=friend

If the transformation for the RPID sphere element was an inclusion set, and its value was the set above, it would mean that the RPID sphere element would be included in any tuple that had a placetype equal to home or a class equal to friend.

When represented in XML, an inclusion set is a XML data type. The content of any element of this type is either the all-tuples element, or a tuples-whose element. The all-tuples element is a short-hand notation for any set that would result in the selection of every tuple in the presence document. The tuples-whose element contains a sequence of an RPID element and its value. Each RPID element/value is an entry in the set. When an RPID element/value is present in the set, it means that presence tuples that are described with that RPID element with that value are selected. Some RPID elements include "from" and "until" qualifiers; these are ignored for the purposes of selection.

Because the inclusion set is a set type, composition rules follow the union operation. This means that if one permission grants access to the sphere element in tuples whose placetype is home, and another permission grants access to the sphere element in tuples whose class is friend, the result is that the sphere element will be provided in any tuple that has a placetype of home or a class of friend.

The default value for any element of this type is empty, meaning that the presence attribute would not be included in any tuples.

The inclusion set allows allows for extensibility by allowing other members of the set which identify a tuple in ways besides the value of a presence attribute within that tuple.

3.3.2 Provide Contact URI

The "provide-contact-uri" permission indicates whether or not the Contact URI for tuples is presented to the watcher. This element is of the inclusion-set type, which means it is a set whose members indicate characteristics that identify which tuples the contact URI should be included in.

OPEN ISSUE: There is no way to define that a contact-uri applies to all tuples without a class label. If we want that, we need to add another selection operation to the inclusion set type, which selects tuples in which a particular presence attribute is, or is not, present.

3.3.3 Provide Activity

This permission controls access to the "activity" element defined in [11]. The name of the element is "provide-activity", and it is of the inclusion set type.

3.3.4 Provide Tuples

This permission controls access to tuples. It indicates which tuples should be present in the document sent to a watcher. It is of the inclusion-set type. The name of the element is "provide-tuples".

When a tuple is included, this means that its basic status and note elements are included. Presence of the contact URI depends on the provide-contact-uri permission, as does the presence of any other presence attributes.

OPEN ISSUE: Because the default for the inclusion-set type is the empty set, no tuples will be included in the presence document by

default, unless they are specifically included. We may want to define a different type that has, as default, some agreed-upon definition of baseline tuples.

3.3.5 Provide Class

This permission controls access to the "class" element within the PIDF document. The name of the element is "provide-class", and it is of the inclusion-set data type.

3.3.6 Provide Contact Type

This permission, "provide-contact-type" controls access to the

"contact-type" element within the PIDF document. It is of the inclusion-set data type.

3.3.7 Idle Detail

This permission, "idle-detail" defines the level of detail present in the "idle" element within the PIDF document, wherever it appears. A separate permission, provide-idle, determines where it will appear.

idle-detail is an enumerated integer type. Its values "no-time", with a value of zero, that indicates that the "idle" element is to be passed on to watchers, but without the specific duration for which the user has been idle, and "full", with a value of one, that indicates that the "idle" element is to be passed onto watchers, and should include a specific duration if available.

The default value for this element is zero, meaning that no durations

are provided by default.

3.3.8 Provide Idle

This permission, "provide-idle", controls access to the RPID "idle" element in the presence document. It is of inclusion-set type.

Because the "provide-idle" and "idle-detail" transformations are separate and orthogonal, it is not possible to define transformations which give a certain level of detail in some tuples, and a different level in other tuples.

> OPEN ISSUE: Is this constraint OK? It wasnt clear how to fix this with the defined data types.

3.3.9 Provide PlaceType

This permission, "provide-placetype" controls access to the "placetype" element within the PIDF document. It is of the inclusion-set type.

> OPEN ISSUE: Do we want any finer grained permissions than just whether to include, or not include, placetype in the presence document?

3.3.10 Provide Privacy

This permission, "provide-privacy" controls access to the "privacy" element within the PIDF document. It is of the inclusion-set type.

3.3.11 Provide Relationship

This permission, "provide-relationship" controls access to the "relationship" element within the PIDF document. It is of the inclusion-set type.

3.3.12 Provide Sphere

This permission, "provide-sphere" controls access to the "sphere" element within the PIDF document. It is of the inclusion-set type.

3.3.13 Provide Unknown Status

It is important that systems be allowed to include proprietary or new presence information, and that users be able to set permissions for that status information, without requiring an upgrade of the presence server and authorization system. For this reason, the "provide-unknown-status" permission is defined. This permission indicates that the unknown presence status with the given name (supplied as mandatory attribute of the "provide-presence-status" element) can be placed in the indicated tuples.

The "provide-unknown-status" element is of the "unknown-inclusion-set" type. This type is identical to "inclusion-set", except elements of this type have to include the mandatory name attribute, identifying the presence status element to which they apply. The value of the name attribute MUST be a qualified element name (meaning that the namespace prefix MUST be included), which will be matched to all unknown child elements of the PIDF "status" element with the same qualified name. In this context, "unknown" means that the presence server is not aware of any schemas that define authorization policies for that element. By definition, this will exclude the "provide-unknown-status" rule from being applied to any of the presence status extensions defined by RPID.

Another consequence of this definition is that the interpretation of the "provide-unknown-status" element can change should the presence server be upgraded with a new schema that defines authorization rules for elements included in a "provide-unknown-status". The "provide-unknown-status" permissions for those elements will then be ignored, resulting in a removal of those elements from presence documents sent to watchers. The system remains privacy safe, but behavior might not be as expected. Developers of systems which allow

clients to set policies are advised to check the capabilities of the server, as defined in [15], before uploading a new authorization document, to make sure that the behavior will be as expected.

The content of the "provide-unknown-status" element indicates the

tuples in which that unknown element will be included. These tuples are selected exactly as defined for elements of the type "inclusion-set" as discussed in Section 3.3.1.

4. Example Document

The following presence authorization document specifies permissions for the user "user@example.com". The permissions indicate that this user should receive all tuples, and within those tuples, the activity element should only be included in tuples whose class is friend.

```
<?xml version="1.0" encoding="UTF-8"?>
<cr:ruleset
 xmlns:cr="urn:ietf:params:xml:ns:common-policy"
 xmlns:rpid="urn:ietf:params:xml:ns:rpid"
 xmlns="urn:ietf:params:xml:ns:pres-rules"
 xmlns:cr="urn:ietf:params:xml:ns:common-policy"
 xmlns:rs="urn:ietf:params:xml:ns:pidf:status:rpid-status"
 xmlns:ts="urn:ietf:params:xml:ns:pidf:rpid-tuple"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <cr:rule id="1">
  <cr:conditions>
   <cr:identity>
    <cr:uri>user@example.com</cr:uri>
```

```
      </cr:identity>
     </cr:conditions>
     <cr:actions>
      <sub-handling>allow</sub-handling>
     </cr:actions>
     <cr:transformations>
      <provide-tuples>
       <all-tuples></all-tuples>
      </provide-tuples>
      <provide-activity>
        <tuples-whose>
          <ts:class>friend</ts:class>
        </tuples-whose>
      </provide-activity>
     </cr:transformations>
    </cr:rule>
   </cr:ruleset>
```


5. XML Schema

```
   <?xml version="1.0" encoding="UTF-8"?>
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:pres-rules"
     xmlns:ts="urn:ietf:params:xml:ns:pidf:rpid-tuple"
     xmlns:rs="urn:ietf:params:xml:ns:pidf:status:rpid-status"
     xmlns:pr="urn:ietf:params:xml:ns:pres-rules"
     xmlns:cr="urn:ietf:params:xml:ns:common-policy"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace="urn:ietf:params:xml:ns:common-policy"/>
```

```
    <xs:import namespace="urn:ietf:params:xml:ns:pidf:status:rpid-status"/>
    <xs:import namespace="urn:ietf:params:xml:ns:pidf:rpid-tuple"/>
    <xs:element name="anonymous" type="xs:boolean"
      substitutionGroup="cr:condition"/>
    <xs:complexType name="inclusion-set">
     <xs:choice>
      <xs:element name="all-tuples"/>
      <xs:sequence>
       <xs:element name="tuples-whose">
        <xs:complexType>
         <xs:sequence>
          <xs:element ref="rs:placetype" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="rs:privacy" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="rs:relationship" minOccurs="0" maxOccurs="unbounded"
          <xs:element ref="rs:sphere" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="ts:class" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="ts:contact-type" minOccurs="0" maxOccurs="unbounded"
         </xs:sequence>
        </xs:complexType>
       </xs:element>
       <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:choice>
    </xs:complexType>
    <xs:element name="provide-contact-uri" type="pr:inclusion-set"
     substitutionGroup="cr:transformation"/>
    <xs:element name="provide-activity" type="pr:inclusion-set"
     substitutionGroup="cr:transformation"/>
    <xs:element name="provide-tuples" type="pr:inclusion-set"
     substitutionGroup="cr:transformation"/>
    <xs:element name="provide-class" type="pr:inclusion-set"
     substitutionGroup="cr:transformation"/>
    <xs:element name="provide-contact-type" type="pr:inclusion-set"
     substitutionGroup="cr:transformation"/>
    <xs:element name="idle-detail" substitutionGroup="cr:transformation">
     <xs:simpleType>
```



```
      <xs:restriction base="xs:token">
       <xs:enumeration value="no-time"/>
       <xs:enumeration value="full"/>
```

```
    </xs:restriction>
   </xs:simpleType>
  </xs:element>
  <xs:element name="provide-idle" type="pr:inclusion-set"
   substitutionGroup="cr:transformation"/>
  <xs:element name="provide-placetype" type="pr:inclusion-set"
   substitutionGroup="cr:transformation"/>
  <xs:element name="provide-privacy" type="pr:inclusion-set"
   substitutionGroup="cr:transformation"/>
  <xs:element name="provide-relationship" type="pr:inclusion-set"
   substitutionGroup="cr:transformation"/>
  <xs:element name="provide-sphere" type="pr:inclusion-set"
   substitutionGroup="cr:transformation"/>
  <xs:element name="sub-handling" substitutionGroup="cr:action">
   <xs:simpleType>
    <xs:restriction base="xs:token">
     <xs:enumeration value="block"/>
     <xs:enumeration value="confirm"/>
     <xs:enumeration value="polite-block"/>
     <xs:enumeration value="allow"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:element>
  <xs:element name="provide-unknown-status" type="pr:unknown-inclusion-set"
   substitutionGroup="cr:transformation"/>
  <xs:complexType name="unknown-inclusion-set">
   <xs:complexContent>
    <xs:extension base="pr:inclusion-set">
     <xs:attribute name="name" type="xs:token" use="required"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:schema>
```

6. Schema Extensibility

It is anticipated that future changes to this specification are accomplished through extensions that define new types of permissions. These extensions MUST exist within a different namespace. Furthermore, the schema defined above and the namespace for elements defined within it MUST NOT be altered by future specifications. Changes in the basic schema, or in the interpretation of elements within that schema, may result in violations of user privacy due to mis-interpretation of documents.

7. XCAP Usage

The following section defines the details necessary for clients to manipulate presence authorization documents from a server using XCAP.

7.1 Application Unique ID

XCAP requires application usages to define a unique application usage ID (AUID) in either the IETF tree or a vendor tree. This specification defines the "rules" AUID within the IETF tree, via the IANA registration in Section 9.

7.2 Structure of Permission Statements

The structure of permission statements is defined in Section 3.

7.3 Additional Constraints

There are no additional constraints defined by this specification.

7.4 Naming Conventions

When a presence agent receives a subscription for some user foo within a domain, it will look for all documents within http://[xcap root services uri]/rules/users/foo, and use all documents found beneath that point to guide authorization policy.

7.5 Authorization Policies

This application usage does not modify the default XCAP authorization policy, which is that only a user can read, write or modify their own documents. A server can allow priveleged users to modify documents that they don't own, but the establishment and indication of such policies is outside the scope of this document.

7.6 XML Schema

The XML schema is defined in Section 5.

8. Security Considerations

Presence authorization policies contain very sensitive information. They indicate which other users are "liked" or "disliked" by a user. As such, when these documents are transported over a network, they SHOULD be encrypted.

Modification of these documents by an attacker can disrupt the service seen by a user, often in subtle ways. As a result, when these documents are transported, the transport SHOULD provide authenticity and message integrity.

In the case where XCAP is used to transfer the document, clients SHOULD use HTTP over TLS, and servers SHOULD define the root services URI as an https URI. The server SHOULD authenticate the client over the resulting TLS connection using HTTP digest.

9. IANA Considerations

There are several IANA considerations associated with this specification.

9.1 XCAP Application Usage ID

This section registers an XCAP Application Usage ID (AUID) according to the IANA procedures defined in [2].

Name of the AUID: pres-rules

Description: Presence rules are documents that describe the permissions that a presentity [13] has granted to users that seek to watch their presence.

9.2 URN Sub-Namespace Registration

This section registers a new XML namespace, per the guidelines in [12]

URI: The URI for this namespace is urn:ietf:params:xml:ns:pres-rules.

Registrant Contact: IETF, SIMPLE working group, (simple@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Presence Rules Namespace</title>
</head>
<body>
```

```
    <h1>Namespace for Permission Statements</h1>
    <h2>urn:ietf:params:xml:ns:pres-rules</h2>
    <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
  </body>
  </html>
  END
```


9.3 XML Schema Registrations

This section registers an XML schema per the procedures in [12].

URI: please assign.

Registrant Contact: IETF, SIMPLE working group, (simple@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).

The XML for this schema can be found as the sole content of Section 5.

Normative References

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

[2] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-01 (work in progress), October 2003.

[3] Sugano, H. and S. Fujimoto, "Presence Information Data Format (PIDF)", draft-ietf-impp-cpim-pidf-08 (work in progress), May 2003.

[4] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C FirstEdition REC-xml-20001006, October 2000.

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

[6] Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[7] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[8] 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.

[9] 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.

[10] Schulzrinne, H., "Common Policy", draft-ietf-geopriv-common-policy-00 (work in progress), February 2004.

[11] Schulzrinne, H., "RPID -- Rich Presence Information Data Format", draft-ietf-simple-rpid-00 (work in progress), July 2003.

[12] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

Informative References

[13] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[14] Day, M., Aggarwal, S., Mohr, G. and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

[15] Rosenberg, J., "An Extensible Markup Language (XML) Representation for Expressing Presence Policy Capabilities", draft-rosenberg-simple-pres-policy-caps-00 (work in progress), February 2004.

[16] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", draft-ietf-simple-presence-10 (work in progress), January 2003.

[17] Jennings, C., Peterson, J. and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, November 2002.

Author's Address

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

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