Network Working Group
Request for Comments: 4827
Category: Standards Track

M. Isomaki
E. Leppanen
Nokia

May 2007

An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Manipulating Presence Document Contents

Status of This Memo

Copyright Notice

Abstract

This document describes a usage of the Extensible Markup Language (XML) Configuration Access Protocol (XCAP) for manipulating the contents of Presence Information Data Format (PIDF) based presence documents. It is intended to be used in Session Initiation Protocol (SIP) based presence systems, where the Event State Compositor can use the XCAP-manipulated presence document as one of the inputs on which it builds the overall presence state for the presentity.

Table of Contents

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, in order to learn its presence information [7]. The presence data model has been specified in [10]. The data model makes a clean separation between person-, service-, and device-related information.

A SIP-based mechanism, SIP PUBLISH method, has been defined for publishing presence state [4]. Using SIP PUBLISH, a Presence User Agent (PUA) can publish its view of the presence state, independently of and without the need to learn about the states set by other PUAs. However, SIP PUBLISH has a limited scope and does not address all the requirements for setting presence state. The main issue is that SIP PUBLISH creates a soft state that expires after the negotiated lifetime unless it is refreshed. This makes it unsuitable for cases where the state should prevail without active devices capable of refreshing the state.

There are three main use cases where setting of permanent presence state that is independent of activeness of any particular device is useful. The first case concerns setting person-related state. The presentity would often like to set its presence state even for periods when it has no active devices capable of publishing available. Good examples are traveling, vacations, and so on. The second case is about setting state for services that are open for communication, even if the presentity does not have a device running that service online. Examples of these kinds of services include e-mail, Multimedia Messaging Service (MMS), and Short Message Service (SMS). In these services, the presentity is provisioned with a server that makes the service persistently available, at least in certain forms, and it would be good to be able to advertise this to the watchers. Since it is not realistic to assume that all e-mail, MMS, or SMS servers can publish presence state on their own (and even if this were possible, such state would almost never change), this has to be done by some other device. And since the availability of the service is not dependent on that device, it would be impractical to require that device to be constantly active just to publish such availability. The third case concerns setting the default state of any person, service, or device in the absence of any device capable of actively publishing such state. For instance, the presentity might want to advertise that his or her voice service is currently closed, just to let the watchers know that such service might be open at some point. Again, this type of default state is independent of any particular device and can be considered rather persistent.

Even though SIP PUBLISH remains the main way of publishing presence state in SIMPLE-based presence systems and is especially well-suited for publishing dynamic state (which presence mainly is), it needs to be complemented by the mechanism described in this document to address the use cases presented above.

XML Configuration Access Protocol (XCAP) [2] allows a client to read, write, and modify application configuration data stored in XML format on a server. The data has no expiration time, so it must be explicitly inserted and deleted. The protocol allows multiple clients to manipulate the data, provided that they are authorized to do so. XCAP is already used in SIMPLE-based presence systems for manipulation of presence lists and presence authorization policies. This makes XCAP an ideal choice for doing device-independent presence document manipulation.

This document defines an XML Configuration Access Protocol (XCAP) application usage for manipulating the contents of presence document. Presence Information Document Format (PIDF) [3] is used as the presence document format, since the event state compositor already has to support it, as it is used in SIP PUBLISH.

Section 3 describes in detail how the presence document manipulated with XCAP is related to soft state publishing done with SIP PUBLISH.

XCAP requires application usages to standardize several pieces of information, including a unique application usage ID (AUID) and an XML schema for the manipulated data. These are specified starting from Section 4.

2. Conventions

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.

Comprehensive terminology of presence and event state publishing is provided in "Session Initiation Protocol (SIP) Extension for Event State Publication" [4].

3. Relationship with Presence State Published Using SIP PUBLISH

The framework for publishing presence state is described in Figure 1. A central part of the framework is the event state compositor element, whose function is to compose presence information received from several sources into a single coherent presence document.

The presence state manipulated with XCAP can be seen as one of the information sources for the compositor to be combined with the soft state information published using SIP PUBLISH. This is illustrated in Figure 1. It is expected that, in the normal case, there can be several PUAs publishing their separate views with SIP PUBLISH, but only a single XCAP manipulated presence document. As shown in the figure, multiple XCAP clients (for instance, in different physical devices) can manipulate the same document on the XCAP server, but this still creates only one input to the event state compositor. The XCAP server stores the XCAP manipulated presence document under the "users" tree in the XCAP document hierarchy. See Section 9 for details and Section 11 for an example.

As individual inputs, the presence states set by XCAP and SIP PUBLISH are completely separated, and it is not possible to directly manipulate the state set by one mechanism with the other. How the compositor treats XCAP-based inputs with respect to SIP PUBLISH-based inputs is a matter of compositor policy, which is beyond the scope of this specification. Since the SIP PUBLISH specification already mandates the compositor to be able to construct the overall presence state from multiple inputs, which may contain non-orthogonal (or in some ways even conflicting) information, this XCAP usage does not impose any new requirements on the compositor functionality.

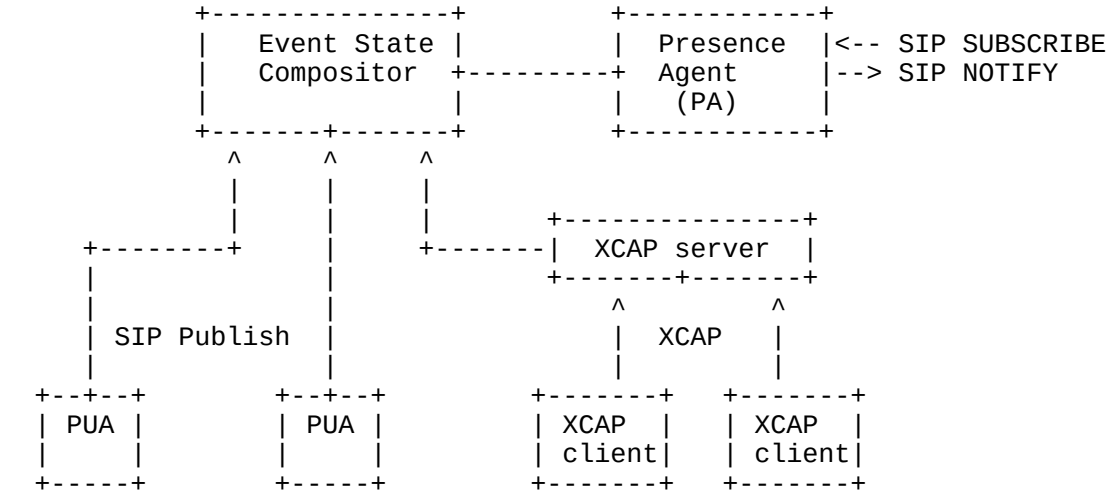

Figure 1: Framework for Presence Publishing and Event State Composition

The protocol interface between XCAP server and the event state compositor is not specified here.

4. Application Usage 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 'pidf-manipulation' AUID within the IETF tree, via the IANA registration in Section 13.

5. MIME Type

The MIME type for this application usage is 'application/pidf+xml'.

6. Structure of Manipulated Presence Information

The XML Schema of the presence information is defined in the Presence Information Data Format (PIDF) [3]. The PIDF also defines a mechanism for extending presence information. See [8], [9], [11], and [12] for currently defined PIDF extensions and their XML Schemas.

The namespace URI for PIDF is 'urn:ietf:params:xml:ns:pidf' which is also the XCAP default document namespace.

7. Additional Constraints

There are no constraints on the document beyond those described in the XML schemas (PIDF and its extensions) and in the description of PIDF [3].

8. Resource Interdependencies

There are no resource interdependencies beyond the possible interdependencies defined in PIDF [3] and XCAP [2] that need to be defined for this application usage.

9. Naming Conventions

The XCAP server MUST store only a single XCAP manipulated presence document for each user. The presence document MUST be located under the "users" tree, using filename "index". See an example in Section 11.

10. Authorization Policies

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

11. Example

The section provides an example of a presence document provided by an XCAP Client to an XCAP Server. The presence document illustrates the situation where a (human) presentity has left for vacation, and before that, has set his presence information so that he is only available via e-mail. In the absence of any published soft state information, this would be the sole input to the compositor forming the presence document. The example document contains PIDF extensions specified in "RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF)" [8] and "CIPID: Contact Information in Presence Information Data Format" [9].

It is assumed that the presentity is a SIP user with Address-of-Record (AOR) sip:someone@example.com. The XCAP root URI for example.com is assumed to be http://xcap.example.com. The XCAP User Identifier (XUI) is assumed to be identical to the SIP AOR, according to XCAP recommendations. In this case, the presence document would be located at http://xcap.example.com/pidf-manipulation/users/sip:someone@example.com/index.

The presence document is created with the following XCAP operation:

```
PUT /pidf-manipulation/users/sip:someone@example.com/index HTTP/1.1
Host: xcap.example.com
Content-Type: application/pidf+xml
...

<?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
           xmlns:rp="urn:ietf:params:xml:ns:pidf:rpid"
           xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
           xmlns:ci="urn:ietf:params:xml:ns:pidf:cipid"
           entity="sip:someone@example.com">

        <tuple id="x8eg92m">
          <status>
            <basic>closed</basic>
          </status>
          <rp:user-input>idle</rp:user-input>
          <rp:class>auth-1</rp:class>
          <contact priority="0.5">sip:user@example.com</contact>
          <note>I'm available only by e-mail.</note>
          <timestamp>2004-02-06T16:49:29Z</timestamp>
        </tuple>

        <tuple id="x8eg92n">
          <status>
```

```
            <basic>open</basic>
          </status>
          <rp:class>auth-1</rp:class>
          <contact priority="1.0">mailto:someone@example.com</contact>
          <note>I'm reading mail a couple of times a week</note>
        </tuple>

        <dm:person id="p1">
           <rp:class>auth-A</rp:class>
           <ci:homepage>http://www.example.com/~someone</ci:homepage>
           <rp:activities>
               <rp:vacation/>
           </rp:activities>
        </dm:person>

      </presence>
```

When the user wants to change the note related to e-mail service, it is done with the following XCAP operation:

```
PUT /pidf-manipulation/users/sip:someone@example.com/index/
~~/presence/tuple%5b@id='x8eg92n'%5d/note HTTP/1.1
If-Match: "xyz"
Host: xcap.example.com
Content-Type: application/xcap-el+xml
...

<note>I'm reading mails on Tuesdays and Fridays</note>
```

12. Security Considerations

A presence document may contain information that is highly sensitive. Its delivery to watchers needs to happen strictly according to the relevant authorization policies. It is also important that only authorized clients are able to manipulate the presence information.

The XCAP base specification mandates that all XCAP servers MUST implement HTTP Digest authentication specified in RFC 2617 [5]. Furthermore, XCAP servers MUST implement HTTP over TLS [6]. It is recommended that administrators of XCAP servers use an HTTPS URI as the XCAP root services' URI, so that the digest client authentication occurs over TLS. By using these means, XCAP client and server can ensure the confidentiality and integrity of the XCAP presence document manipulation operations, and that only authorized clients are allowed to perform them.

13. IANA Considerations

There is an IANA consideration associated with this specification.

13.1. XCAP Application Usage ID

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

Name of the AUID: pidf-manipulation

Description: Pidf-manipulation application usage defines how XCAP is used to manipulate the contents of PIDF-based presence documents.

14. Acknowledgements

The authors would like to thank Jari Urpalainen, Jonathan Rosenberg, Hisham Khartabil, Aki Niemi, Mikko Lonnfors, Oliver Biot, Alex Audu, Krisztian Kiss, Jose Costa-Requena, George Foti, and Paul Kyzivat for their comments.

15. References

15.1. 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)", RFC 4825, May 2007.

[3] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

[4] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, October 2004.

[5] Franks, J., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

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

15.2. Informative References

[7] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[8] Schulzrinne, H., Gurbani, V., Kyzivat, P., and J. Rosenberg, "RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF)", RFC 4480, July 2006.

[9] Schulzrinne, H., "CIPID: Contact Information for the Presence Information Data Format", RFC 4482, July 2006.

[10] Rosenberg, J., "A Data Model for Presence", RFC 4479, July 2006.

[11] Lonnfors, M. and K. Kiss, "Session Initiation Protocol (SIP) User Agent Capability Extension to Presence Information Data Format (PIDF)", Work in Progress, July 2006.

[12] Schulzrinne, H., "Timed Presence Extensions to the Presence Information Data Format (PIDF) to Indicate Status Information for Past and Future Time Intervals", RFC 4481, July 2006.

Authors' Addresses

Markus Isomaki
Nokia
P.O. BOX 100
00045 NOKIA GROUP
Finland

EMail: markus.isomaki@nokia.com

Eva Leppanen
Nokia
P.O. BOX 785
33101 Tampere
Finland

EMail: eva-maria.leppanen@nokia.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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