Network Working Group
Request for Comments: 3858
Category: Standards Track

J. Rosenberg
dynamicsoft

August 2004

An Extensible Markup Language (XML) Based Format for Watcher Information

Status of this Memo

Copyright Notice

Abstract

Watchers are defined as entities that request (i.e., subscribe to) information about a resource. There is fairly complex state associated with these subscriptions. The union of the state for all subscriptions to a particular resource is called the watcher information for that resource. This state is dynamic, changing as subscribers come and go. As a result, it is possible, and indeed useful, to subscribe to the watcher information for a particular resource. In order to enable this, a format is needed to describe the state of watchers on a resource. This specification describes an Extensible Markup Language (XML) document format for such state.

Table of Contents

1. Introduction

Watchers are defined as entities that request (i.e., subscribe to) information about a resource, using the SIP event framework, RFC 3265 [1]. There is fairly complex state associated with these subscriptions. This state includes the identity of the subscriber, the state of the subscription, and so on. The union of the state for all subscriptions to a particular resource is called the watcher information for that resource. This state is dynamic, changing as subscribers come and go. As a result, it is possible, and indeed useful, to subscribe to the watcher information for a particular resource. An important application of this is the ability for a user to find out the set of subscribers to their presentity [11]. This would allow the user to provide an authorization decision for the subscription.

To support subscriptions to watcher information, two components are needed. The first is the definition of a SIP event template-package for watcher information. The other is the definition of a data format to represent watcher information. The former is specified in [2], and the latter is specified here.

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 BCP 14, RFC 2119 [3] and indicate requirement levels for compliant implementations.

This document also uses the terms subscriber, watcher, subscription, notification, watcherinfo subscription, watcherinfo subscriber, and watcherinfo notification with the meanings described in [2].

3. Structure of Watcher Information

Watcher information is an XML document [4] that MUST be well-formed and SHOULD be valid. Watcher information documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying watcherinfo documents and document fragments. The namespace URI for elements defined by this specification is a URN [5], using the namespace identifier 'ietf' defined by [6] and extended by [7]. This URN is:

urn:ietf:params:xml:ns:watcherinfo

A watcher information document begins with the root element tag "watcherinfo". It consists of any number of "watcher-list" sub-elements, each of which is a list of watchers for a particular resource. Other elements from different namespaces MAY be present for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored. There are two attributes associated with the "watcherinfo" element, both of which MUST be present:

version: This attribute allows the recipient of watcherinfo documents to properly order them. Versions start at 0, and increment by one for each new document sent to a watcherinfo subscriber. Versions are scoped within a watcherinfo subscription. Versions MUST be representable using a 32 bit integer. However, versions do not wrap.

state: This attribute indicates whether the document contains the full watcherinfo state, or whether it contains only information on those watchers which have changed since the previous document (partial).

Each "watcher-list" element contains a list of "watcher" elements, each of which describes a watcher on a particular resource. Other elements from different namespaces MAY be present for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored. There are two attributes associated with the "watcher-list" element, both of which MUST be present:

resource: This attribute contains a URI for the resource being watched by that list of watchers. It is mandatory.

package: This attribute contains a token indicating the event package for which watcher information on that resource is being provided. It is mandatory.

The "watcher" element describes a watcher in the watcher list. The value of the "watcher" element is a URI for the watcher. This URI SHOULD be the authenticated identity of the watcher as determined by the server processing the subscription. As such, this URI will usually be an address-of-record (for example, sip:joe@example.com) as opposed to a device address (for example, sip:joe@192.0.2.3). There are three mandatory attributes which MUST be present:

id: A unique identifier for the subscription described by the watcher element. The id MUST be representable using the grammar for token as specified by RFC 3261 [8]. It MUST be unique across all other watchers reported in documents sent in notifications for a particular watcherinfo subscription.

status: The status of the subscription. The meaning of the various statuses are defined in the watcher information event package [2].

event: The event which caused the transition to the current status. The events are defined in the watcher information event package [2].

There are also some optional, informative attributes of the watcher element. These are:

display-name: A textual representation of the name of the subscriber.

expiration: The amount of time, in seconds from the current time, that the subscription will expire.

duration-subscribed: The amount of time, expressed in seconds, between the time the SUBSCRIBE which created the subscription was received, and the current time.

The xml:lang attribute MAY be used with the "watcher" element to specify the language of the "display-name".

The number of watchers present for each resource, and the set of resources listed, depends on the type of data being provided, and to whom.

For example, consider a presence system using watcher information. In one scenario, a user, A, subscribes to the presence of another user, B. A would like to find out about the status of their subscription. To do so, A subscribes to the watcher information for B’s presence. A does not have authorization to learn the status of all watchers for B’s presence. As a result, the watcher information sent to A will contain only one watcher - A themself.

In another scenario, a user, B, wishes to learn the set of people who have subscribed to B’s presence. To do this, B subscribes to the watcher information for B’s presence. Here, B is authorized to see all the watchers of B’s presence. As a result, the watcher information sent to B will contain all watchers of B’s presence.

In the case where an administrator wishes to learn the current status in the system, the watcher information could contain all watchers for all resources.

4. Computing Watcher Lists from the Document

Typically, the watcherinfo NOTIFY will only contain information about those watchers whose state has changed. To construct a coherent view of the total state of all watchers, a watcherinfo subscriber will need to combine NOTIFYs received over time. The watcherinfo subscriber maintains a table for each watcher list it receives information about. Each watcher list is uniquely identified by the URI in the "resource" attribute of the "watcher-list" element. Each table contains a row for each watcher in that watcher list. Each row is indexed by the unique ID for that watcher. It is conveyed in the "id" attribute of the "watcher" element. The contents of each row contain the state of that watcher as conveyed in the "watcher" element. The tables are also associated with a version number. The version number MUST be initialized with the value of the "version" attribute from the "watcherinfo" element in the first document received. Each time a new document is received, the value of the local version number, and the "version" attribute in the new document, are compared. If the value in the new document is one higher than the local version number, the local version number is increased by one, and the document is processed. If the value in the document is more than one higher than the local version number, the local version number is set to the value in the new document, the document is processed, and the watcherinfo subscriber SHOULD generate a refresh request to trigger a full state notification. If the value in the document is less than the local version, the document is discarded without processing.

The processing of the watcherinfo document depends on whether it contains full or partial state. If it contains full state, indicated by the value of the "state" attribute in the "watcherinfo" element, the contents of all tables associated with this watcherinfo subscription are flushed. They are re-populated from the document. A new table is created for each "watcher-list" element, and a new row in each table is created for each "watcher" element. If the watcherinfo contains partial state, as indicated by the value of the "state" attribute in the "watcherinfo" element, the document is used to update the existing tables. For each "watcher-list" element, the watcherinfo subscriber checks to see if a table exists for that list. This check is done by comparing the URI in the "resource" attribute of the "watcher-list" element with the URI associated with the table. If a table doesn’t exist for that list, one is created. For each "watcher" element in the list, the watcherinfo subscriber checks to see whether a row exists for that watcher. This check is done by

comparing the ID in the "id" attribute of the "watcher" element with the ID associated with the row. If the watcher doesn't exist in the table, a row is added, and its state is set to the information from that "watcher" element. If the watcher does exist, its state is updated to be the information from that "watcher" element. If a row is updated or created, such that its state is now terminated, that entry MAY be removed from the table at any time.

5. Example

The following is an example of watcher information for a presentity, who is a professor. There are two watchers, userA and userB.

```
<?xml version="1.0"?>
<watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo"
             version="0" state="full">
  <watcher-list resource="sip:professor@example.net" package="presence">
    <watcher status="active"
             id="8ajksjda7s"
             duration-subscribed="509"
             event="approved" >sip:userA@example.net</watcher>
    <watcher status="pending"
             id="hh8juja87s997-ass7"
             display-name="Mr. Subscriber"
             event="subscribe">sip:userB@example.org</watcher>
  </watcher-list>
</watcherinfo>
```

6. XML Schema

The following is the schema definition of the watcherinfo document format:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       targetNamespace="urn:ietf:params:xml:ns:watcherinfo"
       xmlns:tns="urn:ietf:params:xml:ns:watcherinfo" >
<!-- This import brings in the XML language attribute xml:lang-->
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
             schemaLocation="http://www.w3.org/2001/03/xml.xsd" />
  <xs:element name="watcherinfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:watcher-list" minOccurs="0"
                    maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="version" type="xs:nonNegativeInteger"
```

```
                    use="required"/>
      <xs:attribute name="state" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="full"/>
            <xs:enumeration value="partial"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="watcher-list">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:watcher" minOccurs="0" maxOccurs=
         "unbounded"/>
          <xs:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="resource" type="xs:anyURI" use="required"/>
      <xs:attribute name="package" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="watcher">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:anyURI">
          <xs:attribute name="display-name" type="xs:string"/>
          <xs:attribute name="status" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="pending"/>
                <xs:enumeration value="active"/>
                <xs:enumeration value="waiting"/>
                <xs:enumeration value="terminated"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="event" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="subscribe"/>
                <xs:enumeration value="approved"/>
                <xs:enumeration value="deactivated"/>
                <xs:enumeration value="probation"/>
                <xs:enumeration value="rejected"/>
                <xs:enumeration value="timeout"/>
                <xs:enumeration value="giveup"/>
```

```
              <xs:enumeration value="noresource"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="expiration" type="xs:unsignedLong"/>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="duration-subscribed"
            type="xs:unsignedLong"/>
        <xs:attribute ref="xml:lang"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

7. Security Considerations

Watcher information is sensitive information. The protocol used to distribute it SHOULD ensure privacy, message integrity, and authentication. Furthermore, the protocol should provide access controls which restrict who can see who elses watcher information.

8. IANA Considerations

This document registers a new MIME type, application/watcherinfo+xml, and registers a new XML namespace.

8.1. application/watcherinfo+xml MIME Registration

MIME media type name: application

MIME subtype name: watcherinfo+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [9].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [9].

Security considerations: See Section 10 of RFC 3023 [9] and Section 7 of this specification.

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: This document type has been used to support subscriber authorization functions for SIP-based presence [10] [2].

Additional Information:
Magic Number: None

File Extension: .wif or .xml

Macintosh file type code: "TEXT"

Personal and email address for further information: Jonathan Rosenberg, <jdrosen@jdrosen.net>

Intended usage: COMMON

Author/Change controller: The IETF.

8.2. URN Sub-Namespace Registration for urn:ietf:params:xml:ns:watcherinfo

This section registers a new XML namespace, as per the guidelines in [7].

URI: The URI for this namespace is urn:ietf:params:xml:ns:watcherinfo.

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>Watcher Information Namespace</title>
</head>
<body>
  <h1>Namespace for Watcher Information</h1>
  <h2>urn:ietf:params:xml:ns:watcherinfo</h2>
  <p>See <a href="ftp://ftp.rfc-editor.org/in-notes/rfc3858.txt">
     RFC3858</a>.</p>
```

```
</body>
</html>
END
```

9. Normative References

[1] Roach, A. B., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[2] Rosenberg, J., "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)", RFC 3857, August 2004.

[3] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[4] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, "Extensible Markup language (XML) 1.0 (second edition)," W3C Recommendation REC-xml-20001006, World Wide Web Consortium (W3C), Oct. 2000. Available at http://www.w3.org/XML/.

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

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

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

[8] 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.

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

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

10. Informative References

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

11. Acknowledgements

The authors would like to thank Sean Olson, Steve Donovan, and Cullen Jennings for their detailed comments and assistance with the XML schema.

12. Contributors

The following people were part of the original design team that developed the first version of this specification:

Dean Willis
dynamicsoft
5100 Tennyson Parkway, Suite 1200
Plano, Texas 75024

EMail: dwillis@dynamicsoft.com

Robert Sparks
dynamicsoft
5100 Tennyson Parkway, Suite 1200
Plano, Texas 75024

EMail: rsparks@dynamicsoft.com

Ben Campbell

EMail: ben@nostrum.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003

EMail: schulzrinne@cs.columbia.edu

Jonathan Lennox
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003

EMail: lennox@cs.columbia.edu

Christian Huitema
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

EMail: huitema@microsoft.com

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

EMail: bernarda@microsoft.com

David Gurle
Reuters Corporation

EMail: David.Gurle@reuters.com

Jonathan Lennox contributed the text for the DTD and its usage that were part of earlier versions of this specification.

13. Author's Address

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054

EMail: jdrosen@dynamicsoft.com

14. Full Copyright Statement

Intellectual Property

Acknowledgement

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