SIMPLE J. Rosenberg
Internet-Draft dynamicsoft
Expires: August 16, 2004 February 16, 2004

A Session Initiation Protocol (SIP) Event Package for Modification Events for the Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Managed Documents
draft-ietf-simple-xcap-package-01

Status of this Memo

Copyright Notice

Abstract

This specification defines a Session Initiation Protocol (SIP) event package for finding out about changes to documents managed by the Extensible Markup Language (XML) Configuration Access Protocol (XCAP). XCAP allows a client to manipulate XML documents on a server which contain configuration information for application protocols. Multiple clients can potentially access the same document, in which case the other clients would like to be notified of a change in the document made by another. This event package allows a client to do that.

Table of Contents

1. Introduction

The Extensible Markup Language (XML) Configuration Access Protocol (XCAP) [10] is a protocol that allows clients to manipulate XML documents stored on a server. These XML documents serve as configuration information for application protocols. As an example, resource list [11] subscriptions (also known as presence lists) allow a client to have a single SIP subscription to a list of users, where the list is maintained on a server. The server will obtain presence for those users and report it back to the client. This application requires the server, called a Resource List Server (RLS), to have access to the list of presentities. This list needs to be manipulated by clients so they can add and remove their friends as they desire.

Complexities arise when multiple clients attempt to simultaneously manipulate a document, such as a presence list. Frequently, a client will keep a copy of the current list in memory, so it can render it to users. However, if another client modifies the document, the cached version becomes stale. This information must be made known to all clients which have cached copies of the document, so that they can fetch the most recent one.

This problem is addressed by this specification, which provides a Session Initiation Protocol (SIP) [1]event package [2] for subscribing to changes in documents managed by an XCAP server. This package would be used by any client which is retaining a cached copy of a document obtained by XCAP, so that it can find out when a change has been made, invalidating its cached copy. In fact, the notifications generated by this package indicate the specific change which occurred in the document, so the client can decide whether the change is significant enough to warrant a refetch from the XCAP server.

2. Document Change Event Package

The SIP event framework [2] defines a SIP extension for subscribing to, and receiving notifications of, events. It leaves the definition of many aspects of these events to concrete extensions, known as event packages. This document qualifies as an event package. This section fills in the information required for all event packages by RFC 3265.

2.1 Package Name

The name of this package is "xcap-change". As specified in RFC 3265 [2], this value appears in the Event header field present in SUBSCRIBE and NOTIFY requests.

2.2 Event Package Parameters

The SIP event framework allows event packages to define additional parameters carried in the Event header field. This package defines a single event header parameter, called "doc-component", which specifies a particular document and document component which is being to subscribed to. The request-URI specifies the user whose data is being subscribed to. To subscribe to global data, a user agent would subscribed to a special user name that is configured into the UA. The name "global-xcap-user" is RECOMMENDED, and SHOULD be used if no explicit user name is provisioned. By default, a subscription is for all XCAP data associated with that user. The header field parameter allows the subscription to specify a specific document and document sub-tree.

The format of this header field parameter is a quoted string. The value of this string is the portion of an XCAP URI to the right of the directory for the user, in the case of user data, or to the right of the global directory for global data. The XCAP URI can only refer to a document or to an element within that document. When the URI represents a document, the subscription is to changes anywhere in the document. When it is to an element, the subscription is to changes that occur in the attributes or content of that element, including all children. For example, if a user wishes to subscribe to http://xcap.example.com/services/presence-lists/users/joe/mydir/friends.xml, the event header parameter would be "mydir/friends.xml".

> OPEN ISSUE: There is no way to specify that a subscription is to multiple documents. Multiple subscriptions would be needed for that. Is that limitation OK for now? A filter can fix that down the road.

2.3 SUBSCRIBE Bodies

A SUBSCRIBE request MAY contain a body. The purpose of the body depends on its type. Subscriptions will normally not contain bodies. The Request-URI, which identifies the user whose data is being subscribed to, combined with the event package name and parameter, is sufficient for this package.

One type of body that can be included in a SUBSCRIBE request is a filter document. These filters request that only document change events generate notifications, or would ask for a restriction on the set of data returned in NOTIFY requests. Filter documents are not specified in this document, and at the time of writing, are expected to be the subject of future standardization activity.

Honoring of these filters is at the policy discretion of the notifier.

If the SUBSCRIBE request does not contain a filter, this tells the notifier that no filter is to be applied. The notifier SHOULD send NOTIFY requests at the discretion of its own policy.

2.4 Subscription Duration

Generally speaking, changes to application configuration data are relatively infrequent. Of course, this depends on the type of application, but generally configuration data is static. As a result, notifications are expected infrequently, and subscriptions will typically be held for long periods of time. This would argue for long subscription refresh intervals. For this reason, the default subscription duration is two hours. Of course, a different duration can be requested by a client, or set by a server, using the Expires header field, as per RFC 3265 [2].

2.5 NOTIFY Bodies

As described in RFC 3265 [2], the NOTIFY message will contain bodies that describe the state of the subscribed resource. This body is in a format listed in the Accept header field of the SUBSCRIBE, or a package-specific default if the Accept header field was omitted from the SUBSCRIBE.

In this event package, the body of the notification contains a document change document. This document describes the current version of an XML document managed by XCAP, in addition to the changes in this document from the previous version. Note that a listing of changes from the previous version is only sent in a NOTIFY triggered by a change to the document. NOTIFY requests sent in response to an

initial SUBSCRIBE, or a SUBSCRIBE refresh, only indicate the current version of the XML document. They do not contain the actual full contents of the XML document. In other words, the resource being subscribed to is NOT the XML document itself, but rather, the version history for the document.

> OPEN ISSUE: This is the main issue in this specification. There are three potential scopes. The first is that subscription is to the document itself, in which case a full state update in a NOTIFY contains the current document. The second is a subscription to the revision history, which gives you changes, but not the full state of the document. The third option is to just subscribe to the etag, so that you know that something changed, but the notifications don't tell you anything about what changed. Which do we need? The latter is the simplest, good for the case where third party modification of documents is rare.

All subscribers and notifiers MUST support the "application/xcap-change+xml" data format described in Section 3. The subscribe request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/xcap-change+xml". If the header field is present, it MUST include "application/xcap-change+xml", and MAY include any other types capable of representing changes in XCAP documents.

2.6 Notifier Processing of SUBSCRIBE Requests

This subsection defines package-specific processing at the notifier when it receives a SUBSCRIBE request. General processing rules for requests are covered in Section 8.2 of RFC 3261 [1], in addition to general SUBSCRIBE processing in RFC 3265 [2].

A notifier for this package SHOULD authenticate all subscribers. Generally, subscribers will have a pre-existing relationship with the notifier. This is because the principle application of this package is for a client of XCAP (which will have a relationship with the XCAP server) to find out about changes in cached documents. Therefore, the HTTP Digest mechanism in SIP is a good match for authentication, and MUST be supported by all clients and servers. Note that this authentication mechanism is already mandatory for all SIP-compliant implementations.

Once authenticated, the server SHOULD authorize the subscriber. Generally, this authorization policy SHOULD mirror the authorization policy defined in an XCAP application usage for read access. Thats because this package provides a form of read-access, and the permissions should not differ based on whether the read is performed with XCAP or with a SIP SUBSCRIBE request.

2.7 Notifier Generation of NOTIFY Requests

RFC 3265 details the formatting and structure of NOTIFY messages. However, packages are mandated to provide detailed information on when to send a NOTIFY, how to compute the state of the resource, how to generate neutral or fake state information, and whether state information is complete or partial. This section describes those details for the presence event package.

A notifier MAY send a notification at any time. Typically, it will send one after a document managed by an XCAP server has changed as the result of an XCAP operation. This notification contains an application/xcap-change+xml document that specifies the current version (as a server modification time) for the XML document being subscribed to. It also contains information about what changed - whether a new element or attribute was added, whether an existing one changed, or whether an existing one was deleted, and indicates against which version those changes were made. The xcap-change document also contains a hash of the new XML document.

Notifications sent in response to SUBSCRIBE requests (either initial or refresh), or sent when there is a change in subscription state, will normally only contain the current version of the XML document being subscribed to.

The body of the NOTIFY MUST be sent using one of the types listed in the Accept header field in the most recent SUBSCRIBE request, or using the type "application/xcap-change+xml" if no Accept header field was present.

2.8 Subscriber Processing of NOTIFY Requests

RFC 3265 [2] leaves it to event packages to describe the process followed by the subscriber upon receipt of a NOTIFY request, including any logic required to form a coherent resource state.

In this package, the notifications can be optionally used by the client to determine the state of the XML document being subscribed to. When a client receives a notification, it checks the version against which the changes are relative. If this is not the same as the version currently cached by the client, the client SHOULD use XCAP to fetch the latest version of the document. If it is the same, the client applies the change to its local cache of the document. To apply the changes, the client follows the procedures defined by XCAP [10] as if it were the HTTP server. After applying the changes, the client applies the mandatory XML canonicalization defined in the Canonical XML 1.0 [3] specification, and computes an HMAC [12] using SHA1 over this canonical document, with a key whose value is 0x2238a.

The resulting string is compared with the hash attribute of the xml-change document. If they match, the client can be sure that it has the most up to date version. If they don't match, the client SHOULD fetch the most current version of the document.

Of course, this mechanism for computing the most current document from the hash is optional. A client can elect to ignore the information on what changed and simply fetch the most recent document every time it gets a change indication where the new version is not the same as the one cached by the client.

2.9 Handling of Forked Requests

RFC 3265 [2] requires each package to describe handling of forked SUBSCRIBE requests.

This specification only allows a single dialog to be constructed as a result of emitting an initial SUBSCRIBE request. Section 4.4.9 of RFC 3265 [2] describes the processing that is required to guarantee the creation of a single dialog in response to a SUBSCRIBE request.

2.10 Rate of Notifications

RFC 3265 [2] requires each package to specify the maximum rate at which notifications can be sent. A notifier SHOULD NOT generate notifications at a rate of more than once every five seconds.

2.11 State Agents

RFC 3265 [2] requires each package to consider the role of state agents in the package, and if they are used, to specify how authentication and authorization are done.

State agents play no role in this package.

3. application/xml-change+xml Media Type

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

```
urn:ietf:params:xml:ns:xml-change
```

An xml-change document begins with the root element tag "documents". It consists of any number of "document" sub-elements, each of which conveys information on a particular document. Other elements from different namespaces MAY be present for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored.

Each "document" element consists of zero or more "change" elements, each of which conveys information about a specific change to the document. Other elements from different namespaces MAY be present for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored. There are four attributes associated with the "document" element:

uri: specifies the HTTP URI that identifies the document.

new-etag: specifies the etag of the document after the application of the change. This attribute is mandatory.

previous-etag: specifies the etag of the version of the document against which the change was made. This attribute MUST be present if any change sub-elements are present.

hash: specifies an HMAC of the new document, represented in canonical form. See Section 2.8 for details on how this value is computed. This attribute MUST be present if any change sub-elements are present.

Each "change" element contains text. This text contains the exact value present in the HTTP request that caused the change in the document. If that content was XML, it is represented in the xml-change document as CDATA. There are two attributes associated with this element:

uri: contains the URI that the HTTP request contained in the Request-URI. This attribute is mandatory.

method: contains the method of the HTTP request. This attribute is mandatory.

OPEN ISSUE: Probably it would be better to describe the changes more generically, rather than binding them to the specifics of XCAP. Indeed, if there is any non-determinism in how the server computes the document resulting from an XCAP operation, this approach will not work. We can either use patch or define a specific XML patch format.

3.1 XML Schema

```
   <?xml version="1.0" encoding="UTF-8"?>
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:xml-change"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="urn:ietf:params:xml:ns:xml-change"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="documents">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="tns:document" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="tns:change" type="xs:string" maxOccurs="unbounded"/
>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
            maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="tns:uri" type="xs:anyURI" use="required"/>
         <xs:attribute name="tns:new-etag" type="xs:string" use="required"/>
         <xs:attribute name="tns:previous-etag" type="xs:string"
use="optional"/>
         <xs:attribute name="tns:hash" type="xs:string" use="optional"/>
        </xs:complexType>
       </xs:element>
       <xs:any namespace="##other" processContents="lax" minOccurs="0"
        maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:simpleType name="httpMethod">
     <xs:restriction base="xs:string">
      <xs:enumeration value="PUT"/>
      <xs:enumeration value="DELETE"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:schema>
```

3.2 Example Document

The following is an example of a document compliant to the schema:

```
   <?xml version="1.0" encoding="UTF-8"?>
   <documents xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <document
        uri="http://xcap.example.com/s/presence-lists/users/bill/foo.xml"
        new-etag="asdnasd9asd8asd7"
        previous-etag="s99s99s9s9sjja"
        hash="<hash value>">
       <change method="DELETE"
        uri="http://xcap.example.com/s/presence-lists/
        users/bill/foo.xml?presence-lists/entry[@name="bob"]/uri"/>
     </document>
   </documents>
```

4. Security Considerations

The security considerations for this package are similar to those of XCAP. The configuration data can contain sensitive information, and both the client and the server need to authenticate each other. As such, a notifier for this package MUST support HTTP Digest to authenticate subscribers. Notifiers and subscribers MAY use SIPs S/MIME feature to provide authentication and message integrity.

5. IANA Considerations

There are several IANA considerations associated with this specification.

5.1 SIP Event Package

This specification registers an event package, based on the registration procedures defined in RFC 3265 [2]. The following is the information required for such a registration:

Package Name: xml-change

Package or Template-Package: This is a package.

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

Person to Contact: Jonathan Rosenberg, jdrosen@jdrosen.net.

5.2 application/xcap-change+xml MIME Type

MIME media type name: application

MIME subtype name: xcap-change+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [6].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [6].

Security considerations: See Section 10 of RFC 3023 [6] and Section 4 of this specification.

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: This document type has been used to support manipulation of presence lists [13] using XCAP.

Additional Information:

Magic Number: None

File Extension: .xcd 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.

5.3 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:xcap-change

This section registers a new XML namespace, as per the guidelines in [8]

URI: The URI for this namespace is urn:ietf:params:xml:ns:xcap-change.

Registrant Contact: IETF, SIMPLE working group, (simple@mailman.dynamicsoft.com), 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>XCAP Change Namespace</title>
</head>
<body>
  <h1>Namespace for XCAP Change</h1>
  <h2>urn:ietf:params:xml:ns:change-xml</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
```

```
                END
```

Normative References

[1] 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.

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

[3] Boyer, J., "Canonical XML Version 1.0", W3C REC REC-xml-c14n-20010315, March 2001.

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

[9] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

Informative References

[11] Roach, A., Rosenberg, J. and B. Campbell, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", draft-ietf-simple-event-list-04 (work in progress), June 2003.

[12] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[13] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-01 (work in progress), October 2003.

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.