SIMPLE
Internet-Draft
Expires: January 19, 2006

J. Rosenberg
Cisco Systems

July 18, 2005

An Extensible Markup Language (XML) Document Format for Indicating Changes in XML Configuration Access Protocol (XCAP) Resources

draft-ietf-simple-xcap-diff-01

Status of this Memo

Copyright Notice

Abstract

This specification defines a document format that can be used to describe the differences between versions of resources managed by the Extensible Markup Language (XML) Configuration Access Protocol (XCAP). XCAP diff documents can be delivered to clients using a number of means, including the Session Initiation Protocol (SIP) event package for configuration data. By subscribing to this event package, clients can learn about document changes made by other clients.

Table of Contents

1. Introduction

The Extensible Markup Language (XML) Configuration Access Protocol (XCAP) [8] 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 [12] 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 modification event must be made known to all clients which have cached copies of the document, so that they can fetch the most recent one.

To deal with this problem, clients can use the Session Initiation Protocol (SIP) [10] event package [11] for subscribing to changes in configuration and profile information [9], including application data that resides on an XCAP server. With that package, a user gets notified that a particular document has changed. This notification can include the full content of the new document, or it can be a content indirection [15]. However, in both cases, the transfer of the entire document is ultimately required. This may require a lot of bandwidth, particularly for wireless devices with large documents (such as a resource list [12] with hundreds of users listed). Furthermore, though content indirection can tell a client that a document has changed, it provides it with MIME Content-ID indicating the new version of the document. The MIME Content-ID is not the same

as the entity tag, which is used by XCAP for document versioning. As such, a client cannot easily ascertain whether an indication of a change in a document is due to a change it just made, or due to a change another client made at around the same time.

To resolve this problem, this document defines a data format which can convey changes in XML documents managed by an XCAP server. This data format is an XML document format, called an XCAP diff document. This format can indicate that a document has changed, provide its previous and new entity tags, and optionally include the xcap operation that was performed which resulted in that change. This specification also explains how this format is used in conjunction

with the configuration profile framework.

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 [7] and indicate requirement levels for compliant implementations.

This specification also defines the following additional terms:

Document: When the term document is used without the "XCAP diff" in front of it, it refers to the XCAP document resource about whom the XCAP diff document is reporting a change.

XCAP diff document: The XML document defined by this specification that reports on a set of changes in an XCAP document resource.

Server: Typically an XCAP server, this is a protocol entity that generates XCAP diff documents based on its knowledge of a set of XCAP documents.

Client: Typically an XCAP client and SIP User Agent (UA) that acts as a subscriber to the configuration event package, this is a protocol entity that consumes XCAP diff documents in order to reconstruct the document stored on the server.

3. Structure of an XCAP Diff Document

An XCAP diff document is an XML [2] document that MUST be well-formed and SHOULD be valid. XCAP diff documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying XCAP diff documents and document fragments. The namespace URI for elements defined by this specification is a URN [3], using the namespace identifier 'ietf' defined by [5] and extended by [6]. This URN is:

urn:ietf:params:xml:ns:xcap-diff

An XCAP diff document begins with the root element tag <xcap-diff>. This element has a single mandatory attribute, "xcap-root". The value of this attribute is the XCAP root URI for the documents in which the changes have taken place. A single XCAP diff document can only represent changes in documents within the same XCAP root. The content of the <xcap-diff> element is a sequence of <document> elements. Each <document> element specifies changes in a specific document within the XCAP root. It has one mandatory attribute, "doc-

selector", and a three optional attributes, "new-etag", "previous-etag" and "hash". The "doc-selector" identifies the specific document within the XCAP root for which changes are indicated. Its content MUST be a relative path reference, with the base URI being equal to the XCAP root URI. The "new-etag" attribute provides the etag for the document after the application of the changes, assuming the document exists after those changes. If the change being reported is the deletion of the document, the "new-etag" attribute will not be present. A server MUST include the "new-etag" unless the document does not exist subsequent to the changes reported in the XCAP diff document. If The "previous-etag" attribute provides an identifier for the document instance prior to the change. If the document did not exist prior to the change (that is, the change was the creation of the document), the "previous-etag" is not present. If the server is reporting a specific set of document changes via the <change-log> element described below, a server MUST include the "previous-etag" unless the document did not exist prior to changes reported in the XCAP diff document. If the <change-log> element is not present, the "previous-etag" SHOULD be present. The "previous-etag" and "new-etag" need not have been sequentially assigned etags at the server. An XCAP diff document can describe changes that have occurred over a series of XCAP operations.

The optional "hash" attribute provides an HMAC of the document instance whose etag is "new-etag", once that document is represented in canonical form. See Section 6 for details on how this value is computed. This attribute is optional, and a server MAY elect not to include it. Even if present, a client MAY elect to ignore it.

Each <document> element contains zero or one <change-log> element, followed by any number of elements from another namespace for the purposes of extensibility. Any such unknown elements MUST be ignored by the client. When present, the <change-log> element tells the client the specific set of XCAP operations that can be applied to transform the document from the version whose etag was "previous-etag" to the version whose etag is "new-etag". If the "previous-etag" is not present, the <change-log> element tells the client the specific set of XCAP operations that can be applied to create a document from nothing, and result in the document whose etag is "new-etag". The series of operations in the <change-log> do not have to be the same exact series of operations that occurred at the server. The only requirement is that, if the server includes the <change-log> element, the sequence of events, when executed serially, will result in the transformation of the document with the etag "previous-etag" to the one whose etag is "new-etag". If the <change-log> element is not present, it means that the document has changed in some way, but the XCAP server has elected not to provide the set of changes. In that case, a client can retrieve the latest document if its cached

etag doesn't match the value of "new-etag".

It is important to note that a <document> element with no <change-log> child is not equivalent to a <document> element with a <change-log> child that is itself empty. The latter means that the document has been assigned a new etag but its content is unchanged. The former means that it has been assigned a new etag as a result of a change, but the specific changes are not being reported in the XCAP diff document.

Each <change-log> element contains zero or more <put-event> or <delete-event> elements. It can also contain elements from other namespaces, which allows for extensibility to other events in the future. A client MUST ignore any such elements it does not understand. Each <delete-event> element reports an HTTP DELETE

operation, and each <put-event> element reports an HTTP PUT operation. Both <put-event> and <delete-event> have a single optional attribute, "node-selector", which contains the node selector in the Request URI (after removing any escape coding) of the HTTP PUT or DELETE request. The server MUST include the "node-selector" when the PUT or DELETE operation was against an XML element or attribute. The "node-selector" attribute MUST NOT be present if the PUT or DELETE operation was against the document itself. The <put-event> element also has the mandatory attribute "content-type", which indicates the Content-Type of the HTTP PUT request. The content of the <put-event> element is text. This text contains the body of the HTTP PUT request. If that content was an XML type (including application/xcap-el+xml) that contains angle brackets, it MUST be represented as CDATA. If the content did not contain angle brackets (as is the case with application/xcap-att+xml), it MAY be represented as CDATA.

4. XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:ietf:params:xml:ns:xcap-diff"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="urn:ietf:params:xml:ns:xcap-diff"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="document">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="change-log" minOccurs="0"/>
    <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="doc-selector" type="xs:anyURI" use="required"/>
   <xs:attribute name="new-etag" type="xs:string" use="optional"/>
```



```
   <xs:attribute name="previous-etag" type="xs:string" use="optional"/>
   <xs:attribute name="hash" type="xs:string" use="optional"/>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="xcap-diff">
  <xs:complexType>
   <xs:sequence>
```

```
      <xs:element ref="document"/>
     </xs:sequence>
     <xs:attribute name="xcap-root" type="xs:anyURI" use="required"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="change-log">
    <xs:complexType>
     <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
       <xs:element ref="delete-event"/>
       <xs:element ref="put-event"/>
       <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="put-event">
    <xs:complexType>
     <xs:simpleContent>
      <xs:extension base="xs:string">
       <xs:attribute name="node-selector" type="xs:anyURI" use="optional"/>
       <xs:attribute name="content-type" type="xs:string" use="required"/>
      </xs:extension>
     </xs:simpleContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="delete-event">
    <xs:complexType>
     <xs:attribute name="node-selector" type="xs:anyURI" use="optional"/>
    </xs:complexType>
   </xs:element>
  </xs:schema>
```

5. Example Document

The following is an example of a document compliant to the schema. Line wrapping is for readability purposes only:


```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:xcap-diff xcap-diff.xsd"
 xcap-root="http://xcap.example.com/root">
 <document new-etag="7ahggs"
  doc-selector="resource-lists/users/joe/coworkers"
  previous-etag="8a77f8d">
   <change-log>
     <delete-event
      node-selector="resouce-lists/list[@name="friends"]/ent
ry[@uri="bill@example.com"]"/>
     <put-event
      node-selector="resouce-lists/list[@name="friends"]/ent
ry[@uri="jane@example.com"]"
      content-type="application/xml"><![CDATA[<entry uri="sip:jane@exa
mple.com"><display-name>Jane Doe</display-name></entry>]]>
     </put-event>
   </change-log>
 </document>
</xcap-diff>
```

This example XCAP diff document will transform the example document in Section 3.3 of [14] by removing the entry for Bill Smith and adding one for Jane Doe.

6. Usage with the Config Framework

The framework for user agent profile delivery [9] defines an event package which can be used to subscribe to user, device, application or local-network data that defines the configuration of a client. This data can be present in an XCAP server. Normally, content indirection [15] will be used as the NOTIFY body format, to indicate the specific document that has changed, and should be re-fetched. However, if the client includes an Accept header field including the MIME type "application/xcap-diff+xml", the server has the option of returning documents in this format instead.

When the client performs an initial subscription, the rules in [9] are used to select the set of documents which the subscription applies to. Upon initial subscription, the server does not know which instances of each document (where each instance is identified by an etag) the client currently posessses, if any. Indeed, upon startup, the client will not have any documents. The initial NOTIFY in this case MUST include a <document> element for each document associated with the subscription. The <change-log> for each of those <document> elements MUST be absent. The "previous-etag" attribute

MUST be absent, and the "new-etag" attribute MUST be present and contain the entity tag for the current version of that document resource. An XCAP diff document structured this way is called a "reference" XCAP diff document. It establishes the baseline etags and document URIs for the documents covered by the subscription.

Upon receipt of this document, the client can determine whether its local instance documents, if any, match the etags in the XCAP diff document. If they do not match, the client SHOULD perform a conditional GET for each document. The document URI is constructed by appending the XCAP root in the "xcap-root" attribute of the <xcap-diff> element to the escape coded "doc-selector" from each <document> element. The request is made conditional by including an If-Match header field, with the value of the etag from each <document> element. So long as the documents haven't changed between the NOTIFY and the GET, the client will obtain the reference versions that the server will use for subsequent notifications.

If the conditional GET should fail, the client SHOULD generate a SUBSCRIBE refresh request to trigger a new NOTIFY. The server will always generate a "reference" XML diff document on receipt of a SUBSCRIBE refresh. This establishes a new set of baseline etags, and the client can then attempt to do another fetch. It is anticipated that future extensions to the profile delivery framework will allow a client to include, in its SUBSCRIBE request, an indicator of the current version of the documents it holds. That would obviate the need for a potentially never-ending stream of SUBSCRIBE/GET sequences should the documents be rapidly changing, for some reason.

Once the client has obtained the versions of the documents identified in the reference XML diff, it can process NOTIFY requests on that subscription. To process the NOTIFY requests, it makes sure that its current version matches the version in the "previous-etag" attribute of the <document> element. It then follows the procedures of Section 7.

Once the client has finished applying the instructions to the document, it should end up with the same document the server has. To verify this, the client MAY apply the mandatory XML canonicalization defined in the Canonical XML 1.0 [1] specification, and computes an HMAC [13] using SHA1 over this canonical document, with a key whose value is 0x2238a. The resulting string is compared with the "hash" attribute of the <document> element. If they match, the client can be sure that it has the most up to date version. If they don't match, the client MUST flush its current version of the document from memory. It can then obtain a new XCAP diff reference by sending a

SUBSCRIBE refresh request on the dialog.

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. Furthermore, the server may elect to not send the hash, in which case this check cannot be made.

7. Constructing a Document from the Change Log

When the XCAP diff document contains a <change-log> element for a document, and the client possesses the document instance whose etag matches the "previous-etag" for the document, the client can follow the procedures defined here to obtain the instance document with the etag value of "new-etag". This procedure is relatively straightforward, and is done by having the client emulate XCAP server behavior as defined in [8]

The client starts with the its version of the document whose etag is "previous-etag" as the current document. If there was no "previous-etag", the client starts with no document. The client MUST iterate through each child of <change-log>, in order. For each element, it MUST apply processing depending on the name of the element.

If the element is <delete-event>, the client takes the current document. If the "node-selector" attribute was absent, it deletes the entire document. If the "node-selector" attribute was present, it selects the element or attribute using that node selector, as described in Section 6.3 of [8]. Note that the node selector present in the "node-selector" attribute is not escape coded, and will follow the grammar defined in that section. The selected element or attribute is deleted from the document, and the result becomes the current document. There is no need for the client to run the validity checks or idempotency checks normally performed by the server; a client will always be provided with <delete-event> operations that succeeded at the server.

If the element is <put-event>, the client takes the current document. It then computes the Request URI that was seen by the server, by

concatenating the XCAP root with the "doc-selector" attribute of th <document> element, appending the path separator, and then adding the "node-selector" attribute of the <put-event> element, if present. The client then "acts" as if it were the server, having receive an HTTP PUT request with the Request URI equal to this value prior to escape coding, with a body of Content-Type equal to the value of the "content-type" attribute, and whose body equals the value of the <put-event> element. It follows the logic of Section 8.2 of [8] to apply the PUT, ignorning all validity checks, resource

interdependency computations, error processing and verification of document content. The resulting document becomes the current document. An actual implementation need not literally act as a server; the behavior is defined in these terms to specify what the correct output of the processing has to be.

If the element is unknown to the client, it is skipped.

When each child element of <change-log> has been processed, the current document is equal to the document on the server whose etag equals "new-etag".

8. Security Considerations

XCAP diff documents contain the same information in the documents whose differences they describe. As such, the security considerations associated with those documents apply to XCAP diff documents.

9. IANA Considerations

There are several IANA considerations associated with this specification.

9.1 application/xcap-diff+xml MIME Type

MIME media type name: application

MIME subtype name: xcap-diff+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [4].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [4].

Security considerations: See Section 10 of RFC 3023 [4] and Section 8 of RFCXXXX [[NOTE TO RFC-EDITOR/IANA: Please replace XXXX with the RFC number 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 resource lists [14] using XCAP.

Additional Information:

Magic Number: None

File Extension: .xdf

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.

9.2 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:xcap-diff

This section registers a new XML namespace, as per the guidelines in [6]

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

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>XCAP Diff Namespace</title>
</head>
<body>
  <h1>Namespace for XCAP Diff</h1>
  <h2>urn:ietf:params:xml:ns:xcap-diff</h2>
  <p>See <a href="[URL of published RFC]">RFCXXXX[[NOTE
TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number of this
specification.]]</a>.</p>
</body>
</html>
```

```
END
```

9.3 Schema Registration

This section registers a new XML schema per the procedures in [6].

URI: urn:ietf:params:xml:schema:xcap-diff

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 4.

10. References

10.1 Normative References

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

[2] 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.

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

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

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

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

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

[8] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-07 (work in progress), June 2005.

[9] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-06 (work in progress), February 2005.

10.2 Informative References

[10] 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.

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

[12] Roach, A., Rosenberg, J., and B. Campbell, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", draft-ietf-simple-event-list-07 (work in progress), January 2005.

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

[14] Rosenberg, J., "Extensible Markup Language (XML) Formats for Representing Resource Lists", draft-ietf-simple-xcap-list-usage-05 (work in progress), February 2005.

[15] Burger, E., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", draft-ietf-sip-content-indirect-mech-05 (work in progress), October 2004.

Author's Address

Jonathan Rosenberg
Cisco Systems

600 Lanidex Plaza
Parsippany, NJ 07054
US

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

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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