         An Extensible Markup Language (XML) Configuration Access Protocol (XCAP)
                            Diff Event Package

Abstract

   This document describes an "xcap-diff" SIP (Session Initiation
   Protocol) event package for the SIP Event Notification Framework,
   which clients can use to receive notifications of changes to
   Extensible Markup Language (XML) Configuration Access Protocol (XCAP)
   resources. The initial synchronization information exchange and
   document updates are based on the XCAP Diff format.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc5875.

Table of Contents

1. Introduction

   The SIP events framework [RFC3265] describes subscription and
   notification conventions for the Session Initiation Protocol (SIP)
   [RFC3261]. The Extensible Markup Language (XML)
   [W3C.REC-xml-20060816] Configuration Access Protocol (XCAP) [RFC4825]
   allows a client to read, write, and modify XML-formatted application
   usage data stored on an XCAP server.

   While XCAP allows authorized users or devices to modify the same XML
   document, XCAP does not provide an effective mechanism (beyond
   polling) to keep resources synchronized between a server and a
   client. This memo defines an "xcap-diff" event package that,
   together with the SIP event notification framework [RFC3265] and the
   XCAP diff format [RFC5874], allows a user to subscribe to changes in
   an XML document, and to receive notifications whenever the XML
   document changes.

   There are three basic features that this event package enables:

   First, a client can subscribe to a list of XCAP documents' URLs in a
   collection located on an XCAP server. This allows a subscriber to
   compare server resources with its local resources using the URLs and
   the strong entity tag (ETag) values of XCAP documents, which are
   shown in the XCAP diff format, and to synchronize them.

   Second, this event package can signal a change in those documents in
   one of three ways. The first mode only indicates the event type and
   does not include document contents, so the subscriber uses HTTP
   [RFC2616] to retrieve the updated document. The second mode includes
   document content changes in notification messages, using the XML-
   Patch-Ops [RFC5261] format with minimal notification size. The third
   mode also includes document content changes in notification messages
   with the same XML-Patch-Ops format, but is more verbose, and shows
   the full HTTP version history.

   Third, the client can subscribe to specific XML elements or
   attributes (XCAP components) showing their existing contents in the
   resulting XCAP diff format notification messages. If the requested
   component does not exist but is later created, the notifier sends a
   notification with the component's content. The notifier also sends
   notifications when the subscribed XCAP components are removed, for
   example, after a successful HTTP DELETE request.

2. Terminology

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119, BCP 14
   [RFC2119] and indicate requirement levels for compliant
   implementations.

3. Definitions

   The following terms are used in this document:

   XCAP component: An XML element or an attribute, which can be
      updated, removed, or retrieved with XCAP.

   Aggregating: An XCAP client can update only a single XCAP component
      at a time using HTTP. However, a notifier may be able to
      aggregate a series of these modifications into a single
      notification using XML-Patch-Ops semantics encoded in the XCAP
      diff format.

   This document reuses terminology mostly defined in XCAP [RFC4825] and
   some in WebDAV [RFC4918].

4. XCAP Diff Event Package

4.1. Overview of Operation with Basic Requirements

   To receive "xcap-diff" event package features, the subscriber
   indicates its interest in certain resources by including a URI list
   in the subscription body to the notifier. Each URL in this list MUST
   be an HTTP URL that identifies a collection, an XCAP document, or an
   XCAP component. Collection URLs MUST have a trailing forward slash
   "/", following the conventions of WebDAV [RFC4918]. A collection
   selection includes all documents in that collection and recursively
   all documents in sub-collections. The URL of an XCAP component
   consists of the document URL with the XCAP Node Selector added.
   Although the XCAP Node Selector allows all in-scope namespaces of an
   element to be requested, the client MUST NOT subscribe to namespaces.

   The notifier MUST support XCAP component subscriptions. The notifier
   sends the first notification in response to the subscription, and
   this first notification MUST contain the URLs of the documents and
   XCAP component contents that are part of the subscription. The
   subsequent notifications MAY contain patches to these documents. The
   subscriber can specify how the notifier will signal the changes of
   documents by using the 'diff-processing' event package parameter,
   covered in Section 4.3. Note that the existence of the "diff-

processing" parameter or its value has no influence on XCAP component
subscriptions.

4.2. Event Package Name

The name of this event package is "xcap-diff". As specified in
[RFC3265], this value appears in the Event header field present in
SUBSCRIBE and NOTIFY requests.

4.3. 'diff-processing' Event Package Parameter

With the aid of the optional "diff-processing" Event header field
parameter, the subscriber indicates a preference as to how the
notifier SHOULD indicate change notifications of documents. The
possible values are "no-patching", "xcap-patching", and "aggregate".
All three modes provide information that allows the subscriber to
synchronize its local cache, but only the "xcap-patching" mode
provides intermediate states of the version history. The notifier
SHOULD use the indicated mode if it understands it (as doing so
optimizes network traffic within the capabilities of the receiver).

   The "no-patching" value means that the notifier indicates only the
   document and the event type (creation, modification, and removal)
   in the notification. The notification does not necessarily
   indicate the full HTTP ETag change history. Notifiers MUST
   support the "no-patching" mode as a base-line for
   interoperability. The other, more complex modes are optional.

   The "xcap-patching" value means that the notifier includes all
   updated XCAP component contents and entity tag (ETag) changes made
   by XCAP clients (via HTTP). The client receives the full (HTTP)
   ETag change history of a document.

   The "aggregate" value means that the notifier MAY aggregate
   several individual XCAP component updates into a single XCAP diff
   <document> element. The policy for determining whether or not to
   apply aggregation or to determine how many updates to aggregate is
   locally determined.

   The notifier SHOULD support the "xcap-patching" and "aggregate"
   modes, and thus implement XML-Patch-Ops [RFC5261] diff-generation,
   because this can greatly reduce the required number of
   notifications and overall transmissions.

If the subscription does not contain the "diff-processing" header
field parameter, the notifier MUST default to the "no-patching" mode.

   Note: To see the difference between "xcap-patching" and
   "aggregate" modes, consider a document that has versions "a", "b",
   and "c" with corresponding ETag values "1", "2", and "3". The
   "xcap-patching" mode will include first the change from version
   "a" to "b" with the versions' corresponding "1" and "2" ETags and
   then the change from version "b" to "c" with their "2" and "3"
   ETags. The "aggregate" mode optimizes the change and indicates
   only a single aggregated change from "a" to "c" with the old "1"
   and new "3" ETags. If these changes are closely related, that is,
   the same element has been updated many times, the bandwidth
   savings are larger.

This "diff-processing" parameter is a subscriber hint to the
notifier. The notifier may respond using a simpler mode, but not a
more complex one. Notifier selection of a mode is covered in
Section 4.7. During re-subscriptions, the subscriber MAY change the
diff-processing parameter.

The formal grammar [RFC5234] of the "diff-processing" parameter is:

     diff-processing = "diff-processing" EQUAL (
        "no-patching" /
        "xcap-patching" /
        "aggregate" /
        token)

where EQUAL and token are defined in RFC 3261 [RFC3261].

4.4. SUBSCRIBE Bodies

   The URI list is described by the XCAP resource list format [RFC4826],
   and is included as a body of the initial SUBSCRIBE request. Only a
   simple subset of that format is required, a flat list of XCAP request
   URIs. The "uri" attribute of the <entry> element contains these URI
   values. The subscriber MUST NOT use hierarchical lists or <entry-
   ref> references, etc. (though in the future, semantics may be
   expanded thanks to the functionality in the resource list format).
   In subsequent SUBSCRIBE requests, such as those used for refreshing
   the expiration timer, the subscribed URI list MAY change, in which
   case the notifier MUST use the new list.

   The SUBSCRIBE request MAY contain an Accept header field. If no such
   header field is present, it has a default value of "application/
   xcap-diff+xml". If the header field is present, it MUST include
   "application/xcap-diff+xml", and MAY include any other types.

The SUBSCRIBE request MAY contain the Suppress-If-Match header field
[RFC5839], which directs the notifier to suppress either the body of
a subsequent notification or the entire notification if the ETag
value matches.

If the SUBSCRIBE body contains elements or attributes that the
notifier doesn't understand, the notifier MUST ignore them.

Subscribers need to appropriately populate the Request-URI of the
SUBSCRIBE request, typically set to the URI of the notifier. This
document does not constrain that URI. It is assumed that the
subscriber is provisioned with or has learned the URI of the notifier
of this event package.

The XCAP server will usually be co-located with the SIP notifier, so
the subscriber MAY use relative XCAP Request-URIs. Because relative
Request-URIs are allowed, the notifier MUST know how to resolve these
against the correct XCAP Root URI value.

Figure 1 shows a SUBSCRIBE request and body covering several XCAP
resources: a "resource-list" document, a specific element (XCAP
component) in a "rls-services" document, and a collection in "pidf-
manipulation" application usage. The "Content-Type" header of this
SUBSCRIBE request is "application/resource-lists+xml".

```
SUBSCRIBE sip:tests@xcap.example.com SIP/2.0
...
Accept: application/xcap-diff+xml
Event: xcap-diff; diff-processing=aggregate
Content-Type: application/resource-lists+xml
Content-Length: [XXX]
Expires: 4200

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
 <list>
  <entry uri="resource-lists/users/sip:joe@example.com/index"/>
  <entry uri="rls-services/users/sip:joe@example.com/index/
~~/*/service%5b@uri='sip:marketing@example.com'%5d"/>
  <entry uri="pidf-manipulation/"/>
 </list>
</resource-lists>
```

                    Figure 1: Example subscription body

When subscribing to XCAP components, namespace prefixes of XCAP Node
Selectors MUST be properly resolved to namespace URIs. Section 6.4
of RFC 4825 [RFC4825] describes the conventions when using prefixes

in XCAP Node Selectors. If only XCAP Default Document Namespace is
used, just like in the previous example (where a <service> element is
selected), the query component of the "uri" value is not required.

4.5. Subscription Duration

The default expiration time for subscriptions within this package is
3600 seconds. As per RFC 3265 [RFC3265], the subscriber MAY specify
an alternative expiration timer in the Expires header field.

4.6. NOTIFY Bodies

The format of the NOTIFY message body either is the default of
"application/xcap-diff+xml" or is a format listed in the Accept
header field of the SUBSCRIBE.

In this event package, notification messages contain an XCAP diff
document [RFC5874].

The XCAP diff format [RFC5874] can include the subscribed XCAP
component contents. For documents, the format can also include
corresponding URIs, ETag values, and patching instructions from
version "a" to "b". Removal events (of documents, elements, or
attributes) can be identified too. Except for collection selections,
the "sel" selector values of the XCAP diff format MUST be octet-by-
octet equivalent to the relevant "uri" parameter values of the
<entry> element of the "resource-list" document.

With XCAP component subscriptions, XCAP Node Selectors can contain
namespace prefixes. A notifier MUST then resolve these prefixes to
namespace URIs according to RFC 4825 [RFC4825] conventions. In other
words, notifiers MUST be aware of XCAP Default Document Namespaces
for Application Usages when they locate unprefixed qualified XCAP
elements. Note that the namespace resolving rules of Patch operation
elements <add>, <replace>, and <remove> are described in Section
4.2.1 of [RFC5261].

4.7. Notifier Generation of NOTIFY Requests

During the initial subscription, or if the URI list changes in
SUBSCRIBE refresh requests, the notifier MUST resolve the requested
XCAP resources and their privileges. If there are superfluous
resource selections in the requested URI list, the notifier SHOULD
NOT provide overlapping similar responses for these resources. A
resource for which an authenticated user does not have a read
privilege MUST NOT be included in the XCAP diff format. Note that an
XCAP component that could not be located with XCAP semantics does not
produce an error. Instead, the request remains in a "pending" state,

that is, waiting for this resource to be created (or read access
granted if XCAP Application Usages utilize dynamic access control
lists). Subscriptions to collections have a similar property: once a
new document is created into the subscribed collection, the creation
of a new resource is signaled with the next NOTIFY request.

After the notifier knows the list of authorized XCAP resources, it
generates the first NOTIFY, which contains URI references to all
subscribed, existing documents for which the subscriber has read
privileges, and typically XCAP component(s) of existing content.

After sending the initial notification, the notifier selects a diff-
processing mode for reporting changes. If the subscriber suggested a
mode in the "diff-processing" parameter of the SUBSCRIBE, the
notifier MAY use that requested mode or MAY fall back to a simpler
operational mode, but the notifier MUST NOT use a more complex mode
than the one chosen by the subscriber. From least to most complex,
the order of the modes is the following: "no-patching", "xcap-
patching", "aggregate". Thus, the notifier may respond to an
"aggregate" request using any mode, but cannot reply to an "xcap-
patching" subscription using the "aggregate" mode. Naturally, the
notifier MUST handle a "no-patching" request with the "no-patching"
mode.

In all modes, the notifier MUST maintain the chronological order of
XCAP changes. If several changes to a given resource are presented
in a single notification, the chronological update order MUST be
preserved in the XML document order of the notification body.
Chronological order is preserved to simplify the required subscriber
implementation logic.

While the "aggregate" mode uses bandwidth most efficiently, it
introduces other challenges. The initial synchronization might fail
with rapidly changing resources, because the "aggregate" mode
messages might not include the full version history of a document and
the base XCAP protocol does not support version history retrievals of
documents. When new documents are created in subscribed collections
and the notifier is aggregating patches, the same issue can occur.
In a corner case (such as when the XML prolog changes), the notifier
may not be able to provide patches with the XML-Patch-Ops [RFC5261]
semantics.

If the notifier has to temporarily disable diff generation and send
only the URI references of some changed documents to the subscriber,
it MUST continue with the "xcap-patching" mode afterwards for these
resources, if the initial subscription also started with the "xcap-
patching" mode.

   Note: The diff-generation may be disabled when the NOTIFY body
   becomes impractically large or an intermediate error has happened.
   As the subscriber loses track of the patching operations, it must
   refresh to a "known good" state by downloading current documents.
   Once it has done so, it can re-subscribe, for example, with the
   "aggregate" mode.

In the "aggregate" mode, the notifier chooses how long to wait for
multiple patches to combine and how this combination is done.

In the "xcap-patching" mode, the notifier MAY try to optimize the
diff-generation, for example, by eliminating redundant information
since some XCAP clients will probably not have completely optimized
their HTTP PUT request.

   Note: It is straightforward to change the XCAP client's change
   requests: PUT and DELETE (sent via HTTP) to use XML-Patch-Ops
   semantics. While XCAP does not support patching of all XML node
   types -- for example, namespace declarations cannot be added
   separately -- efficient utilization of XML-Patch-Ops can sometimes
   significantly reduce the bandwidth requirements at the expense of
   extra processing.

After the notifier has reported the existence of an XCAP component,
it MUST also report its removal consistently. For example, the
removal of the parent element of the subscribed element requires the
same signaling since the subscribed element ceases to exist. To
signal the removal of an XCAP component, the notifier sets the
Boolean "exist" attribute value of the <element> or <attribute>
elements to false. Even with rapidly changing resources, the
notifier MUST signal only the latest state: e.g., whether or not the
XCAP component exists.

When the notifier receives a re-subscription, it MUST re-send the
current full XML diff content unless the subscriber has requested a
conditional subscription [RFC5839] by using the header field
Suppress-If-Match: [ETag value]. With a conditional re-subscription,
the notifier MUST also inspect the subscription body when determining
the current subscription state. Since the subscription is based on a
list of XCAP request URIs, it is RECOMMENDED that the notifier does
not consider the order of these URIs when determining the equivalence
to "stored" previous states. If a match to the previous state is not
found, the NOTIFY message MUST contain the full XML diff state
(similar to the initial notification). The notifiers SHOULD
implement the conditional subscription handling with this event
package.

During re-subscriptions, the subscriber may change the value of the
diff-processing parameter. The value change influences only
subsequent notifications, not the notification (if generated)
followed immediately after the (re-)SUBSCRIBE request.

Event packages like this require reliable transfer of NOTIFY
messages. This means that all messages MUST successfully be
transferred or the document will become out of sync, and then patches
will most likely fail (or worse, have unintended consequences). This
"xcap-diff" event package requires, similar to Partial-PIDF-Notify
RFC 5263 [RFC5263], that a notifier MUST NOT send a new NOTIFY
request to the same dialog unless a successful 200-response has been
received for the last sent NOTIFY request. If the NOTIFY request
fails due to a timeout, the notifier MUST remove the subscription.

   Note: This requirement ensures that out-of-order events will not
   happen or that the dialog will terminate after non-resolvable
   NOTIFY request failures. In addition, some of the probable NOTIFY
   error responses (for example, 401, 407, 413) can possibly be
   handled gracefully without tearing down the dialog.

If, for example, the subscriber has selected too many elements to
which to subscribe, such that the notification body would be
impractically large (that is, an intermediate NOTIFY failure), the
notifier MAY discard the <element> element content. The existence of
elements is then indicated with an empty <element> element, and the
content is not shown for those resources. In other words, the
<element> element does not have a child element that would show the
subscribed "full" element content.

4.8. Subscriber Processing of NOTIFY Requests

The first NOTIFY request will usually contain references to HTTP
resources including their strong ETag values. If the subscriber does
not have similar locally cached versions, it will typically start an
unconditional HTTP GET request for those resources. During this HTTP
retrieval time, the subscriber MAY also receive patches to these
documents if it has requested them and if the documents are changing
rapidly. It can happen that the version retrieved by HTTP is not the
same than what is indicated in the initial notification. A
subscriber can then chain the modification list for each document,
and locate the position where the previous ETag value is equal to
that retrieved via HTTP. If an ETag match is not found from the
first change, a subscriber MUST omit all changes up to the point
where it is the same. From that change onwards, the subscriber
applies all reported patches. If the version received via HTTP is
newer than any received via the notifications, the subscriber may not
find an equivalent match of an ETag value from the chain of patches.

This can happen since notifications are reported after HTTP changes
and preferably at some minimum intervals. Also, document removals
can be reported in notifications and/or HTTP retrievals may fail
because of unexisting resources (rapidly changing). In any case, the
subscriber can re-fetch the possible out-of-sync document, wait for
subsequent notifications or refresh the subscription (with "xcap-
patching"), and repeat the described "sync" algorithm until a "full"
sync is achieved.

If the notifier aggregates patches, the previous modification list
may not contain the ETag value retrieved by HTTP simply because of
aggregation optimizations. A similar out-of-sync cycle can happen
when new (subscribed) documents are created that change rapidly. To
avoid such difficulties, the subscriber MAY start the subscription
with the "xcap-patching" mode, and then refresh the subscription with
the "aggregate" mode after the initial sync is achieved. Naturally,
the subscriber can revert back to the "xcap-patching" mode from
"aggregate" at any time and vice versa.

If the subscriber has received a "full" sync and it has detected that
some of the resources are being served with the "xcap-patching" mode
while others are in the "aggregate" mode, it SHOULD refresh the
subscription to the "aggregate" mode.

The notifier MAY at any time temporarily use the "no-patching" mode
for some resources so that the subscriber receives only URI
references of modifications. When the notifier is acting in this
mode, several cycles MAY be needed before an initial "full" sync is
achieved. As the notifier MAY change modes in the middle of a
dialog, the subscriber is always responsible for taking appropriate
actions. Also, as the last resort, the subscriber MAY always disable
the usage of diff-processing by setting the "diff-processing"
parameter to "no-patching".

If a diff format cannot be applied due to patch processing and/or
programming errors (for a list, see Section 5.1 of [RFC5261]), the
subscriber SHOULD refresh the subscription and disable patching by
setting the "diff-processing" parameter to "no-patching". The
subscriber SHOULD NOT reply with a non-200 response since the
notifier cannot make corrections.

During unconditional re-subscriptions, the subscriber MUST stamp the
received state of all previous resources as stale. However, if a
conditional [RFC5839] re-subscription is successful, the subscriber
MUST preserve the current state of resources unless the subscribed
URI list has changed. That is, the subscriber MUST fetch the
resource's state, for example, from some local cache.

4.9. Handling of Forked Requests

   This specification allows only a single dialog to be constructed from
   an initial SUBSCRIBE request. If the subscriber receives forked
   responses to a SUBSCRIBE, the subscriber MUST apply the procedures in
   Section 4.4.9 of RFC 3265 [RFC3265] for handling non-allowed forked
   requests.

4.10. Rate of Notifications

   Notifiers of an "xcap-diff" event package SHOULD NOT generate
   notifications for a single subscription at a rate of more than once
   every five seconds.

4.11. State Agents

   State agents play no role in this package.

5. An Initial Example NOTIFY Document

   Figure 2 shows an example initial XCAP diff format document provided
   by the first NOTIFY request to the SUBSCRIBE example in Figure 1.
   The following is an example Event header field for this SUBSCRIBE
   request:

   Event: xcap-diff; diff-processing=aggregate

   The subscriber requests that the notifier "aggregate" XCAP component
   updates and anticipates that the subsequent notifications will
   contain aggregated patches to these documents.

```
   <?xml version="1.0" encoding="UTF-8"?>
   <d:xcap-diff xmlns:d="urn:ietf:params:xml:ns:xcap-diff"
                xmlns:s="urn:ietf:params:xml:ns:rls-services"
              xcap-root="http://xcap.example.com/root/">

    <d:document new-etag="7ahggs"
              sel="resource-lists/users/sip:joe@example.com/index"/>

    <d:document new-etag="30376adf"
              sel="pidf-manipulation/users/sip:joe@example.com/index"/>

    <d:element sel="rls-services/users/sip:joe@example.com/index/
   ~~/*/service%5b@uri='sip:marketing@example.com'%5d"
            xmlns:rl="urn:ietf:params:xml:ns:resource-lists"
       ><s:service uri="sip:marketing@example.com">
         <s:list name="marketing">
           <rl:entry uri="sip:joe@example.com"/>
           <rl:entry uri="sip:sudhir@example.com"/>
         </s:list>
         <s:packages>
           <s:package>presence</s:package>
         </s:packages>
       </s:service></d:element>

   </d:xcap-diff>
```

           Figure 2: An example initial XCAP diff format document

   Note that the resource-list "index" document included only the new
   ETag value, as the document existed during the subscription time. In
   the "pidf-manipulation" collection, there is only a single document
   for which the user has read privileges. The <service> element exists
   within the rls-services "index" document and its content is shown.
   Note also that the <service> element was located using the Default
   Document Namespace (no prefix in XCAP Node Selector value) although
   it has an "s" prefix in the source document.

6. IANA Considerations

   IANA has added a new event package to the SIP Event Types Namespace
   registry as follows:

```
     Package Name     Type        Contact                    Reference
     ------------     --------    -------                    ---------
     xcap-diff        package     IETF Real-time Applications [RFC5875]
                                  <rai@ietf.org>
```

7. Security Considerations

   This document defines a new SIP event package for the SIP event
   notification framework specified in RFC 3265 [RFC3265]. As such, all
   the security considerations of RFC 3265 apply. The configuration
   data can contain sensitive information, and both the client and the
   server need to authenticate each other. The notifiers MUST
   authenticate the "xcap-diff" event package subscriber using the
   normal SIP authentication mechanisms, for example, Digest as defined
   in Section 22 of RFC 3261 [RFC3261]. The notifiers MUST be aware of
   XCAP User Identifiers (XUI) and how to map the authenticated SIP
   identities unambiguously with XUIs.

   Since XCAP [RFC4825] provides a basic authorization policy for
   resources and since notifications contain content similar to XCAP
   resources, the security considerations of XCAP also apply. The
   notifiers MUST obey the XCAP authorization rules when signalling
   resource changes. In practice, this means following the read
   privilege rules of XCAP resources.

   Denial-of-service attacks against notifiers deserve special mention.
   The following can cause denial of service due to intensive
   processing: subscriptions to a long list of URIs, "pending"
   subscriptions to non-existent documents or XCAP components, and diff-
   generation algorithms that try to optimize the required bandwidth
   usage to extremes.

   The mechanism used for conveying xcap-diff event information MUST
   ensure integrity and SHOULD ensure confidentially of the information.
   An end-to-end SIP encryption mechanism, such as S/MIME described in
   Section 26.2.4 of RFC 3261 [RFC3261], SHOULD be used. If that is not
   available, it is RECOMMENDED that TLS [RFC5246] be used between
   elements to provide hop-by-hop authentication and encryption
   mechanisms described in Sections 26.2.2 ("SIPS URI Scheme") and
   26.3.2.2 ("Interdomain Requests") of RFC 3261 [RFC3261].

8. Acknowledgments

   The author would like to thank Jonathan Rosenberg for his valuable
   comments and for providing the initial event package, and Aki Niemi,
   Pekka Pessi, Miguel Garcia, Pavel Dostal, Krisztian Kiss, Anders
   Lindgren, Sofie Lassborn, Keith Drage, Stephen Hinton, Byron Campen,
   Avshalom Houri, Ben Campbell, Paul Kyzivat, Spencer Dawkins, Pasi
   Eronen, and Chris Newman for their valuable comments. Lisa Dusseault
   critiqued the document during IESG review, raising numerous issues
   that resulted in improved document quality. Further, technical
   writer A. Jean Mahoney devoted countless hours to integrating Lisa's
   comments and cleaning up the technical English usage.

9. References

9.1. Normative References

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2616]   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.

   [RFC3261]   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.

   [RFC3265]   Roach, A., "Session Initiation Protocol (SIP)-Specific
               Event Notification", RFC 3265, June 2002.

   [RFC4825]   Rosenberg, J., "The Extensible Markup Language (XML)
               Configuration Access Protocol (XCAP)", RFC 4825, May 2007.

   [RFC4826]   Rosenberg, J., "Extensible Markup Language (XML) Formats
               for Representing Resource Lists", RFC 4826, May 2007.

   [RFC5234]   Crocker, D. and P. Overell, "Augmented BNF for Syntax
               Specifications: ABNF", STD 68, RFC 5234, January 2008.

   [RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
               (TLS) Protocol Version 1.2", RFC 5246, August 2008.

   [RFC5261]   Urpalainen, J., "An Extensible Markup Language (XML) Patch
               Operations Framework Utilizing XML Path Language (XPath)
               Selectors", RFC 5261, September 2008.

   [RFC5839]   Niemi, A. and D. Willis, "An Extension to Session
               Initiation Protocol (SIP) Events for Conditional Event
               Notification", RFC 5839, May 2010.

   [RFC5874]   Rosenberg, J. and J. Urpalainen, "An Extensible Markup
               Language (XML) Document Format for Indicating a Change in
               XML Configuration Access Protocol (XCAP) Resources",
               RFC 5874, May 2010.

9.2. Informative References

   [RFC4918]   Dusseault, L., "HTTP Extensions for Web
                          Distributed Authoring and Versioning
                          (WebDAV)", RFC 4918, June 2007.

   [RFC5263]   Lonnfors, M., Costa-Requena, J., Leppanen,
                          E., and H. Khartabil, "Session Initiation
                          Protocol (SIP) Extension for Partial
                          Notification of Presence Information",
                          RFC 5263, September 2008.

   [W3C.REC-xml-20060816] Paoli, J., Bray, T., Yergeau, F., Maler, E.,
                          and C. Sperberg-McQueen, "Extensible Markup
                          Language (XML) 1.0 (Fourth Edition)", World
                          Wide Web Consortium FirstEdition REC-xml-
                          20060816, August 2006,
                          <http://www.w3.org/TR/2006/REC-xml-20060816>.

Appendix A. Informative Examples

   These examples illustrate the basic features of the xcap-diff event
   package. Only the relevant header fields are shown. Note also that
   the SIP request URIs of these examples don't correspond to reality.

A.1. Initial Documents on an XCAP Server

   The following documents exist on an XCAP server (xcap.example.com)
   with an imaginary "tests" application usage (there's no Default
   Document Namespace defined in this imaginary application usage).

   http://xcap.example.com/tests/users/sip:joe@example.com/index:

   <?xml version="1.0" encoding="UTF-8"?>
   <doc>
     <note>This is a sample document</note>
   </doc>

   and then

   http://xcap.example.com/tests/users/sip:john@example.com/index:

   <?xml version="1.0" encoding="UTF-8"?>
   <doc>
     <note>This is another sample document</note>
   </doc>

A.2. An Initial Subscription

   The following demonstrates the listing of collection contents and it
   shows only resources where the user has read privileges. The user
   Joe, whose XUI is "sip:joe@example.com", sends an initial
   subscription:

   SUBSCRIBE sip:tests@xcap.example.com SIP/2.0
   ...
   Accept: application/xcap-diff+xml
   Event: xcap-diff; diff-processing=aggregate
   Content-Type: application/resource-lists+xml
   Content-Length: [XXX]

   <?xml version="1.0" encoding="UTF-8"?>
   <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
    <list>
     <entry uri="tests/users/sip:joe@example.com/"/>
    </list>
   </resource-lists>

In addition to the 200 (OK) response, the notifier sends the first
NOTIFY:

```
NOTIFY sip:joe@userhost.example.com SIP/2.0
...
Event: xcap-diff
Content-Type: application/xcap-diff+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
           xcap-root="http://xcap.example.com/">

  <document new-etag="7ahggs"
            sel="tests/users/sip:joe@example.com/index"/>

</xcap-diff>
```

The subscriber learns that the document on this "tests" application
usage is equivalent to its locally cached version, so it does not
act. If the local version had been different, the subscriber would
most likely re-fetch the document.

If the subscriber had requested the "tests/users/" collection, the
notification body would have been the same since Joe has no read
privileges to John's resources (XCAP default behavior).

If the Expires header field had a value "0", the request would be
similar to the PROPFIND method of WebDAV. The syntax and responses
differ, however.

A.3. A Document Addition into a Collection

   Let's say that Joe adds a new document to his collection, using
   either the same client or another client running on a different
   device. He does an HTTP PUT to his application usage collection:

```
PUT /tests/users/sip:joe@example.com/another_document HTTP/1.1
Host: xcap.example.com
....
Content-Type: application/xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<doc>
  <note>This is another sample document</note>
</doc>
```

   This HTTP PUT request results in the XCAP client receiving a strong
   HTTP ETag "terteer" for this new document.

   Then the subscriber receives a notification afterwards:

   NOTIFY sip:joe@userhost.example.com SIP/2.0
   ...
   Event: xcap-diff
   Content-Type: application/xcap-diff+xml
   Content-Length: [XXX]

   <?xml version="1.0" encoding="UTF-8"?>
   <xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
            xcap-root="http://xcap.example.com/">

    <document new-etag="terteer"
            sel="tests/users/sip:joe@example.com/another_document"/>

   </xcap-diff>

   Note that the result is "additive"; it doesn't indicate the already
   indicated "index" document. Only the initial (or refreshed)
   notification contains all document URI references.

   If Joe's client both modifies the documents and refreshes the
   subscriptions, it would typically ignore this notification, since its
   modifications had caused the notification. If the client that
   received this NOTIFY hadn't submitted the document change, it would
   probably fetch this new document.

   If Joe's client refreshes the subscription with the same request body
   as in the initial subscription, the result will include these two
   documents: "index" and "another_document" with their ETags.

A.4. A Series of XCAP Component Modifications

   Now Joe's client uses its XCAP patching capability by doing the
   following:

   PUT /tests/users/sip:joe@example.com/index/~~/doc/foo HTTP/1.1
   Host: xcap.example.com
   ....
   Content-Type: application/xcap-el+xml
   Content-Length: [XXX]

   this is a new element

Since the insertion of the element is successful, Joe's client
receives the new HTTP ETag "fgherhryt3" of the updated "index"
document.

Immediately thereafter, Joe's client issues another HTTP request
(this request could even be pipe-lined):

```
PUT /tests/users/sip:joe@example.com/index/~~/doc/bar HTTP/1.1
Host: xcap.example.com
....
Content-Type: application/xcap-el+xml
Content-Length: [XXX]

<bar>this is a bar element
</bar>
```

The reported new HTTP ETag of "index" is now "dgdgdfgrrr".

And Joe's client issues yet another HTTP request:

```
PUT /tests/users/sip:joe@example.com/index/~~/doc/foobar HTTP/1.1
Host: xcap.example.com
....
Content-Type: application/xcap-el+xml
Content-Length: [XXX]

<foobar>this is a foobar element</foobar>
```

The reported new ETag of "index" is now "63hjjsll".

After awhile, Joe's client receives a notification with an embedded
patch since it has requested "aggregate" diff-processing and the
notifier is capable of producing them:

```
NOTIFY sip:joe@userhost.example.com SIP/2.0
...
Event: xcap-diff
Content-Type: application/xcap-diff+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<d:xcap-diff xmlns:d="urn:ietf:params:xml:ns:xcap-diff"
             xcap-root="http://xcap.example.com/">

 <d:document previous-etag="7ahggs3"
             sel="tests/users/sip:joe@example.com/index"
             new-etag="63hjjsll">
   <d:add sel="*"
```

```
      ><foo>this is a new element</foo><bar>this is a bar element
   </bar><foobar>this is a foobar element</foobar></d:add>
    </d:document>

   </d:xcap-diff>
```

Joe's client applies this patch to the locally cached "index"
document, detects the ETag update, and stores the last ETag value.
Note how several XCAP component modifications were aggregated.

Note also that, if Joe's client did not have a locally cached version
of the reference document, it would have needed to do an HTTP GET
request after the initial notification. If the ETag of the received
resource by HTTP did not match either the previous or new ETag of
this aggregated patch, an out-of-sync condition would be probable.
This issue is not typical, but it can happen. To resolve the issue,
the client could re-fetch the "index" document and/or wait for
subsequent notifications to detect a match. A better and simpler way
to avoid the issue is to refresh the subscription with the "xcap-
patching" mode and later refresh with the "aggregate" mode.

Alternatively, if the notifier's operational mode been "xcap-
patching", the NOTIFY could have been the following:

```
NOTIFY sip:joe@userhost.example.com SIP/2.0
...
Event: xcap-diff
Content-Type: application/xcap-diff+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<d:xcap-diff xmlns:d="urn:ietf:params:xml:ns:xcap-diff"
          xcap-root="http://xcap.example.com/">

 <d:document previous-etag="7ahggs"
             sel="tests/users/sip:joe@example.com/index"
             new-etag="fgherhryt3">
   <d:add sel="*"
    ><foo>this is a new element</foo></d:add></d:document>

 <d:document previous-etag="fgherhryt3"
             sel="tests/users/sip:joe@example.com/index"
             new-etag="dgdgdfgrrr">
   <d:add sel="*"
    ><bar>this is a bar element
</bar></d:add></d:document>

 <d:document previous-etag="dgdgdfgrrr"
```

```
                 sel="tests/users/sip:joe@example.com/index"
                 new-etag="63hjjsll">
      <d:add sel="*"
       ><foobar>this is a foobar element</foobar></d:add></d:document>

   </d:xcap-diff>
```

   If the client had to re-fetch the "index" document after the initial
   notification, it could have skipped some or all of these patches,
   depending on whether the HTTP ETag matched some of these ETags in the
   chain of patches. If the HTTP ETag did not match and the received
   HTTP version is a newer version indicated in later notification(s),
   the sync may then be achieved since the notifier provided the full
   change history in the "xcap-patching" mode.

   Last, the notifier could (temporarily) fall back to the "no-patching"
   mode, which allows the notifier to keep the dialog alive when there
   are too many updates:

```
NOTIFY sip:joe@userhost.example.com SIP/2.0
...
Event: xcap-diff
Content-Type: application/xcap-diff+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
           xcap-root="http://xcap.example.com/">

  <document previous-etag="7ahggs3"
            sel="tests/users/sip:joe@example.com/index"
            new-etag="63hjjsll"/>

  </xcap-diff>
```

   At any time, the notifier may fall back to the "no-patching" mode for
   some or all of the subscribed documents.

A.5. An XCAP Component Subscription

   The user Joe sends an initial subscription for the "id" attribute of
   a <doc> element. The "index" document exists, but the <doc> root
   element does not contain the "id" attribute at the time of the
   subscription.

```
SUBSCRIBE sip:tests@xcap.example.com SIP/2.0
...
Accept: application/xcap-diff+xml
Event: xcap-diff
Content-Type: application/resource-lists+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
 <list>
  <entry uri="tests/users/sip:joe@example.com/index/~~/doc/@id"/>
 </list>
</resource-lists>
```

The first NOTIFY looks like the following since there is nothing to
indicate:

```
NOTIFY sip:joe@userhost.example.com SIP/2.0
...
Event: xcap-diff
Content-Type: application/xcap-diff+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
           xcap-root="http://xcap.example.com/"/>
```

Note that if the "index" document hadn't existed, the first NOTIFY
request would have been the same. The XCAP diff document format
doesn't indicate reasons for non-existing resources.

Afterwards, Joe's client updates the whole document root element
including the attribute "id" (not a typical XCAP operation or a
preferred one, just an illustration here):

```
PUT /tests/users/sip:joe@example.com/index/~~/doc HTTP/1.1
Host: xcap.example.com
....
Content-Type: application/xcap-el+xml
Content-Length: [XXX]

<doc id="bar">This is a new root element</doc>
```

The new HTTP ETag of the "index" document is now "dwawrrtyy".

Then Joe's client gets a notification:

```
NOTIFY sip:joe@userhost.example.com SIP/2.0
...
Event: xcap-diff
Content-Type: application/xcap-diff+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
           xcap-root="http://xcap.example.com/">

 <attribute sel="tests/users/sip:joe@example.com/index/~~/doc/@id"
  >bar</attribute>

</xcap-diff>
```

Note that the HTTP ETag value of the new document is not shown, as it
is irrelevant for this use-case.

Then Joe's client removes the "id" attribute:

```
DELETE /tests/users/sip:joe@example.com/index/~~/doc/@id HTTP/1.1
Host: xcap.example.com
....
Content-Length: 0
```

And the subscriber gets a notification:

```
NOTIFY sip:joe@userhost.example.com SIP/2.0
...
Event: xcap-diff
Content-Type: application/xcap-diff+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
           xcap-root="http://xcap.example.com/">

 <attribute sel="tests/users/sip:joe@example.com/index/~~/doc/@id"
  exists="0"/>

</xcap-diff>
```

The notification indicates that the subscribed attribute was removed
from the document. Naturally, attributes are "removed" if the
element where they belong is removed, for example, by an HTTP DELETE

   request. The component selections indicate only the existence of
   attributes or elements.

A.6. A Conditional Subscription

   The last example is a conditional subscription where a full refresh
   can be avoided when there are no changes in resources. Joe's client
   sends an initial subscription:

   SUBSCRIBE sip:tests@xcap.example.com SIP/2.0
   ...
   Accept: application/xcap-diff+xml
   Event: xcap-diff; diff-processing=xcap-patching
   Content-Type: application/resource-lists+xml
   Content-Length: [XXX]

   <?xml version="1.0" encoding="UTF-8"?>
   <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
    <list>
     <entry uri="tests/users/sip:joe@example.com/"/>
    </list>
   </resource-lists>

   Since there are now two documents in the repository, the first NOTIFY
   looks like the following:

   NOTIFY sip:joe@userhost.example.com SIP/2.0
   ...
   Event: xcap-diff
   SIP-ETag: xggfefe54
   Content-Type: application/xcap-diff+xml
   Content-Length: [XXX]

   <?xml version="1.0" encoding="UTF-8"?>
   <xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
              xcap-root="http://xcap.example.com/">

    <document new-etag="63hjjsll"
              sel="tests/users/sip:joe@example.com/index"/>

    <document new-etag="terteer"
              sel="tests/users/sip:joe@example.com/another_document"/>

   </xcap-diff>

   Note that the NOTIFY request contains the SIP-ETag "xggfefe54". This
   SIP-ETag is placed in the Suppress-If-Match header field of the
   conditional subscription. The "diff-processing" mode also is changed

(or is requested to change):

```
SUBSCRIBE sip:tests@xcap.example.com SIP/2.0
...
Suppress-If-Match: xggfefe54
Accept: application/xcap-diff+xml
Event: xcap-diff; diff-processing=aggregate
Content-Type: application/resource-lists+xml
Content-Length: [XXX]

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
 <list>
  <entry uri="tests/users/sip:joe@example.com/"/>
 </list>
</resource-lists>
```

If the notifier finds a match to the previous stored state when it
evaluates this request, it responds with 204 (No Notification). If
there are no reportable changes as per [RFC5839], NOTIFY request
generation is suppressed. When the notifier can aggregate several
modifications, this re-subscription enables the processing of that
mode thereafter. Indeed, the re-subscription may be quite process-
intensive, especially when there are a large number of relevant
reported resources.

Authors' Addresses

    Jari Urpalainen
    Nokia
    Itamerenkatu 11-13
    Helsinki 00180
    Finland

    Phone: +358 7180 37686
    EMail: jari.urpalainen@nokia.com

    Dean Willis (editor)
    Softarmor Systems LLC
    3100 Independence Pk #311-164
    Plano, TX 75075
    USA

    Phone: +1 214 504 19876
    EMail: dean.willis@softarmor.com