SIP
Internet-Draft
Intended status: Standards Track
Expires: November 6, 2008

J. Urpalainen
Nokia

May 5, 2008

An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Diff Event Package draft-ietf-sip-xcapevent-03

Status of this Memo

This Internet-Draft will expire on November 6, 2008.

Abstract

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

Table of Contents

1. Introduction

The SIP Events framework [RFC3265] describes subscription and notification conventions for the SIP [RFC3261] protocol. 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 the XCAP protocol allows several authorized users or devices to modify the same XML document, XCAP does not provide an effective synchronization mechanism (except polling) to keep resources equivalent 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 [I-D.ietf-simple-xcap-diff], allows a user to subscribe to changes in an XML document, and to receive notifications whenever a change in an XML document takes place.

There are three basic features that this event package enables with the XCAP-Diff [I-D.ietf-simple-xcap-diff] change notification format.

Firstly, it can list a collection [RFC4918] content of an XCAP server, which means in practice listing the URI references of XCAP documents from a collection. This is important when a subscriber is doing an initial synchronization or a comparison of existing server resources to the locally cached XCAP documents, for example. The version-history of document comparisons are based on the strong entity tag (ETag) values of XCAP documents which are also indicated with the XCAP-Diff format.

Secondly, this event package can signal whenever a change is happening in those resources. The changes can be reported with three ways. The simplest model is that only document creations, updates and removals are indicated. The actual contents of those documents are not shown and the subscriber uses the (HTTP) RFC 2616 [RFC2616] protocol for a retrieval of document contents. The two more complex modes allow the changes of documents to be indicated straightaway with the XML-Patch-Ops [I-D.ietf-simple-xml-patch-ops] semantics inside the XCAP-Diff [I-D.ietf-simple-xcap-diff] format. A client can then apply a conditional patch to locally cached documents based on the strong ETag values of documents. The most complex model produces the smallest documents but it doesn't necessarily show the full HTTP version-history information unlike the other, but typically more verbose one.

Lastly, XML element or attribute contents (XCAP components) can be received "in-band", that is straight within the XCAP-Diff notification format. For example, an XCAP element content can be

requested and indicated without the need of a separate HTTP GET request. If this requested node either exists or is later created or modified, the notification body indicates its content. And similarly, the removals of subscribed XCAP components are reported, 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 the XCAP protocol.

Aggregating: While XCAP client can update only a single XCAP component at a time with a relevant HTTP request, a series of these modifications can be aggregated together with the XML-Patch-Ops semantics when they are indicated within the XCAP-Diff format.

This document reuses terminology mostly defined in (XCAP) RFC 4825 [RFC4825] and some in (WebDAV) RFC 4918 [RFC4918].

4. XCAP-Diff Event Package

4.1. Overview of Operation With Basic Requirements

Enabling all of the "xcap-diff" event package features require that the subscriber MUST somehow signal to the notifier the resources it is interested in getting information. These resource selections are indicated simply by sending a URI-list to the notifier in the subscription body. The URIs of this list MUST point to a collection, a document or an XCAP component. When collections are selected, the conventions applied follow the (WebDAV) RFC 4918 [RFC4918] semantics, that is, the subscriber MUST add the forward slash "/" character to the end of the path segment of a URI. A collection selection includes all documents in that particular collection and recursively also all documents in any of the possible sub-collections. The URI of an XCAP component selector consists of the document selector added

with the XCAP Node Selector. Although the XCAP Node Selector allows requesting all in-scope namespaces of an element, subscriptions to them MUST NOT be used.

The first notification of the subscription MUST always contain the URI references of the subscribed, existing documents and/or SHOULD contain the subscribed XCAP element content(s) and/or MUST contain the subscribed XCAP attribute content(s). The notifier MUST thus always support XCAP component subscriptions. The subsequent notifications MAY contain patches to these documents. The notifier MUST support the simplest change notification model, but the XML-Patch-Ops diff-generation is RECOMMENDED to implement. The subscriber MAY control how the notifier will signal the changes of documents. How this can be done, will be shown later in this document.

Whenever a change in a resource or an XCAP component content is indicated inside the XCAP-Diff [I-D.ietf-simple-xcap-diff] format, the subscriber MUST have read privilege to that particular resource.

4.2. Event Package Name

The name of this event package is "xcap-diff". As specified in RFC 3265 [RFC3265], this value appears in the Event header field present in SUBSCRIBE and NOTIFY requests.

4.3. 'diff-processing' Event Package Parameter

With aid of the optional "diff-processing" event header parameter the subscriber directs the notifier how to apply the "XML diffing process" or in other words, how the notifier SHOULD indicate change notifications of documents. The possible values are "no-patching", "xcap-patching" and "aggregate" with the increasing complexity order.

> The "no-patching" value (the simplest operational change notification mode) means that only document creations, modifications and removals are indicated.
>
> The "xcap-patching" value means that all individual XCAP component updates made by XCAP clients along with the entity tag (ETag) changes are indicated.
>
> The "aggregate" value means that the notifier SHOULD aggregate several individual XCAP component updates into a single XCAP-Diff <document> element.

If the subscription does not contain this additional "diff-processing" parameter, the notifier SHOULD send all individual

changes so that the client receives the full (HTTP) ETag change history of a document. In other words, "xcap-patching" is the default operational mode of a notifier. Note that the "no-patching" mode does not necessarily either indicate the full HTTP ETag change history.

Note: If the document history has versions from "a" to "b" to "c" with corresponding ETag values "1", "2" and "3", the "xcap-patching" mode indicates first the change from "a" to "b" with previous "1" and new "2" ETags and also the change from "b" to "c" with previous "2" and new "3" ETags for a particular document. A change from "a" to "b" can for example be an element addition to a document, so it is a single (atomic) change made by XCAP (HTTP) semantics. However, the "aggregate" mode tries to optimize the change and indicates for example only a single aggregated change from "a" to "c" with previous "1" and new "3" ETags. If these changes are closely related, e.g. the same element has been updated many times, the bandwidth savings are naturally larger.

This "diff-processing" parameter is a subscriber hint to the notifier. The notifier MAY fall back to a simpler operational mode but it MUST NOT use a more complex one, when it signals changes. For example, an "aggregate" request MUST be served by the notifier in the "aggregate", "xcap-patching" or "no-patching" modes, and an "xcap-patching" request in the "xcap-patching" and "no-patching" modes. Naturally, the "no-patching" request MUST be served only in the same "no-patching" mode.

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

```
diff-processing = "diff-processing" EQUALS
                  "no-patching" /
                  "xcap-patching" /
                  "aggregate" / token
```

4.4. SUBSCRIBE Bodies

The list of the requested URIs are described by the XCAP resource list format specified in RFC 4826 [RFC4826], and it is included as a body of the SUBSCRIBE request that creates the subscription. Only a simple subset of that format is required, a flat list of XCAP R-URIs. The "uri" attribute of the <entry> element contains these URI values. The usage of hierarchical lists and <entry-ref> references, etc. MUST NOT be used. However, using this format allows adding some future semantics to these subscriptions. In subsequent SUBSCRIBE requests, such as those used for refreshing the expiration timer, the subscribed URI-list MAY change.

Subscribers need to appropriately populate the Request-URI of the SUBSCRIBE request, typically set to the URI of the notifier. This document does not provide any constraints to it. It is assumed that the subscriber is provisioned or has learned the URI of the notifier of this event package. This specification assumes that a subscriber populates initial SUBSCRIBE requests with the URI of that notifier.

It is anticipated that the XCAP server will be collocated with the SIP notifier, so the subscriber MAY use relative XCAP R-URIs. This means that the notifier has then been provisioned somehow the XCAP Root URI value, for example. A future specification(s) MAY be written for the more complex use-cases, this memo describes only the most simple one.

> Note: It is worth noting that the notifier and the XCAP server can be separated (running on different hosts). However, it requires that the notifier has some means (in real-time) to be signalled about content changes of documents on an XCAP server. Also theoretically, a single notifier could serve multiple XCAP servers and absolute XCAP R-URIs could then be used.

Figure 1 shows an example of a subscription body of several XCAP resources: a "resource-list" document, a specific element in a "rls-services" document and a collection in "pidf-manipulation" Application Usage. For all of these resources the client MUST have read privilege in order to actually receive them in a NOTIFY request. The "Content-Type" header of this SUBSCRIBE request is "application/resource-lists+xml".

```
<?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

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

As described in RFC 3265 [RFC3265], 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 an XCAP diff document [I-D.ietf-simple-xcap-diff]. 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 XCAP-Diff format [I-D.ietf-simple-xcap-diff] can indicate the full element and attribute content of XML documents, and for documents the corresponding URIs, the ETag values and patching instructions from version "a" to "b". Also the removals of documents, elements and attributes can be shown. With other than 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.

4.7. Notifier Generation of NOTIFY Requests

During the initial subscription the notifier MUST first resolve the requested XCAP resources. If the subscribed body contains elements or attributes that it doesn't understand, they MUST be ignored by the notifier. If there are superfluous resource selections in the requested URI-list, the notifier SHOULD NOT provide overlapping similar responses for these resources. Only the resources where the authenticated user has read privilege, MUST be included in the XCAP-Diff format. Note that for example, an XCAP component which 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. Subscriptions to collections have a similar property: once a new document is created into the subscribed collection, the creation of a new resource is notified with the next NOTIFY request.

After the list of authorized XCAP resources are known, the notifier generates the first full response. This initial notification body contains URI references to subscribed existing documents and/or XCAP component(s) actual existing content.

After sending the initial notification, the notifier MUST start the

follow-up of the subscribed XCAP component or document updates made by XCAP (HTTP) clients. The diff-processing parameter directs how the notifier reports changes. Regardless of these operational modes, the same end result is achieved, the resources MAY be kept in-sync. Some intermediate states of the version-history of resources MAY be lost by these notifications, but the chronological order of XCAP changes MUST be maintained. The same rule applies if several changes for a given resource are indicated in a single notification, the chronological order MUST follow the XML document order in the XCAP-Diff document.

While with the most complex patching mode "aggregate" the bandwidth usage is the most efficient, it introduces other challenges. The initial synchronization MAY fail with rapidly changing resources, because the "aggregate" mode doesn't necessarily indicate the full version-history of a document and the base XCAP protocol does not support version-history retrievals of documents. Secondly, when new documents are created into the subscribed collections and the notifier is "aggregating" patches, the same issue MAY occur. In a corner-case, the notifier may not be able to provide patches with the XML-Patch-Ops [I-D.ietf-simple-xml-patch-ops] semantics. Therefore, 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.

The notifiers MAY decide the appropriate diff-generation (aggregation) logic themselves, for example how long to wait for subsequent patches if there's already something to signal.

When the notifier is acting in the "xcap-patching" mode, it MAY also disable the diff-generation temporarily for some reason for certain resources, for example when the NOTIFY body becomes impractically large or an intermediate error has happened. All XCAP clients may not try to optimize changes to its extremes, so even when acting in the "xcap-patching" operational mode the notifier MAY try to optimize the diff-generation.

> Note: It is straightforward to change the XCAP HTTP request to the XML-Patch-Ops semantics. While XCAP does not support patching of all XML node types, for example namespace declarations can not be added separately, utilization of XML-Patch-Ops can sometimes significantly reduce the bandwidth requirements in the expense of extra processing.

When the notifier has reported that some XCAP components exist in a document, it MUST also report their removals consistently. For

example, the removal of the parent element of the subscribed element requires the same signalling since the subscribed element ceases to exist after the removal. The removal of an XCAP component is signalled by setting the Boolean "exist" attribute value to false of the <element> or <attribute> elements. Even with rapidly changing resources the notifier MUST signal only the last existing state: whether the XCAP component exists or not.

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

When the notifier process receives a re-subscription it MUST re-send the current full XML-Diff content unless the NOTIFY message can be suppressed with the conditional subscription [I-D.ietf-sip-subnot-etags] semantics by using the header Suppress-If-Match: [ETag value]. With a conditional re-subscription the notifier MUST compare also the subscription body when determining the current subscription state. Since the subscription is based on a list of XCAP R-URIs it is RECOMMENDED when determining the equivalence to "stored" previous states, that the order of appearance of these URIs is not significant. Once 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.

Event packages like this require in practice a reliable transfer of NOTIFY messages. This means that all messages MUST successfully be transferred as otherwise patching will most likely fail or at least the document contents becomes to be out-of-sync. This "xcap-diff" event package requires, similar to Partial-PIDF-Notify [I-D.ietf-simple-partial-notify] that the notifiers 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 condition, the notifier MUST remove the subscription.

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

If for example, the subscriber has selected too many elements to subscribe, so that the notification body would become impractically large (e.g. 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 not have a child element then which 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 (if it has requested them) to these documents if the documents are changing frequently. It can thus happen that the subscriber receives newer versions of documents (with HTTP) than what was indicated in the initial notification. If patches are received in these notifications and if all atomic XCAP modifications are indicated with both previous and new ETags of each resource, it is easy to chain the modification list for a document and possibly omit some or all of the patches based on the received ETag (with HTTP) of a document. Indeed, there's still a chance that the received version by HTTP is newer than any of those versions indicated in the notification so that an equivalent match of an ETag value is not found from the chain of patches. This can happen since notifications are reported after the actual HTTP changes and preferably at some minimum intervals. In such a case, the subscriber SHOULD either wait for subsequent notifications or refresh the subscription and repeat the described "sync" algorithm until a match is achieved.

The notifier MAY at any time disable (temporarily) the diff-processing of some resources so that only URI references of modifications are received. So even when the notifier is acting in this simpler diff-processing ("xcap-patching") mode, several cycles MAY be needed before an initial "full" sync is achieved. As the notifier MAY also disable this diff-processing in the middle of a dialog, the subscriber is always, at any time responsible to make the appropriate, similar actions. Also as the last resort the subscriber MAY always disable the usage of diff-processing.

If the subscription is started straightaway with the "aggregate" mode which doesn't necessarily show the full version-history information, the previous "sync" algorithm breaks more easily. Indeed, it is successful if the received (HTTP) ETag matches either with the previous or the new ETag of the reported aggregated patch. This failure MAY successfully be resolved by re-fetching the out-of-sync document, waiting for subsequent notifications or by refreshing the subscription, but the same issue MAY still repeat. However, in such

a case or in general, the safer way to avoid this out-of-sync issue is to start the subscription with the "xcap-patching" mode, and afterwards refresh the subscription to the "aggregate" mode.

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.

If a diff format can not be applied because of some patch processing and/or programming errors, see Section 5.1 of [I-D.ietf-simple-xml-patch-ops], the subscriber SHOULD refresh the subscription and disable the usage of patching. The subscriber SHOULD NOT reply with a non-200 response when this kind of error happens, since the notifier could hardly make any corrective actions.

During re-subscriptions the received state of all previous resources MUST be stamped as stale except when a conditional [I-D.ietf-sip-subnot-etags] re-subscription is successful. Then the current state of resources MUST be preserved unless the subscribed URI-list has changed, i.e. the resource's state MUST then be fetched for example from some local cache.

4.9. Handling of Forked Requests

This specification only allows a single dialog to be constructed as a result of emitting an initial SUBSCRIBE request. In case a SUBSCRIBE request is forked and the subscriber receives forked responses, the subscriber MUST apply the procedures indicated in Section 4.4.9 of RFC 3265 [RFC3265] for handling non-allowed forked requests.

4.10. Rate of Notifications

Notifiers of "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 document provided by the first NOTIFY request. The subscriber used the list as in the example in Figure 1. An example event header of this SUBSCRIBE request:

Event: xcap-diff; diff-processing=aggregate

The subscriber requests the notifier to actually "aggregate" XCAP component updates together. It is anticipated that the subsequent notifications would contain aggregated patches to these documents.

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

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

 <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:d="urn:ietf:params:xml:ns:xcap-diff"
          xmlns="urn:ietf:params:xml:ns:rls-services"
          xmlns:rl="urn:ietf:params:xml:ns:resource-lists"
    ><service uri="sip:marketing@example.com">
      <list name="marketing">
        <rl:entry uri="sip:joe@example.com"/>
        <rl:entry uri="sip:sudhir@example.com"/>
      </list>
      <packages>
        <package>presence</package>
      </packages>
    </service></d:element>

</xcap-diff>
```

Figure 2: An example initial XCAP-Diff 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 where the user has read privilege. The <services> element exists within the rls-services "index" document and its content is shown.

6. IANA Considerations

This specification instructs IANA to add a new event package to the

SIP Event Types Namespace registry. The new data to be added is:

Package Name	Type	Contact	Reference
xcap-diff	package	IETF SIP Working Group <sip@ietf.org>	[RFCXXXX]

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 identities (XUI) and how to map the authenticated SIP identities unambiguously with XUIs.

Since XCAP [RFC4825] provides basic authorization policy for resources and as notifications contain similar fragment content of 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 the notifiers deserve a special mentioning. Subscriptions to a long list of URIs MAY be too process-intensive. The "pending" subscriptions to un-existing documents or XCAP components impose the same challenge as well as the diff-generation algorithms, at least when they try to optimize the required bandwidth usage to extremes.

The mechanism used for conveying this 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 [RFC4346] be used between elements to provide hop-by-hop authentication and encryption mechanisms described in Section 26.2.2 "SIPS URI Scheme" and Section 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 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 and Paul Kyzivat for their valuable comments.

9. References

9.1. Normative References

[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.

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

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

[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.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[I-D.ietf-simple-xcap-diff] Rosenberg, J. and J. Urpalainen, "An Extensible Markup Language (XML) Document Format for Indicating A Change in XML Configuration Access Protocol (XCAP) Resources", draft-ietf-simple-xcap-diff-08 (work in progress), February 2008.

[I-D.ietf-simple-xml-patch-ops] Urpalainen, J., "An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath)

Selectors", draft-ietf-simple-xml-patch-ops-04 (work in progress), November 2007.

[I-D.ietf-sip-subnot-etags]
Niemi, A., "An Extension to Session Initiation Protocol (SIP) Events for Conditional Event Notification", draft-ietf-sip-subnot-etags-02 (work in progress), February 2008.

9.2. Informative References

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

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

[I-D.ietf-simple-partial-notify]
Lonnfors, M., Costa-Requena, J., Leppanen, E., and H. Khartabil, "Session Initiation Protocol (SIP) extension for Partial Notification of Presence Information", draft-ietf-simple-partial-notify-10 (work in progress), January 2008.

Appendix A. Informative Examples

In all following examples only the very relevant headers for this event package or HTTP requests are shown. These examples illustrate the basic features of this "xcap-diff" event package. Note also that the SIP R-URIs of these examples don't correspond to the reality, i.e. that they typically change within a dialog.

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 (AU) (there's no Default Document Namespace defined in this imaginary AU).

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 user Joe whose XUI is "sip:joe@example.com" does 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 first NOTIFY looks like:

```
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 realizes that there's only a single document on this "tests" Application Usage and it has already an equivalent locally cached version so it doesn't do any actions. Had it a different version locally, it would most likely re-fetch the document.

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

So this demonstrates the listing of a collection contents and it shows only resources where the user Joe has read privilege. If the Expires header has a value "0" this is effectively a similar request to the PROPFIND method of WebDAV, the syntax's and responses differ, however.

A.3. A Document Addition Into a Collection

Let's say that Joe adds a new document to his collection, it can be another client running on a different device where the subscriber runs or it is on the same. So he does an HTTP PUT to his AU 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>
```

As a result to this HTTP PUT request the XCAP client gets 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 this result is "additive", it doesn't indicate the already indicated "index" document. Only the initial (or refreshed) notification contains all document URI references.

If it is the same device doing modifications and the subscription, the subscriber typically ignores this event. If it were an another device it would probably fetch this new document.

If the subscriber decides now to refresh 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 some of the Joe's client decides to utilize XCAP patching capability, so it does 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]

<foo>this is a new element</foo>
```

Again the XCAP client receives the new HTTP ETag "fgherhryt3" of the updated "index" document as the insertion of the element is successful.

Immediately thereafter the XCAP client does again (even pipe-lining could work):

```
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 yet again the XCAP client does:

```
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 the subscriber receives then 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>
```

So the subscriber applies this patch to the locally cached "index" document and also detects the ETag update (and stores the last ETag value). Note that how several XCAP component modifications were aggregated together. Note also that if the client hadn't had a locally cached version of the reference document, it would have needed to do a HTTP GET request after the initial notification. If the ETag of the received resource by HTTP did not then match with either the previous or new ETag of this aggregated patch, an out-of-sync would be probable. So the client could try to re-fetch the "index" document and resolve the issue and/or wait for subsequent notifications to detect a match. Another, and a more certain and simpler way to avoid the issue, is to refresh the subscription with the "xcap-patching" mode and later refresh to the "aggregate" mode. That said, it is not typical that this issue occurs, but it can happen and the client (subscriber) is about to handle the consequences.

Had the "xcap-patching" mode been the operational mode of the notifier, so as an alternative, the NOTIFY response could have been:

```
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>
```

Note that if the client had to do a re-fetch of the "index" document after the initial notification, it would be easy to skip some or all of these patches depending on the received resource version by HTTP, that is, when the HTTP ETag matches with some of these ETags in the chain of patches. If not, the received HTTP version must be a newer version which will be indicated in later notification(s) and the sync MAY then be achieved if the notifier is able to still provide the full change history.

And lastly the notifier could (temporarily) fall back to the "no-patching" mode:

```
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:d="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>
```

This fall back allows to keep the dialog alive, for example when there are really (too) many rapidly changing updates happening. So at any time, the notifier may fall back to this simplest mode for some (or all) of the subscribed documents.

A.5. An XCAP Component Subscription

The user Joe does an initial subscription for the "id" affribute 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>
```

In addition to the 200 OK response, the first NOTIFY looks like since there's 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 format doesn't indicate reasons for non-existing resources.

Afterwards Joe's XCAP client updates the whole document root element including the attribute "id" (not a typical XCAP operation nor 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 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"
  >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 XCAP 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 gets removed e.g. by a HTTP DELETE request. The component selections indicate only the existence of attributes or elements.

A.6. A Conditional Subscription

The last example is about a conditional subscription with which the regeneration of a full refresh can be avoided when there are no changes in resources. The user Joe does 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>
```

In addition to the 200 OK response, the first NOTIFY looks like (since there are now two documents at the repository):

```
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 responds with the SIP-ETag "xggfefe54". So a subscription refresh where the "diff-processing" mode is changed (or is requested to change), looks like:

```
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>
```

So the notifier evaluates this request and if it finds a match to the previous stored state it responds with 204 (No Notification). If there are no reportable changes as per [I-D.ietf-sip-subnot-etags] the whole NOTIFY request generation is being suppressed. When the notifier is capable of aggregating several modifications, this re-subscription effectively 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.

Author's Address

Jari Urpalainen
Nokia
Itamerenkatu 11-13
Helsinki 00180
Finland

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

Full Copyright Statement

Intellectual Property