SIMPLE
Internet-Draft
Expires: April 23, 2004

H. Khartabil
M. Lonnfors
J. Costa-Requena
E. Leppanen
Nokia

October 24, 2003

Functional Description of Event Notification Filtering
draft-khartabil-simple-filter-funct-01

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on April 23, 2004.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

The SIP event notification framework describes the usage of the Session Initiation Protocol (SIP) for subscriptions and notifications of changes to a state of a resource. The document does not describe a mechanism of how filtering of event notification information can be achieved.

This document describes the operations a subscriber performs in order to define filtering rules, how to handle responses to subscriptions carrying filtering rules, and how to handle notifications with filtering rules applied to them. The document also describes how the

notifier behaves when receiving such filtering rules and how a notification is constructed.

The format of the filter is outside the scope of this document.

Table of Contents

1. Conventions

In this document, the key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant implementations.

2. Introduction

SIP event notification is described in [5]. It defines a general framework for sending subscriptions and receiving notifications in SIP based systems. It introduces the concept of event packages, which are concrete applications of the general event framework to a specific usage of events.

Filtering is a mechanism for controlling the content of event notifications. Additionally, the subscriber may specify the rules for when a notification should be sent to it. Requirements for such mechanisms are defined in [10] and [11].

The filtering mechanism is expected to be particularly valuable to users of mobile wireless access devices. The characteristics of the devices typically include high latency, low bandwidth, low data processing capabilities, small display, and limited battery power. Such devices can benefit from the ability to filter the amount of information generated at the source of the event notification.

It is stated in [5] that the notifier may send a NOTIFY at any time, but typically it is sent when the state of the resource changes. It also states that the notifications would contain the complete and current state of the resource authorized for a certain subscriber to see. The format of such resource state information is package specific. In this memo, we assume that the NOTIFY for any package contains an XML document.

There is a separate approach for the rate limiting of SIP event notifications, namely the throttling mechanism [12]. This throttling mechanism allows the subscriber to specify a maximum rate at which event notifications are generated by a single notifier. The solution for such requirements is out of scope for this document.

This document presents a mechanism for filtering whereby a subscriber describes its preference of when notifications are to be sent to it and what they are to contain. It also describes how the notifier functions when generating notifications by taking into account filters and default functionality of the package/service.

The XML format for defining the filter criteria is described in [9].

3. Client Operation

3.1 Transport Mechanism

Transportation of the filter criteria to the server is achieved by inserting the XML document, as defined in [9], in the body of the SUBSCRIBE request. Alternatively, the XML document can be uploaded to the server using means outside the scope of this document.

3.2 SUBSCRIBE Bodies

SIP entities compliant with this specification MUST support the content type 'application/simple-filter+xml'.

3.3 Subscriber Generating SUBSCRIBE Requests

This section presents additional functionality required from the subscriber when filters are used in the bodies of the SUBSCRIBE requests. Normal operations of services, e.g., as defined in [4], [8] and [6] are otherwise followed.

As defined in [5], the SUBSCRIBE message MAY contain a body. This body would serve the purpose of carrying filtering information. Honouring those filters is at the discretion of the notifier and might depend on local policies.

No content in the body of a SUBSCRIBE indicates to the notifier that no filter is being requested so that the notifier is instructed to send all the NOTIFY requests using the notifier's own or service specific policy. Note that e.g. in the list case [6] the filter criteria might have been uploaded to the server beforehand (by means outside the scope of this document).

If the body of the SUBSCRIBE includes the filter criteria, the body MUST be of the MIME-Type 'application/simple-filter+xml'.

3.3.1 Structure of Filter Criteria

Multiple filter criteria MAY be included in one SUBSCRIBE. This is achieved by including multiple <filter> elements in the filter criteria [9]. Each <filter> element may include a URI attribute.

A SUBSCRIBE request destined to a list URI [6] MAY include multiple criteria specific to individual resources. This is achieved by including multiple <filter> elements with different URIs of resources in each of those elements. This resource specific filter criteria overrides any list specific filter criteria, if any. The list specific filter criteria may or may not include a URI.

3.3.2 Request-URI vs. Filter Criteria URI

The URI in the filter criteria defines the target resource, e.g. in the Presence service case; it is the presentity's presence information to which the filter criteria is applied. The subscriber MAY choose to leave the URI in the filter criteria undefined. If the URI is not defined within the filter criteria, the filter criteria apply to the resource identified in the Request-URI. Similarly, the subscriber MAY define a filter criteria URI. If the Request-URI is a list URI [6], the filter criteria URI MUST be the list URI, a sub-list URI or resource who's URI is one of the URIs that result from a lookup, by an RLS, on the Request-URI. If not, the filter criteria may be ignored or may be rejected.

3.3.3 Changing Filter Criteria within a Dialog

The client MAY reset or change the filter criteria by re-issuing a new SUBSCRIBE request within the existing dialog. A SUBSCRIBE within the exiting dialog that does not contain a filter criteria is assumed to prolong existing filter criteria. This means that filter criteria are persistent and are only explicitly removed.

A client requiring removal of a filter criteria may do so by using the 'removed="true"' attribute as defined in [9].

In the case the URI in the filter criteria is that of a list, a client may override the existing filter criteria with a filter criteria for an individual resource, that is part of the list subscribed to earlier, by issuing a new SUBSCRIBE within the existing dialog and including a filter criteria specific for that individual resource. The new filter criteria need not include the original filter criteria since a filter is only removed in the manner indicated above.

3.3.4 Subscriber Interpreting SIP responses

The SUBSCRIBE request will be confirmed with a final response. A 200-class responses indicate that the subscription has been accepted, and that a NOTIFY will be sent immediately. A "200" response indicates that the subscription has been accepted and the filter criteria is accepted. A "202" response merely indicates that the subscription has been understood, the content type has been accepted, and that authorization may or may not have been granted. A "202" response also indicates that the filter has not been accepted yet. The acceptance of the filter criteria MAY arrive in a subsequent NOTIFY.

A non-200 class final responses indicate that no subscription or

dialog has been created, and no subsequent NOTIFY message will be sent. All non-200 class final responses have the same meanings and handling as described in [3] and [5].

Specifically, a "415" response indicates that the MIME type 'application/simple-filter+xml' is not understood by the notifier. A "488" response indicates that the content type (filter) is understood but some aspects of it were either not understood or not accepted.

3.4 Subscriber Processing of NOTIFY Requests

If the 2xx response was returned for the SUBSCRIBE, the NOTIFY that follows MAY contain a body that describes the present state of the resource after the filter criteria have been applied.

If the NOTIFY indicates that a subscription has been terminated [5], the subscription is assumed to be terminated. Behaviour in such events is also described in [5].

If the subscription is indicated as active, NOTIFY requests are handled as described in package specific documents and [5].

4. Resource List Server Behaviour

The Resource List Server is defined in [6]. This section describes how such entity behaves in the presence of a filter criteria in a subscription to a list.

4.1 Request-URI vs. Filter Criteria URI

If the URI is not defined within the filter criteria, the filter criteria apply to the resource identified in the Request-URI.

If no URI is indicated by the filter criteria, the filter criteria apply to all the notifications that the RLS issues. If the URI indicated by the filter criteria is a list URI, the filter criteria apply to all the notifications that the RLS issues.

If the URI indicated by the filter criteria is for one resource who's URI is one of the URIs that result from a lookup, by the RLS, on the Request-URI, the filter criteria for that particular resource are extracted and propagated in the SUBSCRIBE request sent to that resource. (It is possible to have more than one filter criteria in a SUBSCRIBE body, and therefore a filter criteria specific to a resource must be extracted and only that is propagated. For example, if the Request-URI in a SUBSCRIBE has the value "sip:mybuddies@mydomain.com" where "bob@otherdomain.com" is a resource belonging to that list, and the URI in a filter criteria is

"sip:bob@otherdomain.com", the filter criteria specific for Bob are extracted and placed in the body of the SUBSCRIBE sent to "bob@otherdomain.com").

If the URI indicated by the filter criteria is for one resource who's URI is NOT one of the URIs that result from a lookup, by the RLS, on the Request-URI, the filter criteria are propagated to all the fanned out subscriptions. This is to accommodate the scenario where the subscriber knows that there are sub-lists in the list that the original subscription was sent to, and the subscriber wishes to set a filter criteria for a resource in that sub-list.

5. Server Operation

5.1 NOTIFY Bodies

SIP entities compliant with this specification MUST support content-type 'application/simple-filter+xml'.

5.2 Notifier Processing SUBSCRIBE Requests

This section presents addition functionality required from the notifier when filters are used in the bodies of the SUBSCRIBE requests. Normal package specific functionality are otherwise followed.

The notifier will examine the content type header and will return a 415 response if it does not understand the content type 'application/simple-filter+xml'.

A 200-class responses indicate that the subscription has been accepted, and the NOTIFY will be sent immediately. A "200" response indicates that the subscription has been accepted, the user is authorized and the filter is accepted. A "202" response merely indicates that the subscription has been understood, but the authorization may or may not have been granted. A "202" response also indicates that the filter criteria have not been accepted yet. The acceptance of the filter criteria MAY arrive in a subsequent NOTIFY.

Procedures described in section Section 5.4 are followed if an error is encountered.

5.2.1 Request-URI vs. Filter Criteria URI

The subscriber may have chosen to leave the URI in the filter criteria undefined. If the URI is not defined within the filter criteria, the filter criteria apply to the resource identified in the Request-URI.

Similarly, the subscriber may have chosen to include a URI in the filter criteria. In this case, the filter criteria apply to all notifications send for this subscription. If the Request-URI and the URI in the filter criteria mismatch, the filter criteria may be ignored or may be rejected.

5.3 Notifier Generating NOTIFY Requests

Upon receiving the SUBSCRIBE with the filter criteria, the notifier SHOULD retain the filter criteria as long as the subscription persists. The filter MAY be incorporated within an existing subscription (in an active dialog) by sending a re-SUBSCRIBE that includes the filter criteria in the body.

If the response sent to the SUBSCRIBE was the 202 and the 202 was chosen because the filter could not be accepted that time, the NOTIFY MAY be used to terminate the subscription if the filter was found unacceptable.

As described in [5], the NOTIFY message MAY contain a body that describes the state of the resource. This body is in one of the formats listed in the Accept header of the SUBSCRIBE, or the package specific default if the Accept header is omitted.

5.3.1 Generation of NOTIFY Contents

If the NOTIFY being sent is the immediate one sent after a 2xx response to the original SUBSCRIBE, its contents MUST be populated according to the filter criteria. If applying the filter criteria results in not sending a NOTIFY, the NOTIFY must be sent with empty contents. A notifier may also choose not to obey the filter criteria in the first NOTIFY and instead send the configured value authorised for that subscriber set by the notifier using means outside the scope of this document).

The input to the content filter is a package specific XML document, e.g. [2] and [7] derived according to the package specific specifications, ([4] and [8]).

The content is filtered according to the expressions in the <what> element of the filter criteria. The expression indicates the delivered XML elements and/or attributes. Prefixes of the namespaces of the items of the XML document to be filtered must be expanded before applying the filter to the items.

The expression directly states the XML elements and attributes to be delivered in the NOTIFY, along with their values. In addition to the selected contents also the namespaces of all the selected items are included in the NOTIFY. The XML elements and/or attributes indicated by the expression in the <what> element must be items that the

subscriber is authorised to see. If not, the notifier policy dictates the behaviour of the notifier (notifier can either ignore the filter, parts of the filter, or reject the filter completely). Implementers need to carefully consider such an implementation decision; the subscriber may not be aware of the authorised contents and therefore most likely will include in the filter a criteria requesting unauthorised contents. It is therefore RECOMMENDED that notifiers just ignores the parts of the filter criteria where it is requesting unauthorised info. I.e. The filter criteria in the <filter> element where the unauthorised contents are requested is ignored. If polite blocking is used by the notifier, the notifier may choose to ignore the filter, by choosing to deliver notifications containing bogus information in the unauthorised elements or attributes.

The resultant XML document MUST be well formed and valid according to the XML schema. This means that all mandatory elements and attributes along with their values MUST be included in the XML document regardless of the expression. In other words, if the results of applying a filter criteria on an XML document results in a non-valid XML document, the notifier MUST add elements and attributes, along with their values, from the original XML document into the newly formulated one in order for it to be a valid one. Thissituation can occur if the notfier did not closely examine the filter before accepting it

5.3.2 Handling of Notification Triggering Rules

There can be several <trigger> elements inside one <filter> element. If the criteria for any of the <trigger> elements are satisfied, a NOTIFY SHOULD be generated.

The items (XML elements and/or attributes) indicated by the expression in the <changed> element, <added> element or <removed> element must be items that the subscriber is authorised to access. If not, the notifier policy dictates the behaviour of the notifier (notifier can either ignore the filter, parts of the filter, or reject the filter completely).

5.4 Handling Abnormal Cases

In case of an invalid filter criteria definition the XML document of the filter criteria is not aligned with the XML schema of the filter criteria [9]. The notifier rejects the SUBSCRIBE request with a "488" response. A warning header in the response may give better indication why the filters were not accepted. If the subscription was accepted with a "202" response but the invalid filter criteria was discovered after that, a NOTIFY with a subscription-state of value 'terminated' is sent. An event-reason-value "badfilter", introduced

here, of subexp-params [5] MAY be included.

In case of an erroneous expression in the filter definition the notifier either ignores the filter definition or terminates the subscription.

If a <what> or <trigger> element is empty, the notifier proceeds as if the element did not exist.

6. IANA Considerations

A new event-reason-value "badfilter" is defined to represent the event where the filter criteria is not well formed and/or not accepted.

OPEN ISSUE: IANA registration template must be added later. [13]

7. Examples

The following chapters include filtering examples for Presence and Watcher Information. The format of filter criteria is according to [9].

7.1 Presence Specific Examples

This chapter describes three use cases where the presence information filtering solution is utilised [4]. In the first use case the watcher is interested in getting messaging specific information of a certain presentity. In the second use case the watcher is interested in getting information about the communication means and contact addresses the presentity is currently available for communication on. The third case shows how a presentity can request triggers to receive notifications

Below is the Presentity's presence information in PIDF [2]. It includes two tuples: one for the instant messaging and another for the voice related information.

```
<?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
    xmlns:rpids="urn:example-comietf:params:ns:sip-rpids"
      entity="sip:presentity@domain.com">
         <tuple id="432sd">
            <status>
               <basic>closed</basic>
            </status>
            <rpid:class>im</rpid:class>
            <contact>im:presentity@domain.com</contact>
         </tuple>
         <tuple id="thr76jk">
            <status>
               <basic>open</basic>
            </status>
            <rpid:class>voice</rpid:class>
            <contact>tel:2224055555@domain.com</contact>
         </tuple>
      </presence>
```

7.1.1 Subscriber Requests Messaging Related Information

The subscriber initiates a subscription to the presentity's messaging (MMS, IM and SMS) related presence information. The subscription includes the content limiting filter.

The filtered content is indicated with an expression. This expression selects all the upper, lower and parallel level PIDF XML elements that match the criteria (this means the tuple and its root element). That criteria are: <label> elements that have values beginning with "MMS", "SMS" or "IM".

In this case, the notification includes the contents of the tuple that has the value "IM" in its <label> element.

SUBSCRIBE request from the subscriber including filter:

```
SUBSCRIBE sip:presentity@domain.com
Via: SIP/2.0/TCP 10.0.0.1:5060;branch=xjfdsjfk
To: <sip:presentity@domain.com>
From: <sip:watcher@domain.com>;tag:12341111
Call-ID: 121212@10.0.0.1
Cseq: 1 SUBSCRIBE
Expires: 3600
Event: Presence
Contact: <sip:watcher@client.domain.com>
Content-Type: application/simple-filter+xml
```

```
   Content-Length: ...

   <?xml version="1.0" encoding="UTF-8"?>
   <filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
         <filter id="mess" uri="sip:presentity@domain.com">
              <what>
                        <condition base-element="//tuple" type="xml-
element">rpid:class="IM" or rpid:class="SMS" or rpid:class="MMS"</condition>
                        <include level="self" type="xml-element">//tuple</
include>
                        <include level="descendant">//tuple</include>
                        <include level="ancestor">//tuple</include>
            </what>
          </filter>
   </filter-set>


   Notification to the subscriber:

   NOTIFY sip:watcher@client.domain.com SIP/2.0
   Via: SIP/2.0/TCP presence.domain.com:5060;branch=xjfder
   To: <sip:watcher@domain.com>;tag:12341111
   From: <sip:presentity@domain.com>;tag:232321
   Call-ID: 121212@10.0.0.1
   Cseq: 1 NOTIFY
   Event: Presence
   Subscription-State: active; expires=3599
   Contact: sip:presentity@server.domain.com
   Content-Type: application/cpim-pidf+xml
   Content-Length: ...

   <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
      xmlns:rpids="urn:example-com:ietf:params:ns:sip-rpids"
      entity="sip:presentity@domain.com">
         <tuple id="432sd">
            <status>
               <basic>closed</basic>
            </status>
            <rpid:class>im</rpid:class>
            <contact>im:presentity@domain.com</contact>
         </tuple>
      </presence>
```

7.1.2 Subscriber Fetches Information about "Open" Communication Means

The subscriber makes a subscription to the presentity's available communication means. The subscription includes the content limiting

filter.

The filtered content is indicated with an expression. This expression selects all the upper, lower and parallel level PIDF XML elements that match the criteria. That criteria are: the <basic> element's value is "Open". There is also need to indicate that only the tuples which have contact address information are selected.

In this case the notification returns the contents of the tuple that has both the value "open" inside the <status> element and the existing <contact> elements.

SUBSCRIBE request from the subscriber including filter:

```
SUBSCRIBE sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/TCP 10.0.0.1:5060;branch=xjfdsjfk
To: <sip:presentity@domain.com>
From: <sip:watcher@domain.com>;tag:12341111
Call-ID: 121212@10.0.0.1
Cseq: 1 SUBSCRIBE
Expires: 3600
Event: Presence
Contact: <sip:watcher@client.domain.com>
Content-Type: application/simple-filter+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
   <filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
      <filter id="open-mean" uri="sip:presentity@domain.com">
            <what>
                     <condition base-element="//tuple" type="xml-
element">basic="open"</condition>
                     <include level="self" type="xml-element">//tuple</
include>
                     <include level="descendant">//tuple</include>
                     <include level="ancestor">//tuple</include>
          </what>
           </filter>
   </filter-set>
```

Notification to the subscriber:

```
NOTIFY sip:watcher@client.domain.com SIP/2.0
Via: SIP/2.0/TCP presence.domain.com:5060;branch=xjfder
To: <sip:watcher@domain.com>;tag:12341111
From: <sip:presentity@domain.com>;tag:232321
Call-ID: 121212@10.0.0.1
Cseq: 1 NOTIFY
Event: Presence
Subscription-State: active; expires=3599
Contact: sip:presentity@server.domain.com
Content-Type: application/cpim-pidf+xml
```

```
   Content-Length: ...
```

```
   <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
      xmlns:rpids="urn:example-com:ietf:params:ns:sip-rpids"
      entity="sip:presentity@domain.com">
         <tuple id="thr76jk">
            <status>
               <basic>open</basic>
            </status>
               <rpid:class>voice</rpid:class>
               <contact>tel:2224055555@domain.com</contact>
         </tuple>
      </presence>
```

7.1.3 Subscriber Requests Notifications when Presentity's Status Changes

The subscriber subscribes to the presentity, specifying in the filter that it wants notifications only when the <basic>element has changed to value 'open'

SUBSCRIBE request from the subscriber including filter:

```
SUBSCRIBE sip:presentity@domain.com
Via: SIP/2.0/TCP 10.0.0.1:5060;branch=xjfdsjfk
To: <sip:presentity@domain.com>
From: <sip:watcher@domain.com>;tag:12341111
Call-ID: 121212@10.0.0.1
Cseq: 1 SUBSCRIBE
Expires: 3600
Event: Presence
Contact: <sip:watcher@client.domain.com>
Content-Type: application/simple-filter+xml
Content-Length: ...
```

```
<?xml version="1.0" encoding="UTF-8"?>
   <filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
     <filter id = "open-mean" uri="sip:presentity@domain.com">
       <trigger>
            <changed changed-from="closed" changed-to="open">//presence/
tuple/status/basic</changed>
        </trigger>
     </filter>
   </filter-set>
```

Assuming a 2nd PIDF document is created with both tuples having status of closed:

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

```
      <presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
```

```
      xmlns:rpids="urn:example-com:ietf:params:ns:sip-rpids"
      entity="sip:presentity@domain.com">
         <tuple id="432sd">
            <status>
            <basic>closed</basic>
            </status>
               <rpid:class>im</rpid:class>
               <contact>im:presentity@domain.com</contact>
         </tuple>
         <tuple id="thr76jk">
            <status>
               <basic>closed</basic>
            </status>
            <rpid:class>voice</rpid:class>
            <contact>tel:2224055555@domain.com</contact>
         </tuple>
      </presence>
```

A NOTIFY is not sent to the subscriber in this case.

Now, a 3rd PIDF document is created when IM status changes to OPEN:

```
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
   xmlns:rpids="urn:example-com:ietf:params:ns:sip-rpids"
   entity="sip:presentity@domain.com">
      <tuple id="432sd">
         <status>
            <basic>open</basic>
         </status>
         <rpid:class>im</rpid:class>
         <contact>im:presentity@domain.com</contact>
      </tuple>
      <tuple id="thr76jk">
         <status>
            <basic>closed</basic>
         </status>
         <rpid:class>voice</rpid:class>
         <contact>tel:2224055555@domain.com</contact>
      </tuple>
   </presence>
```

Notification to the subscriber is sent in this case:

```
NOTIFY sip:watcher@client.domain.com SIP/2.0
Via: SIP/2.0/TCP presence.domain.com:5060;branch=xjfder
To: <sip:watcher@domain.com>;tag:12341111
```

```
From: <sip:presentity@domain.com>;tag:232321
Call-ID: 121212@10.0.0.1
Cseq: 1 NOTIFY
Event: Presence
Subscription-State: active; expires=3599
Contact: sip:presentity@server.domain.com
Content-Type: application/cpim-pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
   xmlns:rpids="urn:example-com:ietf:params:ns:sip-rpids"
   entity="sip:presentity@domain.com">
      <tuple id="432sd">
         <status>
            <basic>closed</basic>
         </status>
         <rpid:class>im</rpid:class>
         <contact>im:presentity@domain.com</contact>
      </tuple>
      <tuple id="thr76jk">
         <status>
            <basic>open</basic>
         </status>
         <rpid:class>voice</rpid:class>
         <contact>tel:2224055555@domain.com</contact>
      </tuple>
   </presence>
```

7.2 Watcher Information Specific Examples

The examples in this section use the presence event package with the winfo template-package [8].

Watcher information to a Presentity:

```
<?xml version="1.0"?>
  <watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo"
version="0" state="full">
  <watcher-list resource="sip:presentity@domain.com"
               package="presence">
      <watcher status="active"
         id="sr8fdsj"
         duration-subscribed="509"
         expiration="20"
         event="approved">sip:watcherA@example.com"</watcher>
      <watcher status="pending"
         id="sr8fdsj"
         duration-subscribed="501"
         expiration="100"
         event="subscribe">sip:watcherB@example.com"</watcher>
      <watcher status="terminated"
         id="sr8fdsj"
         duration-subscribed="500"
         expiration="0"
         event="rejected">sip:watcherC@example.com"</watcher>
      <watcher status="active"
         id="sr8fdsj"
         duration-subscribed="20"
         expiration="30"
         event="approved">sip:watcherD@domain.com"</watcher>
  </watcher-list>
  </watcherinfo>
```

7.2.1 Watcher Subscriber Makes Subscription to Gets All the Information about Active Watchers

SUBSCRIBE request from the presentity including the filter:

```
SUBSCRIBE sip:presentity@domain.com
Via: SIP/2.0/TCP 10.0.0.1:5060;branch=xjfdsjfk
To: <sip:presentity@domain.com>
From: <sip:presentity@domain.com>;tag:12341111
Call-ID: 121212@10.0.0.1
Cseq: 1 SUBSCRIBE
Expires: 3600
Event: Presence.winfo
Contact: sip:presentity@client.domain.com
Content-Type: application/simple-filter+xml
Content-Length: ...
```

```
      <?xml version="1.0" encoding="UTF-8"?>
        <filter-set xmlns="urn:ietf:params:xml:ns:simple-filter"
pkgdef="winfo">
            <filter id = "active" uri="sip:presentity@domain.com">
              <what>
                 <condition base-element="//watcher">@package="presence" and
@status="active"</condition>
                   <include>//watcher</include>
                   <include level="ancestor">//watcher</include>
             </what>
           </filter>
        </filter-set>
```

Notification to the subscriber:

```
NOTIFY sip:presentity@client.domain.com SIP/2.0
Via: SIP/2.0/TCP presence.domain.com:5060;branch=xjfder
To: sip:presentity@domain.com;tag:12341111
From: sip:presentity@domain.com;tag:232321
Call-ID: 121212@10.0.0.1
Cseq: 1 NOTIFY
Contact: sip:presentity@server.domain.com
Event: Presence.winfo

Content-Type: application/watcherinfo+xml
Content-Length: ...

<?xml version="1.0"?>
  <watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo"
version="0" state="full">
  <watcher-list resource="sip:presentity@domain.com"
                package="presence">
      <watcher status="active"
         id="sr8fdsj"
         duration-subscribed="509"
         expiration="20"
         event="approved">sip:watcherA@example.com"</watcher>
      <watcher status="active"
         id="sr8fdsj"
         duration-subscribed="20"
         expiration="30"
         event="approved">sip:watcherD@domain.com"</watcher>
  </watcher-list>
  </watcherinfo>
```

7.2.2 Watcher Subscriber Requests Information of Watchers with Specific

SUBSCRIBE request from the presentity including the filter:

```
SUBSCRIBE sip:presentity@domain.com
Via: SIP/2.0/TCP 10.0.0.1:5060;branch=xjfdsjfk
To: <sip:presentity@domain.com>;tag:12341111
From: <sip:presentity@domain.com>
Call-ID: 121212@10.0.0.1
Cseq: 1 SUBSCRIBE
Expires: 0
Event: Presence.winfo
Contact: <sip:presentity@client.domain.com>
Content-Type: application/simple-filter+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
  <ev-filter-set xmlns="urn:ietf:params:xml:ns:simple-filter"
pkgdef="winfo">
      <ev-filter id = "history" uri="sip:presentity@domain.com">
        <what>
           <condition base-element="//watcher">@package="presence" and
@duration-subscribed>500</condition>
             <include>//watcher</include>
             <include level="ancestor">//watcher</include>
        </what>
      </ev-filter>
  </ev-filter-set>
```

Notification to the subscriber:

```
NOTIFY sip:presentity@client.domain.com SIP/2.0
Via: SIP/2.0/TCP presence.domain.com:5060;branch=xjfder
To: sip:presentity@domain.com;tag:12341111
From: sip:presentity@domain.com;tag:232321
Call-ID: 121212@10.0.0.1
Cseq: 1 NOTIFY
Contact: sip:presentity@server.domain.com
Event: Presence.winfo

Content-Type: application/watcherinfo+xml
Content-Length: ...

<?xml version="1.0"?>
  <watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo"
version="0" state="full">
  <watcher-list resource="sip:presentity@domain.com"
                package="presence">
      <watcher status="active"
         id="sr8fdsj"
```

```
                duration-subscribed="509"
                expiration="20"
```

```
            event="approved">sip:watcherA@example.com"</watcher>
         <watcher status="pending"
            id="sr8fdsj"
            duration-subscribed="501"
            expiration="100"
            event="subscribe">sip:watcherB@example.com"</watcher>
      </watcher-list>
      </watcherinfo>
```

7.2.3 Watcher Subscriber Requests Specific Watcher Info On Specific Triggers

This filter selects watcher information notifications [7] to be sent when the pending subscription status has changed from 'pending' to 'terminated'. In the notification, only the watchers that have a status of 'terminated' and an event of 'rejected' are included.

SUBSCRIBE request from the Watcher Subscriber including the filter:

```
   SUBSCRIBE sip:presentity@domain.com
   Via: SIP/2.0/TCP 10.0.0.1:5060;branch=xjfdsjfk
   To: <sip:presentity@domain.com>;tag:12341111
   From: <sip:presentity@domain.com>
   Call-ID: 121212@10.0.0.1
   Cseq: 1 SUBSCRIBE
   Expires: 0
   Event: Presence.winfo
   Contact: <sip:presentity@client.domain.com>
   Content-Type: application/simple-filter+xml
   Content-Length: ...
```

```
<?xml version="1.0" encoding="UTF-8"?>
  <ev-filter-set xmlns="urn:ietf:params:xml:ns:simple-winfo-filter"
pkgdef="winfo">
      <ev-filter id = "filter-X" uri="sip:presentity@domain.com">
          <what>
              <condition base-element="//watcher">@package="presence" and
@status="active" and event="rejected"</condition>
              <include>//watcher</include>
              <include level="ancestor">//watcher</include>
       </what>
          <trigger>
                <changed changed-from="pending" changed-to="terminated">//
@status</changed>
            </trigger>
       </ev-filter>
  </ev-filter-set>
```

A 2nd Winfo document is created due to some change:

```
    <?xml version="1.0"?>
        <watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo"
      version="0" state="full">
        <watcher-list resource="sip:presentity@domain.com"
                     package="presence">
            <watcher status="active"
               id="sr8fdsj"
               duration-subscribed="509"
               expiration="20"
               event="approved">sip:watcherA@example.com</watcher>
            <watcher status="terminated"
               id="sr8fdsj"
               duration-subscribed="501"
               expiration="100"
               event="rejected">sip:watcherB@example.com</watcher>
            <watcher status="terminated"
               id="sr8fdsj"
               duration-subscribed="500"
               expiration="0"
               event="rejected">sip:watcherC@example.com</watcher>
            <watcher status="active"
               id="sr8fdsj"
               duration-subscribed="20"
               expiration="30"
               event="approved">sip:watcherD@domain.com</watcher>
       </watcher-list>
        </watcherinfo>
```

Notification to the subscriber, taking into account the <trigger> and <what> elements:

```
      NOTIFY sip:presentity@client.domain.com SIP/2.0
      Via: SIP/2.0/TCP presence.domain.com:5060;branch=xjfder
      To: sip:presentity@domain.com;tag:12341111
      From: sip:presentity@domain.com;tag:232321
      Call-ID: 121212@10.0.0.1
      Cseq: 1 NOTIFY
      Contact: sip:presentity@server.domain.com
      Event: Presence.winfo

      Content-Type: application/watcherinfo+xml
      Content-Length: ...

    <?xml version="1.0"?>
        <watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo"
      version="0" state="full">
        <watcher-list resource="sip:presentity@domain.com"
                     package="presence">
```

```
            <watcher status="terminated"
```

```
            id="sr8fdsj"
            duration-subscribed="501"
            expiration="100"
            event="rejected">sip:watcherB@example.com</watcher>
         <watcher status="terminated"
            id="sr8fdsj"
            duration-subscribed="500"
            expiration="0"
            event="rejected">sip:watcherC@example.com</watcher>
      </watcher-list>
      </watcherinfo>
```

8. Security Considerations

The presence of filters in the body in a SIP message has a significant effect on the ways in which the request is handled at a server. As a result, it is especially important that messages containing this extension be authenticated and authorized.

Processing of requests and looking up filter criteria requires set operations and searches, which can require some amount of computation. This enables a DoS attack whereby a user can send requests with substantial numbers messages with large contents, in the hopes of overloading the server. To counter this, the server SHOULD only allow filters with no more than about 20 occurances of the <what>, <changed>, <added> and <removed> elements.

Requests can reveal sensitive information about a UA's capabilities. If this information is sensitive, it SHOULD be encrypted using SIP S/MIME capabilities.

All package specific security measures MUST be followed.

9. Acknowledgements

The authors would like to thank Tim Moran, Sreenivas Addagatla and Juha Kalliokulju for their valuable input.

References

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

[2] Sugano, H., "CPIM Presence Information Data Format", draft-ietf-impp-cpim-pidf-07.txt, December 2002.

[3] Rosenberg, J., "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[4] Rosenberg, J., "Session Initiation Protocol (SIP) Extensions for Presence", draft-ietf-simple-presence-10.txt, January 2003.

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

[6] Roach, A., "A Session Initiation Protocol (SIP) Event Notification Extension for Collections", draft-ietf-simple-event-list-01.txt, March 2003.

[7] Rosenberg, J., "An Extensible Markup Language (XML) Based Format for Watcher Information", draft-ietf-simple-winfo-format-04.txt, January 2003.

[8] Rosenberg, J., "A Watcher Information Event Template-Package for SIP", draft-ietf-simple-winfo-package-05.txt, January 2003.

[9] Khartabil, H., "An Extensible Markup Language (XML) Based Format for Event Notification Filtering", draft-khartabil-simple-filter-format-01.txt, October 2003.

[10] Moran, T., "Requirements for Presence Specific Event Notification Filtering", draft-ietf-simple-pres-filter-reqs-01.txt, June 2003.

[11] Kiss, K., "Requirements for Filtering of Watcher Information", draft-ietf-simple-winfo-filter-reqs-00.txt, April 2003.

[12] Niemi, A., "Requirements for Limiting the Rate of Event Notifications", draft-niemi-sipping-event-throttle-reqs-01.txt, February 2003.

[13] Mealling, M., "The IETF XML Registry", draft-mealling-iana-xmlns-registry-04.txt, June 2002.

[14] Bray, T., "Exensible Markup Language (XML) 1.0 (Second Edition)", W3C CR CR-xml11-20011006, October 2000.

Authors' Addresses

Hisham Khartabil
Nokia
P.O. Box 321
Helsinki
Finland

Phone: +358 7180 76161
EMail: hisham.khartabil@nokia.com

Mikko Lonnfors
Nokia
Itamerenkatu 00180
Helsinki
Finland

Phone: + 358 50 4836402
EMail: mikko.lonnfors@nokia.com

Jose Costa-Requena
Nokia
P.O. Box 321
FIN-00045 NOKIA GROUP
FINLAND

Phone: +358 71800 8000
EMail: jose.costa-requena@nokia.com

Eva Leppanen
Nokia
P.O BOX 785
Tampere
Finland

Phone: +358 7180 77066
EMail: eva-maria.leppanen@nokia.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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