SIP
Internet Draft
Expires: 23 August 2003

Hisham Khartabil
Eva Leppanen

Nokia

24 February 2003

Event Notification Filtering for Watcher Info

draft-khartabil-simple-winfo-filter-00.txt

STATUS OF THIS MEMO

Copyright Notice

Abstract

Watcher information template-package for the SIP event framework refers to the set of users subscribed to a particular resource within a particular event package. The Watcher Information I-D does not describe a mechanism of how filtering of watcher information can be achieved.

This document defines a watcher info filter. It provides the mechanism whereby a watcherinfo subscriber can specify the watcher info event filtering rules, using XML, for the notifier and how that

notifier is to behave when receiving such filter.

Table of Contents

1.0 Terminology....3
2.0 Introduction....3
2.1 Motivation For Using XML....4
3.0 Watcherinfo Package Specific Filters....4
3.1 Package ID....4
3.3 Transport mechanism....4
3.4 Structure of XML-Encoded winfo Filter Criteria....4
3.4.1 The <ev-filter-set> Root Element....5
3.4.2 The <ev-filter> Element....5
3.4.3 The <what> Element....5
5.0 Client and server operation....6
5.1 Client Operation....6
5.1.1 SUBSCRIBE Bodies....6
5.1.2 Subscriber Generating SUBSCRIBE requests....6
5.1.2.1 To-Header vs. Filter URI....6
5.1.2.2 Changing The Filter Criteria Within A Dialog....7
5.1.2.3 Subscriber Interpreting SIP responses....7
5.1.3 Subscriber processing NOTIFY requests....7
5.2 Server Operation....8
5.2.1 NOTIFY Bodies....8
5.2.2 Notifier processing SUBSCRIBE Requests....8
5.2.3 Notifier Generation of NOTIFY requests....8
5.2.3.1 Generating NOTIFY Contents....9
5.2.4 Handling of abnormal cases....9
6.0 IANA Considerations....10
7.0 Examples....10
7.1 Presentity makes subscription to get all the information about active watchers....11
7.2 Presentity makes subscription to get information about defined, new and unauthorized watchers....12
7.3 Presentity requests history information of watchers....13
8.0 XML Schema....14
9.0 Security Considerations....15
10.0 Acknowledgements....16
11.0 References....16
AuthorsÆ Addresses....17
Full Copyright Statement....17
Acknowledgement....18

1.0 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED" "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1].

The document uses the terms as defined in [2] and [3].

2.0 Introduction

Watcher information template-package for the SIP event framework refers to the set of users subscribed to a particular resource within a particular event package [2].

SIP event notification is described in [4]. 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 [5].

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 [2] that Watcherinfo notifications are generated every time there is any change in the state of the watcher information. It also states that Watcherinfo notifications triggered from a change in watcher state only contain information on the watcher whose state has changed. The format of such information defined in [3].

This document presents a mechanism for the filtering of the watcherinfo information. The document specifically describes how the subscriber is able to both limit the content and indicate more specifically the preferred content of the notifications using XML and Xpath [14] technologies. It also describes how the notifier functions when generating notifications by taking into account filters, authorisation rules and default functionality of the watcherinfo notification service.

The watcherinfo specific filtering solution presented here is compatible with the watcherinfo event template-package [2].

Requirements for the watcherinfo filter criteria can be found [5] and [13].

Events Notification RFC [4] and the SIMPLE working group both identify filtering work as potential work.

2.1 Motivation For Using XML

When trying to select the schema and protocol for defining the filter criteria, it can be found that there are several alternatives available. The best practice is to modularise the problem and work on smaller modules (the protocol and the filter information schema). The selection of the schema for defining the filter criteria requires a readable syntax with enough flexibility to define the semantics.

Among the multiple possibilities (i.e. XML, Perl, TCL, etc), XML provides a generic framework where a schema for the filter logic including a namespace allowing the definition of the filter criteria. Thus, an XML schema provides an extensible mechanism where the logic required for the filtering functionality can be built. XML provides extensibility based on "namespaces" associated with a globally unique URI.

The XML also enables the use of Xpath [14] for referencing items in an XML document (which in the case is watcher information in the format of watcherinfo).
In addition, by using XML the need for reliance on the transport protocol in order to interpret the filter criteria can be eliminated.

3.0 Watcherinfo Package Specific Filters

This section describes the technologies and information needed to provide a filter specification relevant to watcherinfo filtering.

3.1 Package ID

The package ID is æwinfoÆ. This ID is carried in the event-header of the SUBSCRIBE as well as within the XML document carrying filter criteria.

3.3 Transport mechanism

Transportation of the filter criteria to the server is achieved by inserting the filter XML document in the body of a SIP SUBSCRIBE request. Alternatively, the document can be uploaded to the server using a transport specific for data manipulation.

3.4 Structure of XML-Encoded winfo Filter Criteria

Winfo Filter Criteria is an XML document [7] that MUST be well-formed and SHOULD be valid. Winfo Filter Criteria XML documents MUST have the XML declaration and it SHOULD contain an encoding declaration in the XML declaration e.g. "<?XML version='1.0' encoding='UTF-8'?>". This specification makes use of XML namespaces for identifying the XML schema of the Winfo Filter Criteria documents and document fragments.

The namespace identifier for elements defined by this specification is a URN [8]; using the namespace identifier 'ietf' defined by [9] and extended by [10]. This urn is:

urn:ietf:params:xml:ns:simple-winfo-filter+xml.

This namespace declaration indicates the namespace on which the filter criteria are based on.

3.4.1 The <ev-filter-set> Root Element

The root element of the filter criteria is <ev-filter-set>. The <ev-filter-set> MAY have a 'pkgdef' attribute that defines the æwinfoÆ as package for the filter criteria.

The <ev-filter-set> root element MUST contain the namespace mentioned above. This element MUST contain one or more <ev-filter> elements.

3.4.2 The <ev-filter> Element

The <ev-filter> element is used to specify the content of an individual filter.

<ev-filter> MUST have an 'id' attribute. The value of the 'id' attribute MUST be unique within the <ev-filter-set> root element. <ev-filter> MAY have an 'uri' attribute. The value of the 'uri' attribute is the URI of a watcher-list resource.

The <ev-filter> element MUST contain a <what> element.

3.4.3 The <what> Element

The <what> element is used to specify the content to be delivered to the watcherinfo subscriber. It does not specify the exact content but rules that are used to construct that information.

The <what> element MUST contain the 'report' attribute. The value of the 'report' is either 'default'. The 'default' value means that the whole content of selected watcher elements are delivered.

The æstateÆ attribute is optional and indicates whether the document to be delivered to the watcherinfo subscriber contains the ôfullö watcherinfo state, or whether it contains only information on those

watchers which have changed since the previous document (ôpartialö) [2]. The delivered content depends on the XPath rules and the selected state. If this attribute is not present, the default value is a matter of local policy [2].

OPEN ISSUE: the æreportÆ attribute may be later removed if it is asserted that no other value (e.g. ôonly-selectedö) is needed.

3.4.3.1 Defining the delivered content

It must be possible in the filter criteria to be able to select watchers from which watcher information is delivered. The selection is based on any XML elements and attributes of the default watcherinfo format [3].

The preferred watcher information is indicated using the capabilities of XPath [14]. The XPath clause is stated in the value of the <What> element.

4.0 Client and server operation

4.1 Client Operation

4.1.1 SUBSCRIBE Bodies

SIP entities compliant with this specification MUST support content-type æapplication/simple-winfo-filter+xmlÆ.

4.1.2 Subscriber Generating SUBSCRIBE requests

This section presents additional functionalities required from the subscriber when filters are used in SUBSCRIBE requests bodies. Normal operations as defined in [2] are then followed.

As defined in [2], a SUBSCRIBE for winfo MAY contain a body. This body, as defined in this document, would serve the purpose of filtering. Honouring of these filters is at the policy discretion of the notifier.

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 NOTIFY requests using the notifierÆs default watcherinfo subscription filtering policy as described in [2].

If present, the body MUST be of MIME-Type æapplication/simple-winfo-filter+xmlÆ.

4.1.2.1 To-Header vs. Filter URI

The Event package "winfo" in Event-header of the SUBSCRIBE request specifies that the filter rules apply to the watcherinfo of a watcher-list resource whose URI appears in the filter.

If no watcher-list resource URI is indicated by the filter, the filter rules apply to the watcher-list of the resource identified in the To-header.

If the resource URI indicated by the filter is not the same as the one in the To-header, that filterÆs rules do not apply and are ignored. The filter does not affect Watcherinfo documents sent to the watcherinfo subscriber.

4.1.2.2 Changing The 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. The SUBSCRIBE provides the "replace" semantics. This means that all filters are replaced. A SUBSCRIBE within the exiting dialogs that does not contain a filter is assumed to be removing all filter criteria.

The filter MAY be incorporated within an existing subscription (an active dialog) by the subscriber sending a re-SUBSCRIBE that includes the filter in the body.

4.1.2.3 Subscriber Interpreting SIP responses

The SUBSCRIBE request will be confirmed with a final response. 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, the user is authorized and that the filter 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 acceptation of the filter MAY arrive in a subsequent NOTIFY.

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 responses have the same meanings and handling as described in RFC 3261 [11] and RFC 3265[4]. Specifically, a 415 response indicates that the server does not understand the MIME-type æapplication/simple-pres-filter+xmlÆ. A 488 response indicates that the content (filter) is understood but some aspects of it were either not understood or not accepted.

4.1.3 Subscriber processing NOTIFY requests

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

If the NOTIFY indicates that a subscription has been terminated [4], the subscription is assumed to be terminated. Behaviour in such events is also described in [4].

If the subscription is indicated as active, NOTIFY requests are handled as described in [2] and [4].

4.2 Server Operation

4.2.1 NOTIFY Bodies

SIP entities compliant with this specification MUST support content-type æapplication/simple-winfo-filter+xmlÆ.

4.2.2 Notifier processing SUBSCRIBE Requests

This section presents addition functionalities required from the notifier when filters are used in SUBSCRIBE requests bodies. Normal operations as defined in [2] are then followed.

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

If the notifier accepts the content-type, it then starts parsing the body. If the body is understood but some aspects of the contents (filter) were not understood and accepted, then procedures described in section 4.2.4 are followed.

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, the user is authorized and that the filter is accepted. A 202 response merely indicates that the subscription has been understood, and that authorization may or may not have been granted. A 202 response also indicates that the filter has not been accepted yet. The acceptation of the filter MAY arrive in a subsequent NOTIFY.

The filter MAY be incorporated within an existing subscription (an active dialog) by the notifier receiving a re-SUBSCRIBE that includes the filter in the body.

4.2.3 Notifier Generation of NOTIFY requests

Upon receiving the SUBSCRIBE with the filtering rules, the notifier SHOULD retain the filter as long as the subscription persists. The filter SHOULD be used as long as the subscription is active.

If the response sent to the SUBSCRIBE was a 202, and the 202 was chosen because the filter could not be accepted in time, the NOTIFY MAY be used to terminate the subscription if the filter was not found acceptable. See Section 4.2.4.

As described in [2], the NOTIFY message MAY contain a body that describes the state of the watchers. This body MUST be in one of the formats listed in the Accept header of the SUBSCRIBE, or the presence specific default æapplication/watcherinfo+xmlÆ if the Accept header is omitted.

4.2.3.1 Generating NOTIFY Contents

The input to the filter is a watcherinfo document [3] derived according to the normal functionality defined in [2].

Before applying the Xpath expression to the watcherinfo document, the æstateÆ attribute is considered. If the value is ôpartialö, the information of watchers, whose state has not changed since the last document, is removed before any further processing. If the value is ôfullö, all watchersÆ information is included in the next processing step.

If the æstateÆ attribute is omitted from the filter, the notifierÆs local policies are applied to determine the watchers whose information is filtered.

The content is then filtered according to the XPath expression. The XPath expression indicates the delivered watcher elements.

The 'report' attribute indicates the delivered content. The value ôdefaultö of the 'report' attribute means that all the values and attributes of the selected watcher elements are delivered. In addition to the selected watcher elements, the namespace definitions are also conveyed.

4.2.4 Handling of abnormal cases

In case of an invalid filter definition û the filter XML document is not aligned with the filter XML schema, the notifier rejects the SUBSCRIBE request with a 488. A warning header in the response may give a better indication why the filters are not accepted.

If the subscription was accepted with a 202 response and the filter is being evaluated after that, a NOTIFY with a subscription-state of terminated is sent. An event-reason-value of ôbadfilterö MAY be included.

The above paragraph uses the extension to the event-reason-value of subexp-params as defined in [6].

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

In case of indication of empty content by the XPath clause the presence agent functions according to the normal procedure for the content filtering described in this document.

5.0 IANA Considerations

A new content type "application/simple-winfo-filter+xml" is defined to represent an XML MIME for the filter criteria of Presence.

This specification follows the guidelines of RFC3023 [12].

OPEN ISSUE: IANA registration template must be added later.

6.0 Examples

The examples in this section use the ôpresenceö event package with the ôwinfoö template-package.

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>
    </watcherinfo>
```

6.1 Presentity makes subscription to get 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@dom.com>
Content-Type: application/simple-winfo-filter+xml
Content-Length: à

<?xml version="1.0" encoding="UTF-8"?>
  <ev-filter-set xmlns="urn:ietf:params:xml:ns:simple-winfo-filter"
pkgdef=öwinfoö>
      <ev-filter id = "active" uri=ösip:presentity@domain.comö>
        <what report="default" state="partial">
          //*[(.                .                /@package="presence"
and                                  /                                    /
@status="active")]/ancestor-or-
self:                                                                 :*
         </what>
       </ev-filter>
  </ev-filter-set>
```

Notification to the subscriber:

```
NOTIFY sip:presentity@dom.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: à

<?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>
     </watcherinfo>
```

6.2 Presentity makes subscription to get information about defined, new and unauthorized 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@dom.com>
Content-Type: application/simple-winfo-filter+xml
Content-Length: à

<?xml version="1.0" encoding="UTF-8"?>
  <ev-filter-set xmlns="urn:ietf:params:xml:ns:simple-winfo-filter"
pkgdef=öwinfoö>
      <ev-filter id = "filter_X" uri=ösip:presentity@domain.comö>
        <what report="default" state="partial">
           //*[(.                .                /@package="presence" and
((@status="terminated" and
@event="rejected") or @event="subscribe" or contains(.,"domain.com"))]/
ancestor-or-
self:                                                              :
*
         </what>
       </ev-filter>
  </ev-filter-set>
```

Notification to the subscriber:

```
NOTIFY sip:presentity@dom.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: à

   <?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="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>
     </watcherinfo>
```

6.3 Presentity requests history information of 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>;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@dom.com>
Content-Type: application/simple-winfo-filter+xml
Content-Length: à

<?xml version="1.0" encoding="UTF-8"?>
  <ev-filter-set xmlns="urn:ietf:params:xml:ns:simple-winfo-filter"
pkgdef=öwinfoö>
       <ev-filter id = "history" uri=ösip:presentity@domain.comö>
         <what report="default" state="full">
            //*[(.                  .                 /@package="presence" and
(@duration-subscribed>500)]/ancestor-or-
self:      :*
        </what>
      </ev-filter>
  </ev-filter-set>

Notification to the subscriber:

NOTIFY sip:presentity@dom.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: à

<?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>
  </watcherinfo>
```

7.0 XML Schema

```
The XML Schema for the watcher information filter.
******************

XML Schema Implementation (Normative)
<?xml version="1.0" encoding="UTF-8"?>

<!-- ************************************************************ -->

<xsd:schema targetNamespace="urn:ietf:params:xml:ns:simple-winfo-
filter"
xmlns:xsd=http://www.w3.org/2001/XMLSchema/>
<xsd:import namespace="http://www.w3.org/XML/1998/namespace"
  schemaLocation="http://www.w3.org/2001/xml.xsd"> <!ùlang -->

<xsd:annotation>
  <xsd:documentation xml:lang="en">
    XML Schema Definition for SIP Event Filtering for Winfo.
  </xsd:documentation>
</xsd:annotation>

<!-- ************************************************************ -->

<xsd:element name="ev-filter-set" type="EventFilterSetType"/>

<!--
```

```
   The "pkgdef" attribute is useful to identify the event package
   -->

   <xsd:complexType name="EventFilterSetType">
```

```
    <xsd:sequence>
      <xsd:element name="ev-filter" type="EventFilterType"
                   minOccurs="1"    maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="pkgdef"    type="xsd:string" use="optional"/>
  </xsd:complexType>

  <!--
     An event filter corresponds to the SIP URI indicated by its "uri"
  attribute. The URI defines the watcher-list resource. -->

  <xsd:complexType name="EventFilterType">
    <xsd:sequence>
      <xsd:element name="what" type="WhatType" use="required"/>
    </xsd:sequence>
    <xsd:attribute name="id"  type="xsd:string" use="required"/>
    <xsd:attribute name="uri" type="xsd:anyURI" use="optional"/>
    <xsd:attribute name="state" type="StateType" use="optional"/>

  </xsd:complexType>


  <xsd:complexType name="WhatType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
     <xsd:attribute name="report" type="ReportType" use="required"/>
      </xsd:extension base>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:simpleType name="ReportType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="default"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="StateType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="partial"/>
      <xsd:enumeration value="full"/>
    </xsd:restriction>
  </xsd:simpleType>


  <!-- ******************************************************** -->

  </xsd:schema>

  <!-- *
```

8.0 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 rules.

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

All watcherinfo related security measures discussed in [2] MUST be followed.

9.0 Acknowledgements

The authors would like to thank Jyrki Aarnos and Juha Kalliokulju and Tim Moran for their valuable comments.

10.0 References

[1] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[2] Rosenberg, J., "A Watcher Information Event Template-Package for SIP", draft-ietf-simple-winfo-package-05.txt. Internet Draft, January 2003, Work in progress.

[3] Rosenberg, J., "An XML Based Format for Watcher Information", draft-ietf-impp-cpim-pidf-05.txt. Internet Draft, January 2003. Work in progress.

[4] Roach, A., "SIP-Specific Event Notification", RFC 3265, Internet Engineering Task Force, June 2002.

[5] Kiss, K. et al, " Requirements for Filtering of Watcher Information", draft-kiss-simple-winfo-filter-reqs-00.txt. Internet Draft, February 2003, Work in progress.

[6] Khartabil, H. et al, ôEvent Notification Filtering for Presence", draft-khartabil-simple-presence-filter-00.txt. Internet Draft, January 2003. Work in progress.

[7] W. W. W. C. (W3C), "Extensible markup language (xml) 1.0." The XML 1.0 spec can be found at http://www.w3.org/TR/1998/REC-xml-19980210.

[8] R. Moats, "URN syntax," RFC 2141, Internet Engineering Task Force, May 1997.

[9] R. Moats, "A URN namespace for IETF documents," RFC 2648, Internet Engineering Task Force, Aug. 1999.

[10] M. Mealling, "The IETF XML registry", draft-mealling-iana-xmlns-registry-04.txt. Internet Draft. November 2001. Work in progress.

[11] J. Rosenberg, et al. ôSIP: Session Initiation Protocolö. RFC 3261, Internet Engineering Task Force, June 2002.

[12] M. Murata, S. S. Laurent, and D. Kohn, "XML media types," RFC 3023, Internet Engineering Task Force, Jan. 2001.

[13] K.Kiss, ôRequirements for Presence Service based on 3GPP specifications and wireless environment characheristicsö draft-kiss-simple-presence-wireless-reqs-01.txt. Internet Draft, October 2002. Work in progress.

[14] W. W. W. C. (W3C), "XML Path Language (Xpath)". W3C Recommendation. <link - http://www.w3.org/TR/xpath >

AuthorsÆ Addresses

Hisham Khartabil
Nokia

P.O. Box 321
FIN-00045
NOKIA GROUP
FINLAND

Email: hisham.khartabil@nokia.com
Tel: + 358 7180 76161

Eva Leppanen
Nokia

P.O Box 785
FIN-33101 Tampere
FINLAND

Tel: +358 7180 77066
Email: eva-maria.leppanen@nokia.com

Full Copyright Statement

Acknowledgement

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