Network Working Group
Request for Comments: 4661
Category: Standards Track

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

September 2006

An Extensible Markup Language (XML)-Based Format for Event Notification Filtering

Status of This Memo

Copyright Notice

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 whereby filtering of event notification information can be achieved. Filtering is a mechanism for defining the preferred notification information to be delivered and for specifying triggers that cause that information to be delivered. In order to enable this, a format is needed to enable the subscriber to describe the state changes of a resource that cause notifications to be sent to it and what those notifications are to contain. This document presents a format in the form of an XML document.

Table of Contents

1. Introduction

The SIP event notification framework [2] 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 whereby filtering of event notification information can be achieved.

Filtering is a mechanism for defining the preferred notification information, referred to as content, to be delivered and for specifying the rules for when that information should be delivered.

The filtering mechanism is expected to be particularly valuable and primarily applicable 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. However, implementers need to be aware of the computational burden on the source of the event notification. This is discussed further in Section 8.

The structure of the filter criteria is described using the XML schema. The filter criteria is presented as an XML document. The XML document contains the user's preference as to when notifications are to be sent to it and what they are to contain. The scope of the "when" part is triggering.

The triggering is defined as enabling a subscriber to specify triggering rules for notifications where the criteria are based on changes of the event package [2] specific state information, e.g., for the presence information document [15], the change in the value of the <status> element.

The functionality of the filtering regarding the SIP event notifications is specified in [3].

2. 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.

Throughout the document, the "resulting XML document" refers to the final XML document that carries state information to be delivered to the subscriber after the filters have been applied to it.

"Content" refers to the XML document that appears in a notification reflecting the state of a resource.

3. Structure of XML-Encoded Simple-Filter

A simple-filter is an XML document [8] that MUST be well formed and MUST be valid according to schemas, including extension schemas, available to the validater, and applicable to the XML document. The simple-filter documents MUST be based on XML 1.0 and MUST be encoded using UTF-8.

The namespace identifier for elements defined by this specification is a URN [5], which uses the namespace identifier 'ietf' defined by [6] and extended by [4]. This urn is: urn:ietf:params:xml:ns:simple-filter.

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

3.1. MIME Type for Simple-Filter Document

The MIME type for the simple-filter document is "application/simple-filter+xml". Any transport protocol (SIP [12], for example) used to carry the filters that also carries payload type information MUST identify the payload as MIME type "application/simple-filter+xml" (for example, a Content-Type header field).

3.2. The <filter-set> Root Element

The root element of the filter criteria is <filter-set>.

The <filter-set> element contains the namespace definition mentioned above. With the optional 'package' attribute, it is possible to define the package to which the filter criteria is applied. This might be especially useful in cases where the XML document containing the filter criteria is separated from the events that make use of it or from the protocol that usually carries it.

The <filter-set> element may contain one <ns-bindings> element.

The <filter-set> element contains one or more <filter> elements.

3.3. The <ns-bindings> Element

The <ns-bindings> element is used to bind namespaces to local prefixes used in expressions that select elements or attributes using

the syntax in Section 5. Those prefixes apply to the <include>, <exclude>, <changed>, <added>, and <removed> elements.

The <ns-bindings> element contains one or more <ns-binding> elements. Each <ns-binding> element has a 'prefix' attribute. The value of the 'prefix' attribute is a prefix used to qualify the elements pointed to by the expression. The <ns-binding> element also has a 'urn' attribute that identifies the namespace that the prefix represents.

3.4. The <filter> Element

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

The <filter> element has an 'id' attribute. The value of the 'id' attribute MUST be unique within the <filter-set> element. The <filter> MAY have a 'uri' attribute. The value of the 'uri' attribute is the URI of the resource to which the filter applies. The <filter> MAY have a 'domain' attribute. The value of the 'domain' attribute is the domain of the resources to which the filter applies. The 'uri' attribute and the 'domain' attribute MUST NOT appear together in the <filter>.

The URI of the resource is useful in cases where the 'event list' extension [17] is used with a package. Since a subscription to an event package may be addressed to an event list, the 'uri' attribute allows the subscriber to define a filter specific to an individual resource within that list. That resource may be another list. The 'uri' attribute may, of course, carry the URI of the list itself. If the <filter> does not contain the 'uri' attribute, the filter applies to the resource identified in the subscription request.

The 'domain' attribute carries a domain. In this case, the filter applies to resources whose URI has a domain part matching that domain. This can be used when a subscription is for a resource that is an event list with many resources from differing domains.

URI matching is done according to the matching rules defined for a particular scheme. When matching domains, the user part of the URI is ignored for matching purposes.

The <filter> MAY have a 'remove' attribute that together with the 'id' attribute indicates the existing filter to be removed. The value of the 'remove' attribute is of the type "Boolean". The default value is "false".

The <filter> MAY have an 'enabled' attribute that indicates whether a filter is enabled or disabled. The value of the 'enabled' attribute is of the type "Boolean". The default value is "true".

The <filter> element MAY contain a <what> element and MAY contain one or more <trigger> elements, but it MUST contain either the <what> element or the <trigger> element when the filter is being enabled for the first time. When a filter is disabled by setting the 'enabled' attribute to "false", the <what> and <trigger> elements MAY be omitted. Similarly, when a filter is re-enabled by setting the 'enabled' attribute to "true", the <what> and <trigger> elements MAY be omitted.

Filter contents can be changed by changing the contents in the <what> and <trigger> elements and maintaining the value of the filter 'id' attribute.

3.5. The <what> Element

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

The <what> element may contain one or more <include> elements and one or more <exclude> elements. When more than one <include> element has been defined, the results are additive. That is, each <include> element adds an element or attribute to the resulting XML document. When more than one <exclude> element has been defined, each <exclude> element value depletes the contents of the resulting XML document.

3.5.1. The <include> Element

The <include> element is used to select the content to be delivered. Its value can identify an XML element, an attribute, or a namespace of an XML document to be filtered. This is indicated using the 'type' attribute.

Note that the resulting XML document MUST be valid. Therefore, in addition to including the elements identified with the <include> element value, all other mandatory XML elements and/or attributes must be incorporated in the resulting XML document in order to make it valid. This, in practice, means that a subscriber defining a filter only needs to <include> optional elements and/or attributes, but may <include> mandatory elements and/or attributes as well. There are also cases where a filter selects an attribute that belongs to an optional element. In this case, the optional element needs to appear in the resulting XML document.

The syntax defined in Section 5 (see the definition of "selection") MUST be used. The following example selects the <basic> element defined in the PIDF [13]. This results in the selection of the <basic> element as well as all the ancestors, i.e., <status> and <tuple>.

<include type="xpath">/presence/tuple/status/basic</include>.

3.5.2. The <exclude> Element

The <exclude> element is used to define exceptions to the set of XML elements and/or attributes selected by the <include> elements. Thus, XML elements (including their lower-level "child" elements) and/or attributes defined by the <exclude> element are not delivered. This is most useful when an <include> element identifies a namespace.

The <exclude> element has the optional 'type' attribute (see the definition of the 'type' in Section 3.5.3).

Note that the resulting XML document MUST be valid. Therefore, if the step in applying the <exclude> element value to an XML document results in an invalid document according to the schema, that step MUST be reversed, resulting in the elements and/or attributes being re-introduced into the resulting XML document with their previous values in order to make it valid. This, in practice, means that a subscriber defining a filter only needs to <exclude> optional elements and/or attributes, but SHOULD NOT <exclude> mandatory elements and/or attributes.

The syntax MUST follow Section 5.

3.5.3. The 'type' Attribute

The 'type' attribute is used to describe the value of the <include> and <exclude> elements. The following values are pre-defined: "xpath" and "namespace". The 'type' attribute is optional, and, if omitted, the default value is "xpath".

The "xpath" value is used when the <include> or <exclude> element contains a value following the syntax in Section 5 that selects an element or an attribute.

The "namespace" value is used when the <include> or <exclude> element contains a value of a namespace. The value is the URI of the namespace. The resulting XML document is comprised of the elements defined within the namespace.

3.6. The <trigger> Element

The <trigger> element is used to identify the package-specific changes that a resource has to encounter before the content is delivered to the subscriber. It can appear more than once in a <filter> element. Multiple appearances of this element denote the "OR" operation. This means that updates to a resource that satisfy any of the <trigger> elements criteria constitute the content to be delivered.

The <trigger> element MAY contain the <changed>, <added>, or <removed> elements, but it MUST contain at least one of the three elements. Any combination of the 3 elements is possible. Multiple appearances of those elements within a <trigger> element denotes the "AND" operation. This means that updates to a resource that satisfy ALL of the <changed>, <added>, and <removed> elements' criteria within the <trigger> element constitute the content to be delivered.

3.6.1. The <changed> Element

The <changed> element is used to identify the XML element or attribute, from the package-specific XML document, whose value MUST change from that of the "previous XML document", in order to activate the trigger and cause the content to be delivered. Previous XML document" in this context refers to the raw version of the most recent XML document that was sent to the subscriber, before the filters were applied to it. The XML element or attribute MUST be expressed using the syntax defined in Section 5 for the "reference" ABNF.

The <changed> element MAY contain the 'from' attribute, the 'to' attribute, the 'by' attribute, or any combination of the three. The absence of all of those attributes means a change of any sort to the value of the element or attribute activates the trigger. An update to the element or attribute value with an identical value is not a change.

Comparison of a change is done according to the element or attribute's lexical rules.

3.6.1.1. The 'from' Attribute

A trigger is active when the XML element or attribute identified with the <changed> element has changed from the value indicated by this attribute to a different value.

3.6.1.2. The 'to' Attribute

A trigger is active when the XML element or attribute identified with the <changed> element has changed to the value indicated by this attribute from a different value.

3.6.1.3. The 'by' Attribute

A trigger is active when the XML element or attribute identified with the <changed> element has changed by at least the amount indicated by this attribute from a different value. That is, the 'by' attribute applies only to numerical values and indicates a delta with respect to the current value that an attribute or element (identified in the <changed> element) needs to change before it is selected. For example, if the 'by' attribute is set to 2 and the current value of the element/attribute is 6, the element/attribute is selected when it reaches (or exceeds) the value 8 or when it decreases to 4 or a lower value.

3.6.1.4. Combination of Attributes

Any combination of the 'from', 'to', and 'by' attributes in the <changed> element is possible. For example, if the 'from' attribute is combined with the 'to' attribute, it is interpreted to mean that the trigger is active when the XML element or attribute identified with the <changed> element has changed from the 'from' value to the 'to' value. Note that if the 'by' attribute is used in combination with the other attributes, the other attribute types MUST match the 'by' type of decimal.

3.6.2. The <added> Element

The <added> element triggers content delivery when the XML element it identifies has been added to the document being filtered (that is, this instance of that element appears in the current document, but not in the previous document). It can be used, for example, to learn of new services and/or capabilities subscribed to by the user, or services and/or capabilities that the user has now allowed the subscriber to see. The XML element or attribute MUST be expressed using the syntax defined in Section 5 for the "reference" ABNF.

Note that if a filter includes both the content filter (<what>) part and the <added> element, then the definitions of the <what> part SHOULD also cover the added elements. Otherwise, the content is delivered without the items defined in the <added> element.

3.6.3. The <removed> Element

The <removed> element triggers content delivery when the XML element it identifies has been removed from the document being filtered (that is, this instance of that element appeared in the previous document, but not in this document). The XML element or attribute MUST be expressed using the syntax defined in Section 5 for the "reference" ABNF.

4. XML Schema Extensibility

The simple-filter document is meant to be extended. An extension takes place by defining a new set of elements in a new namespace, governed by a new schema. Every extension MUST have an appropriate XML namespace assigned to it. The XML namespace of the extension MUST be different from the namespaces defined in this specification. The extension MUST NOT change the syntax or semantics of the schemas defined in this document. All XML tags and attributes that are part of the extension MUST be appropriately qualified so as to place them within that namespace and MUST be designed such that receivers can safely ignore such extensions.

This specification defines explicit places where new elements or attributes from an extension can be placed. These are explicitly indicated in the schemas by the <any> and <anyAttribute> elements. Extensions to this specification MUST specify where their elements can be placed within the document.

As a result, a document that contains extensions will require multiple schemas in order to determine its validity - a schema defined in this document, along with those defined by extensions present in the document. Because extensions occur by adding new elements and attributes governed by new schemas, the schemas defined in this document are fixed and would only be changed by a revision to this specification. Such a revision, should it take place, would endeavor to allow documents compliant to the previous schema to remain compliant to the new one. As a result, the schemas defined here don’t provide explicit schema versions, as this is not expected to be needed.

5. Syntax for Referencing XML Items and Making Logical Expressions

The ABNF [10] is used to describe the syntax for the expressions. The syntax is defined to be XPATH [9] compatible but has only a restricted set of capabilities of the XPATH. More information about the meaning of the items of the syntax can be found in [9]. The "abbreviated syntax" of the "node test" is used in the references ("reference"). The expression in the syntax relates to the

predicate, comparison, and logical expressions of the XPATH. If an XPATH expression evaluates to more than one element at a certain step, the filter applies to all the elements that are evaluated. That is, if a filter including an element evaluates to 2 elements, both elements are included as a result.

```
selection = reference [expression]
expression = "[" (elem-expr / attr-expr)
                    1*[oper (elem-expr / attr-expr)] "]"
elem-expr = (elem-path / "." / "..") compar value
elem-path = (element / "*") 1*["/" / "*" / element] ["*" / element]
attr-expr = [elem-path "/"] attribute compar value

reference = elem-reference / attr-reference
elem-reference =  "/" 1*("/" / "/*" / ("/" element))
attr-reference = reference attribute

oper = "and" / "or"
compar = "=" / "<" / ">"
element = [ns] string
attribute = "@" [ns] string
ns = string ":"
string = <any sequence of data supported by XML in names of XML
element, and/or attribute or prefixes of namespaces>
value = <any sequence of data supported by XML as a value of the
XML element and/or attribute>
```

When identifying XML elements or attributes, the value may consist of two parts: the XML element/attribute selector and the condition (comparison and logical expressions). The XML element selector appears first followed by the condition part in square brackets. In the XML element selector part, the XML elements may be referenced by giving the full hierarchical path as: "/presence/tuple/status/basic", by denoting the selection to cover any hierarchical level by its name as: "//tuple/status/basic", or using the wildcard "*", denoting any value in a certain level as "/*/watcher".

Example references are listed as follows:

- Selecting an element by using an XML element as a condition:
  - //*[status/basic="open"]
  - /presence/tuple[*/basic="open"]

- Selecting an element by using XML attributes as a condition:
  - //watcher[@duration-subscribed<500]
  - /*/watcher[@event="rejected"]

- Selecting an element by using two XML elements as a condition:
  - //tuple[status/basic="open" and type="device"]

- Selecting an attribute:
  - //watcher/@duration-subscribed

In some cases, due to the design of the XML schema, the XPATH-like expression results in identification of more than one element with the same name (the XPATH expression may not have uniquely identified an element at every step). In those cases, all elements identified are selected.

When evaluating XPATH location steps, namespace expansion follows XPATH 1.0 [9] semantics, i.e., if the QName does not have a prefix, then the namespace URI in the expanded name is null. With non-default namespaces, expansion is done according to the given <ns-bindings> definitions. When a default namespace is used in the document, the <ns-binding> element SHOULD be used to define an equal URI with some prefix in order to have a valid XPATH evaluation in location steps.

6. Examples

The XML Schema for the XML document examples is specified in Section 7.

6.1. Filter Criteria Using <what> Element

A user wishes to get to know his friend’s availability and willingness for messaging (SMS, IM, and MMS) in order to know whether the friend is able to receive a message, the address to contact, and the type of the message to be used.

This example shows how to define a content filter. The <basic> element as well as all parent elements are selected based on a condition defined by a logical expression. The condition is <class> elements that have a value "MMS", "SMS", or "IM".

The <class> element is defined in [14] as an extension to PIDF [13].

```
<?xml version="1.0" encoding="UTF-8"?>
<filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
  <ns-bindings>
    <ns-binding prefix="pidf" urn="urn:ietf:params:xml:ns:pidf"/>
    <ns-binding prefix="rpid"
                        urn="urn:ietf:params:xml:ns:pidf:rpid"/>
  </ns-bindings>
  <filter id="123" uri="sip:presentity@example.com">
```

```
      <what>
        <include type="xpath">
          /pidf:presence/pidf:tuple[rpid:class="IM" or rpid:class="SMS"
          or rpid:class="MMS"]/pidf:status/pidf:basic
        </include>
      </what>
    </filter>
  </filter-set>
```

6.2. Filter Criteria Using <trigger> Element

A user requires to be informed when his colleague becomes available by some communication means. The user gets the full presence state of the colleague when a certain PIDF [13] tuple <basic> status changes from "closed" to "open".

```
<?xml version="1.0" encoding="UTF-8"?>
<filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
  <ns-bindings>
    <ns-binding prefix="pidf" urn="urn:ietf:params:xml:ns:pidf"/>
  </ns-bindings>
  <filter id="123" uri="sip:presentity@example.com">
    <trigger>
      <changed from="CLOSED" to="OPEN">
        /pidf:presence/pidf:tuple/pidf:status/pidf:basic
      </changed>
</trigger>
</filter>
</filter-set>
```

6.3. Filter Criteria Using <what> and <trigger> Elements

A user wishes to get information about pending and waiting subscriptions in order to be able to authorise watchers to see his presence information.

The filter selects watcher information notifications [16] to be sent when a subscription status has changed to "pending" or "waiting". In the notification, only the watchers that have a status of "pending" or "waiting" are included.

```
<?xml version="1.0" encoding="UTF-8"?>
<filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
  <ns-bindings>
    <ns-binding prefix="wi"
                        urn="urn:ietf:params:xml:ns:watcherinfo"/>
  </ns-bindings>
  <filter id="123" uri="sip:presentity@example.com">
```

```
      <what>
        <include>
          /wi:watcherinfo/wi:watcher-list/wi:watcher[@status="pending"
          or @status="waiting"]
        </include>
      </what>
      <trigger>
        <changed to="pending">
          /wi:watcherinfo/wi:watcher-list/wi:watcher/@status
        </changed>
      </trigger>
      <trigger>
        <changed to="waiting">
          /wi:watcherinfo/wi:watcher-list/wi:watcher/@status
        </changed>
      </trigger>
    </filter>
  </filter-set>
```

6.4. Content Filter Using Namespaces

A user turns her terminal on, and the terminal automatically fetches general presence status and information about communication means from a certain pre-defined set of her buddies.

The filter is defined to select XML elements belonging to the PIDF namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
  <filter id="123" uri="sip:buddylist@example.com">
    <what>
      <include type="namespace">
        urn:ietf:params:xml:ns:pidf
      </include>
    </what>
  </filter>
</filter-set>
```

6.5. Content Filter Using Only <include> Elements

A user wants to know if a group of his friends is available for gaming. He orders notifications about the current status and future changes of the game-specific presence information.

This filter is defined to select the game-specific tuple to be delivered.

```
<?xml version="1.0" encoding="UTF-8"?>
<filter-set xmlns="urn:ietf:params:xml:ns:simple-filter" >
  <ns-bindings>
    <ns-binding prefix="game-ext"
                        urn="urn:ietf:params:xml:ns:game-ext"/>
  </ns-bindings>
  <filter id="123">
    <what>
      <include>
        /pidf:presence/pidf:tuple/
        pidf:status[game-ext:label="game-X"]
      </include>
    </what>
  </filter>
</filter-set>
```

6.6. Two Content Filters as Filter Criteria

The user is interested in getting up-to-date information about the communication means and contact addresses of his friends. The user also wants to get more information (e.g., location) about one of the friends in the list, named Bob. The PIDF element <note> is filtered out, i.e., excluded. The list was predefined as buddies@example.com.

```
<?xml version="1.0" encoding="UTF-8"?>
<filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
  <ns-bindings>
    <ns-binding prefix="pidf" urn="urn:ietf:params:xml:ns:pidf"/>
    <ns-binding prefix="rpid"
                        urn="urn:ietf:params:xml:ns:pidf:rpid"/>
  </ns-bindings>
  <filter id="8439" uri="sip:buddies@example.com">
    <what>
      <include>
        /pidf:presence/pidf:tuple[rpid:class="service"]/pidf:status/
        pidf:basic
      </include>
    </what>
    </filter>
  <filter id="999" uri="sip:bob@example.com">
    <what>
      <include type="namespace">
        urn:ietf:params:xml:ns:pidf
      </include>
      <exclude>
        /pidf:presence/pidf:tuple/pidf:note
      </exclude>
    </what>
```

```
    </filter>
  </filter-set>
```

7. XML Schema for Filter Criteria

XML Schema Implementation (Normative)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:simple-filter"
       xmlns="urn:ietf:params:xml:ns:simple-filter"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       elementFormDefault="qualified">

  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
                  schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <xs:annotation>
    <xs:documentation xml:lang="en">
      XML Schema Definition for Filter Criteria.
    </xs:documentation>
  </xs:annotation>

  <xs:element name="filter-set" type="FilterSetType"/>

    <xs:complexType name="FilterSetType">
      <xs:sequence>
        <xs:element name="ns-bindings" type="NSBindings"
                          minOccurs="0" maxOccurs="1"/>
        <xs:element name="filter" type="FilterType"
                          minOccurs="1"
      maxOccurs="unbounded"/>
        </xs:sequence>
      <xs:attribute name="package" type="xs:string" use="optional"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

  <xs:complexType name="NSBindings">
    <xs:sequence>
      <xs:element name="ns-binding" type="NSBinding"
                        minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="NSBinding">
    <xs:attribute name="prefix" type="xs:string" use="required"/>
    <xs:attribute name="urn" type="xs:anyURI" use="required"/>
  </xs:complexType>
```

```
<xs:complexType name="FilterType">
  <xs:sequence>
    <xs:element name="what" type="WhatType"
                        minOccurs="0" maxOccurs="1"/>
    <xs:element name="trigger" type="TriggerType"
                        minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id"  type="xs:string" use="required"/>
  <xs:attribute name="uri" type="xs:anyURI" use="optional"/>
  <xs:attribute name="domain" type="xs:string" use="optional"/>
  <xs:attribute name="remove" type="xs:boolean" use="optional"
                       default="false"/>
  <xs:attribute name="enabled" type="xs:boolean" use="optional"
                       default="true"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:complexType name="WhatType">
  <xs:sequence>
    <xs:element name="include" type="InclType"
                        minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="exclude" type="ExclType"
                        minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="InclType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="type" type="TypeType"
                           default="xpath" use="optional"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="ExclType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="type" type="TypeType"
                           default="xpath" use="optional"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
```

```
   </xs:complexType>

     <xs:simpleType name="TypeType">
       <xs:restriction base="xs:string">
         <xs:enumeration value="xpath"/>
         <xs:enumeration value="namespace"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:complexType name="TriggerType">
       <xs:sequence>
       <xs:element name="changed" type="ChangedType"
                            minOccurs="0" maxOccurs="unbounded"/>
       <xs:element name="added" type="xs:string"
                            minOccurs="0" maxOccurs="unbounded"/>
       <xs:element name="removed" type="xs:string"
                            minOccurs="0" maxOccurs="unbounded"/>
       <xs:any namespace="##other" processContents="lax"
                           minOccurs="0" maxOccurs="unbounded"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="ChangedType">
       <xs:simpleContent>
         <xs:extension base="xs:string">
           <xs:attribute name="from" type="xs:anySimpleType"
                                 use="optional"/>
           <xs:attribute name="to" type="xs:anySimpleType"
                                 use="optional"/>
           <xs:attribute name="by" type="xs:decimal"
                                 use="optional"/>
           <xs:anyAttribute namespace="##other" processContents="lax"/>
         </xs:extension>
       </xs:simpleContent>
     </xs:complexType>

   </xs:schema>
```

8. Security Considerations

The filters in the body in a SIP message have 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 authorised. Authentication can be achieved using the Digest Authentication mechanism described in [12]. The authorisation decision is based on the permissions that the resource (notifier) has given to the watcher. An example of such an auhorisation policy can be found in [18].

Requests can reveal sensitive information about a UA's capabilities. If this information is sensitive, it SHOULD be encrypted using SIP S/MIME capabilities [11].

All filtering-related security measures discussed in [2] MUST be followed along with package-specific security.

9. IANA Considerations

This document registers a new MIME type, "application/ simple-filter+xml", and registers a new XML namespace.

This specification follows the guidelines of RFC3023 [7].

9.1. application/simple-filter+xml MIME TYPE

MIME media type: application

MIME subtype name: simple-filter+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml, as specified in RFC 3023 [7].

Encoding considerations: Same as encoding considerations of application/xml, as specified in RFC 3023 [7].

Security considerations: See section 10 of RFC 3023 [7] and section Section 8 of this document.

Interoperability considerations: none.

Published specification: This document.

Applications that use this media type: This document type has been used to support the SIP-based Event notification framework and its packages.

Additional information:

Magic number: None

File extension: .cl or .xml

Macintosh file type code: "TEXT"

Personal and email address for further information: Hisham Khartabil (hisham.khartabil@telio.no)

Intended Usage: COMMON

Author/change controller: The IETF

9.2. URN Sub-Namespace Registration for urn:ietf:params:xml:ns:simple-filter

This section registers a new XML namespace, as per guidelines in the IETF XML Registry [4].

URI: The URI for this namespace is urn:ietf:params:xml:ns:simple-filter.

Registrant Contact: IETF, SIMPLE working group, Hisham Khartabil (hisham.khartabil@telio.no)

9.3. Schema Registration

This section registers a new XML schema per the procedures in [4].

URI: urn:ietf:params:xml:ns:simple-filter

Registrant Contact: IETF, SIMPLE working group, Hisham Khartabil (hisham.khartabil@telio.no).

The XML for this schema can be found as the sole content of Section 7.

10. Acknowledgements

The authors would like to thank Jonathan Rosenberg, Henning Schulzrinne, Tim Moran, Jari Urpalainen, Sreenivas Addagatla, Miguel-Angel Garcia Martin, Mary Barnes, Paul Kyzivat, Robert Sparks, and Elwyn Davies for their valuable input and comments.

11. References

11.1. Normative References

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

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

[3] Khartabil, H., Leppanen, E., Lonnfors, M., and J. Costa-Requena, "Functional Description of Event Notification Filtering", RFC 4660, September 2006.

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

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

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

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

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

[9] Clark, J., "XML Path Language (XPath) Version 1.0", W3C REC REC-xpath-19991116, November 1999.

[10] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[11] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

11.2. Informative References

[12] 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.

[13] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

[14] Schulzrinne, H., Gurbani, V., Kyzivat, P., and J. Rosenberg, "RPID -- Rich Presence Extensions to the Presence Information Data Format (PIDF)", RFC 4480, July 2006.

[15] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[16] Rosenberg, J., "An Extensible Markup Language (XML) Based Format for Watcher Information", RFC 3858, August 2004.

[17] Roach, A. B., Campbell, B., and J. Rosenberg, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", RFC 4663, September 2006.

[18] Rosenberg, J., "Presence Authorization Rules", Work in Progress, June 2006.

Authors' Addresses

Hisham Khartabil
Telio
P.O. Box 1203 Vika
Oslo
Norway

Phone: +47 2167 3544
EMail: hisham.khartabil@telio.no

Eva Leppanen
Nokia
P.O BOX 785
Tampere
Finland

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

Mikko Lonnfors
Nokia
P.O BOX 321
Helsinki
Finland

Phone: + 358 71800 8000
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

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).