Session Initiation Protocol (SIP) Event Notification Extension for Notification Throttling
draft-niemi-sipping-event-throttle-05

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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 September 5, 2007.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This memo specifies a throttle mechanism for limiting the rate of Session Initiation Protocol (SIP) event notifications. This mechanism can be applied in subscriptions to all SIP event packages, but the mechanism is especially designed to be used in combination with a subscription to a Resource List Server (RLS).
Table of Contents

1. Introduction ................................................. 3
2. Definitions and Document Conventions ......................... 4
3. Overview ..................................................... 4
   3.1. Use Case ............................................. 4
   3.2. Requirements .......................................... 5
   3.3. Event Throttle Model ................................... 6
   3.4. Basic Operation ....................................... 7
4. Operation of Event Throttles ................................... 7
   4.1. Negotiating the Use of Throttle .......................... 8
   4.2. Setting the Throttle .................................... 8
      4.2.1. Subscriber Behavior .............................. 8
      4.2.2. Notifier Behavior ................................ 8
   4.3. Selecting the Throttle Interval ......................... 9
   4.4. Buffer Policy Description .............................. 9
      4.4.1. Partial State Notifications ...................... 9
      4.4.2. Full State Notifications .......................... 10
   4.5. Estimated Bandwidth Savings ............................ 10
5. Syntax ...................................................... 11
   5.1. "throttle" Header Field Parameter ....................... 11
   5.2. Augmented BNF Definitions .............................. 11
6. IANA Considerations .......................................... 11
7. Security Considerations ..................................... 11
8. Acknowledgements ............................................ 12
9. References ................................................ 12
   9.1. Normative References ................................. 12
   9.2. Informative References ................................ 12
Author’s Address ............................................... 13
Intellectual Property and Copyright Statements ............... 14
1. Introduction

The SIP events framework [RFC3265] defines a generic framework for subscriptions to and notifications of events related to SIP systems. This framework defines the methods SUBSCRIBE and NOTIFY, and introduces the concept of an event package, which is a concrete application of the SIP events framework to a particular class of events.

One of the things the SIP events framework mandates is that each event package specification defines an absolute maximum on the rate at which notifications are allowed to be generated by a single notifier. Such a limit is provided in order to reduce network congestion.

All of the existing event package specifications include a maximum notification rate recommendation, ranging from once in every five seconds [RFC3856], [RFC3680], [RFC3857] to once per second [RFC3842].

Per the SIP events framework, each event package specification is also allowed to define additional throttle mechanisms which allow the subscriber to further limit the rate of event notification. So far none of the event package specifications have defined such a mechanism.

The resource list extension [I-D.ietf-simple-event-list] to the SIP events framework also deals with rate limiting of event notifications. The extension allows a subscriber to subscribe to a heterogenous list of resources with a single SUBSCRIBE request, rather than having to install a subscription for each resource separately. The event list subscription also allows rate limiting, or throttling of notifications, by means of the Resource List Server (RLS) buffering notifications of resource state changes, and sending them in batches. However, the event list mechanism provides no means for the subscriber to set the interval for the throttling; it is strictly an implementation decision whether batching of notifications is supported, and by what means.

This document defines an extension to the SIP events framework that allows a subscriber to set a throttle to event notifications generated by the notifier. The requirements and model for generic event throttles are further discussed in Section 3. A throttle is simply a timer value that indicates the minimum time period allowed between two notifications. As a result of this throttle, a compliant notifier will limit the rate at which it generates notifications.

This mechanism is applicable to any event subscription, but it is mainly intended for use with an event list subscription.
2. Definitions and Document Conventions

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

Indented passages such as this one are used in this document to provide additional information and clarifying text. They do not contain normative protocol behavior.

3. Overview

There are many applications that potentially would make use of a throttle mechanism. This chapter only illustrates one, albeit the main use case, in which a mobile device uses the event list subscription in combination with the event throttling mechanism to limit the amount of traffic it may expect to receive.

3.1. Use Case

A presence application in a mobile device contains a list of 100 buddies or presentities. In order to decrease the processing and network load of watching 100 presentities, the presence application has employed a Resource List Server (RLS) with the list of buddies, and therefore only needs a single subscription to the RLS in order to receive notification of the presence state of the resource list.

In order to control the buffer policy of the RLS, the presence application sets a throttle interval via the event throttle extension. Alternatively, the presence application could set a default throttle for the resource list, via a list manipulation interface, e.g., using the XML Configuration Access Protocol (XCAP) [I-D.ietf-simple-xcap].

The RLS will buffer notifications that do not comply with the throttle interval, and batch all of the buffered state changes together in a single notification when allowed by the throttle. The throttle applies to the overall resource list, which means that there is a hard cap imposed by the throttle to the amount of traffic the presence application can expect to receive.

For example, with a throttle of 20 seconds, the presence application can expect to receive a notification every 20 seconds at a maximum.

The presence application can also modify the throttle during the lifetime of the subscription. For example, if the User Interface
(UI) of the application shows inactivity for a period of time, it can throttle the event list subscription little by little until it is completely squelched. After all, if the user isn’t actively access the buddy list, there is no reason to keep it absolutely current either.

Currently, a subscription refresh is needed in order to update the throttle interval. However, this is highly inefficient, since each refresh automatically generates a (full-state) notification carrying the latest resource state. There is work [I-D.niemi-sip-subnot-etags] ongoing to solve these inefficiencies.

3.2. Requirements

REQ1: The subscriber must be able to set using a throttle mechanism the minimum time period between two notifications in a specific subscription.

REQ2: It must be possible to use of the throttle mechanism in subscriptions to all events.

REQ3: It must be possible to use the throttle mechanism together with any event filtering mechanism.

REQ4: The notifier must be allowed to use a throttling policy in which the minimum time period between two notifications is longer than the one given by the subscriber.

For example, due to congestion reasons, local policy at the notifier could temporarily dictate a throttling policy that in effect increases the subscriber-configured minimum time period between two notifications.

REQ5: The throttle mechanism must provide a reasonable resolution for setting the minimum period between two notifications. At a minimum, the throttling mechanism must include discussion of the situation resulting from a minimum time period which exceeds the subscription duration, and should provide mechanisms for avoiding this situation.

REQ6: A throttle must be possible to be installed, adapted, or removed in the course of an active subscription.

REQ7: A throttle mechanism must allow for the application of authentication and integrity protection mechanisms to subscriptions invoking that mechanism.
Note that Section 7 contains further discussion on the security implications of the throttle mechanism.

### 3.3. Event Throttle Model

The notifier is responsible for sending out event notifications upon state changes of the subscribed resource. We can model the notifier as consisting of three components: the event state resource(s), the Resource List Server (RLS) (or any other notifier), a notification buffer, and finally the subscriber, or watcher of the event state, as shown in Figure 1.

![Figure 1: Model for the Resource List Server (RLS) Supporting Throttling](image)

In short, the RLS reads event state changes from the event state resource, either by creating a backend subscription, or by other means; it packages them into event notifications, and submits them into the output buffer. The rate at which this output buffer drains is controlled by the subscriber via the event throttle mechanism.
When a set of notifications are batched together, the way in which overlapping resource state is handled depends on the type of the resource state:

In theory, there are many buffer policies that the notifier could implement. However, we only concentrate on two practical buffer policies in this specification, leaving additional ones for further study and out of the scope of this work. These two buffer policies depend on the mode in which the notifier is operating.

**Full-state:** Last (most recent) full state notification of each resource is sent out, and all others in the buffer are discarded. This policy applies to those event packages that carry full-state notifications.

**Partial-state:** The state deltas of each buffered partial notification per resource are merged, and the resulting notification is sent out. This policy applies to those event packages that carry partial-state notifications.

### 3.4. Basic Operation

A subscriber that wants to limit the rate of event notification in a specific event list subscription does so by suggesting a throttle as part of the SUBSCRIBE message. The throttle indicating the minimum time allowed between transmission of two consecutive notifications in a subscription is given as an Event header parameter in the SUBSCRIBE request.

Note that the witnessed time between two consecutive received notifications may not conform to the set throttle for a number of reasons. For example, network jitter and retransmissions may result in the subscriber receiving the notifications in lesser intervals than what the throttle recommends.

A notifier that supports the throttle mechanism will comply with value given in the throttle, and adjust its rate of notification accordingly.

Throttled notifications will have exactly the same properties as the un-throttled ones, with the exception that they will not be generated more frequent than what the throttle allows.

### 4. Operation of Event Throttles
4.1. Negotiating the Use of Throttle

A subscriber that wishes to apply a throttle to notifications in a subscription constructs a SUBSCRIBE request that includes a proposed throttle interval in a "throttle" Event header field parameter.

A compliant notifier will reflect back the possibly adjusted throttle interval in a "throttle" Subscription-State header field parameter of the subsequent NOTIFY requests.

A notifier that does not understand the event-throttle extension, will not reflect the "throttle" parameter in the NOTIFY requests; the absence of this parameter serves as a hint to the subscriber that no throttling is supported by the notifier.

Otherwise, the indicated throttle value is adopted by the notifier, and the notification rate is adjusted accordingly.

4.2. Setting the Throttle

4.2.1. Subscriber Behavior

In general, the way in which a subscriber generates SUBSCRIBE requests and processes NOTIFY requests is according to RFC 3265 [RFC3265].

A subscriber that wishes to throttle the notifications in a subscription includes a "throttle" Event header parameter in the SUBSCRIBE request, indicating in seconds the desired throttle value. The value of this parameter is an integral number of seconds in decimal.

The notifier is allowed to lower the suggested throttle interval. The adjusted throttle value will be reflected back in the Subscription-State header field of the subsequent NOTIFY requests, which the subscriber MUST take as the current, possibly adjusted throttle interval for the subscription.

There are two main consequencies for the subscriber when applying the throttle mechanism: state transitions may be lost, and event notifications may be delayed. If either of these side effects constitute a problem to the application that is to utilize event throttles, developers are instructed not to use the mechanism.

4.2.2. Notifier Behavior

In general, the way in which a notifier processes SUBSCRIBE requests and generates NOTIFY requests is according to RFC 3265 [RFC3265].
A notifier that supports the event-throttle extension extracts the value of the "throttle" Event header parameter, and uses it as the suggested minimum time allowed between two notifications. This value can be adjusted by the notifier, as defined in Section 4.3.

A compliant notifier MUST NOT generate notifications more frequent than what the throttle allows for, except when generating the notification either upon receipt of a SUBSCRIBE request (the first notification) or upon termination of the subscription (the last notification). Such notifications reset the throttle timer, even though they do not need to abide by it.

Retransmissions of NOTIFY requests are not affected by the throttle, i.e., the throttle only applies to the generation of new transactions. In other words, the throttle is reset only after the previous transaction has completed.

4.3. Selecting the Throttle Interval

Special care needs to be taken when selecting the throttle value. Using the throttle syntax it is possible to insist both very short and very long throttles to be applied to the subscription. For example, a throttle could potentially set a minimum time value between notifications that exceeds the subscription expiration value. Such a configuration would effectively quench the notifier, resulting in exactly two notifications to be generated.

The notifier is responsible for adjusting the proposed throttle value based on its local policy. The notifier MAY lower the throttle value, e.g., because of lowering the subscription expiration. The notifier MUST include the adjusted throttle value in the Subscription-State header field’s "throttle" parameter in each of the NOTIFY requests. In addition, different event packages MAY define additional constraints to the allowed throttle intervals. Such constraints are out of the scope of this specification.

4.4. Buffer Policy Description

4.4.1. Partial State Notifications

With partial notifications, the notifier will always need to keep both a copy of the current full state of the resource F, as well as the last successfully communicated full state view F’ of the resource in a specific subscription. The construction of a partial notification then involves creating a diff of the two states, and generating a notification that contains that diff.

When a throttle is applied to the subscription, it is important that
F’ is replaced with F only when the throttle is reset. Additionally, the notifier implementation SHOULD check to see that the size of an accumulated partial state notification is smaller than the full state, and if not, the notifier SHOULD send the full state notification instead.

4.4.2. Full State Notifications

With full state notifications, the notifier only needs to keep the full state of the resource, and when that changes, send the resulting notification over to the subscriber.

When a throttle is applied in the subscription, the notifier receives the state changes of the resource, and generates a notification. If there is a pending notification, the notifier simply replaces that notification with the new notification, discarding the older state.

4.5. Estimated Bandwidth Savings

It is difficult to estimate the total bandwidth savings accrued by using the throttle mechanism over a subscription, since such estimates will vary depending on the usage scenarios. However, it is easy to see that given a subscription where several full state notification would have normally been sent in any given throttle interval, a throttled subscription would only send a single notification during the same interval, yielding bandwidth savings of several times the notification size.

With partial-state notifications, drawing estimates is further complicated by the fact that the states of consecutive updates may or may not overlap. However, even in the worst case scenario, where each partial update is to a different part of the full state, a throttled notification merging all of these n partial states together should at a maximum be the size of a full-state update. In this case, the bandwidth savings are approximately n times the size of the NOTIFY header.

It is also true that there are several compression schemes available that have been designed to save bandwidth in SIP, e.g., SigComp [RFC3320] and TLS compression [RFC3943]. However, such compression schemes are complementary rather than competing mechanisms to the throttle mechanism. After all, they can both be applied simultaneously, and in such a way that the compound savings are as good as the sum of applying each one alone.
5. Syntax

This section describes the syntax extensions required for the throttle mechanism.

5.1. "throttle" Header Field Parameter

The "throttle" parameter is added to the rule definitions of the Event header field and the Subscription-State header field in the SIP Events [RFC3265] grammar. Usage of this parameter is described in section Section 4.2.

5.2. Augmented BNF Definitions

This section describes the Augmented BNF [RFC2234] definitions for the new syntax elements. Note that we derive here from the ruleset present in both SIP Events [RFC3265] and SIP [RFC3261], adding additional alternatives to the alternative sets of "event-param" and "subexp-params" defined therein.

```
event-param    =/  throttle-param
subexp-params  =/  throttle-param
throttle-param =   "throttle" EQUAL delta-seconds
```

6. IANA Considerations

This specification registers a new SIP header field parameter, defined by the following information which is to be added to the Header Field Parameters and Parameter Values sub-registry under http://www.iana.org/assignments/sip-parameters.

<table>
<thead>
<tr>
<th>Header Field</th>
<th>Parameter Name</th>
<th>Predefined</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>throttle</td>
<td>No</td>
<td>[RFCxxxx]</td>
</tr>
<tr>
<td>Subscription-State</td>
<td>throttle</td>
<td>No</td>
<td>[RFCxxxx]</td>
</tr>
</tbody>
</table>

(Note to the RFC Editor: please replace "xxxx" with the RFC number of this specification, when assigned.)

7. Security Considerations

Naturally, the security considerations listed in SIP events [RFC3265], which the throttle mechanism extends, apply in entirety. In particular, authentication and message integrity SHOULD be applied to subscriptions with the event-throttle extension.
8. Acknowledgements

Thanks to Pekka Pessi, Dean Willis, Eric Burger, Alex Audu, Alexander Milinski, Jonathan Rosenberg, Cullen Jennings and Adam Roach for support and/or review of this work.

9. References

9.1. Normative References

[I-D.ietf-simple-event-list]


9.2. Informative References

[I-D.ietf-simple-xcap]

[I-D.niemi-sip-subnot-etag]

Internet-Draft               Event Throttle                   March 2007


Author’s Address

Aki Niemi
Nokia
P.O. Box 407
NOKIA GROUP, FIN 00045
Finland

Phone: +358 50 389 1644
Email: aki.niemi@nokia.com