

SIPPING  
Internet-Draft  
Expires: April 24, 2004

J. Rosenberg  
dynamicsoft  
H. Schulzrinne  
Columbia University  
R. Mahy, Ed.  
Cisco Systems, Inc.  
October 25, 2003

**An INVITE Initiated Dialog Event Package for the Session  
Initiation Protocol (SIP)  
draft-ietf-sipping-dialog-package-03.txt**

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 24, 2004.

Copyright Notice

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

Abstract

This document defines a dialog event package for the SIP Events architecture, along with a data format used in notifications for this package. The dialog package allows users to subscribe to another user, and receive notifications about the changes in state of INVITE initiated dialogs that the user is involved in.



## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Terminology . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Dialog Event Package . . . . .	<a href="#">4</a>
<a href="#">3.1</a>	Event Package Name . . . . .	<a href="#">4</a>
<a href="#">3.2</a>	Event Package Parameters . . . . .	<a href="#">4</a>
<a href="#">3.3</a>	SUBSCRIBE Bodies . . . . .	<a href="#">5</a>
<a href="#">3.4</a>	Subscription Duration . . . . .	<a href="#">6</a>
<a href="#">3.5</a>	NOTIFY Bodies . . . . .	<a href="#">6</a>
<a href="#">3.6</a>	Notifier Processing of SUBSCRIBE Requests . . . . .	<a href="#">6</a>
<a href="#">3.7</a>	Notifier Generation of NOTIFY Requests . . . . .	<a href="#">7</a>
<a href="#">3.7.1</a>	The Dialog State Machine . . . . .	<a href="#">8</a>
<a href="#">3.7.2</a>	Applying the state machine . . . . .	<a href="#">10</a>
<a href="#">3.8</a>	Subscriber Processing of NOTIFY Requests . . . . .	<a href="#">12</a>
<a href="#">3.9</a>	Handling of Forked Requests . . . . .	<a href="#">12</a>
<a href="#">3.10</a>	Rate of Notifications . . . . .	<a href="#">12</a>
<a href="#">3.11</a>	State Agents . . . . .	<a href="#">13</a>
<a href="#">4.</a>	Dialog Information Format . . . . .	<a href="#">13</a>
<a href="#">4.1</a>	Structure of Dialog Information . . . . .	<a href="#">13</a>
<a href="#">4.1.1</a>	Dialog Element . . . . .	<a href="#">14</a>
<a href="#">4.1.2</a>	State . . . . .	<a href="#">15</a>
<a href="#">4.1.3</a>	Duration . . . . .	<a href="#">15</a>
<a href="#">4.1.4</a>	Replaces . . . . .	<a href="#">15</a>
<a href="#">4.1.5</a>	Referred-By . . . . .	<a href="#">16</a>
<a href="#">4.1.6</a>	Route-Set . . . . .	<a href="#">16</a>
<a href="#">4.1.6.1</a>	Local and Remote elements . . . . .	<a href="#">16</a>
<a href="#">4.1.6.1.1</a>	Identity . . . . .	<a href="#">16</a>
<a href="#">4.1.6.1.2</a>	Target . . . . .	<a href="#">17</a>
<a href="#">4.1.6.1.3</a>	Session Description . . . . .	<a href="#">17</a>
<a href="#">4.1.6.1.4</a>	CSeq . . . . .	<a href="#">17</a>
<a href="#">4.2</a>	Constructing Coherent State . . . . .	<a href="#">17</a>
<a href="#">4.3</a>	Schema . . . . .	<a href="#">18</a>
<a href="#">4.4</a>	Example . . . . .	<a href="#">21</a>
<a href="#">5.</a>	Security Considerations . . . . .	<a href="#">25</a>
<a href="#">6.</a>	IANA Considerations . . . . .	<a href="#">25</a>
<a href="#">6.1</a>	application/dialog-info+xml MIME Registration . . . . .	<a href="#">25</a>
<a href="#">6.2</a>	URN Sub-Namespace Registration for urn:ietf:params:xml:ns:dialog-info . . . . .	<a href="#">26</a>
<a href="#">6.3</a>	Schema Registration . . . . .	<a href="#">26</a>
<a href="#">7.</a>	Acknowledgements . . . . .	<a href="#">27</a>
	Normative References . . . . .	<a href="#">27</a>
	Informative References . . . . .	<a href="#">28</a>
	Authors' Addresses . . . . .	<a href="#">28</a>
	Intellectual Property and Copyright Statements . . . . .	<a href="#">30</a>



## 1. Introduction

The SIP Events framework [1] defines general mechanisms for subscription to, and notification of, events within SIP networks. It introduces the notion of a package, which is a specific "instantiation" of the events mechanism for a well-defined set of events. Packages have been defined for user presence [14], watcher information [15], and message waiting indicators [16], amongst others. Here, we define an event package for INVITE initiated dialogs. Dialogs refer to the SIP relationship established between two SIP peers [2]. Dialogs can be created by many methods, although RFC 3261 defines only one - the INVITE method. RFC 3265 defines the SUBSCRIBE and NOTIFY methods, which also create dialogs. However, the usage of this package to model transitions in the state of those dialogs is out of the scope of this specification.

There are a variety of applications enabled through the knowledge of INVITE dialog state. Some examples include:

**Automatic Callback:** In this basic Public Switched Telephone Network (PSTN) application, user A calls user B. User B is busy. User A would like to get a callback when user B hangs up. When B hangs up, user A's phone rings. When A picks it up, they here ringing, and are being connected to B. To implement this with SIP, a mechanism is required for B to receive a notification when the dialogs at A are complete.

**Presence-Enabled Conferencing:** In this application, a user A wishes to set up a conference call with users B and C. Rather than scheduling it, it is to be created automatically when A, B and C are all available. To do this, the server providing the application would like to know whether A, B and C are "online", not idle, and not in a phone call. Determining whether or not A, B and C are in calls can be done in two ways. In the first, the server acts as a call stateful proxy for users A, B and C, and therefore knows their call state. This won't always be possible, however, and it introduces scalability, reliability, and operational complexities. Rather, the server would subscribe to the dialog state of those users, and receive notifications as it changes. This enables the application to be provided in a distributed way; the server need not reside in the same domain as the users.

**IM Conference Alerts:** In this application, a user can get an IM sent to their phone whenever someone joins a conference that the phone is involved in. The IM alerts are generated by an application separate from the conference server.



In general, the dialog package allows for construction of distributed applications, where the application requires information on dialog state, but is not co-resident with the end user on which that state resides.

## **2. 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](#) [9] and indicate requirement levels for compliant implementations.

## **3. Dialog Event Package**

This section provides the details for defining a SIP Events package, as specified by [1].

### **3.1 Event Package Name**

The name of this event package is "dialog". This package name is carried in the Event and Allow-Events header, as defined in [1].

### **3.2 Event Package Parameters**

This package defines four Event Package parameters. They are call-id, to-tag, from-tag, and include-session-description. If a subscription to a specific dialog is requested, all of the first three of these parameters MUST be present. They identify the dialog that is being subscribed to. The to-tag is matched against the local tag, the from-tag is matched against the remote tag, and the call-id is matched against the Call-ID. The include-session-description parameter indicates if the subscriber would like to receive the session descriptions associated with the subscribed dialog or dialogs.

It is also possible to subscribe to the set of dialogs created as a result of a single INVITE sent by a UAC. In that case, the call-id and to-tag MUST be present. The to-tag is matched against the local tag, and the call-id is matched against the Call-ID.

The ABNF for these parameters is shown below. It refers to many constructions from the ABNF of [RFC3261](#), such as word, callid, EQUAL, DQUOTE, and token.

```
call-id      = "call-id" EQUAL ( token / DQUOTE callid DQUOTE )  
              ;; NOTE: any DQUOTES inside callid MUST be escaped!  
from-tag     = "from-tag" EQUAL token  
to-tag       = "to-tag" EQUAL token
```



with-sessd = "include-session-description"

Any callids which contain embedded double-quotes MUST escape those double-quotes using the backslash-quoting mechanism. Note that the call-id parameter may need to be expressed as a quoted string. This is because the ABNF for callid and word (which is used by callid) allow for some characters (such as "@", "[", and ":") which are not allowed within a token.

### **3.3 SUBSCRIBE Bodies**

A SUBSCRIBE for a dialog package MAY contain a body. This body defines a filter to apply to the subscription. Filter documents are not specified in this document, and at the time of writing, are expected to be the subject of future standardization activity.

A SUBSCRIBE for a dialog package MAY be sent without a body. This implies the default subscription filtering policy. The default policy is:

- o If the Event header field contained dialog identifiers, notifications are generated every time there is a change in the state of any matching dialogs for the user identified in the request URI of the SUBSCRIBE.
- o If there were no dialog identifiers in the Event header field, notifications are generated every time there is any change in the state of any dialogs for the user identified in the request URI of the SUBSCRIBE with the following exceptions. If the target (Contact) URI of a subscriber is equivalent to the remote target URI of a specific dialog, then the dialog element for that dialog is suppressed for that subscriber. (The subscriber is already a party in the dialog directly, so these notifications are superfluous.) If no dialogs remain after suppressing dialogs, the entire notification to that subscriber is suppressed and the version number in the dialog-info element is not incremented for that subscriber. Implicit filtering for one subscriber does not affect notifications to other subscribers.
- o Notifications do not normally contain full state; rather, they only indicate the state of the dialog whose state has changed. The exception is a NOTIFY sent in response to a SUBSCRIBE. These NOTIFYS contain the complete view of dialog state.
- o The notifications contain the identities of the participants in the dialog, and the dialog identifiers. Additional information, such as the route set, CSeq numbers, SDP information, and so on, are not included normally unless explicitly requested and/or



explicitly authorized.

### **3.4 Subscription Duration**

Dialog state changes fairly quickly; once established, a typical phone call lasts a few minutes (this is different for other session types, of course). However, the interval between new calls is typically infrequent. As such, we arbitrarily choose a default duration of one hour. Clients SHOULD specify an explicit duration.

There are two distinct use cases for dialog state. The first is when a subscriber is interested in the state of a specific dialog or dialogs (and they are authorized to find out about just the state of those dialogs). In that case, when the dialogs terminate, so too does the subscription. In these cases, the value of the subscription duration is largely irrelevant, and SHOULD be longer than the typical duration of a dialog, about two hours would cover most dialogs.

In another case, a subscriber is interested in the state of all dialogs for a specific user. In these cases, a shorter interval makes more sense. The default is one hour for these subscriptions.

### **3.5 NOTIFY Bodies**

As described in [RFC 3265](#) [1], the NOTIFY message will contain bodies that describe the state of the subscribed resource. This body is in a format listed in the Accept header field of the SUBSCRIBE, or a package-specific default if the Accept header field was omitted from the SUBSCRIBE.

In this event package, the body of the notification contains a dialog information document. This document describes the state of one or more dialogs associated with the subscribed resource. All subscribers and notifiers MUST support the "application/dialog-info+xml" data format described in [Section 4](#). The subscribe request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/dialog-info+xml". If the header field is present, it MUST include "application/dialog-info+xml", and MAY include any other types capable of representing dialog state.

Of course, the notifications generated by the server MUST be in one of the formats specified in the Accept header field in the SUBSCRIBE request.

### **3.6 Notifier Processing of SUBSCRIBE Requests**

The dialog information for a user contains sensitive information.



Therefore, all subscriptions SHOULD be authenticated and then authorized before approval. All implementors of this package MUST support the digest authentication mechanism as a baseline. Authorization policy is at the discretion of the administrator, as always. However, a few recommendations can be made.

It is RECOMMENDED that, if the policy of user B is that user A is allowed to call them, dialog subscriptions from user A be allowed. However, the information provided in the notifications does not contain any dialog identification information; merely an indication of whether the user is in at least one call, or not. Specifically, they should not be able to find out any more information than if they sent an INVITE. (This concept of a "virtual" dialog is discussed more in [Section 3.7.2](#), and an example of such a notification body is shown below.)

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  version="0" state="full"
  entity="sip:alice@example.com">
  <dialog id="as7d900as8">
    <state>confirmed</state>
  </dialog>
</dialog-info>
```

It is RECOMMENDED that if a user agent registers with the address-of-record X, that this user agent authorize subscriptions that come from any entity that can authenticate itself as X. Complete information on the dialog state SHOULD be sent in this case. This authorization behavior allows a group of devices representing a single user to all become aware of each other's state. This is useful for applications such as single-line-extension.

Note that many implementations of "shared-lines" have a feature which allows details of calls on a shared address-of-record to be made private. This is a completely reasonable authorization policy which could result in notifications which contain only the id attribute of the dialog element and the state element when shared-line privacy is requested, and notifications with more complete information when shared-line privacy is not requested.

### **[3.7](#) Notifier Generation of NOTIFY Requests**

Notifications are generated for the dialog package when an INVITE request is sent, when a new dialog comes into existence at a UA, or when the state or characteristics of an existing dialog changes. Therefore, a model of dialog state is needed in order to determine precisely when to send notifications, and what their content should



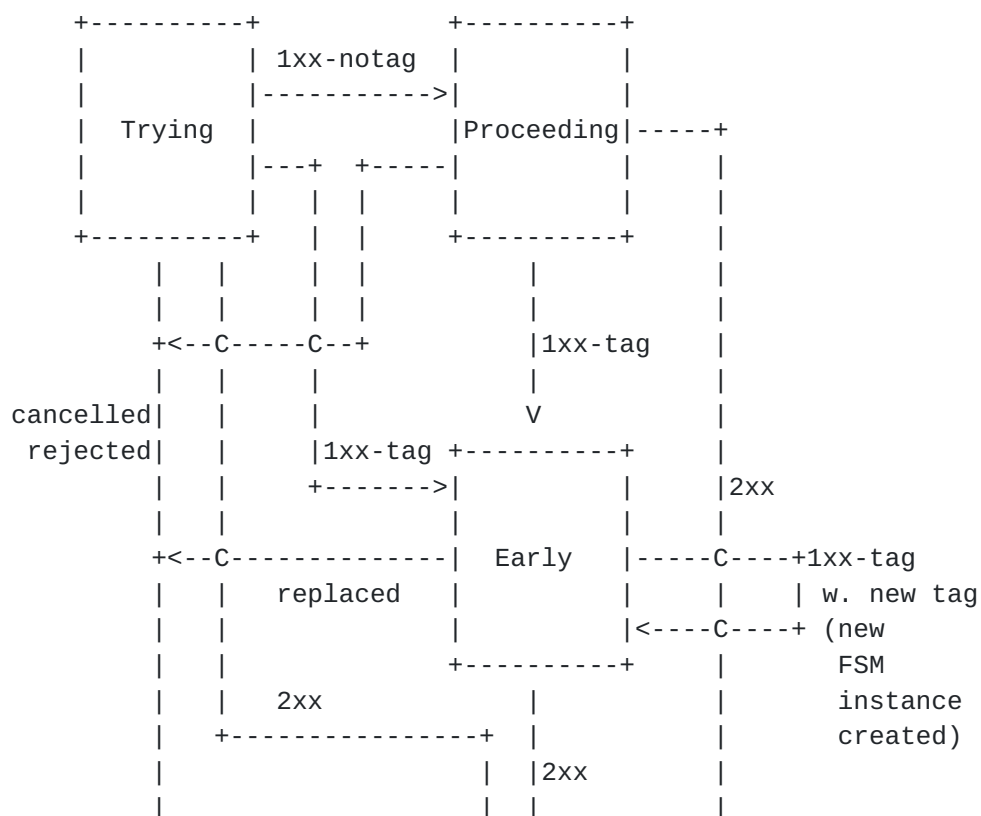
be. The SIP specification has a reasonably well defined lifecycle for dialogs. However, it is not explicitly modelled. This specification provides an explicit model of dialog state through a finite state machine.

It is RECOMMENDED that NOTIFY requests only contain information on the dialogs whose state or participation information has changed. However, if a notifier receives a SUBSCRIBE request, the triggered NOTIFY SHOULD contain the state of all dialogs that the subscriber is authorized to see.

### 3.7.1 The Dialog State Machine

Modelling of dialog state is complicated by two factors. The first is forking, which can cause a single INVITE to generate many dialogs at a UAC. The second is the differing views of state at the UAC and UAS. We have chosen to handle the first issue by extending the dialog FSM to include the states between transmission of the INVITE and the creation of actual dialogs through receipt of 1xx and 2xx responses. As a result, this specification supports the notion of dialog state for dialogs before they are fully instantiated.

We have also chosen to use a single FSM for both UAC and UAS.





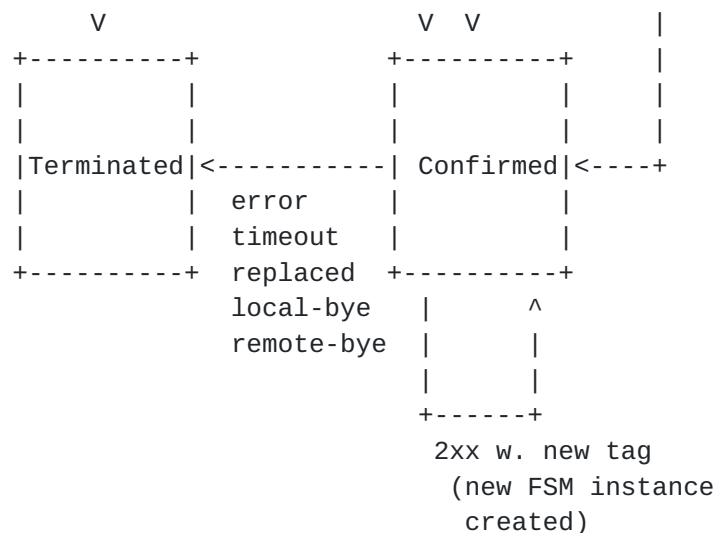


Figure 3

The FSM for dialog state is shown in Figure 3. The FSM is best understood by considering the UAC and UAS cases separately.

The FSM is created in the "trying" state when the UAC sends an INVITE request. Upon receipt of a 1xx without a tag, the FSM transitions to the "proceeding" state. Note that there is no actual dialog yet, as defined by the SIP specification. However, there is a "half-dialog", in the sense that two of the three components of the dialog ID are known (the call identifier and local tag). If a 1xx with a tag is received, the FSM transitions to the early state. The full dialog identifier is now defined. Had a 2xx been received, the FSM would have transitioned to the "confirmed" state.

If, after transitioning to the "early" or "confirmed" states, the UAC receives another 1xx or 2xx respectively with a different tag, another instance of the FSM is created, initialized into the "early" or "confirmed" state respectively. The benefit of this approach is that there will be a single FSM representing the entire state of the invitation and resulting dialog when dealing with the common case of no forking.

If the UAC should send a CANCEL, and then subsequently receive a 487 to its INVITE transaction, all FSMs spawned from that INVITE transition to the "terminated" state with the event "cancelled". If the UAC receives a new invitation (with a Replaces [13] header) which replaces the current Early or Confirmed dialog, all INVITE transactions spawned from the replaced invitation transition to the "terminated" state with the event "replaced". If the INVITE transaction terminates with a non-2xx response for any other reason, all FSMs spawned from that INVITE transition to the terminated state



with the event "rejected".

Once in the confirmed state, the call is active. It can transition to the terminated state if the UAC sends a BYE or receives a BYE (corresponding to the "local-bye" and "remote-bye" events as appropriate), if a mid-dialog request generates a 481 or 408 response (corresponding to the "error" event), or a mid-dialog request generates no response (corresponding to the "timeout" event).

From the perspective of the UAS, when an INVITE is received, the FSM is created in the "trying" state. If it sends a 1xx without a tag, the FSM transitions to the "proceeding" state. If a 1xx is sent with a tag, the FSM transitions to the "early" state, and if a 2xx is sent, it transitions to the "confirmed" state. If the UAS should receive a CANCEL request and then generate a 487 response to the INVITE (which can occur in the proceeding and early states), the FSM transitions to the terminated state with the event "cancelled". If the UAS should generate any other non-2xx final response to the INVITE request, the FSM transitions to the terminated state with the event "rejected". If the UAS receives a new invitation (with a Replaces [[13](#)] header) which replaces the current Confirmed dialog, the replaced invitation transition transitions to the "terminated" state with the event "replaced". Once in the "confirmed" state, the other transitions to the "terminated" state occur for the same reasons they do in the case of UAC.

There should never be a transition from the "trying" state to the "terminated" state with the event "cancelled", since the SIP specification prohibits transmission of CANCEL until a provisional response is received. However, this transition is defined in the FSM just to unify the transitions from trying, proceeding, and early to the terminated state.

OPEN ISSUE: Since there is only one possible event to cause transitions to the Proceeding (1xx-notag), Early (1xx-tag), and Confirmed (2xx) states, the only events which provide any additional information are the events for transitions to Terminated (error, timeout, cancelled, local-bye, remote-bye and replaced). Of these, timeout may not be relevant, since it is often indistinguishable from "rejected" (for example, a 408) or "error". Likewise it is unclear if there is any value in distinguishing "local-bye" from "cancelled"; perhaps we should use a single event, such as "local-hangup" instead.

### **3.7.2 Applying the state machine**

The notifier MAY generate a NOTIFY request on any event transition of



the FSM. Whether it does or not is policy dependent. However, some general guidelines are provided.

When the subscriber is unauthenticated, or is authenticated, but represents a third party with no specific authorization policies, it is RECOMMENDED that subscriptions to an individual dialog, or to a specific set of dialogs, is forbidden. Only subscriptions to all dialogs (i.e., there are no dialog identifiers in the Event header field) are permitted. In that case, actual dialog states across all dialogs will not be reported. Rather, a single "virtual" dialog FSM be used, and event transitions on that FSM be reported.

If there is any dialog at the UA whose state is "confirmed", the virtual FSM is in the "confirmed" state. If there are no dialogs at the UA in the confirmed state, but there is at least one in the "early" state, the virtual FSM is in the "early" or "confirmed" state. If there are no dialogs in the confirmed or early states, but there is at least one in the "proceeding" state, the virtual FSM is in the "proceeding", "early" or "confirmed" state. If there are no dialogs in the confirmed, early, or proceeding states, but there is at least one in the "trying" state, the virtual FSM is in the "trying", "proceeding", "early" or "confirmed" state. The choice about which state to use depends on whether the UA wishes to let unknown users that their phone is ringing, as opposed to in an active call.

It is RECOMMENDED that, in the absence of any preference, "confirmed" is used in all cases (as shown in the example in [Section 3.6](#)). Furthermore, it is RECOMMENDED that the notifications of changes in the virtual FSM machine not convey any information except the state of the FSM and its event transitions - no dialog identifiers (which are ill-defined in this model in any case). The use of this virtual FSM allows for minimal information to be conveyed. A subscriber cannot know how many calls are in progress, or with whom, just that there exists a call. This is the same information they would receive if they simply sent an INVITE to the user instead; a 486 response would indicate that they are on a call.

When the subscriber is authenticated, and has authenticated itself with the same address-of-record that the UA itself uses, if no explicit authorization policy is defined, it is RECOMMENDED that all state transitions on dialogs that have been subscribed to (which is either all of them, if no dialog identifiers were present in the Event header field, or a specific set of them identified by the Event header field parameters) be reported, along with complete dialog IDs.

The notifier MAY generate a NOTIFY request on any change in the characteristics associated with the dialog. Since these include CSeq



numbers and SDP, receipt of re-INVITES and UPDATE requests [3] which modify this information MAY trigger notifications.

OPEN ISSUE: Is there a good reason to include CSeqs at all? Can anyone come up with a use case? This seems to contradict the "Rate of Notifications" section, and I can come up with some good examples where this would be VERY BAD. For example, say Alice sends an invitation to Bob, and then, on the same dialog, subscribes to his dialog package, requesting CSeq information. Every notification updates the CSeq which in turn generates another notification, causing an infinite flood of messages.

### **3.8 Subscriber Processing of NOTIFY Requests**

The SIP Events framework expects packages to specify how a subscriber processes NOTIFY requests in any package specific ways, and in particular, how it uses the NOTIFY requests to construct a coherent view of the state of the subscribed resource.

Typically, the NOTIFY for the dialog package will only contain information about those dialogs whose state has changed. To construct a coherent view of the total state of all dialogs, a subscriber to the dialog package will need to combine NOTIFYS received over time.

Notifications within this package can convey partial information; that is, they can indicate information about a subset of the state associated with the subscription. This means that an explicit algorithm needs to be defined in order to construct coherent and consistent state. The details of this mechanism are specific to the particular document type. See [Section 4.2](#) for information on constructing coherent information from an application/dialog-info+xml document.

### **3.9 Handling of Forked Requests**

Since dialog state is distributed across the UA for a particular user, it is reasonable and useful for a SUBSCRIBE request for dialog state to fork, and reach multiple UA.

As a result, a forked SUBSCRIBE request for dialog state can install multiple subscriptions. Subscribers to this package MUST be prepared to install subscription state for each NOTIFY generated as a result of a single SUBSCRIBE.

### **3.10 Rate of Notifications**

For reasons of congestion control, it is important that the rate of



notifications not become excessive. As a result, it is RECOMMENDED that the server not generate notifications for a single subscriber at a rate faster than once every 5 seconds.

Editors Note: This seems too slow to me. I think 1 second is probably reasonable.

### **3.11 State Agents**

Dialog state is ideally maintained in the user agents in which the dialog resides. Therefore, the elements that maintain the dialog are the ones best suited to handle subscriptions to it. However, in some cases, a network agent may also know the state of the dialogs held by a user. As such, state agents MAY be used with this package.

## **4. Dialog Information Format**

Dialog information is an XML document [4] that MUST be well-formed and SHOULD be valid. Dialog information documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying dialog information documents and document fragments. The namespace URI for elements defined by this specification is a URN [5], using the namespace identifier 'ietf' defined by [6] and extended by [7]. This URN is:

urn:ietf:params:xml:ns:dialog-info

A dialog information document begins with the root element tag "dialog-info".

### **4.1 Structure of Dialog Information**

A dialog information document starts with a dialog-info element. This element has three mandatory attributes:

version: This attribute allows the recipient of dialog information documents to properly order them. Versions start at 0, and increment by one for each new document sent to a subscriber. Versions are scoped within a subscription. Versions MUST be representable using a 32 bit integer.

state: This attribute indicates whether the document contains the full dialog information, or whether it contains only information on those dialogs which have changed since the previous document (partial).



entity: This attribute contains a URI that identifies the user whose dialog information is reported in the remainder of the document. This user is referred to as the "observed user".

The dialog-info element has a series of zero or more dialog sub-elements. Each of those represents a specific dialog.

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
              version="0" state="full"
              entity="sip:alice@example.com">
</dialog-info>
```

#### **4.1.1 Dialog Element**

The dialog element reports information on a specific dialog or "half-dialog". It has a single mandatory attribute, id. The id attribute provides a single string that can be used as an identifier for this dialog or "half-dialog". This is a different identifier than the dialog ID defined in [RFC 3261](#) [2], but related to it.

For a caller, the id is created when an INVITE request is sent. When a 1xx with a tag, or a 2xx is received, the dialog is formally created. The id remains unchanged. However, if an additional 1xx or 2xx is received, resulting in the creation of another dialog (and resulting FSM), that dialog is allocated a new id.

For a callee, the id is created when an INVITE outside of an existing dialog is received. When a 2xx or a 1xx with a tag is sent, creating the dialog, the id remains unchanged.

The id MUST be unique amongst all dialogs at a UA.

There are a number of optional attributes which provide identification information about the dialog:

call-id: This attribute is a string which represents the call-id component of the dialog identifier. (Note that single and double quotes inside a call-id must be escaped using &quote; for " and &apos; for ' .)

OPEN ISSUE: Is it legal to include escaped quotes in XML attributes?

local-tag: This attribute is a string which represents the local-tag component of the dialog identifier.



remote-tag: This attribute is a string which represents the remote-tag component of the dialog identifier. The remote tag attribute won't be present if there is only a "half-dialog", resulting from the generation of an INVITE for which no final responses or provisional responses with tags has been received.

direction: This attribute is either initiator or recipient, and indicates whether the observed user was the initiator of the dialog, or the recipient of the INVITE that created it.

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
              version="0" state="partial"
              entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" direction="initiator">
    ...
  </dialog>
</dialog-info>
```

The sub-elements of the dialog element provide additional information about the dialog. Some of these sub-elements provide more detail about the dialog itself, while the local and remote sub-elements describe characteristics of the participants involved in the dialog. The only mandatory sub-element is the state element.

#### [4.1.1.2](#) State

The state element indicates the state of the dialog. Its value is an enumerated type describing one of the states in the FSM above. It has an optional event attribute that can be used to indicate the event which caused any transition into the terminated state, and an optional code attribute that indicates the response code associated with any transition caused by a response to the original INVITE.

```
<state event="rejected" code="486">terminated</state>
```

#### [4.1.1.3](#) Duration

The duration element contains the amount of time, in seconds, since the FSM was created.

```
<duration>145</duration>
```

#### [4.1.1.4](#) Replaces

The replaces element is used to correlate a new dialog with one it replaced as a result of an invitation with a Replaces header. This element is present in the replacement dialog only (the newer dialog)



and contains attributes with the call-id, local-tag, and remote-tag of the replaced dialog.

```
<replaces call-id="hg287s98s89" local-tag="6762h7" remote-tag="09278hsb"/>
```

#### [4.1.1.5](#) Referred-By

The referred-by element is used to correlate a new dialog with a REFER [\[12\]](#) request which triggered it. The element is present in a dialog which was triggered by a REFER request which contained a Referred-By [\[11\]](#) header and contains the (optional) display name attribute and the Referred-By URI as its value.

```
<referred-by display="Bob">sip:bob@example.com</referred-by>
```

#### [4.1.1.6](#) Route-Set

The route-set element conveys an ordered list of hop elements which represents the complete route set of the dialog (not including the local and remote target URIs) from the perspective of the notifier.

OPEN ISSUE: Does any one want/need this?

```
<route-set>
  <hop>sip:proxy1.example.net;lr</hop>
  <hop>sip:proxy2.example.com;lr</hop>
</route-set>
```

##### [4.1.1.6.1](#) Local and Remote elements

The local and remote elements are sub-elements of the dialog element which contain information about the local and remote participants respectively. They both have a number of optional sub-elements which indicate the identity conveyed by the participant, the target URI, the feature-tags of the target, and the session-description of the participant.

##### [4.1.1.6.1.1](#) Identity

The identity element indicates a local or remote URI, as defined in [\[2\]](#) as appropriate. It has an optional attribute, display-name, that contains the display name from the appropriate URI.

Note that multiple identities (for example a sip: URI and a tel: URI) could be included if they all correspond to the participant.

```
<identity display="Anonymous">sip:anonymous@anonymous.invalid</identity>
```



#### [4.1.6.1.2](#) Target

The target contains the local or remote target URI as constructed by the user agent for this dialog, as defined in [RFC 3261](#) [2] in a "uri" attribute.

It can contain a list of Contact header parameters in param sub-elements (such as those defined in [10]). The param element contains a required pname attribute and an optional pval attribute (some parameters merely exist and have no explicit value). The param element itself has no contents.

```
<target uri="sip:alice@pc33.example.com">
  <param pname="isfocus"/>
  <param pname="class" pval="personal"/>
</target>
```

#### [4.1.6.1.3](#) Session Description

The session-description element contains the session description used by the observed user for its end of the dialog. This element should generally NOT be included in the notifications, unless explicitly requested by the subscriber. It has a single attribute, type, which indicates the MIME media type of the session description.

#### [4.1.6.1.4](#) CSeq

The cseq element contains the most recent value of the CSeq header used by the UA in an outgoing request on the dialog. This element should generally NOT be included in the notifications, unless explicitly requested by the subscriber. If no CSeq has yet been defined, the value of the element is -1.

OPEN ISSUE: Is this really useful?

### [4.2](#) Constructing Coherent State

The dialog information subscriber maintains a table for the list of dialogs. The table contains a row for each dialog. Each row is indexed by an ID, present in the "id" attribute of the "dialog" element. The contents of each row contain the state of that dialog as conveyed in the document. The table is also associated with a version number. The version number MUST be initialized with the value of the "version" attribute from the "dialog-info" element in the first document received. Each time a new document is received, the value of the local version number, and the "version" attribute in the new document, are compared. If the value in the new document is one higher than the local version number, the local version number is



increased by one, and the document is processed. If the value in the document is more than one higher than the local version number, the local version number is set to the value in the new document, and the document is processed. If the document did not contain full state, the subscriber SHOULD generate a refresh request to trigger a full state notification. If the value in the document is less than the local version, the document is discarded without processing.

The processing of the dialog information document depends on whether it contains full or partial state. If it contains full state, indicated by the value of the "state" attribute in the "dialog-info" element, the contents of the table are flushed. They are repopulated from the document. A new row in the table is created for each "dialog" element. If the document contains partial state, as indicated by the value of the "state" attribute in the "dialog-info" element, the document is used to update the table. For each "dialog" element in the document, the subscriber checks to see whether a row exists for that dialog. This check is done by comparing the ID in the "id" attribute of the "dialog" element with the ID associated with the row. If the dialog doesn't exist in the table, a row is added, and its state is set to the information from that "dialog" element. If the dialog does exist, its state is updated to be the information from that "dialog" element. If a row is updated or created, such that its state is now terminated, that entry MAY be removed from the table at any time.

### 4.3 Schema

The following is the schema for the application/dialog-info+xml type:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:ietf:params:xml:ns:dialog-info"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="urn:ietf:params:xml:ns:dialog-info"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <!-- This import brings in the XML language attribute xml:lang-->
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
  <xs:element name="dialog-info">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:dialog" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



```
</xs:sequence>
<xs:attribute name="version" type="xs:nonNegativeInteger"
  use="required"/>
<xs:attribute name="state" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="full"/>
      <xs:enumeration value="partial"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="entity" type="xs:anyURI" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="dialog">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tns:state" minOccurs="1" maxOccurs="1"/>
      <xs:element name="duration" type="xs:nonNegativeInteger"
        minOccurs="0" maxOccurs="1"/>
      <xs:element name="replaces" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:attribute name="call-id" type="xs:string"
            use="required"/>
          <xs:attribute name="local-tag" type="xs:string"
            use="required"/>
          <xs:attribute name="remote-tag" type="xs:string"
            use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="referred-by" type="tns:nameaddr"
        minOccurs="0" maxOccurs="1"/>
      <xs:element name="route-set" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="hop" type="xs:string" minOccurs="1"
              maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="local" type="tns:participant"
        minOccurs="0" maxOccurs="1"/>
      <xs:element name="remote" type="tns:participant"
        minOccurs="0" maxOccurs="1"/>
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```



```
<xs:attribute name="call-id" type="xs:string"
  use="optional"/>
<xs:attribute name="local-tag" type="xs:string"
  use="optional"/>
<xs:attribute name="remote-tag" type="xs:string"
  use="optional"/>
<xs:attribute name="direction" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="initiator"/>
      <xs:enumeration value="recipient"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:complexType name="participant">
  <xs:sequence>
    <xs:element name="identity" type="nameaddr"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="target" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="param" minOccurs="0"
            maxOccurs="unbounded">
            <xs:complexType>
              <xs:attribute name="pname" type="xs:string"
                use="required"/>
              <xs:attribute name="pval" type="xs:string"
                use="optional"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:attribute name="uri" type="xs:string" use="required"/>
  </xs:sequence>
  <xs:element name="session-description" type="tns:sessd"
    minOccurs="0" maxOccurs="1"/>
  <xs:element name="cseq" type="xs:nonNegativeInteger"
    minOccurs="0" maxOccurs="1"/>
  <xs:any namespace="##other" processContents="lax"
    minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="nameaddr">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="display-name" type="xs:string"
```



```
        use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="sessd">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="state">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="event" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="cancelled"/>
                <xs:enumeration value="rejected"/>
                <xs:enumeration value="replaced"/>
                <xs:enumeration value="local-bye"/>
                <xs:enumeration value="remote-bye"/>
                <xs:enumeration value="error"/>
                <xs:enumeration value="timeout"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="code" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:positiveInteger">
                <xs:minInclusive value="100"/>
                <xs:maxInclusive value="699"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

#### 4.4 Example

For example, if a UAC sends an INVITE that looks like, in part:



```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@example.com>
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.example.com>
Content-Type: application/sdp
Content-Length: 142
```

[SDP not shown]

The XML document in a notification from Alice might look like:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  version="0"
  state="full"
  entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
    local-tag="1928301774" direction="initiator">
    <state>trying</state>
  </dialog>
</dialog-info>
```

If the following 180 response is received:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@example.com>;tag=456887766
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@host.example.com>
```

The XML document in a notification might look like:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  version="1"
  state="full"
  entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
    local-tag="1928301774" remote-tag="456887766">
```



```
        direction="initiator">
    <state>early</state>
</dialog>
</dialog-info>
```

If it receives a second 180 with a different tag:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@example.com>;tag=hh76a
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:jack@host.example.com>
```

This results in the creation of a second dialog:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
    version="2"
    state="full"
    entity="sip:alice@example.com">
    <dialog id="as7d900as8" call-id="a84b4c76e66710"
        local-tag="1928301774" remote-tag="456887766"
        direction="initiator">
        <state>early</state>
    </dialog>
    <dialog id="as7d900as8" call-id="a84b4c76e66710"
        local-tag="1928301774" remote-tag="hh76a"
        direction="initiator">
        <state>early</state>
    </dialog>
</dialog-info>
```

If a 200 OK is received on the second dialog, it moves to confirmed:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
    version="3"
    state="partial"
    entity="sip:alice@example.com">
    <dialog id="as7d900as8" call-id="a84b4c76e66710"
        local-tag="1928301774" remote-tag="hh76a"
        direction="initiator">
        <state>confirmed</state>
```



```
</dialog>
</dialog-info>
```

32 seconds later, the other early dialog terminates because no 2xx is received for it. This implies that it was successfully cancelled, and therefore the following notification is sent:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  version="4"
  state="partial"
  entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
    local-tag="1928301774" remote-tag="hh76a"
    direction="initiator">
    <state event="cancelled">terminated</state>
  </dialog>
</dialog-info>
```

EDITORS NOTE: should provide another example with a richer notification

```
<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:dialog-info"
  version="1" state="full">
  <dialog id="123456">
    <state>confirmed</state>
    <duration>274</duration>
    <local>
      <identity display="Alice">sip:alice@example.com</identity>
      <target uri="sip:alice@pc33.example.com">
        <param pname="isfocus"/>
        <param pname="class" pval="personal"/>
      </target>
    </local>
    <remote>
      <identity display="Bob">sip:bob@example.org</identity>
      <target uri="sip:bobster@phone21.example.org"/>
    </remote>
  </dialog>
</dialog-info>
```



## **5. Security Considerations**

Subscriptions to dialog state can reveal sensitive information. For this reason, [Section 3.6](#) discusses authentication and authorization of subscriptions, and provides guidelines on sensible authorization policies. All implementations of this package MUST support the digest authentication mechanism.

Since the data in notifications is sensitive as well, end-to-end SIP encryption mechanisms using S/MIME MAY be used to protect it.

## **6. IANA Considerations**

This document registers a new MIME type, application/dialog-info+xml and registers a new XML namespace.

### **6.1 application/dialog-info+xml MIME Registration**

MIME media type name: application

MIME subtype name: dialog-info+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in [RFC 3023](#) [8].

Encoding considerations: Same as encoding considerations of application/xml as specified in [RFC 3023](#) [8].

Security considerations: See [Section 10 of RFC 3023](#) [8] and [Section 5](#) of this specification.

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: This document type has been used to support SIP applications such as call return and auto-conference.

Additional Information:

Magic Number: None

File Extension: .dif or .xml



Macintosh file type code: "TEXT"

Personal and email address for further information: Jonathan Rosenberg, <jdrosen@jdrosen.net>

Intended usage: COMMON

Author/Change controller: The IETF.

## **6.2 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:dialog-info**

This section registers a new XML namespace, as per the guidelines in [7].

URI: The URI for this namespace is  
urn:ietf:params:xml:ns:dialog-info.

Registrant Contact: IETF, SIPING working group, <sipping@ietf.org>, Jonathan Rosenberg <jdrosen@jdrosen.net>.

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml1-basic/xhtml1-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1"/>
  <title>Dialog Information Namespace</title>
</head>
<body>
  <h1>Namespace for Dialog Information</h1>
  <h2>urn:ietf:params:xml:ns:dialog-info</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

## **6.3 Schema Registration**

This specification registers a schema, as per the guidelines in in [7].



URI: please assign.

Registrant Contact: IETF, SIPING Working Group  
(sipping@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).

XML: The XML can be found as the sole content of [Section 4.3](#).

## 7. Acknowledgements

The authors would like to thank Sean Olson for his comments.

## Normative References

- [1] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", [RFC 3265](#), June 2002.
- [2] 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.
- [3] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", [RFC 3311](#), October 2002.
- [4] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C REC REC-xml-20001006, October 2000.
- [5] Moats, R., "URN Syntax", [RFC 2141](#), May 1997.
- [6] Moats, R., "A URN Namespace for IETF Documents", [RFC 2648](#), August 1999.
- [7] Mealling, M., "The IETF XML Registry", [draft-mealling-iana-xmlns-registry-04](#) (work in progress), July 2002.
- [8] Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types", [RFC 3023](#), January 2001.
- [9] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [10] Rosenberg, J., "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", [draft-ietf-sip-callee-caps-00](#) (work in progress), June 2003.
- [11] Sparks, R., "The SIP Referred-By Mechanism",



[draft-ietf-sip-referredby-01](#) (work in progress), February 2003.

- [12] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", [RFC 3515](#), April 2003.
- [13] Dean, R., Biggs, B. and R. Mahy, "The Session Initiation Protocol (SIP) 'Replaces' Header", [draft-ietf-sip-replaces-03](#) (work in progress), March 2003.

#### Informative References

- [14] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", [draft-ietf-simple-presence-10](#) (work in progress), January 2003.
- [15] Rosenberg, J., "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)", [draft-ietf-simple-wininfo-package-05](#) (work in progress), January 2003.
- [16] Mahy, R., "A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)", [draft-ietf-sipping-mwi-02](#) (work in progress), March 2003.

#### Authors' Addresses

Jonathan Rosenberg  
dynamicsoft  
600 Lanidex Plaza  
Parsippany, NJ 07054  
US

Phone: +1 973 952-5000  
EMail: [jdrosen@dynamicsoft.com](mailto:jdrosen@dynamicsoft.com)  
URI: <http://www.jdrosen.net>

Henning Schulzrinne  
Columbia University  
M/S 0401  
1214 Amsterdam Ave.  
New York, NY 10027  
US

EMail: [schulzrinne@cs.columbia.edu](mailto:schulzrinne@cs.columbia.edu)  
URI: <http://www.cs.columbia.edu/~hgs>



Rohan Mahy (editor)  
Cisco Systems, Inc.  
5617 Scotts Valley Dr  
Scotts Valley, CA 95066  
USA  
  
EMail: rohan@cisco.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.