

Internet Engineering Task Force
Internet Draft

SIPPING WG
J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia U.

[draft-ietf-sipping-dialog-package-00.txt](#)

June 24, 2002

Expires: December 2002

A Session Initiation Protocol (SIP) Event Package for Dialog State

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/lid-abstracts.txt>

To view the list Internet-Draft Shadow Directories, see <http://www.ietf.org/shadow.html>.

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.

Internet Draft

Dialog Package

June 24, 2002

Table of Contents

1	Introduction	4
2	Dialog Event Package	5
2.1	Event Package Name	5
2.2	Event Package Parameters	5
2.3	SUBSCRIBE Bodies	5
2.4	Subscription Duration	5
2.5	NOTIFY Bodies	6
2.6	Notifier Processing of SUBSCRIBE Requests	6
2.7	Notifier Generation of NOTIFY Requests	7
2.7.1	The Dialog State Machine	7
2.7.2	Applying the state machine	9
2.8	Subscriber Processing of NOTIFY Requests	10
2.9	Handling of Forked Requests	11
2.10	Rate of Notifications	11
2.11	State Agents	11
3	Dialog Information Format	11
3.1	Structure of Dialog Information	12
3.1.1	Dialog Element	12
3.1.2	State	13
3.1.3	Local URI	13
3.1.4	Remote URI	13
3.1.5	Local Session Description	13
3.1.6	Remote Session Description	13
3.1.7	Remote Target	13
3.1.8	Local CSeq	14
3.1.9	Remote CSeq	14
3.1.10	Duration	14
3.2	Constructing Coherent State	14
3.3	Schema	15
3.4	Example	18
4	Security Considerations	20
5	IANA Considerations	20
5.1	application/dialog-info+xml MIME Registration	20
5.2	URN Sub-Namespace Registration for urn:ietf:params:xml:ns:dialog-info	21

6	Acknowledgements	22
7	Authors Addresses	22
8	Normative References	22
9	Informative References	23

[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 [[8](#)], watcher information [[9](#)], and message waiting indicators [[10](#)], amongst others. Here, we define an event package for INVITE initiated dialogs. Dialogs refer to the SIP relationship established between two SIP peers [[2](#)].

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

Automatic Callback: In this basic 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. In VoIP, this requires A 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 subscriber 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](#) Dialog Event Package

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

[2.1](#) 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 [\[1\]](#).

[2.2](#) Event Package Parameters

This package does not define any event package parameters.

[2.3](#) SUBSCRIBE Bodies

A SUBSCRIBE for a dialog package MAY contain a body. This body defines a filter to apply to the subscription.

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

- o 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.
- 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, remote target URI, CSeq numbers, SDP information, and so on, are not included normally unless explicitly requested and/or explicitly authorized.

OPEN ISSUE: It might be useful to define a filter within this document that would allow a subscriber to ask about a specific dialog.

[2.4](#) 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, and RECOMMEND that clients 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 (and they are authorized to find out about just the state of that dialog). In that case, when the dialog terminates, 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 call legs for a specific user. In these cases, a shorter interval makes more sense. The default is one hour for these subscriptions.

[2.5](#) NOTIFY Bodies

The body of the notification contains a dialog information document. The format of this document is described in [Section 3](#). Its MIME type is "application/dialog-info+xml". All subscribers MUST support this

format, and MUST list its type in any Accept header in the SUBSCRIBE. When no Accept header is present in the SUBSCRIBE, its default value is "application/dialog-info+xml".

Other dialog information formats might be defined in the future. In that case, the subscriptions MAY indicate support for other formats. However, they MUST always support and list "application/dialog-info+xml" as an allowed format.

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

[2.6](#) Notifier Processing of SUBSCRIBE Requests

The dialog information for a user contains very sensitive information. Therefore, all subscriptions SHOULD be authenticated and then authorized before approval. 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 a user is that 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 one or more calls, or not. Specifically, they should not be able to find out any more information than if they sent an

INVITE.

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.

[2.7](#) Notifier Generation of NOTIFY Requests

Notifications are generated for the dialog package 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 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.

[2.7.1](#) 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. We have also chosen to use a single FSM for both UAC and UAS.

The FSM for dialog state is shown in Figure 1. 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 "1xx-notag" event), 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

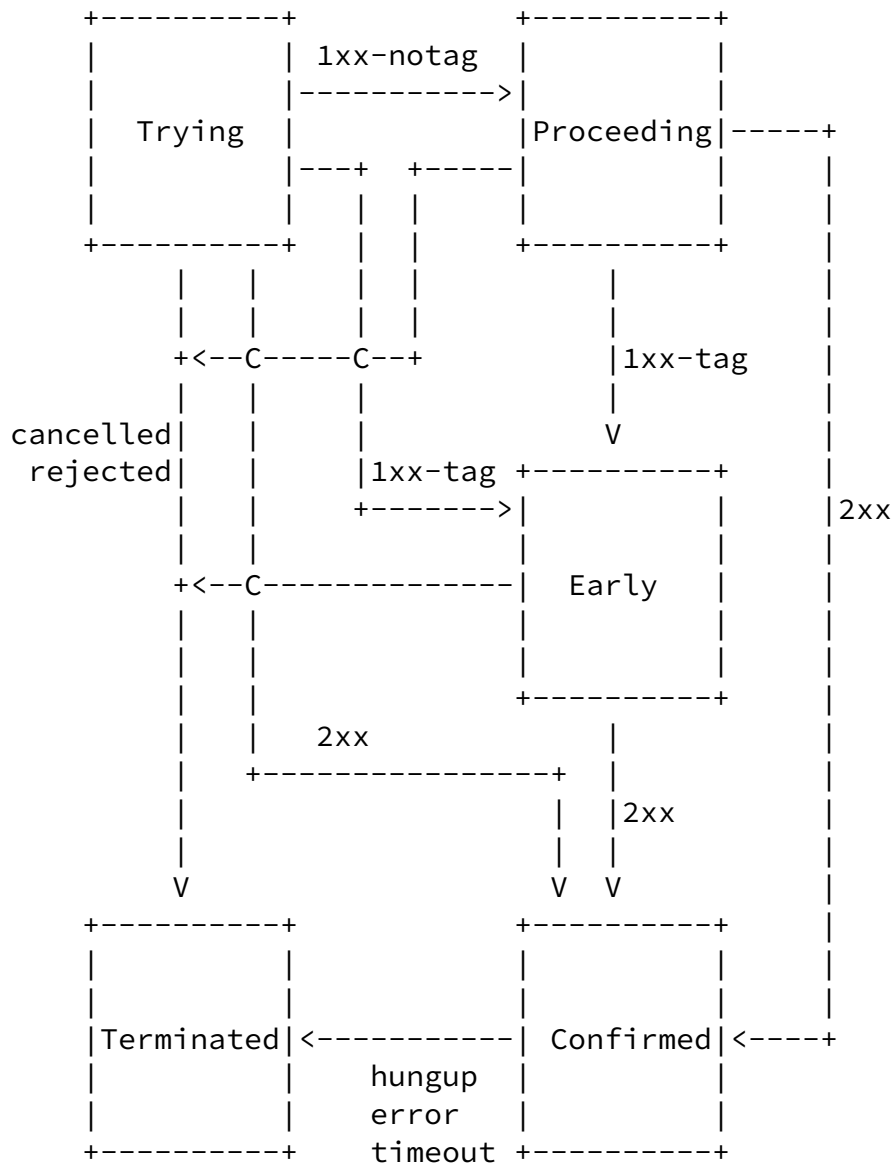


Figure 1: Dialog finite state machine

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 with a different tag, another instance of the FSM is created, initialized into the "early" or "confirmed" state depending on the response code. 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 "canceled". 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 "hungup" event), 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". Once in the "confirmed" state, the 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.

[2.7.2](#) Applying the state machine

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

Internet Draft

Dialog Package

June 24, 2002

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 actual dialog states across all dialogs 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" 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" 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" state. 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.

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 all dialogs 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 [[11](#)] which modify this information MAY trigger notifications.

[2.8](#) 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 3.2](#) for information on constructing coherent information from an application/dialog-info+xml document.

[2.9](#) 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.

[2.10](#) 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.

[2.11](#) 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. Therefore, the usage of state agents is NOT RECOMMENDED for this package.

[3](#) Dialog Information Format

Dialog information is an XML document [[3](#)] 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 [4], using the namespace identifier 'ietf' defined by [5] and extended by [6]. This URN is:

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

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

[3.1](#) 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.

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

[3.1.1](#) Dialog Element

The dialog element reports information on a specific 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. This is a different identifier than the formal dialog ID defined in SIP [2]. This identifier exists before the dialog exists, and remains unchanged when the actual dialog is established.

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.

`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 generation of a request that can

`create` a dialog.

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

The sub-elements of the dialog element provide additional information about the dialog. The only mandatory one is `status`.

[3.1.2](#) 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 the transition into the current state, and an optional `code` attribute that indicates the response code associated with the transition, assuming the event was `1xx-tag`, `1xx-notag`, or `2xx`.

[3.1.3](#) Local URI

The `local-uri` element indicates the local URI, as defined in [2]. It has an optional attribute, `display-name`, that contains the display

name from the local URI.

[3.1.4 Remote URI](#)

The `remote-uri` element indicates the remote URI, as defined in [2]. It has an optional attribute, `display-name`, that contains the display name from the remote URI.

[3.1.5 Local Session Description](#)

The `local-session-description` element contains the session description used by the notifier 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.

[3.1.6 Remote Session Description](#)

The `remote-session-description` element contains the session description used by the peer of the notifier 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.

[3.1.7 Remote Target](#)

J. Rosenberg et. al.

[Page 13]

Internet Draft

Dialog Package

June 24, 2002

The `remote-target` contains the `remote-target` URI as constructed by the user agent for this dialog, as defined in RFC BBBB [2]. This element should generally not be included in notifications, unless explicitly requested by the subscriber.

[3.1.8 Local CSeq](#)

The `local-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`.

[3.1.9 Remote CSeq](#)

The remote-cseq element contains the most recent value of the CSeq header seen by the UA in an incoming 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.

[3.1.10](#) Duration

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

[3.2](#) 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, the document is processed, and 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.

[3.3](#) Schema

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:ietf:params:xml:ns:dialog-info"
  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: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:complexType>
  </xs:element>
  <xs:attribute name="entity" type="xs:anyURI" use="required"/>
</xs:schema>
```

```
<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="local-uri" type="tns:nameaddr"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="remote-uri" type="tns:nameaddr"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="local-session-description" type="tns:sessd"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="remote-session-description" type="tns:sessd"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="remote-target" type="tns:nameaddr"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="local-cseq" type="tns:cseq" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="remote-cseq" type="tns:cseq" 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: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:element name="state">
<xs:complexType>
  <xs:simpleContent>
    <xs:restriction base="xs:string">
      <xs:enumeration value="trying"/>
      <xs:enumeration value="proceeding"/>
      <xs:enumeration value="early"/>
      <xs:enumeration value="confirmed"/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>

```

```
<xs:enumeration value="terminated"/>
<xs:attribute name="event" use="optional">
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="1xx-notag"/>
    <xs:enumeration value="1xx-tag"/>
    <xs:enumeration value="2xx"/>
    <xs:enumeration value="cancelled"/>
    <xs:enumeration value="rejected"/>
    <xs:enumeration value="hungup"/>
    <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:nonNegativeInteger">
    <xs:maxInclusive value="699"/>
    <xs:minInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:restriction>
</xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:complexType name="nameaddr">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="display-name" type="xs:string"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name="sessd">
  <xs:simpleContent>
    <xs:restriction base="xs:string">
      <xs:attribute name="type" type="xs:string"/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name="cseq">
  <xs:simpleContent>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="-1"/>
    </xs:restriction>
```

```
</xs:simpleContent>
</xs:complexType>
```

```
</xs:schema>
```

[3.4](#) Example

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

```
INVITE sip:callee@foo.com SIP/2.0
From: sip:caller@bar.com;tag=123
To: sip:callee@foo.com
Call-ID: 987@1.2.3.4
```

The XML document in a notification might look like:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  version="0"
  state="full"
  entity="sip:caller@bar.com">
  <dialog id="as7d900as8" call-id="987@1.2.3.4"
    local-tag="123" direction="initiator">
    <state>trying</state>
  </dialog>
</dialog-info>
```

If a 1xx with a tag is received, 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:caller@bar.com">
  <dialog id="as7d900as8" call-id="987@1.2.3.4"
    local-tag="123" remote-tag="456" direction="initiator">
    <state>early</state>
```

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

If it receives a second lxx, with a different tag, 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:caller@bar.com">
  <dialog id="as7d900as8" call-id="987@1.2.3.4"
    local-tag="123" remote-tag="456" direction="initiator">
    <state>early</state>
  </dialog>
  <dialog id="as7d900as8" call-id="987@1.2.3.4"
    local-tag="123" remote-tag="789" 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:caller@bar.com">
  <dialog id="as7d900as8" call-id="987@1.2.3.4"
    local-tag="123" remote-tag="789" direction="initiator">
    <state event="2xx">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:caller@bar.com">
  <dialog id="as7d900as8" call-id="987@1.2.3.4"
    local-tag="123" remote-tag="456" direction="initiator">
    <state event="cancelled">terminated</state>
  </dialog>
</dialog-info>
```

[4](#) Security Considerations

Subscriptions to dialog state can reveal very sensitive information. For this reason, this specification recommends authentication and authorization of subscriptions, and provides guidelines on sensible authorization policies.

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

[5](#) IANA Considerations

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

[5.1](#) 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](#) [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 4](#) 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.

[5.2](#) URN Sub-Namespace Registration for urn:ietf:params:xml:ns:dialog-info

This section registers a new XML namespace, as per the guidelines in [\[6\]](#).

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

Registrant Contact: IETF, SIMPLE working group,
<simple@mailman.dynamicsoft.com>, 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/xhtml-basic/xhtml-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>application/dialog-info+xml</h2>

```

J. Rosenberg et. al.

[Page 21]

Internet Draft

Dialog Package

June 24, 2002

```
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

[6](#) Acknowledgements

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

7 Authors Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: schulzrinne@cs.columbia.edu

8 Normative References

[1] A. Roach, "SIP-specific event notification," Internet Draft, Internet Engineering Task Force, Mar. 2002. Work in progress.

[2] J. Rosenberg, H. Schulzrinne, et al. , "SIP: Session initiation protocol," Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

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

[4] R. Moats, "URN syntax," [RFC 2141](#), Internet Engineering Task Force, May 1997.

J. Rosenberg et. al.

[Page 22]

Internet Draft

Dialog Package

June 24, 2002

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

[6] M. Mealling, "The IANA XML registry," Internet Draft, Internet

Engineering Task Force, Nov. 2001. Work in progress.

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

9 Informative References

[8] J. Rosenberg, "Session initiation protocol (SIP) extensions for presence," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[9] J. Rosenberg, "A session initiation protocol (SIP)event template-package for watcher information," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[10] R. Mahy, "A message summary and message waiting indication event package for the session initiation protocol (SIP)," Internet Draft, Internet Engineering Task Force, June 2002. Work in progress.

[11] J. Rosenberg, "The session initiation protocol UPDATE method," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

Full Copyright Statement

Copyright (c) The Internet Society (2002). 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 assigns.

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.

