Network Working Group
Request for Comments: 4235
Category: Standards Track

J. Rosenberg
Cisco Systems
H. Schulzrinne
Columbia University
R. Mahy, Ed.
SIP Edge LLC

November 2005

An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)

Status of this Memo

Copyright Notice

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 to receive notification of the changes in state of INVITE-initiated dialog usages in which the subscribed-to user is involved.

Table of Contents

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 [16], watcher information [17], and message waiting indicators [18], amongst others. This document defines an event package for INVITE-initiated dialog usages. 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 [1] defines the SUBSCRIBE and NOTIFY methods, which also create new dialog usages. However, using this package to model state for non-session dialog usages is out of the scope of this specification.

A variety of applications are enabled through knowledge of INVITE dialog usage state. Some examples include:

Automatic Callback: In this basic Public Switched Telephone Network (PSTN) application, user A calls user B but 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 up, they hear ringing, while they are being connected to B. To implement this with SIP, a mechanism is required for A to receive a notification when the dialogs at B are complete.

Presence-Enabled Conferencing: In this application, user A wishes to set up a conference call with users B and C. Rather than being scheduled, the call is 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. In the second way, the server subscribes to the dialog state of those users and receives notifications as this state 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 receive an Instant Message (IM) on 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.

This document also defines two new callee capability [10] feature parameters:

- "sip.byeless", which indicates that a SIP user agent (UA) is not capable of terminating a session itself (for example, in some announcement or recording services, or in some call centers) in which the UA is no longer interested in participating; and

- "sip.rendering", which positively describes whether the user agent is rendering any of the media it is receiving. These feature parameters are useful in many of the same applications that motivated the dialog package, such as conferencing, presence, and the shared-line example described in Section 6.2.

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 in [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 fields, as defined in [1].

3.2. Event Package Parameters

This package defines four Event Package parameters: call-id, to-tag, from-tag, and include-session-description. If a subscription to a specific dialog is requested, the first three of these parameters MUST be present, to 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 whether the subscriber would like to receive the session descriptions associated with the subscribed dialog usage or usages.

It is also possible to subscribe to the set of dialogs created as a result of a single INVITE sent by a UAC (user agent client). 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 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"
```

If any call-ids contain embedded double-quotes, those double-quotes MUST be escaped 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 the callid production and the word production, which is used by callid (both from RFC 3261 [1]), allow some characters (such as "@", "[", and ":") that are not allowed within a token.

3.3. SUBSCRIBE Bodies

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

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

- If the Event header field contained dialog identifiers, a notification is 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.

- If there were no dialog identifiers in the Event header field, a notification is 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.

- Notifications do not normally contain full state; rather, they only indicate the state of the dialog(s) whose state has changed. The exceptions are a NOTIFY sent in response to a SUBSCRIBE, and a NOTIFY that contains no dialog elements. These NOTIFYs contain the complete view of dialog state.

- The notifications contain the identities of the participants in the dialog, the target URIs, and the dialog identifiers. Session descriptions are not included unless explicitly requested and 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 long. 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 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; it SHOULD be longer than the typical duration of a dialog. We recommend a default duration of two hours, which is likely to 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 in a package-specific default format 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 it 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. The 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. 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>
```

A user agent that registers with the address-of-record X SHOULD 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 become aware of each other's state. This is useful for applications such as single-line-extension, also known as shared lines.

Note that many implementations of "shared-lines" have a feature that allows details of calls on a shared address-of-record to be made private. This is a completely reasonable authorization policy that could result in notifications that 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 (user agent client) and UAS (usage agent server). We have chosen to handle the first issue by extending the dialog finite state machine (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.

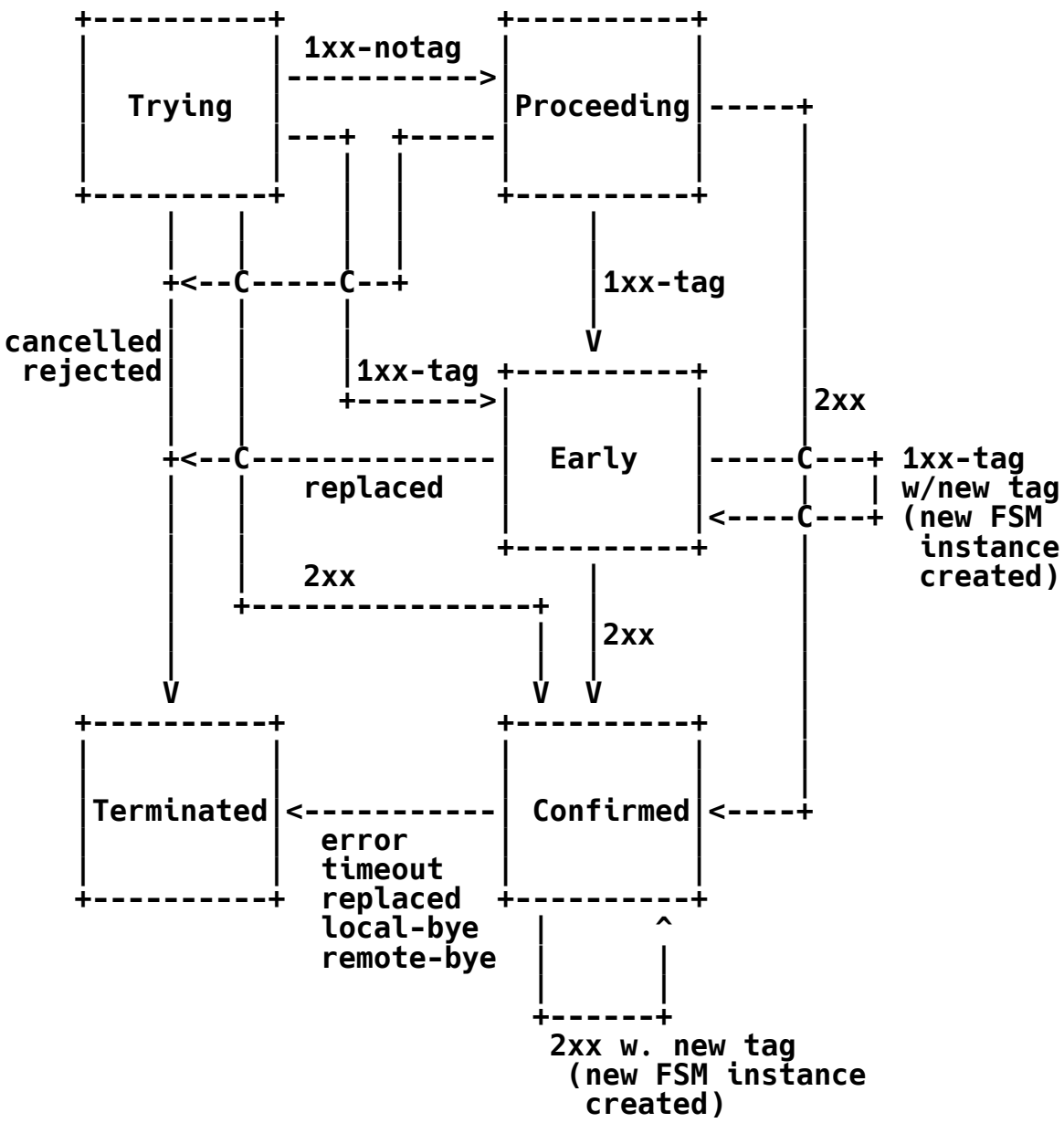

Figure 3

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 (the call identifier and local tag) are known. 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 in the common case of no forking.

If the UAC sends a CANCEL and then subsequently receives 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) that 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 receives a CANCEL request and then generates 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 generates 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 field) that replaces the current Confirmed dialog, the replaced invitation 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 states to the Terminated state.

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 it 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 be 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 will be used, and event transitions on that FSM will 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 of state to use depends on whether the UA wishes to let unknown users know that their phone is ringing, as opposed to being 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 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 (Busy Here) 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 be reported, along with complete dialog IDs. This means either all of the dialogs, if no dialog identifiers were present in the Event header field, or the specific set of dialogs identified by the Event header field parameters.

The notifier SHOULD generate a NOTIFY request on any change in the characteristics associated with the dialog. Since these include Contact URIs, Contact parameters, and session descriptions, receipt of re-INVITEs and UPDATE requests [3] that modify this information MAY trigger notifications.

3.8. Subscriber Processing of NOTIFY Requests

The SIP Events framework expects packages to specify how a subscriber processes NOTIFY requests in package-specific ways. In particular, a package should specify 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 contain information about only 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 4.3 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 to reach multiple UAs.

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 be excessive. It is RECOMMENDED that the server not generate notifications for a single subscriber faster than once every 1 second.

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. 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 non-negative 32 bit integer.

- state: This attribute indicates whether the document contains the full dialog information, or whether it contains only information on those dialogs that 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. An example:

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

4.1.1. Dialog Element

The dialog element reports information about 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 response with a tag, or a 2xx response 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 current dialogs at a UA.

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

- call-id: This attribute is a string that 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 ' for ' .)

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

- remote-tag: This attribute is a string that 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.

4.1.2. State Element

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 that 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>
```

4.1.3. Duration Element

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

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

4.1.4. Replaces Element

The "replaces" element is used to correlate a new dialog with one it replaced as a result of an invitation with a Replaces header field. 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"/>
```

4.1.5. Referred-By Element

The "referred-by" element is used to correlate a new dialog with a REFER [12] request that triggered it. The element is present in a dialog that was triggered by a REFER request that contained a Referred-By [11] header field 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>
```

4.1.6. Local and Remote Elements

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

4.1.6.1. Identity Element

The "identity" element indicates a local or remote URI, as defined in [2] as appropriate. It has an optional attribute, display, 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. To avoid repeating identity information in each request, the subscriber can assume that the identity URIs are the same as in previous notifications if no identity elements are present in the corresponding local or remote element. If any identity elements are present in the local or remote part of a notification, the new list of identity tags completely supersedes the old list in the corresponding part.

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

4.1.6.2. Target Element

The "target" contains the local or remote target URI 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 two required attributes, pname and pval. Boolean parameters are represented by the explicit pval values, "true" and "false" (for example, when a feature parameter is explicitly negated). Parameters that have no value at all are represented by the explicit pval value "true". The param element itself has no contents. To avoid repeating Contact information in each request, the subscriber can assume that the target URI and parameters are the same as in previous notifications if no target element is present in the corresponding local or remote element. If a target element is present in the local or remote part of a notification, the new target tag and list of parameter tags completely supersedes the old target and parameter list in the corresponding part. Note that any quoting (including extra angle-bracket quoting used to quote string values in [10]) or backslash escaping MUST be removed before being placed in a pval attribute. Any remaining single quotes, double quotes, and ampersands MUST be properly XML escaped.

```
<target uri="sip:alice@pc33.example.com">
  <param pname="isfocus" pval="true"/>
  <param pname="class" pval="business"/>
  <param pname="description" pval="Alice's desk &amp;amp; office"/>
  <param pname="sip.rendering" pval="no"/>
</target>
```

4.1.6.3. Session Description Element

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 it was explicitly requested by the subscriber. It has a single attribute, "type", which indicates the MIME media type of the session description. To avoid repeating session description information in each request, the subscriber can assume that the session description is the same as in previous notifications if no session description element is present in the corresponding local or remote element.

4.2. Sample Notification Body

```
<?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" pval="true"/>
            <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>
```

4.3. Constructing Coherent State

The dialog information subscriber maintains a table listing the dialogs, with a row for each dialog. Each row is indexed by an ID that is present in the "id" attribute of the "dialog" element. Each row contains 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 is compared to the "version" attribute in the new document. 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 (SUBSCRIBE) 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 and then 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 compares the ID in the "id" attribute of the "dialog" element with the ID associated with the row. If the dialog does not 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.

4.4. 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: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: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="tns: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="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="uri" type="xs:string"
                                      use="required"/>
          </xs:complexType>
        </xs:element>
        <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>
```

5. Definition of New Media Feature Parameters

This section defines two new media feature parameters that are useful as input to user presence, in conferencing applications, and in applications like the shared-line example described in Section 6.2. These feature parameters are especially useful in combination with the dialog package, as they allow an authorized third party to become aware of these characteristics.

5.1. The "sip.byeless" Parameter

The "sip.byeless" media feature parameter is a new boolean parameter, defined in this document, that provides a positive indication that the user agent setting the parameter is unable to terminate sessions on its own (for example, by sending a BYE request). For example,

continuous announcement services and certain recording services are unable to determine when it would be desirable to terminate a session, and therefore they do not have the ability to terminate sessions at all. Also, many human call centers are configured so that they never terminate sessions. (This is to prevent call center agents from accidentally disconnecting the caller). (Note that per [10], this parameter name must be preceded by a "+" character when used in a SIP Contact header field.)

```
Contact: <sip:recording-service@host.example.net>
    ;automaton;+sip.byeless
```

5.2. The "sip.rendering" Parameter

The "sip.rendering" media feature parameter is a new string parameter, defined in this document, that can provide a positive indication whether the user agent setting the parameter is currently rendering any of the media it is receiving in the context of a specific session. It MUST only be used in a Contact header field in a dialog created using the INVITE request.

This parameter has three legal values: "yes", "no", and "unknown". The value "yes" indicates positive knowledge that the user agent is rendering at least one of the streams of media that it is receiving. The value "no" indicates positive knowledge that the user agent is rendering none of the media that it is receiving. The value "unknown" indicates that the user agent does not know whether the media associated with the session is being rendered (which may be the case if the user agent is acting as a 3pcc (Third Party Call Control) [19] controller).

The "sip.rendering" parameter is useful in applications such as shared appearances, conference status monitoring, or as an input to user presence.

```
Contact: <sip:musak-onhold@host.example.net>
  ;automaton;+sip.rendering="no"
```

6. Examples

6.1. Basic Example

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

```
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 response is received on the second dialog, the dialog 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 response has been 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>
```

6.2. Emulating a Shared-Line Phone System

The following example shows how a SIP telephone user agent can provide detailed state information and also emulate a shared-line telephone system (the phone "lies" about having a dialog while it is merely offhook).

Idle:

```
<?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>
```

Seized:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="1" state="partial"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8">
    <state>trying</state>
  </dialog>
</dialog-info>
```

Dialing:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="2" state="partial"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" direction="initiator">
    <state>trying</state>
    <local>
      <identity display="Alice Smith">
         sip:alice@example.com
      </identity>
      <target uri="sip:alice@pc33.example.com"/>
    </local>
    <remote>
      <identity>sip:bob@example.net</identity>
    </remote>
  </dialog>
</dialog-info>
```

Ringing:

```
<?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="07346y131" direction="initiator">
    <state code="180">early</state>
    <remote>
      <target uri="sip:bobster@host2.example.net"/>
    </remote>
  </dialog>
</dialog-info>
```

Answered (by voicemail):

```
<?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="07346y131" direction="initiator">
    <state reason="cancelled">terminated</state>
  </dialog>
  <dialog id="zxcvbnm3" call-id="a84b4c76e66710"
          local-tag="1928301774"
          remote-tag="8736347" direction="initiator">
    <state code="200">confirmed</state>
    <remote>
      <target uri="sip:bob-is-not-here@vm.example.net">
        <param pname="actor" pval="msg-taker"/>
        <param pname="automaton" pval="true"/>
        <param pname="+sip.byeless" pval="true"/>
      </target>
    </remote>
  </dialog>
</dialog-info>
```

Alice would rather talk to Bob's assistant (Cathy Jones) than to Bob's voicemail. She indicates this preference by pressing a key (perhaps "0" in North America or "9" in Europe). Bob's voicemail system then acts on this keypress by transferring [20] Alice's call to Cathy's AOR.

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="5" state="partial"
             entity="sip:alice@example.com">
  <dialog id="zxcvbnm3" call-id="a84b4c76e66710"
          local-tag="1928301774"
          remote-tag="8736347" direction="initiator">
    <state reason="replaced">terminated</state>
  </dialog>
  <dialog id="sfhjsjk12" call-id="o34oii1"
          local-tag="8903j4"
          remote-tag="78cjkus" direction="receiver">
    <state reason="replaced">confirmed</state>
    <replaces call-id="a84b4c76e66710"
          local-tag="1928301774"
          remote-tag="8736347"/>
    <referred-by>
      sip:bob-is-not-here@vm.example.net
    </referred-by>
    <local>
      <target uri="sip:alice@pc33.example.com"/>
        <param pname="+sip.rendering" pval="yes"/>
    </local>
    <remote>
      <identity display="Cathy Jones">
         sip:cjones@example.net
      </identity>
      <target uri="sip:line3@host3.example.net">
        <param pname="actor" pval="attendant"/>
        <param pname="automaton" pval="false"/>
      </target>
    </remote>
  </dialog>
</dialog-info>
```

Alice and Cathy talk, Cathy adds Alice to a local conference:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="6" state="partial"
             entity="sip:alice@example.com">
  <dialog id="sfhjsjk12" call-id="o34oii1"
          local-tag="8903j4"
          remote-tag="78cjkus" direction="receiver">
    <state>confirmed</state>
    <remote>
      <target uri="sip:confid-34579@host3.example.net">
        <param pname="isfocus" pval="true"/>
      </target>
    </remote>
  </dialog>
</dialog-info>
```

Alice puts Cathy on hold:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="7" state="partial"
             entity="sip:alice@example.com">
  <dialog id="sfhjsjk12" call-id="o34oii1"
          local-tag="8903j4"
          remote-tag="78cjkus" direction="receiver">
    <state>confirmed</state>
    <local>
      <target uri="sip:alice@pc33.example.com"/>
        <param pname="+sip.rendering" pval="no"/>
      </target>
    </local>
  </dialog>
</dialog-info>
```

Cathy hangs up:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="8" state="partial"
             entity="sip:alice@example.com">
  <dialog id="sfhjsjk12" call-id="o34oii1"
          local-tag="8903j4"
          remote-tag="78cjkus" direction="receiver">
    <state reason="remote-bye">terminated</state>
  </dialog>
  <dialog id="08hjh1345">
    <state>trying</state>
  </dialog>
</dialog-info>
```

Alice hangs up:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="9" state="full"
             entity="sip:alice@example.com">
</dialog-info>
```

6.3. Minimal Dialog Information with Privacy

The following example shows the same user agent providing minimal information to maintain privacy for services like automatic callback.

Onhook:

```
<?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>
```

Offhook: (implementation/policy choice for Alice to transition to this "state" when "seized", when Trying, when Proceeding, or when Confirmed.)

```
<?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="1">
    <state>confirmed</state>
  </dialog>
</dialog-info>
```

Onhook: (implementation/policy choice for Alice to transition to this "state" when terminated, or when no longer "seized")

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="2" state="full"
             entity="sip:alice@example.com">
</dialog-info>
```

7. 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. User agents that implement the dialog package SHOULD also implement SIP over TLS [15] and the sips: scheme.

8. IANA Considerations

This document registers a new MIME type, application/dialog-info+xml; a new XML namespace; and two new media feature parameters in the SIP tree.

8.1. MIME Registration for application/dialog-info+xml Type

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 7 of this specification.

Interoperability considerations: none.

Published specification: This document.

Applications that 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: .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.

8.2. URN Sub-Namespace Registration for urn:ietf:params:xml:ns:dialog-info

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

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

Registrant Contact: The IESG, <iesg@ietf.org>
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>urn:ietf:params:xml:ns:dialog-info</h2>
  <p>See <a href="ftp://ftp.rfc-editor.org/in-notes/rfc4235.txt">
       RFC4235</a>.</p>
</body>
</html>
END
```

8.3. Schema Registration

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

URI: urn:ietf:params:xml:schema:dialog-info

Registrant Contact: The IESG, <iesg@ietf.org>

XML: The XML can be found as the sole content of Section 4.4.

8.4. Media Feature Parameter Registration

This section registers two new media feature tags, per the procedures defined in RFC 2506 [14]. The tags are placed into the sip tree, which is defined in [10].

8.4.1. Media Feature Tag sip.byeless
Media feature tag name sip.byeless

ASN.1 Identifier 19

Summary of the media feature indicated by this tag: This feature tag is a boolean flag. When set it indicates that the device is incapable of terminating a session autonomously.

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of an application, such as an announcement service, recording service, conference, or call center.

Examples of typical use: Call centers and media services.

Related standards or documents: RFC 4235
Security Considerations: This media feature tag can be used in ways that affect application behaviors or may reveal private information. For example, a conferencing or other application may decide to terminate a call prematurely if this media feature tag is set. Therefore, if an attacker can modify the values of this tag, they may be able to affect the behavior of applications. As a result of this, applications that utilize this media feature tag SHOULD provide a means for ensuring its integrity. Similarly, this feature tag should only be trusted as valid when it comes from the user or user agent described by the tag. As a result, protocols for conveying this feature tag SHOULD provide a mechanism for guaranteeing authenticity.

8.4.2. Media Feature Tag sip.rendering

Media feature tag name: sip.rendering

ASN.1 Identifier: 20

Summary of the media feature indicated by this tag: This feature tag contains one of three string values indicating if the device is rendering any media from the current session ("yes"), none of the media from the current session ("no"), or if this status is not known to the device ("unknown").

Values appropriate for use with this feature tag: String.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application, for describing the state of a device (such as a phone or PDA) during a multimedia session.

Examples of typical use: Conferencing, telephone shared-line emulation, and presence applications.

Related standards or documents: RFC 4235

Security Considerations: This media feature tag can be used in ways that affect application behaviors or may reveal private information. For example, a conferencing or other application may decide to terminate a call prematurely if this media feature tag is set to "no". Therefore, if an attacker can modify the values of this tag, they may be able to affect the behavior of applications. As a result of this, applications that utilize this media feature tag SHOULD provide a means for ensuring its integrity. Similarly, this feature tag should only be trusted as valid when it comes from the user or user agent described by the tag. As a result, protocols for conveying this feature tag SHOULD provide a mechanism for guaranteeing authenticity.

9. Acknowledgements

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

10. References

10.1. Normative References

[1] Roach, A.B., "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] Paoli, J., Sperberg-McQueen, C., Bray, T., and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C FirstEdition 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", BCP 81, RFC 3688, January 2004.

[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., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[11] Sparks, R., "The Session Initiation Protocol (SIP) Referred-By Mechanism", RFC 3892, September 2004.

[12] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[13] Mahy, R., Biggs, B., and R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header", RFC 3891, September 2004.

[14] Holtman, K., Mutz, A., and T. Hardie, "Media Feature Tag Registration Procedure", BCP 31, RFC 2506, March 1999.

[15] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

10.2. Informative References

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

[17] Rosenberg, J., "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)", RFC 3857, August 2004.

[18] Mahy, R., "A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)", RFC 3842, August 2004.

[19] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

[20] Sparks, R., "Session Initiation Protocol Call Control - Transfer", Work in Progress, July 2005.

Authors' Addresses

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@cisco.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
URI: http://www.cs.columbia.edu/~hgs

Rohan Mahy (editor)
SIP Edge LLC

EMail: rohan@ekabal.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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