                  A Multi-party Application Framework for SIP

1 Abstract

     This document defines a framework and requirements for multi-party
     applications in SIP. To enable discussion of multi-party
     applications we define an abstract call model for describing the
     media relationships required by many of these applications. The
     model and actions described here are specifically chosen to be
     independent of the SIP signaling and/or mixing approach chosen to
     actually setup the media relationships. In addition to its dialog
     manipulation aspect, this framework includes requirements for
     communicating related information and events such as conference and
     session state, and session history. This framework also describes
     other goals which embody the spirit of SIP applications as used on
     the Internet.

2 Conventions used in this document

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

SIP Multiparty Framework

Table of Contents

3 Motivation and Background

The Session Initiation Protocol [SIP] was defined for the initiation, maintenance, and termination of sessions or calls between one or more users. However, despite its origins as a large-scale multiparty conferencing protocol, SIP is used today primarily for point to point calls. This two-party configuration is the focus of the SIP specification and most of its extensions.

This document defines a framework and requirements for multi-party applications in SIP. Most multi-party applications manipulate SIP dialogs (also known as call legs) to cause participants in a conversation to perceive specific media relationships. In other protocols that deal with the concept of calls, this manipulation is known as call control. In addition to its dialog manipulation aspect, "call control" also includes communicating information and events related to manipulating calls, including information and events dealing with session state and history, conference state, user state, and even message state.

3.1 Goals

Based on input from the SIP community, the authors compiled the following set of goals for SIP call control and multiparty applications:

- Define Primitives, Not Services. Allow for a handful of robust yet simple mechanisms which can be combined to deliver features and services. Throughout this document we refer to these simple mechanisms as "primitives". Primitives should be sufficiently robust that when they are combined they can be used to build lots of services. However, the goal is not to define a provably complete set of primitives. Note that while the IETF will NOT standardize behavior or services, it may define example services for informational purposes, as in [service examples].

- Participant oriented. The primitives should be designed to provide services which are oriented around the experience of the participants. The authors observe that end users of features and services usually don't care how a media relationship is setup. Their ultimate experience is based only on the resulting media and other externally visible characteristics.

- Signaling Model independent: Support both a central control and a peer-to-peer feature invocation model (and combinations of the two). baseline SIP already supports a centralized control model described in [3pcc], and the SIP community has expressed a great deal of interest in peer-to-peer or distributed call control. Some such

primitives are already defined in [REFER] and [Replaces].

- Mixing Model independent: The bulk of interesting multiparty
  applications involve mixing or combining media from multiple
  participants. This mixing can be performed by one or more of the

participants, or by a centralized mixing resource. The experience
of the participants should not depend on the mixing model used.
While most examples in this document refer to audio mixing, the
framework applies to any media type. In this context a "mixer"
refers to combining media in an appropriate, media-specific way.

- Invoker oriented. Only the user who invokes a feature or a service
needs to know exactly which service is invoked or why. This is good
because it allows new services to be created without requiring new
primitives from all the participants; and it allows for much simpler
feature authorization policies, for example, when participation
spans organizational boundaries. As discussed in section 4.7, this
also avoids exponential state explosion when combining features.
The invoker only has to manage a user interface or API to prevent
local feature interactions. All the other participants simply need
to manage the feature interactions of a much smaller number of
primitives.

- Primitives make full use of URIs. URIs are a very powerful
mechanism for describing users and services. They represent a
plentiful resource which can be extremely expressive and easily
routed, translated, and manipulated--even across organizational
boundaries. URIs can contain special parameters and informational
headers which need only be relevant to the owner of the namespace
(domain) of the URI. Just as a user who selects an http: URL need
not understand the significance and organization of the web site it
references, a user may encounter a SIP URL which translates into an
email-style group alias, which plays a pre-recorded message, or runs
some complex call-handling logic.

- Make use of SIP headers and SIP event packages to provide SIP
entities with information about their environment. These should
include information about the status / handling of dialogs on other
user agents, information about the history of other contacts
attempted prior to the current contact, the status of participants,
the status of conferences, user presence information, and the status
of messages.

- Encourage service decomposition, and design to make use of
standard components using well-defined, simple interfaces. Sample
components include a SIP mixer, recording service, announcement
server, and voice dialog server. (This is not an exhaustive list).

- Include authentication, authorization, policy, logging, and
accounting mechanisms to allow these primitives to be used safely
among mutually untrusted participants. Some of these mechanisms may
be used to assist in billing, but no specific billing system will be
endorsed.

- Permit graceful fallback to baseline SIP. Definitions for new SIP
  call control extensions/primitives MUST describe a graceful way to
  fallback to baseline SIP behavior. Support for one primitive MUST
  NOT imply support for another primitive.

   - There is no desire or goal to reinvent traditional models, such as
   the model used the [H.450] family of protocols, [JTAPI], or the
   [CSTA] call model, as these other models do not share the design
   goals presented in this document.

4 **Key Concepts**

4.1 **"Conversation Space" Model**

   This document introduces the concept of an abstract "conversation
   space" (essentially as a set of participants who believe they are
   all communicating among one another). Each conversation space
   contains one or more participants.

   Participants are SIP User Agents which send original media to or
   terminate and receive media from other members of the conversation
   space. Logically, every participant in the conversation space has
   access to all the media generated in that space (this is strictly
   true if all participants share a common media type). A SIP User
   Agent which does not contribute or consume any media is NOT a
   participant; nor is a user agent which merely forwards, transcodes,
   mixes, or selects media originating elsewhere in the conversation
   space. [Note that a conversation space consists of zero or more SIP
   calls or SIP conferences. A conversation space is similar to the
   definition of a "call" in some other call models.]

   Participants may represent human users or non-human users (referred
   to as robots or automatons in this document). Some participants may
   be hidden within a conversation space. Some examples of hidden
   participants include: robots which generate tones, images, or
   announcements during a conference to announce users arriving and
   departing, a human call center supervisor monitoring a conversation
   between a trainee and a customer, and robots which record media for
   training or archival purposes.

   Participants may also be active or passive. Active participants are
   expected to be intelligent enough to leave a conversation space when
   they no longer desire to participate. (An attentive human
   participant is obviously active.) Some robotic participants (such
   as a voice messaging system, an instant messaging agent, or a voice
   dialog system) may be active participants if they can leave the
   conversation space when there is no human interaction. Other robots
   (for example our tone generating robot from the previous example)
   are passive participants. A human participant "on-hold" is passive.

   An example diagram of a conversation space can be shown as a
   "bubble" or ovals, or as a "set" in curly or square brace notation.
   Each set, oval, or "bubble" represents a conversation space. Hidden

participants are shown in lowercase letters.

{ A , B }   [A , B]
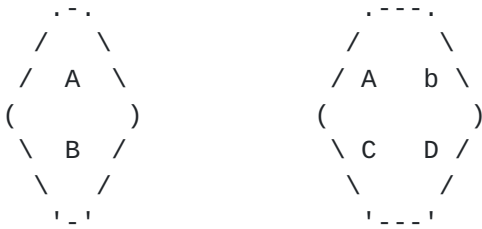
```
                    SIP Multiparty Framework


       .-.                      .----.
      /   \                    /      \
     /  A  \                  / A   b  \
    (       )                (          )
     \  B  /                  \ C   D  /
      \   /                    \      /
       '-'                      '---'



```
Comparison with Related Definitions

    In SIP, a call is "an informal term that refers to some
    communication between peers, generally set up for the purposes of a
    multimedia conversation." Obviously we cannot discuss normative
    behavior based on such an intentionally vague definition. The
    concept of a conversation space is needed because the SIP definition
    of call is not sufficiently precise for the purpose of describing
    the user experience of multiparty features.

    Do any other definitions convey the correct meaning? SIP, and [SDP]
    both define a conference as "a multimedia session identified by a
    common session description." A session is defined as "a set of
    multimedia senders and receivers and the data streams flowing from
    senders to receivers." Both of these definitions are heavily
    oriented toward multicast sessions with little differenciation among
    participants. As such, neither is particularly useful for our
    purposes. In fact, the definition of "call" in some call models is
    more similar to our definition of a conversation space.

    Some examples of the relationship between conversation spaces, SIP
    call legs, and SIP sessions are listed below. In each example, a
    human user will perceive that there is a single call.

        A simple two-party call is a single conversation space, a single
        session, and a single call-leg.

        A locally mixed three-way call is two sessions and two call-
        legs. It is also a single conversation space.

        A simple dial-in audio conference is a single conversation
        space, but is represented by as many call-legs and sessions as
        there are human participants.

        A multicast conference is a single conversation space, a single
        session, and as many call-legs as participants.

4.2 **Signaling Models**

    Obviously to make changes to a conversation space, you must be able
    to use SIP signaling to cause these changes. Specifically there

must be a way to manipulate SIP dialogs (call legs) to move
participants into and out of conversation spaces. Although this is
not as obvious, there also must be a way to manipulate SIP dialogs
to include non-participant user agents which are otherwise involved

in a conversation space (ex: B2BUAs, 3pcc controllers, mixers,
transcoders, translators, or relays).

Implementations may setup the media relationships described in the
conversation space model using the approach described in [3pcc]. The
3pcc approach relies on only the following 3 primitive operations:

    Create a new call-leg (INVITE)
    Modify a call-leg   (reINVITE)
    Destroy a call-leg   (BYE)

The main advantage of the 3pcc approach is that it only requires
very basic SIP support from end systems to support call control
features. As such, third-party call control is a natural way to
handle protocol conversion and mid-call features. It also has the
advantage and disadvantage that new features can/must be implemented
in one place only (the controller), and neither requires enhanced
client functionality, nor takes advantage of it.

In addition, a peer-to-peer approach is discussed at length in this
draft. The primary drawback of the peer-to-peer model is additional
end system complexity. The benefits of the peer-to-peer model
include:
- state remains at the edges
- call signaling need only go through participants involved
  (there are no additional points of failure)
- peers can take advantage of end-to-end message integrity or
  encryption
- setup time is shorter (fewer messages and round trips
  are required)

The peer-to-peer approach relies on additional "primitive"
operations, some of which are identified here.

    Replace an existing dialog
    Join a new dialog with an existing dialog [Join]
    Fork a new dialog with an existing dialog
    Locally do media forking (multi-unicast)
    Ask another UA to send a request on your behalf

Many of the features, primitives, and actions described in this
document also require some type of media mixing, combining, or
selection as described in the next section.

4.3 Mixing Models

SIP permits a variety of mixing models, which are discussed here
briefly. This topic is discussed more thoroughly in [conf-models].
For brevity, only the two most popular conferencing models are
significantly discussed in this document (local and centralized

mixing). Applications of the conversation spaces model to multicast
and multi-unicast (full unicast mesh) conferences are left as an
exercise for the reader. Note that a distributed full mesh
conference can be used for basic conferences, but does not easily

allow for more complex conferencing actions like splitting, joining, and forking.

Call control features should be designed to allow a mixer (local or centralized) to decide when to reduce a conference back to a 2-party call, or drop all the participants (for example if only two automatons are communicating). The actual heuristics used to release calls are beyond the scope of this document, but may depend on properties in the conversation space, such as the number of active, passive, or hidden participants; and the send-only, receive-only, or send-and-receive orientation of various participants.

4.3.1 (Single) End System Mixing

The first model we call "end system mixing". In this model, user A calls user B, and they have a conversation. At some point later, A decides to conference in user C. To do this, A calls C, using a completely separate SIP call. This call uses a different Call-ID, different tags, etc. There is no call set up directly between B and C. No SIP extension or external signaling is needed. A merely decides to locally join two call-legs.

```
    B     C
     \   /
      \ /
       A
```
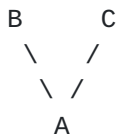
A receives media streams from both B and C, and mixes them. A sends a stream containing A's and C's streams to B, and a stream containing A's and B's streams to C. Basically, user A handles both signaling and media mixing.

4.3.2 Centralized Mixing

In a centralized mixing model, all participants have a pairwise SIP and media relationship with the mixer. Three applications of centralized mixing are also discussed below.

[diagram]

4.3.2.1 Dial-In Conference Servers

Dial-In conference servers closely mirror dial-in conference bridges in the traditional PSTN. A dial-in conference server acts as a normal SIP UA. Users call it, and the server maintains point to point SIP relationships with each user that calls in. The server takes the media from the users who dial into the same conference, mixes them, and sends out the appropriate mixed stream to each participant separately.

As in other applications of centralized mixing, the conference is
identified by the request URI of the calls from each participant.
This provides numerous advantages from a services and routing point
of view. For example, one conference on the server might be known as

sip:conference34@servers.com. All users who call
sip:conference34@servers.com are mixed together. Dial-In conference
servers are usually associated with pre-arranged conferences.
However, the same model applies to ad-hoc conferences. An ad-hoc
conference server creates the conference state when the first user
joins, and destroys it when the last one leaves. The SIP interface
is identical to the pre-arranged case.

4.3.2.2 Ad-hoc Centralized Conferences

In an ad-hoc centralized conference, two users A and B start with a
normal SIP call. At some point later, they decide to add a third
party. Instead of using end system mixing, they would prefer to use
a central SIP mixer. Initially, A calls B. At some point, B decides
to add user C to the call, and begins the transition to a conference
server. The first step in this process is the discovery of a
conference server that supports ad-hoc conferences. This can be done
through static configuration, or through any of a number of standard
service discovery protocols, such as the Service Location Protocol
[SLP]. Once the server is discovered, a conference ID is chosen. The
first participant to send an INVITE to this URL creates the initial
conference state in the server. SIP dialogs are manipulated (using
any combination of 3pcc or peer-to-peer signaling) so that each
participant is sending media to the conference server. It is also
possible to transition from a end system mixed conference (even one
with a complex connection topology), to a centralized conference
server.

4.3.2.3 Dial-Out Conferences

Dial-out conferences are a simple variation on dial-in conferences.
Instead of the users joining the conference by sending an INVITE to
the server, the server chooses the users who are to be members of
the conference, and then sends them the INVITE. Typically dial out
conferences are pre-arranged, with specific start times and an
initial group membership list. However, there are other means for
the dial-out server to determine the list of participants, including
user presence [13]. Once the users accept or reject the call from
the dial out server, the behavior of this system is identical to the
dial-in server case.

4.3.3 Multicast and Multi-unicast conferences

In these models, all endpoints send media to all other endpoints.
Consequently every endpoint mixes their own media from all the other
sources, and sends their own media to every other participant.

[diagrams]

4.3.3.1 Large-Scale Multicast Conferences

Large-scale multicast conferences were the original motivation for both the Session Description Protocol [SDP] and SIP. In a large-scale multicast conference, one or more multicast addresses are

allocated to the conference. Each participant joins that multicast groups, and sends their media to those groups. Signaling is not sent to the multicast groups. The sole purpose of the signaling is to inform participants of which multicast groups to join. Large-scale multicast conferences are usually pre-arranged, with specific start and stop times. However, multicast conferences do not need to be pre-arranged, so long as a mechanism exists to dynamically obtain a multicast address.

4.3.3.2 Centralized Signaling, Distributed Media

In this conferencing model, there is a centralized controller, as in the dial-in and dial-out cases. However, the centralized server handles signaling only. The media is still sent directly between participants, using either multicast or multi-unicast. Multi-unicast is when a user sends multiple packets (one for each recipient, addressed to that recipient). This is referred to as a "Decentralized Multipoint Conference" in [H.323].

4.3.3.3 Full Distributed Unicast Conferencing

In this conferencing model, each participant has both a pairwise media relationship and a pairwise SIP relationship with every other participant (a full mesh). This model requires a mechanism to maintain a consistent view of distributed state across the group. This is a classic hard problem in computer science. Also, this model does not scale well for large numbers of participants. bascause for <n> participants the number of media and SIP relationships is approximately n-squared. As a result, this model is not generally available in commercial implementations; to the contrary it is primarily the topic of research or experimental implementations. Note that this model assumes peer-to-peer signaling.

4.4 Conveying Information and Events

Participants should have access to information about the other participants in a conversation space, so that this information can be rendered to a human user or processed by an automaton. Although some of this information may be available from the Request-URI or To, From, Contact, or other SIP headers, another mechanism of reporting this information is necessary.

Many applications are driven by knowledge about the progress of calls and conferences. In general these types of events allow for the construction of distributed applications, where the application requires information on dialog and conference state, but is not necessarily co-resident with an endpoint user agent or conference

server. For example, a mixer involved in a conversation space may
wish to provide URLs for conference status, and/or conference/floor
control.

SIP Multiparty Framework

The SIP [Events] architecture 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.

New event packages should be able to
provide the status of a user's call-legs (dialogs), provide the
status of conferences and its participants, provide user presence
information, and provide the status of user's messages. While this
is not an exhaustive list, these are sufficient to enable the sample
features described in this document.

A conference event package allows users to subscribe to information
about an entire conference or conversation space. This conference
state could be provided by a conference server or mixing component
(described in a later section) if centralized mixing is used, or
gathered from relevant peers and merged into a cohesive set of
state. Notifications would convey information about the
pariticipants such as: the SIP URL identifying each user, their
status in the space (active, declined, departed), URLs to invoke
other features (such as sidebar conversations), links to other
relevant information (such as floor control policies), and if floor
control policies are in place, the user's floor control status. A
dialog event package would provide information about all the dialogs
the target user is maintaining, what conversations the user in
participating in, and how these are correlated. Concrete proposals
for conference events and dialog events are described in [dialog-
pkg] and [conf-pkg] respectively.

Note that user presence has a close relationship with these two
proposed event packages. It is fundamental to the presence model
that the information used to obtain user presence is constructed
from any number of different input sources. Examples of such sources
include SIP REGISTER requests and uploads of presence documents.
These two packages can be considered another mechanism that allows a
presence agent to determine the presence state of the user.
Specifically, a user presence server can act as a subscriber for the
call-leg and conference packages to obtain additional information
that can be used to construct a presence document.

The multi-party architecture should also provide a mechanism to get
information about the status /handling of a dialog (for example,
information about the history of other contacts attempted prior to
the current contact). Finally, the architecture should provide
ample opportunities to present informational URIs which relate to
calls, conversations, or dialogs in some way. For example, consider
the SIP Call-Info header, or Contact headers returned in a 300-class
response. Frequently additional information about a call or dialog

can be fetched via non-SIP URIs. For example, consider a web page
for package tracking when calling a delivery company, or a web page
with related documentation when joining a dial-in conference. The
use of URIs in the multiparty framework is discussed in more detail
in Section 4.6.

4.5 Componentization and Decomposition

This framework proposes a decomposed component architecture with a
very loose coupling of services and components. This means that a
service (such as a conferencing server or an auto-attendant) need
not be implemented as an actual server. Rather, these services can
be built by combining a few basic components in straightforward or
arbitrarily complex ways.

Since the components are easily deployed on separate boxes, by
separate vendors, or even with separate providers, we achieve a
separation of function that allows each piece to be developed in
complete isolation. We can also reuse existing components for new
applications. This allows rapid service creation, and the ability
for services to be distributed across organizational domains
anywhere in the Internet.

For many of these components it is also desirable to discover their
capabilities, for example querying the ability of a mixer to host a
10 dialog conference, or to reserve resources for a specific time.
These actions could be provided in the form of URLs, provided there
is an a priori means of understanding their semantics. For example
if there is a published dictionary of operations, a way to query the
service for the available operations and the associated URLs, the
URL can be the interface for providing these service operations.
This concept is described in more detail in the context of dialog
operations in section 4.6

4.5.1 Media Intermediaries

Media Intermediaries are not participants in any conversation space,
although an entity which is also a media translator may also have a
colocated participant component (for example a mixer which also
announces the arrival of a new participant; the announcement portion
is a participant, but the mixer itself is not). Media
intermediaries should be as transparent as possible to the end
users--offering a useful, fundamental service; without getting in
the way of new features implemented by participants. Some common
media intermediaries are desribed below.

4.5.1.1 Mixer

A SIP mixer is a component that combines media from all dialogs in
the same conversation in a media specific way. For example, the
default combining for an audio conference would be an N-1
configuration, while the same mixer might interleave text messages
on a per-line basis.

Conventions for specifying a mixing or conferencing service in a SIP
URI are proposed in [ms-uri].

4.5.1.2 **Transcoder**

A transcoder translates media from one encoding or format to another
(for example, GSM voice to G.711, MPEG2 to H.261, or text/html to
text/plain).

4.5.1.3 **Media Relay**

A media relay terminates media and simply forwards it to a new
destination without changing the content in any way. Sometimes
media relays are used to provide source IP address anonymity, to
facilitate middlebox traversal, or to provide a trusted entity where
media can be forcefully disconnected.

4.5.2 **Queue Server**

A queue server is a location where calls can be entered into one of
several FIFO (first-in, first-out) queues. A queue server would
subscribe to the presence of groups or individuals who are
interested in its queues. When detecting that a user is available
to service a queue, the server redirects or transfers the last call
in the relevant queue to the available user. On a queue-by-queue
basis, authorized users could also subscribe to the call state
(dialog information) of calls within a queue. Authorized users
could use this information to effectively pluck (take) a call out of
the queue (for example by sending an INVITE with a Replaces header
to one of the user agents in the queue).

4.5.3 **Parking Place**

A parking place is a location where calls can be terminated
temporarily and then retrieved later. While a call is "parked", it
can receive media "on-hold" such as music, announcements, or
advertisements. Such a service could be further decomposed such
that announcements or music are handled by a separate component.

4.5.4 **Announcements and Voice Dialogs**

An announcement server is a server which can play digitized media
(frequently audio), such as music or recorded speech. These servers
are typically accessible via SIP, HTTP, or RTSP. An analogous
service is a recording service which stores digitized media. A
convention for specifying announcements in SIP URIs is described in
[ms-uri]. Likewise the same server could easily provide a service
which records digitized media.

A "voice dialog" is a model of spoken interactive behavior between a
human and an automaton which can include synthesized speech,

digitized audio, recognition of spoken and DTMF key input, recording
of spoken input, and interaction with call control. Dialogs
frequently consist of forms or menus. Forms present information and
gather input; menus offer choices of what to do next.

Spoken dialogs are a basic building block of applications which use voice. Consider for example that a voice mail system, the conference-id and passcode collection system for a conferencing system, and complicated voice portal applications all require a voice dialog component.

4.5.4.1. **Text-to-Speech and Automatic Speech Recognition**

Text-to-Speech (TTS) is a service which converts text into digitized audio. TTS is frequently integrated into other applications, but when separated as a component, it provides greater opportunity for broad reuse. Various interfaces to access standalone TTS services via HTTP, [CATS], and SIP ([app-components], and [ms-uri]) have been proposed.

Automatic Speech Recognition (ASR) is a service which attempts to decipher digitized speech based on a proposed grammar. Like TTS, ASR services can be embedded, or exposed so that many applications can take advantage of such services. Various IP interfaces to ASR, such as CATS, have been proposed.

4.5.4.2. **VoiceXML**

[VoiceXML] is a W3C recommendation that was designed to give authors control over the spoken dialog between users and applications. The application and user take turns speaking: the application prompts the user, and the user in turn responds. Its major goal is to bring the advantages of web-based development and content delivery to interactive voice response applications. We believe that VoiceXML represents the ideal partner for SIP in the development of distributed IVR servers. VoiceXML is an XML based scripting language for describing IVR services at an abstract level. VoiceXML supports DTMF recognition, speech recognition, text-to-speech, and playing out of recorded media files. The results of the data collected from the user are passed to a controlling entity through an HTTP POST operation. The controller can then return another script, or terminate the interaction with the IVR server.

A VoiceXML server also need not be implemented as a monolithic server. Below is a diagram of a VoiceXML browser which is split into media and non-media handling parts. The VoiceXML interpreter handles SIP dialog state and state within a VoiceXML document, and sends requests to the media component over another protocol (for example [RTSP] or CATS).

```
            +-------------+
            |             |
```

```
                              | VoiceXML    |
                              | Interpreter |
                              | (signaling) |
                              +-------------+
```
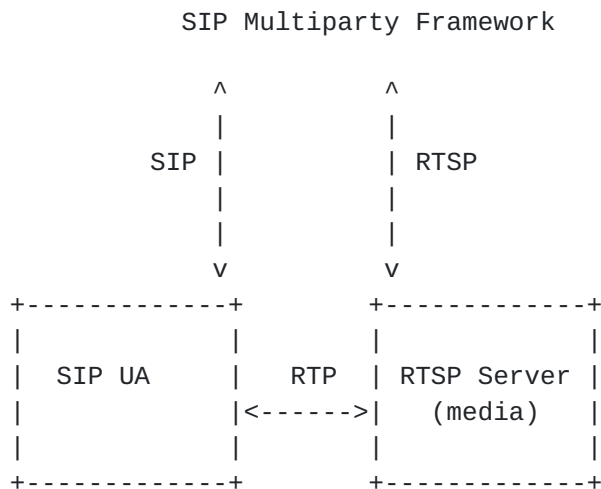
```
                    SIP Multiparty Framework


                         ^              ^
                         |              |
                    SIP  |              |  RTSP
                         |              |
                         |              |
                         v              v
              +-------------+    +-------------+
              |             |    |             |
              |   SIP UA    |    | RTSP Server |
              |             |RTP |             |
              |             |<------>|  (media)  |
              |             |    |             |
              +-------------+    +-------------+



               Figure : Decomposed VoiceXML Server
```

More details about the integration of SIP with VoiceXML are provided
in [sip-vxml]

4.6 Use of URIs

All naming in SIP uses URIs. URIs in SIP are used in a plethora of
contexts: the Request-URI; Contact, To, From, and *-Info headers;
application/uri bodies; and embedded in email, web pages, instant
messages, and ENUM records. The request-URI identifies the user or
service that the call is destined for.

SIP URIs embedded in informational SIP headers, SIP bodies, and non-
SIP content can also specify methods, special parameters, headers,
and even bodies. For example:

sip:bob@babylon.biloxi.com;method=BYE?Call-ID=13413098
  &To=<sip:bob@biloxi.com>;tag=879738
  &From=<sip:alice@atlanta.com>;tag=023214

sip:bob@babylon.biloxi.com;method=REFER?
  Refer-To=<http://www.atlanta.com/~alice>

Throughout this draft we discuss call control primitive operations.
One of the biggest problems is defining how these operations may be
invoked. There are a number of ways to do this. One way is to
define the primitives in the protocol itself such that SIP methods
(for example REFER) or SIP headers (for example Replaces) indicate a
specific call control action. Another way to invoke call control
primitives is to define a specific Request-URI naming convention.
Either these conventions must be shared between the client (the
invoker) and the server, or published by or on behlf of the server.
The former involves defining URL construction techniques (e.g. URL
parameters and/or token conventions) as proposed in [ms-uri]. The
```

latter technique usually involves discovering the URI via a SIP
event package, a web page, a business card, or an Instant Message.
Yet another means to acquire the URLs is to define a dictionary of
primitives with well-defined semantics and provide a means to query

the named primitives and corresponding URLs that may be invoked on
the service or dialogs.

## 4.6.1 Naming Users in SIP

An address-of-record, or public SIP address, is a SIP (or SIPS) URI
that points to a domain with a location server that can map the URI
to set of Contact URIs where the user might be available.  Typically
the Contact URIs are populated via registration.

```
    Address of Record        Contacts

    sip:bob@biloxi.com    ->  sip:bob@babylon.biloxi.com:5060
                              sip:bbrown@mailbox.provider.net
                              sip:+1.408.555.6789@mobile.net
```

[Caller-prefs] defines a set of additional parameters to the Contact
header that define the characteristics of the user agent at the
specified URI. For example, there is a mobility parameter which
indicates whether the UA is fixed or mobile. When a user agent
registers, it places these parameters in the Contact headers to
characterize the URIs it is registering. This allows a proxy for
that domain to have information about the contact addresses for that
user.

When a caller sends a request, it can optionally include the Accept-
Contact and Reject-Contact headers which request certain handling by
the proxy in the target domain. These headers contain preferences
that describe the set of desired URIs to which the caller would like
their request routed. The proxy in the target domain matches these
preferences with the Contact characteristics originally registered
by the target user. The target user can also choose to run
arbitrarily complex "Find-me" feature logic on a proxy in the target
domain.

There is a strong asymmetry in how preferences for callers and
callees can be presented to the network. While a caller takes an
active role by initiating the request, the callee takes a passive
role in waiting for requests. This motivates the use of callee-
supplied scripts and caller preferences included in the call
request. This asymmetry is also reflected in the appropriate
relationship between caller and callee preferences. A server for a
callee should respect the wishes of the caller to avoid certain
locations, while the preferences among locations has to be the
callee's choice, as it determines where, for example, the phone
rings and whether the callee incurs mobile telephone charges for
incoming calls.

SIP User Agent implementations are encouraged to make intelligent

decisions based on the type of participants (active/passive, hidden,
human/robot) in a conversation space. This information is conveyed
in a SIP URI parameter and communicated using an appropriate SIP
header or event body. For example, a music on hold service may take
the sensible approach that if there are two or more unhidden

participants, it should not provide hold music; or that it will not
send hold music to robots.

Multiple participants in the same conversation space may represent
the same human user. For example, the user may use one participant
for video, chat, and whiteboard media on a PC and another for audio
media on a SIP phone. In this case, the address-of-record is the
same for both user agents, but the Contacts are different. In
addition, human users may add robot participants which act on their
behalf (for example a call recording service, or a calendar
reminder). Call Control features in SIP should continue to function
as expected in such an environment.

4.6.2 **Naming Services with SIP URIs.**

   A critical piece of defining a session level service that can be
   accessed by SIP is defining the naming of the resources within that
   service. This point cannot be overstated.

   In the context of SIP control of application components, we take
   advantage of the fact that the standard SIP URI has a user part.
   Most services may be thought of as user automatons that participate
   in SIP sessions. It naturally follows that the user address, or the
   left-hand-side of the URI, should be utilized as a service
   indicator.

   For example, media servers commonly offer multiple services at a
   single host address. Use of the user part as a service indicator
   enables service consumers to direct their requests without
   ambiguity. It has the added benefit of enabling media services to
   register their availability with SIP Registrars just as any "real"
   SIP user would. This maintains consistency and provides enhanced
   flexibility in the deployment of media services in the network.

   There has been much discussion about the potential for confusion if
   media services URIs are not readily distinguishable from other types
   of SIP UA's. The use of a service namespace provides a mechanism to
   unambiguously identify standard interfaces while not constraining
   the development of private or experimental services.

   In SIP, the request-URI identifies the user or service that the call
   is destined for. The great advantage of using URIs (specifically,
   the SIP request URI) as a service identifier comes because of the
   combination of two facts. First, unlike in the PSTN, where the
   namespace (dialable telephone numbers) are limited, URIs come from
   an infinite space. They are plentiful, and they are free. Secondly,
   the primary function of SIP is call routing through manipulations of
   the request URI. In the traditional SIP application, this URI

represents people. However, the URI can also represent services, as
we propose here. This means we can apply the routing services SIP
provides to routing of calls to services. The result - the problem
of service invocation and service location becomes a routing
problem, for which SIP provides a scalable and flexible solution.

SIP Multiparty Framework

Since there is such a vast namespace of services, we can explicitly name each service in a finely granular way. This allows the distribution of services across the network.

Consider a conferencing service, where we have separated the names of ad-hoc conferences from scheduled conferences, we can program proxies to route calls for ad-hoc conferences to one set of servers, and calls for scheduled ones to another, possibly even in a different provider. In fact, since each conference itself is given a URI, we can distribute conferences across servers, and easily guarantee that calls for the same conference always get routed to the same server. This is in stark contrast to conferences in the telephone network, where the equivalent of the URI - the phone number - is scarce. An entire conferencing provider generally has one or two numbers. Conference IDs must be obtained through IVR interactions with the caller, or through a human attendant. This makes it difficult to distribute conferences across servers all over the network, since the PSTN routing only knows about the dialed number.

In the case of a dialog server, the voice dialog itself is the target for the call. As such, the request URI should contain the identifier for this spoken dialog. This is consistent with the Request-URI service invocation model of [RFC 3087](). This URL can be in one of two formats. In the first, the VoiceXML script is identified directly by an HTTP URL. In the second, the script is not specified. Rather, the dialog server uses its configuration to map the incoming request to a specific script.

Since the request URI could indicate a request for a variety of different services, of which a dialog server is only one type, this example request URI first begins with a service identifier, that indicates the basic service required. For VoiceXML scripts, this identification information is a URL-encoded version of the URL which references the script to execute, or if not present, the dialog server uses server-specific configuration to determine which script to execute.

   Examples of URLs that invoke VoiceXML dialogs are:
   (line folding for clarity only)

   sip:dialog.vxml.http%3a//dialogs.server.com/script32.vxml
    @vxmlservers.com

   sip:dialog.vxml@vxmlservers.com

The first of these indicates that the dialog server (located at vxmlservers.com) should invoke a VoiceXML script fetched from [http://dialogs.server.com/script32.vxml](). Since the user part of the

SIP URL cannot contain the : character, this must be escaped to %3a.

These types of conventions are not limited to application component servers. An ordinary SIP User Agent can have a special URIs as well, for example, one which is automatically answered by a

speakerphone. Since URIs are so plentiful, using a separate URI for
this service does not exhaust a valuable resource. The requested
service is clear to the user agent receiving the request. This URI
can also be included as part of another feature (for example, the
Intercom feature described in Section 6.1.6). This feature can be
specified with a SIP user parameter, since are part of the userpart
of a SIP URI.

Likewise a Request URI can fully describe an announcement service
through the use of the user part of the address and additional URI
parameters. In our example, the user portion of the address,
"annc", specifies the announcement service on the media server.
The two URI parameters "play=" and "early=" specify the audio
resource to play and whether early media is desired.

    sip:annc@ms2.carrier.net;
     play=http://audio.carrier.net/allcircuitsbusy.au;early=yes

    sip:annc@ms2.carrier.net;
     play=file://fileserver.carrier.net/geminii/yourHoroscope.wav

In practical applications, it is important that an invoker does not
necessarily apply semantic rules to various URIs it did not create.
Instead, it should allow any arbitrary string to be provisioned, and
map the string to the desired behavior. The administrator of a
service may choose to provision specific conventions or mnemonic
strings, but the application should not require it. In any large
installation, the system owner is likely to have pre-existing rules
for mnemonic URIs, and any attempt by an application to define its
own rules may create a conflict. Implementations should allow an
arbitrary mix of URLs from these schemes, or any other scheme that
renders valid SIP URIs to be provisioned, rather than enforce only
one particular scheme.

For example, a voicemail application can be built using very
different sets of URI conventions, as illustrated below:

       URI Identity   Example Scheme 1
                                 Example Scheme 2
                                     Example Scheme 3

       Deposit with   sip:sub-rjs-deposit@vm.wcom.com
       standard greeting   sip:677283@vm.wcom.com
                                   sip:rjs@vm.wcom.com;mode=deposit

       Deposit with on   sip:sub-rjs-deposit-busy.vm.wcom.com
       phone greeting   sip:677372@vm.wcom.com
                                   sip:rjs@vm.wcom.com;mode=3991243

```
       Deposit with       sip:sub-rjs-deposit-sg@vm.wcom.com
       special greeting       sip:677384@vm.wcom.com
                                 sip:rjs@vm.wcom.com;mode=sg
```

```
    Retrieve - SIP      sip:sub-rjs-retrieve@vm.wcom.com
    authentication          sip:677405@vm.wcom.com
                                sip:rjs@vm.wcom.com;mode=retrieve


    Retrieve - prompt   sip:sub-rjs-retrieve-inpin.vm.wcom.com
    for PIN in-band         sip:677415@vm.wcom.com
                                sip:rjs@vm.wcom.com;mode=inpin
```

As we have shown, SIP URIs represent an ideal, flexbile mechanism
for describing and naming service resources, be they queues,
conferences, voice dialogs, announcements, voicemail treatments, or
phone features.

4.7 Invoker Independence

Only the invoker of features in SIP need to know exactly which
feature they are invoking. One of the primary benefits of this
approach is that combinations of features should work in SIP call
control. For example, let us examine the combination of a
"transfer" of a call which is "conferenced".

Alice calls Bob. Alice silently "conferences in" her robotic
assistant Albert as a hidden party. Bob transfers Alice to Carol.
If Bob asks Alice to Replace her leg with a new one to Carol then
both Alice and Albert should be communicating with Carol
(transparently).

Using the peer-to-peer model, this combination of features works
fine if A is doing local mixing (Alice replaces Bob's call-leg with
Carol's), or if A is using a central mixer (the mixer replaces Bob's
call leg with Carol's). A clever implementation using the 3pcc
model can generate similar results.

New extensions to the SIP Call Control Framework should attempt to
preserve this property.

4.8 Billing issues

Billing in the PSTN is typically based on who initiated a call. At
the moment billing in a SIP network is neither consistent with
itself, nor with the PSTN. (A billing model for SIP should allow
for both PSTN-style billing, and non-PSTN billing.) The example
below demonstrates one such inconsistency.

Alice places a call to Bob. Alice then blind transfers Bob to Carol
through a PSTN gateway. In current usage of REFER and BYE/Also, Bob
may be billed for a call he did not initiate (his UA originated the
outgoing call leg however). This is not necessarily a terrible

thing, but it demonstrates a security concern (Bob must have
appropriate local policy to prevent fraud). Also, Alice may wish to
pay for Bob's session with Carol. There should be a way to signal
this in SIP.

Likewise a Replacement call may maintain the same billing
relationship as a Replaced call, so if Alice first calls Carol, then
asks Bob to Replace this call, Alice may continue to receive a bill.

Further work in SIP billing should define a way to set or discover
the direction of billing.

5 Catalog of call control actions and sample features

Call control actions can be categorized by the dialogs upon which
they operate. The actions may involve a single or multiple dialogs.
These dialogs can be early or established. Multiple dialogs may be
related in a conversation space to form a conference or other
interesting media topologies.

It should be noted that it is desirable to provide a means by which
a party can discover the actions which may be performed on a dialog.
The interested party may be independent or related to the dialogs.
One means of accomplishing this is through the ability to define and
obtain URLs for these actions as described in section 4.6.

Below are listed several call control "actions" which establish or
modify dialogs and relate the participants in a conversation space.
The names of the actions listed are for descriptive purposes only
(they are not normative). This list of actions is not meant to be
exhaustive.

In the examples, all actions are initiated by the user "Alice"
represented by UA "A".

5.1 Early Dialog Actions

The following are a set of actions that may be performed on a single
early dialog. These actions can be thought of as a set of remote
control operations. For example an automaton might perform the
operation on behalf of a user. Alternatively a user might use the
remote control in the form of an application to perform the action
on the early dialog of a UA which may be out of reach. All of these
actions correspond to telling the UA how to respond to a request to
establish an early dialog. These actions provide useful
functionality for PDA, PC and server based applications which desire
the ability to control a UA.

5.1.1 Remote Answer

A dialog is in some early dialog state such as 180 Ringing. It may
be desirable to tell the UA to answer the dialog. That is tell it
to send a 200 Ok response to establish the dialog.

5.1.2 **Remote Forward or Put**

It may be desirable to tell the UA to respond with a 3xx class
response to forward an early dialog to another UA.

5.1.3 Remote Busy or Error Out

It may be desirable to instruct the UA to send an error response
such as 486 Busy Here.

5.2 Single Dialog Actions

There is another useful set of actions which operate on a single
established dialog. These operations are useful in building
productivity applications for aiding users to control their phone.
For example a CRM application which sets up calls for a user
eliminating the need for the user to actually enter an address.
These operations can also be thought of a remote control actions.

5.2.1 Remote Dial

This action instructs the UA to initiate a dialog. This action can
be performed using the REFER method.

5.2.2 Remote On and Off Hold

This action instructs the UA to put an established dialog on hold.
Though this operation can be conceptually be performed with the
REFER method, there is no semantics defined as to what the referred
party should do with the SDP. There is no way to distinguish between
the desire to go on or off hold.

5.2.3 Remote Hangup

This action instructs the UA to terminate an early or established
dialog. A REFER request with the following Refer-To URI performs
this action. Note: this URL is not properly escaped.

```
sip:bob@babylon.biloxi.com;method=BYE?Call-ID=13413098
  &To=<sip:bob@biloxi.com>;tag=879738
  &From=<sip:alice@atlanta.com>;tag=023214
```

5.3 Multi-dialog actions

These actions apply to a set of related dialogs.

5.3.1 Transfer

The conversation space changes as follows:

        before   after

```
        { A , B }  -->   { C , B }

   A replaces itself with C.
```

   To make this happen using the peer-to-peer approach, "A" would send
   two SIP requests. A shorthand for those requests is shown below:
        REFER B Refer-To:C
        BYE B

   To make this happen instead using the 3pcc approach, the controller
   sends requests represented by the shorthand below:
        INVITE C (w/SDP of B)
        reINVITE B (w/SDP of C)
        BYE A

   Features enabled by this action:
   - blind transfer
   - transfer to a central mixer (some type of conference or forking)
   - transfer to park server (park)
   - transfer to music on hold or announcement server
   - transfer to a "queue"
   - transfer to a service (such as Voice Dialogs service)
   - transition from local mixer to central mixer

[5.3.2](#) Take

   The conversation space changes as follows:

        { B , C } -->   { B , A }

   A forcibly replaces C with itself. In most uses of this primitive,
   A is just "un-replacing" itself.

   Using the peer-to-peer approach, "A" sends:
        INVITE B Replaces: <call leg between B and C>

   Using the 3pcc approach (all requests sent from controller)
        INVITE A (w/SDP of B)
        reINVITE B (w/SDP of A)
        BYE C

   Features enabled by this action:
   - transferee completes an attended transfer
   - retrieve from central mixer (not recommended)
   - retrieve from music on hold or park
   - retrieve from queue
   - call center take
   - voice portal resuming ownership of a call it originated
   - answering-machine style screening (pickup)
   - pickup of a ringing call (i.e. early dialog)

   Note: that pick up of a ringing call has perhaps some interesting
   additional requirements. First of all it is an early dialog as
   opposed to an established dialog. Secondly the party which is to

pickup the call may only wish to do so only while it is an early
dialog. That is in the race condition where the ringing UA accepts
just before it receives signaling from the party wishing to take the

call, the taking party wishes to yield or cancel the take. The goal
is to avoid yanking an answered call from the called party.

5.3.3 Add

The conversation space changes as follows:

```
    { A , B } -->    { A, B, C }
```

A adds C to the conversation.

Using the peer-to-peer approach, adding a party using local mixing
requires no signaling. To transition from a 2-party call or a
locally mixed conference to centrally mixing A could send the
following requests:
```
    REFER B  Refer-To: mixer
    INVITE mixer
    BYE B
```

To add a party to a central mixer:
```
    REFER C  Refer-To: mixer
          or
    REFER mixer  Refer-To: C
```

Using the 3pcc approach to transition to centrally mixed, the
controller would send:
```
    INVITE mixer leg 1 (w/SDP of A)
    INVITE mixer leg 2 (w/SDP of B)
    INVITE C (late SDP)
    reINVITE A (w/SDP of mixer leg 1)
    reINVITE B (w/SDP of mixer leg 2)
    INVITE mixer leg3 (w/SDP of C)
```

To add a party to a central mixer:
```
    INVITE C (late SDP)
    INVITE mixer (w/SDP of C)
```

Features enabled:
- standard conference feature
- call recording
- answering-machine style screening (screening)

5.3.4 Local Join

The conversation space changes like this:

```
    { A, B}  , {A, C}  -->  {A, B, C}
```

```
        or like this
```

```
{ A, B}  , {C, D}  -->  {A, B, C, D}
```

A takes two conversation spaces and joins them together into a
single space.

Using the peer-to-peer approach, A can mix locally, or REFER the participants of both conversation spaces to the same central mixer (as in 5.3)

For the 3pcc approach, the call flows for inserting participants, and joining and splitting conversation spaces are tedious yet straightforward, so these are left as an exercise for the reader.

Features enabled:
- standard conference feature
- leaving a sidebar to rejoin a larger conference

5.3.5 Insert

The conversation space changes like this:

        { B , C } --> {A, B, C }

A inserts itself into a conversation space.

A proposed mechanism for signaling this using the peer-to-peer approach is to send a new header in an INVITE with "joining" semantics. For example:
        INVITE B Join: <call id of B and C>

If B accepted the INVITE, B would accept responsibility to setup the call legs and mixing necessary (for example: to mix locally or to transfer the participants to a central mixer)

Features enabled:
- barge-in
- call center monitoring
- call recording

5.3.6 Split
    { A, B, C, D } --> { A, B } , { C, D }

If using a central mixer with peer-to-peer
REFER C Refer-To: mixer (new URI)
REFER D Refer-To: mixer (new URI)
BYE C
BYE D

Features enabled:
- sidebar conversations during a larger conference

5.3.7 Near-fork

A participates in two conversation spaces simultaneously:

        { A, B } --> { B , A } & { A , C }

   A is a participant in two conversation spaces such that A sends the
   same media to both spaces, and renders media from both spaces,
   presumably by mixing or rendering the media from both. We can
   define that A is the "anchor" point for both forks, each of which is
   a separate conversation space.

   This action is purely local implementation (it requires no special
   signaling). Local features such as switching calls between the
   background and foreground are possible using this media
   relationship.

5.3.8 Far fork

   The conversation space diagram...

       { A, B } --> { A , B } & { B , C }

   A requests B to be the "anchor" of two conversation spaces.

   For an example of using 3pcc to setup media forking, see [Media
   forking]. The session descriptions for forking are quite complex.
   Controllers should verify that endpoints can handle forked-media, by
   using some type of Requires header token.

   Two ways to setup this media relationship using peer-to-peer call
   control have been proposed:
   - the anchor receives a REFER with requires forked-media (implicit)
   - the anchor receives an INVITE with an explicit header (explicit)

   Features enabled:
   - barge-in
   - voice portal services
   - whisper
   - hotword detection
   - sending DTMF somewhere else

6 Security Considerations

   Call Control primitives provide a powerful set of features that can
   be dangerous in the hands of an attacker. To complicate matters,
   call control primitives are likely to be automatically authorized
   without direct human oversight.

   The class of attacks which are possible using these tools include
   the ability to eavesdrop on calls, disconnect calls, redirect calls,
   render irritating content (including ringing) at a user agent, cause

an action that has billing consequences, subvert billing (theft-of-
service), and obtain private information. Call control extensions
must take extra care to describe how these attacks will be
prevented.

We can also make some general observations about authorization and trust with respect to call control. The security model is dramatically dependent on the signaling model chosen (see section 4.2)

Let us first examine the security model used in the 3pcc approach. All signaling goes through the controller, which is a trusted entity. Traditional SIP authentication and hop-by-hop encrpytion and message integrity work fine in this environment, but end-to-end encrpytion and message integrity may not be possible.

When using the peer-to-peer approach, call control actions and primitives can be legitimately initiated by a) an existing participant in the conversation space, b) a former participant in the conversation space, or c) an entity trusted by one of the participants. For example, a participant always initiates a transfer; a retrieve from Park (a take) is initiated on behalf of a former participant; and a barge-in (insert or far-fork) is initiated by a trusted entity (an operator for example).

Authenticating requests by an existing participant or a trusted entity can be done with baseline SIP mechanisms. In the case of features initiated by a former participant, these should be protected against replay attacks by using a unique name or identifier per invocation. The Replaces header exhibits this behavior as a by-product of its operation (once a Replaces operation is successful, the call-leg being Replaced no longer exists). For other requests, a "one-time" Request-URI may be provided to the feature invoker.

To authorize call control primitives that trigger special behavior (such as an INVITE with Replace, Join, or Fork semantics), the receiving user agent may have trouble finding appropriate credentials with which to challenge or authorize the request, as the sender may be completely unknown to the receiver, except through the introduction of a third party. These credentials need to be passed transitively in some way or fetched in an event body, for example.

7 Appendix A: Example Features

Primitives are defined in terms of their ability to provide features. These example features should require an amply robust set of services to demonstrate a useful set of primitives. They are described here briefly. Note that the descriptions of these features are non-normative. Some of these features are used as examples in section 6 to demonstrate how some features may require certain media relationships. Note also that this document describes a mixture of

both features originating in the world of telephones, and features
which are clearly Internet oriented.

7.1 **Example Feature Definitions:**

SIP Multiparty Framework

Call Waiting - Alice is in a call, then receives another call.
Alice can place the first call on hold, and talk with the other
caller. She can typically switch back and forth between the
callers.

Blind Transfer - Alice is in a conversation with Bob. Alice asks
Bob to contact Carol, but makes no attempt to contact Craol
independently. In many implementations, Alice does not verify Bob's
success or failure in contacting Carol.

Attended Transfer - The transferring party establishes a session
with the transfer target before completing the transfer.

Consultative transfer - the transferring party establishes a session
with the target and mixes both sessions together so that all three
parties can participate, then disconnects leaving the transferee and
transfer target with an active session.

Conference Call - Three or more active, visible participants in the
same conversation space.

Call Park - A call participant parks a call (essentially puts the
call on hold), and then retrieves it at a later time (typically from
another location).

Call Pickup - A party picks up a call that was ringing at another
location. One variation allows the caller to choose which location,
another variation just picks up any call in that user's "pickup
group".

Music on Hold - When Alice places a call with Bob on hold, it
replaces its audio with streaming content such as music,
announcements, or advertisements.

Call Monitoring - A call center supervisor joins an in-progress call
for monitoring purposes.

Barge-in - Carol interrupts Alice who has a call in-progress call
with Bob. In some variations, Alice forcibly joins a new
conversation with Carol, in other variations, all three parties are
placed in the same conversation (basically a 3-way conference).

Hotline - Alice picks up a phone and is immediately connected to the
technical support hotline, for example.

Autoanswer - Calls to a certain address or location answer
immediately via a speakerphone.

Intercom - Alice typically presses a button on a phone which
immediately connects to another user or phone and casues that phone

to play her voice over its speaker. Some variations immediately
setup two-way communications, other variations require another
button to be pressed to enable a two-way conversation.

SIP Multiparty Framework

Speakerphone paging - Alice calls the paging address and speaks.
Her voice is played on the speaker of every idle phone in a
preconfigured group of phones.

Speed dial - Alice dials an abbreviated number, or enters an alias,
or presses a special speed dial button representing Bob. Her action
is interpreted as if she specified the full address of Bob.

Call Return - Alice calls Bob. Bob misses the call or is
disconnected before he is finished talking to Alice. Bob invokes
Call return which calls Alice, even if Alice did not provide her
real identity or location to Bob.

Inbound Call Screening - Alice doesn't want to receive calls from
Matt. Inbound Screening prevents Matt from disturbing Alice. In
some variations this works even if Matt hides his identity.

Outbound Call Screening - Alice is paged and unknowingly calls a
PSTN pay-service telephone number in the Carribean, but local policy
blocks her call, and possibly informs her why.

Call Forwarding - Before a call-leg is accepted it is redirected to
another location, for example, because the originally intended
recipient is busy, does not answer, is disconnected from the
network, configured all requests to go soemwhere else.

Message Waiting - Bob calls Alice when she steps away from her
phone, when she returns a visible or audible indicator conveys that
someone has left her a voicemail message. The message waiting
indication may also convey how many messages are waiting, from whom,
what time, and other useful pieces of information.

Do Not Disturb - Alice selects the Do Not Disturb option. Calls to
her either ring briefly or not at all and are forwarded elsewhere.
Some variations allow specially authorized callers to override this
feature and ring Alice anyway.

Distinctive ring - Incoming calls have different ring cadences or
sample sounds depending on the From party, the To party, or other
factors.

Automatic Callback: Alice calls Bob, but Bob is busy. Alice would
like Bob to call her automatically when he is available. When Bob
hangs up, alice's phone rings. When Alice answers, Bob's phone
rings. Bob answers and they talk.

Find-Me - Alice sets up complicated rules for how she can be reached
(possibly using [CPL], [presence] or other factors). When Bob calls
Alice, his call is eventually routed to a temporary Contact where
Alice happens to be available.

Whispered call waiting - Alice is in a conversation with Bob. Carol
    calls Alice. Either Carol can "whisper" to Alice directly ("Can you

get lunch in 15 minutes?"), or an automaton whispers to Alice informing her that Carol is trying to reach her.

Voice message screening - Bob calls Alice. Alice is screening her calls, so Bob hears Alice's voicemail greeting. Alice can hear Bob leave his message. If she decides to talk to Bob, she can take the call back from the voicemail system, otherwise she can let Bob leave a message. This emulates the behavior of a home telephone answering machine

Presence-Enabled Conferencing: Alice wants to set up a conference call with Bob and Cathy when they all happen to be available (rather than scheduling a predefined time). The server providing the application monitors their status, and calls all three when they are all "online", not idle, and not in another call.

IM Conference Alerts: A user receives an notification as an Instant Message whenever someone joins a conference they are also in.

Single Line Extension -- A group of phones are all treated as "extensions" of a single line. A call for one rings them all. As soon as one answers, the others stop ringing. If any extension is actively in a coversation, another extension can "pick up" and immediately join the conversation. This emulates the behavior of a home telephone line with multiple phones.

Click-to-dial - Alice looks in her company directory for Bob. When she finds Bob, she clicks on a URL to call him. Her phone rings (or possibly answers automatically), and when she answers, Bob's phone rings.

Pre-paid calling - Alice pays for a certain currency or unit amount of calling value. When she places a call, she provides her account number somehow. If her account runs out of calling value during a call her call is disconnected or redirected to a service where she can purchase more calling value.

Voice Portal - A service that allows users to access a portal site using spoken dialog interaction. For example, Alice needs to schedule a working dinner with her co-worker Carol. Alice uses a voice portal to check Carol's flight schedule, find a restauraunt near her hotel, make a reservation, get directions there, and page Carol with this information.

7.2 Implementation of these features

```
Example Features:
Call Hold                 [Offer/Answer] for SIP
Call Waiting                  Local Implementation
```

```
Blind Transfer             [cc-transfer]
Attended Transfer          [cc-transfer]
Consultative transfer      [cc-transfer]
Conference Call            [conf-models]
```

SIP Multiparty Framework

       Call Park   *[examples]
       Call Pickup   *[examples]
       Music on Hold   *[examples]
       Call Monitoring   *Insert
       Barge-in   *Insert or Far-Fork
       Hotline   Local Implementation
       Autoanswer   Local URI convention
       Speed dial   Local Implementation
       Intercom   *Speed dial + autoanswer
       Speakerphone paging   *Speed dial + autoanswer
       Call Return   Proxy feature
       Inbound Call Screening   Proxy or Local implementation
       Outbound Call Screening Proxy feature
       Call Forwarding   Proxy or Local implementation
       Message Waiting   [msg-waiting]
       Do Not Disturb   [presence]
       Distinctive ring   *Proxy or Local implementation
       Automatic Callback   2 person presence-based conference
       Find-Me   Proxy service based on presence
       Whispered call waiting   Local implementation
       Voice message screening *
       Presence-based Conferencing*call when presence = available
       IM Conference Alerts   subscribe to conference status
       Single Line Extension   *
       Click-to-dial   *
       Pre-paid calling   *
       Voice Portal   *

7.2.1 Call Park

   Call park requires the ability to: put a dialog some place,
   advertise it to users in a pickup group and to uniquely identify it
   in a means that can be communicated (including human voice). The
   dialog can be held locally on the UA parking the dialog or
   alternatively transferred to the park service for the pickup group.
   The parked dialog then needs to be labeled (e.g. orbit 12) in a way
   that can be communicated to the party that is to pick up the call.
   The UAs in the pick up group discovers the parked dialog(s) via
   [call-leg] from the park service. If the dialog is parked locally
   the park service merely aggregates the parked call states from the
   set of UAs in the pickup up group.

7.2.2 Call Pickup

   There are two different features which are called call pickup. The
   first is the pickup of a parked dialog. The UA from which the
   dialog is to be picked up subscribes to the call state [call-leg] of
   the park service or the UA which has locally parked the dialog.
   Dialogs which are parked should be labeled with an identifier. The

labels are used by the UA to allow the user to indicate which dialog
is to be picked up. The UA picking up the call invoked the URL in
the call state which is labeled as replace-remote.

The other call pickup feature involves picking up an early dialog
(typically ringing). This feature uses some of the same primitives
as the pick up of a parked call. The call state of the UA ringing
phone is advertised using [call-leg]. The UA which is to pickup the
early dialog subscribes either directly to the ringing UA or to a
service aggregating the states for UAs in the pickup group. The
call state identifies early dialogs. The UA uses the call state(s)
to help the user choose which early dialog that is to be picked up.
The UA then invokes the URL in the call state labeled as replace-
remote.

7.2.3 Music on Hold

Music on hold can be implemented a number of ways. One way is to
transfer the held call to a holding service. When the UA wishes to
take the call off hold it basically performs a take on the call from
the holding service. This involves subscribing to call state on the
holding service and then invoking the URL in the call state labeled
as replace-remote.

Alternatively music on hold can be performed as a local mixing
operation. The UA holding the call can mix in the music from the
music service via RTP (i.e. an additional dialog) or RTSP or other
streaming media source. This approach is simpler (i.e. the held
dialog does not move so there is less chance of loosing them) from a
protocol perspective, however it does use more LAN bandwidth and
resources on the UA.

7.2.4 Call Monitoring

Call monitoring is a [join] operation. The monitoring UA sends a
Join to the dialog it wants to listen to. It is able to discover
the dialog via the call state [call-leg] on the monitored UA. The
monitoring UA sends SDP in the INVITE which indicates receive only
media {offer/answer}. IN addition the monitoring UA should indicate
that it wants to receive a mix (see Error! Reference source not
found.). As the UA is monitoring only it does not matter whether
the UA indicates it wishes the send stream be mix or point to point.

7.2.5 Barge-in

Barge-in works the same as call monitoring except that it must
indicate that the send media stream to be mixed so that all of the
other parties can hear the stream from UA barging in.

7.2.6 Intercom

The UA initiates a dialog using INVITE in the ordinary way [bis].
The calling UA then signals the paged UA to answer the call. The
calling UA may discover the URL to answer the call via the call

state [call-leg] of the called UA. The called UA accepts the INVITE
with a 200 Ok and automatically enables the speakerphone.

Alternatively this can be a local decision for the UA to answer
based upon called party identification.

7.2.7 Speakerphone paging

Speakerphone paging can be implemented using either multicast or
through a simple multipoint mixer. In the multicast solution the
paging UA sends a multicast INVITE [bis] with send only media in the
[SDP] (see also [offer/answer]). The automatic answer and enabling
of the speakerphone is a locally configured decision on the paged
UAs. The paging UA sends RTP via the multicast address indicated in
the SDP.

The multipoint solution is accomplished by sending an INVITE to the
multipoint mixer. The mixer is configured to automatically answer
the dialog. The paging UA then sends [REFER] requests for each of
the UAs that are to become paging speakers (The UA is likely to send
out a single REFER which is parallel forked by the proxy server).
The UAs performing as paging speakers are configured to
automatically answer based upon caller identification (e.g. To
field, URI or Referred-To headers).

7.2.8 Distinctive ring

The target UA either makes a local decision based on information in
an incoming INVITE (To, From, Contact, Request-URI) or trusts an
Alert-Info header provded by the caller or inserted by a trusted
proxy. In the latter case, the UA fetches the content described in
the URI (typically via http) and renders it to the user.

7.2.9 Voice message screening

At first, this is the same as call monitoring. In this case the
voicemail service is one of the UAs. The UA screening the message
monitors the call on the voicemail service, and also subscribes to
call-leg information. If the user screening their messages decides
to answer, they perform a Take from the voicemail system (for
example, send an INVITE with Replaces to the UA leaving the message)

7.2.10 Single Line Extension

Incoming calls ring all the extensions through basic parallel
forking [bis]. Each extension subscribes to call-leg events from
each other extension. While one user has an active call, any other
UA extension can insert itself into that conversation (it already
knows the call-leg information)in the same way as barge-in.

7.2.11 Click-to-dial

The application or server which hosts the click-to-dial application

captures the URL to be dialed and can setup the call using 3pcc or
can send a [REFER] request to the UA which is to dial the address.
As users sometimes change their mind or wish to give up listing to a

ringing or voicemail answered phone, this application illustrates the need to also have the ability to remotely hangup a call.

7.2.12 Pre-paid calling

For prepaid calling, the user's media always passes through a device which is trusted by the pre-paid provider. This may be the other endpoint (for example a PSTN gateway). In either case, an intermediary proxy or B2BUA can periodically verify the amount of time available on the pre-paid account, and use the session-timer extension to cause the trusted endpoint (gateway) or intermediary (media relay) to send a reINVITE before that time runs out. During the reINVITE, the SIP intermediary can reverify the account and insert another session-timer header.

Note that while most pre-paid systems on the PSTN use an IVR to collect the account number and destination, this isn't strictly necessary for a SIP-originated prepaid call. SIP requests and SIP URIs are sufficiently expressive to convey the final destination, the provider of the prepaid service, the location from which the user is calling, and the prepaid account they want to use. If a pre-paid IVR is used, the mechanism described below (Voice Portals) can be combined as well.

7.2.13 Voice Portal

A voice portal is essentially a complex collection of voice dialogs used to access interesting content. One of the most desirable call control features of a Voice Portal is the ability to start a new outgoing call from within the context of the Portal (to make a restauraunt reservation, or return a voicemail message for example). Once the new call is over, the user should be able to return to the Portal by pressing a special key, using some DTMF sequence (ex: a very long pound or hash tone), or by speaking a hotword (ex: "Main Menu").

In order to accomplish this, the Voice Portal starts with the following media relationship:

    { User , Voice Portal }

The user then asks to make an outgoing call. The Voice Portal asks the User to perform a Far-Fork. In other words the Voice Portal wants the following media relationship:

    { Target , User } & { User , Voice Portal }

The Voice Portal is now just listening for a hotword or the appropriate DTMF. As soon as the user indicates they are done, the Voice Portal Takes the call from the old Target, and we are back to

the original media relationship.

This feature can also be used by the account number and phone number collection menu in a pre-paid calling service. A user can press a DTMF sequence which presents them with the a

8 **References**

[SIP] M. Handley, E. Schooler, and H. Schulzrinne, "SIP: Session Initiation Protocol", RFC2543, Internet Engineering Task Force, Nov 1998.

[RFC2119] S. Bradner, "Key words for use in RFCs to indicate requirement   levels," Request for Comments (Best Current Practice) 2119, Internet   Engineering Task Force, Mar. 1997.

[REFER] R. Sparks, "The Refer Method", Internet Draft <draft-ietf-sip-refer-02>, IETF, October 30, 2001, Work in progress.

[3pcc] J. Rosenberg, J. Peterson, H. Schulzrinne, G. Camarillo, "Third Party Call Control in SIP", Internet Draft <draft-rosenberg-sip-3pcc-02.txt>, IETF; March 2001. Work in progress

[transfer] R. Sparks, "SIP Call Control - Transfer", Internet Draft <draft-ietf-sip-cc-transfer-04.txt>, IETF; Feb. 2001. Work in progress.

[Replaces] B. Biggs, R. Dean, R. Mahy, "The SIP Replaces Header", Internet Draft <draft-ietf-sip-replaces-00.txt>, IETF, Nov. 2001. Work in progress.

[conf-models] J. Rosenberg, H. Schulzrinne, "Models for Multi Party Conferencing in SIP", Internet Draft <draft-rosenberg-sip-conferencing-models-00.txt>, IETF; Nov. 2000. Work in progress.

[service examples] A. Johnston, R. Sparks, C. Cunningham, S. Donovan, K. Summers, "SIP Service Examples" Internet Draft <draft-ietf-sip-service-examples-03.txt>, IETF, June 2002, Work in progress.

[Join] R. Mahy, D. Petrie, "The SIP Join and Fork Headers", Internet Draft <draft-mahy-sipping-join-and-fork-00.txt>, IETF, November 2001, Work in progress.

[RTP] H. Schulzrinne , S. Casner , R. Frederick , V. Jacobson , "RTP: A Transport Protocol for Real-Time Applications", Request for Comments (Standards Track)1889, IETF, January 1996

[SDP] H. Schulzrinne M. Handley, V. Jacobson, "SDP: Session Description Protocol", Request for Comments (Standards Track) 2327, Internet Engineering Task Force, April 1998

SIP Multiparty Framework

[events] A. Roach, "SIP-Specific Event Notification",Internet Draft
<draft-ietf-sip-events-03.txt>, IETF, February 2002, Work in
progress.

[offer/answer] J. Rosenberg, H. Schulzrinne, "An Offer/Answer Model
with SDP", Internet Draft <draft-ietf-mmusic-sdp-offer-answer-
01.txt>, IETF, February 21, 2002, Work in progress.

[caller prefs] J. Rosenberg, "SIP Caller Preferences and Callee
Capabilities",Internet Draft <draft-ietf-sip-callerprefs-05.txt>,
IETF, November 21, 2001, Work in progress.

[msg waiting] R. Mahy, I. Slain, "Message Waiting in SIP",Internet
Draft <draft-mahy-sip-message-waiting-02.txt>, IETF, July 2001, Work
in progress.

[Presence] Rosenberg et al., "SIP Extensions for Presence", Internet
Draft <draft-ietf-simple-presence-04.txt>, IETF, November 21, 2001,
Work in progress.

[visited] D. Oran, H. Schulzrinne, "The Visited Header",Internet
Draft <>, IETF, date, Work in progress.

[app components] , "",Internet Draft <>, IETF, date, Work in
progress.

[ms-uri] J. Van Dyke, E. Burger, "SIP URI Conventions for Media
Servers",Internet Draft <draft-burger-sipping-msuri-01.txt>, IETF,
November 21, 2001, Work in progress.

[call-pkg] J. Rosenberg, H. Schulzrinne, "SIP Event Packages for
Call Leg and Conference State", Internet Draft <draft-rosenberg-sip-
call-package-00.txt>, IETF, July 13, 2001, Work in progress.

[enum] , "",Internet Draft <>, IETF, date, Work in progress.

[http] R. Fielding et al, "Hypertext Transfer Protocol --
HTTP/1.1", Request for Comments (Standards Track) 2616, Internet
Engineering Task Force, June 1999

[rtsp] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming
Protocol (RTSP)", Request for Comments (Standards Track) 2326,
Internet Engineering Task Force, April 1998

[mrcp] S. Shanmugham, P. Monaco, B. Eberman, "MRCP: Media Resource
Control Protocol", Internet Draft <draft-shanmugham-mrcp-01.txt>,
IETF, November 20, 2001, Work in progress.

[VoiceXML] S. McGlashan et al, "Voice Extensible Markup Language
(VoiceXML) Version 2.0", W3C Working Draft, 23 October 2001, Work in

progress.

[H.323]

[tel URL]

[caller-prefs]

[session timer]

[service context]

[avt tones]

[GSM]

[MPEG2]

[G.711]

[H.261]

[H.450]

[JTAPI]

[CSTA]

[mrcp-sip] , "",Internet Draft <draft-robinson-mrcp-sip-00.txt>, IETF, date, Work in progress.

[distributed full mesh conf]

[Media forking] M. Shankar, "SIP Forked Media", Internet Draft <draft-shankar-sip-forked-media-00.txt>, IETF, Feb. 2001. Work in progress.

[PHONECTL] R. Dean, Belkind, B. Biggs, "PHONECTL: A Protocol for Remote Phone Control", Internet Draft <draft-dean-phonectl-03.txt>, IETF, Jan. 2001. Work in progress.

9 Changes since -00

- Removed many media-specific references.

- Condensed discussion on mixing models, and VoiceXML discussion.

- Moved the sample feature discussion to an Appendix

10

To Do

- Add diagrams to section 4.3.2 and 4.3.3

- Convert to XML

   - Fix references

SIP Multiparty Framework

   - Propose to move Appendix A (sample features to service flows)

   - Align with terminology with conferencing drafts

   - Show roadmap for related drafts

     Other frameworks and requirements
         Conferencing framework
         Conferencing models
         Framework for markup

     Extensions
         REFER
         Replaces
         Join
         Caller prefs

     Packages
         conference-package
         dialog package

     Usage Drafts
         3pcc
         cc-transfer

     Informational Drafts
         Service flows

   - Define some semantics for authorization rules. For example one
   could define a dictionary of primitives and/or perhaps define sets
   or classes of these primitives, then configure who is allowed to use
   them

11

   Acknowledgments

   Thanks to all who attended the SIP interim meeting in February 2001
   for their support of the ideas behind this document.

12

   Author's Addresses

   Rohan Mahy
   Cisco Systems
   170 West Tasman Dr, MS: SJC-21/3/3
   Phone: +1 408 526 8570
   Email: rohan@cisco.com

   Ben Campbell

dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
Email: bcampbell@dynamicsoft.com

SIP Multiparty Framework

Alan Johnston
WorldCom
100 S. 4th Street
St. Louis, Missouri 63104
Email: alan.johnston@wcom.com

Daniel G. Petrie
Pingtel Corp.
400 W. Cummings Park
Suite 2200
Woburn, MA 01801
Phone: +1 781 938 5306
Email: dpetrie@pingtel.com

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

Robert J. Sparks
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024
Email: rsparks@dynamicsoft.com

SIP Multiparty Framework