Internet Engineering Task Force SIP WG
INTERNET-DRAFT
G. Camarillo, E. Burger, H. Schulzrinne, A. van Wijk
draft-ietf-sipping-transc-3pcc-00.ps
Ericsson, SnowShore Networks, Columbia University, Viataal
February 3, 2004
Expires: August, 2004

Transcoding Services Invocation in the Session Initiation Protocol Using Third Party Call Control

Status of this Memo

Copyright Notice

Abstract

This document describes how to invoke transcoding services using SIP and third party call control. This way of invocation meets the requirements for SIP regarding transcoding services invocation to support deaf, hard of hearing and speech-impaired individuals.

Contents

1 Introduction

The framework for transcoding with SIP [1] describes how two SIP UAs can discover imcompatibilities that prevent them from establishing a session (e.g., lack of support for a common codec or for a common media type). When such incompatibilities are found, the UAs need to invoke transcoding services to successfully establish the session. 3pcc (third party call control) [2] is one way to perform such invocation.

2 General Overview

In the 3pcc model for transcoding invocation, a transcoding server that provides a particular transcoding service (e.g., speech-to-text) is identified by a URI. A UA that wishes to invoke that service sends an INVITE request to that URI establishing a number of media streams. The way the transcoder manipulates and manages the contents of those media streams (e.g., the text received over the text stream is transformed into speech and sent over the audio stream) is service specific.

All the call flows in this document use SDP. The same call flows could be used with another session description protocol that provided similar session description capabilities.

3 Third Party Call Control Flows

Given two UAs (A and B) and a transcoding server (T), the invocation of a transcoding service consists of establishing two sessions; A-T and T-B. How these sessions are established depends on which party, the caller (A) or the callee (B), invokes the transcoding services. Section 3.2 deals with callee invocation and Section 3.3 deals with caller invocation.

In all our 3pcc flows we have followed a general principle; a 200 (OK) response from the transcoding service has to be received before contacting the callee. This tries to ensure that the transcoding service will be available when the callee accepts the session.

Still, the transcoding service does not know the exact type of transcoding it will be performing until the callee accepts the session. So, there are always chances of failing to provide transcoding services after the callee has accepted the session. A system with tough requirements could use preconditions to avoid this situation. When preconditions are used, the callee is not alerted until everything is ready for the session.

3.1 Terminology

All the flows in this document follow the naming convention below:

SDP A: A session description generated by A. It contains, among other things, the transport address/es (IP address and port number) where A wants to receive media for each particular stream.

SDP B: A session description generated by B. It contains, among other things, the transport address/es where B wants to receive media for each particular stream.

SDP A+B: A session description that contains, among other things, the transport address/es where A wants to receive media and the transport address/es where B wants to receive media.

SDP TA: A session description generated by T and intended for A. It contains, among other things, the transport address/es where T wants to receive media from A.

SDP TB: A session description generated by T and intended for B. It contains, among other things, the transport address/es where T wants to receive media from B.

SDP TA+TB: A session description generated by T that contains, among other things, the transport address/es where T wants to receive media from A and the transport address/es where T wants to receive media from B.

3.2 Callee's Invocation

In this scenario, B receives an INVITE from A, and B decides to introduce T in the session. Figure 1 shows the call flow for this scenario.

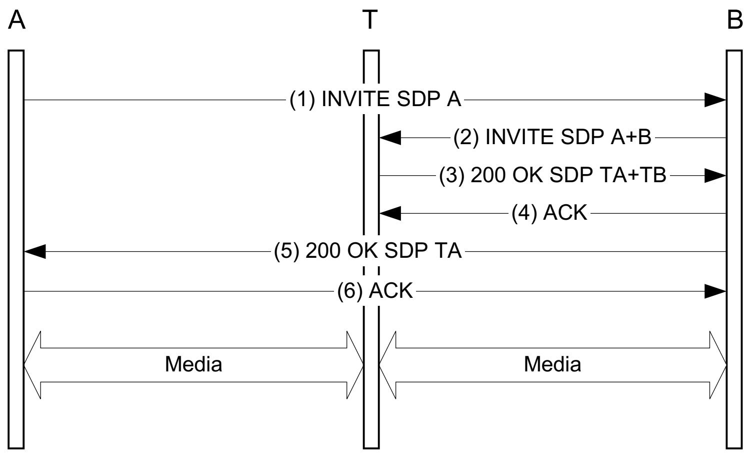

Figure 1: Callee's invocation of a transcoding service

In Figure 1, A can both hear and speak and B is a deaf user with a speech impairment. A proposes to establish a session that consists of an audio stream (1). B wants to send and receive only text, so it invokes a transcoding service T that will perform both speech-to-text and text-to-speech conversions (2). The session descriptions of Figure 1 are partially shown below.

(1) INVITE SDP A

```
m=audio 20000 RTP/AVP 0
c=IN IP4 A.domain.com
```

(2) INVITE SDP A+B

```
m=audio 20000 RTP/AVP 0
c=IN IP4 A.domain.com
m=text 40000 RTP/AVP 96
c=IN IP4 B.domain.com
a=rtpmap:96 t140/1000
```

(3) 200 OK SDP TA+TB

```
m=audio 30000 RTP/AVP 0
c=IN IP4 T.domain.com
m=text 30002 RTP/AVP 96
c=IN IP4 T.domain.com
a=rtpmap:96 t140/1000
```

(5) 200 OK SDP TA

```
m=audio 30000 RTP/AVP 0
c=IN IP4 T.domain.com
```

Four media streams (i.e., two bi-directional streams) have been established at this point:

1. Audio from A to T.domain.com:30000
2. Text from T to B.domain.com:40000
3. Text from B to T.domain.com:30002
4. Audio from T to A.domain.com:20000

When either A or B decide to terminate the session, B will send a BYE to T indicating that the session is over.

If the first INVITE (1) received by B is empty (no session description), the call flow is slightly different. Figure 2 shows the messages involved.

B may have different reasons for invoking T before knowing A's session description. B may want to hide its capabilities, and therefore it wants to return a session description with all the codecs B supports plus all the codecs T supports. Or T may provide recording services (besides transcoding), and B wants T to record the conversation, regardless of whether or not transcoding is needed.

This scenario (Figure 2) is a bit more complex than the previous one. In INVITE (2), B still does not have SDP A, so it cannot provide T with that information. When B finally receives SDP A in (6), it has to send it to T. B sends an empty INVITE to T (7) and gets a 200 OK with SDP TA+TB (8). In general, this SDP TA+TB can be different than the one that was sent in (3). That is why B needs to send the updated SDP TA to A in (9). A then sends a possibly updated SDP A (10) and B sends it to T in (12). On the other hand, if T happens to return the same SDP TA+TB in (8) as in (3), B can skip messages (9), (10) and (11). So, implementors of transcoding services are encouraged to return the same session description in (8) as in (3) in this type of scenario. The session descriptions of this flow are shown below:

(2) INVITE SDP A+B

```
m=audio 20000 RTP/AVP 0
c=IN IP4 0.0.0.0
m=text 40000 RTP/AVP 96
c=IN IP4 B.domain.com
a=rtpmap:96 t140/1000
```

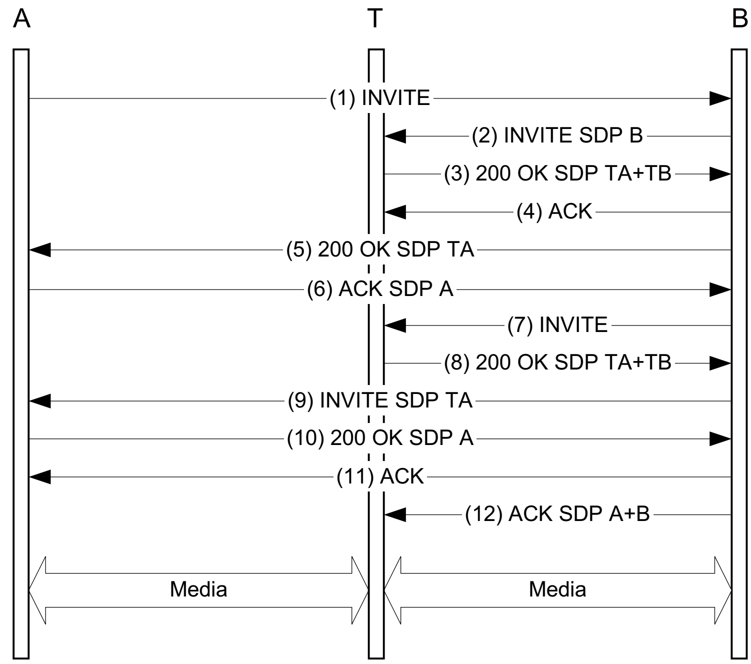

Figure 2: Callee's invocation after initial INVITE without SDP

(3) 200 OK SDP TA+TB

```
m=audio 30000 RTP/AVP 0
c=IN IP4 T.domain.com
m=text 30002 RTP/AVP 96
c=IN IP4 T.domain.com
a=rtpmap:96 t140/1000
```

(5) 200 OK SDP TA

```
m=audio 30000 RTP/AVP 0
c=IN IP4 T.domain.com
```

(6) ACK SDP A

```
m=audio 20000 RTP/AVP 0
c=IN IP4 A.domain.com
```

(8) 200 OK SDP TA+TB

```
m=audio 30004 RTP/AVP 0
c=IN IP4 T.domain.com
m=text 30006 RTP/AVP 96
```

```
c=IN IP4 T.domain.com
a=rtpmap:96 t140/1000
```

(9) INVITE SDP TA

```
m=audio 30004 RTP/AVP 0
c=IN IP4 T.domain.com
```

(10) 200 OK SDP A

```
m=audio 20002 RTP/AVP 0
c=IN IP4 A.domain.com
```

(12) ACK SDP A+B

```
m=audio 20002 RTP/AVP 0
c=IN IP4 A.domain.com
m=text 40000 RTP/AVP 96
c=IN IP4 B.domain.com
a=rtpmap:96 t140/1000
```

Four media streams (i.e., two bi-directional streams) have been established at this point:

1. Audio from A to T.domain.com:30004
2. Text from T to B.domain.com:40000
3. Text from B to T.domain.com:30006
4. Audio from T to A.domain.com:20002

3.3 Caller's Invocation

In this scenario, A wishes to establish a session with B using a transcoding service. A uses 3pcc to set up the session between T and B. The call flow we provide here is slightly different than the ones in [2]. In [2], the controller establishes a session between two user agents, which are the ones deciding the characteristics of the streams. Here, A wants to establish a session between T and B, but A wants to decide how many and which types of streams are established. That is why A sends its session description in the first INVITE (1) to T, as opposed to the media-less initial INVITE recommended by [2]. Figure 3 shows the call flow for this scenario.

We do not include the session descriptions of this flow, since they are very similar to the ones in Figure 2. In this flow, if T returns the same SDP TA+TB in (8) as in (2), messages (9), (10) and (11) can be skipped.

3.4 Receiving the Original Stream

Sometimes, as pointed out in the requirements for SIP in support of deaf, hard of hearing and speech-impaired individuals [3], a user wants to receive both the original stream (e.g., audio)

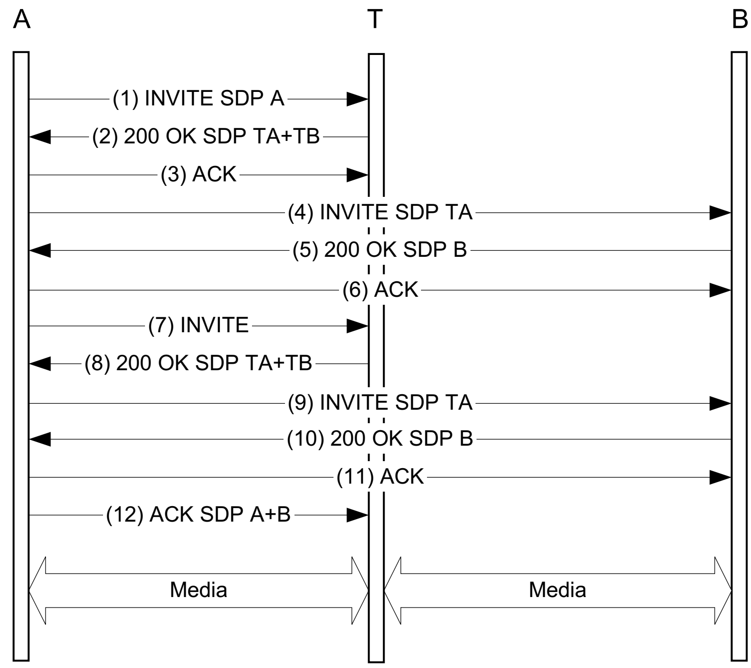

Figure 3: Caller's invocation of a transcoding service

and the transcoded stream (e.g., the output of the speech-to-text conversion). There are various possible solutions for this problem. One solution consists of using the SDP group attribute with FID semantics [4]. FID allows requesting that a stream is sent to two different transport addresses in parallel, as shown below:

```
a=group:FID 1 2
m=audio 20000 RTP/AVP 0
c=IN IP4 A.domain.com
a=mid:1
m=audio 30000 RTP/AVP 0
c=IN IP4 T.domain.com
a=mid:2
```

The problem with this solution is that the majority of the SIP user agents do not support FID. Moreover, only a small fraction of the few UAs that do support FID, support sending simultaneous copies of the same media stream at the same time. In addition, FID forces both copies of the stream to use the same codec.

So, we recommend that T (instead of a user agent) replicates the media stream. The transcoder T receiving the following session description performs speech-to-text and text-to-speech conversions between the first audio stream and the text stream. In addition, T copies the first audio stream to the second audio stream and sends it to A.

```
m=audio 40000 RTP/AVP 0
```

```
c=IN IP4 B.domain.com
m=audio 20000 RTP/AVP 0
c=IN IP4 A.domain.com
a=recvonly
m=text 20002 RTP/AVP 96
c=IN IP4 A.domain.com
a=rtpmap:96 t140/1000
```

3.5 Transcoding Services in Parallel

Transcoding services sometimes consist of human relays (e.g., a person performing speech-to-text and text-to-speech conversions for a session). If the same person is involved in both conversions (i.e., from A to B and from B to A), he or she has access to all the conversation. In order to provide some degree of privacy, sometimes two different persons are allocated to do the job (i.e., one person handles A->B and the other B->A). This type of disposition is also useful for automated transcoding services, where one machine converts text to synthetic speech (text-to-speech) and a different machine performs voice recognition (speech-to-text).

The scenario just described involves four different sessions; A-T1, T1-B, B-T2 and T2-A. Figure 4 shows the call flow where A invokes T1 and T2.

(1) INVITE SDP AT1

```
m=text 20000 RTP/AVP 96
c=IN IP4 A.domain.com
a=rtpmap:96 t140/1000
a=sendonly
m=audio 20000 RTP/AVP 0
c=IN IP4 0.0.0.0
a=recvonly
```

(2) INVITE SDP AT2

```
m=text 20002 RTP/AVP 96
c=IN IP4 A.domain.com
a=rtpmap:96 t140/1000
a=recvonly
m=audio 20000 RTP/AVP 0
c=IN IP4 0.0.0.0
a=sendonly
```

(3) 200 OK SDP T1A+T1B

```
m=text 30000 RTP/AVP 96
c=IN IP4 T1.domain.com
a=rtpmap:96 t140/1000
a=recvonly
m=audio 30002 RTP/AVP 0
```

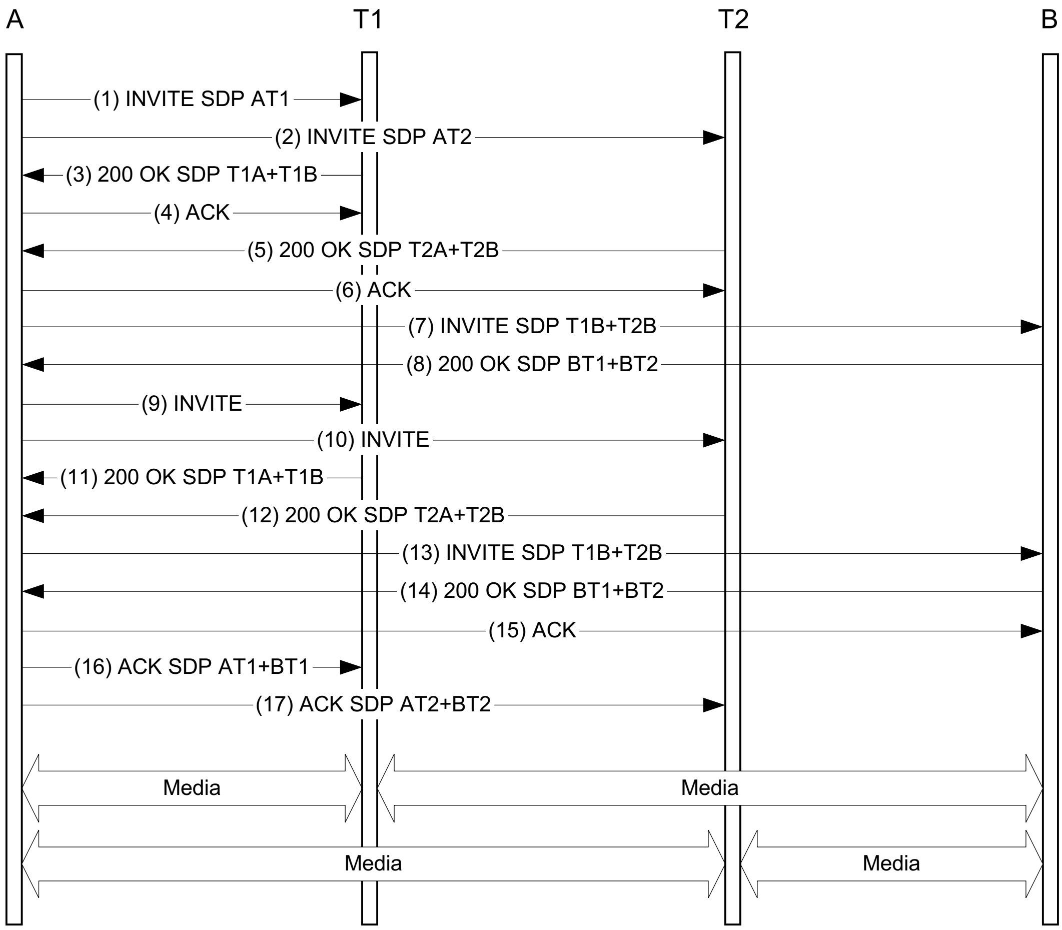

Figure 4: Transcoding services in parallel

```
c=IN IP4 T1.domain.com
a=sendonly
```

(5) 200 OK SDP T2A+T2B

```
m=text 40000 RTP/AVP 96
c=IN IP4 T2.domain.com
a=rtpmap:96 t140/1000
a=sendonly
m=audio 40002 RTP/AVP 0
c=IN IP4 T2.domain.com
a=recvonly
```

(7) INVITE SDP T1B+T2B

```
m=audio 30002 RTP/AVP 0
c=IN IP4 T1.domain.com
a=sendonly
m=audio 40002 RTP/AVP 0
c=IN IP4 T2.domain.com
a=recvonly
```

(8) 200 OK SDP BT1+BT2

```
m=audio 50000 RTP/AVP 0
c=IN IP4 B.domain.com
a=recvonly
m=audio 50002 RTP/AVP 0
c=IN IP4 B.domain.com
a=sendonly
```

(11) 200 OK SDP T1A+T1B

```
m=text 30000 RTP/AVP 96
c=IN IP4 T1.domain.com
a=rtpmap:96 t140/1000
a=recvonly
m=audio 30002 RTP/AVP 0
c=IN IP4 T1.domain.com
a=sendonly
```

(12) 200 OK SDP T2A+T2B

```
m=text 40000 RTP/AVP 96
c=IN IP4 T2.domain.com
a=rtpmap:96 t140/1000
```

```
a=sendonly
m=audio 40002 RTP/AVP 0
c=IN IP4 T2.domain.com
a=recvonly
```

Since T1 have returned the same SDP in (11) as in (3) and T2 has returned the same SDP in (12) as in (5), messages (13), (14) and (15) can be skipped.

(16) ACK SDP AT1+BT1

```
m=text 20000 RTP/AVP 96
c=IN IP4 A.domain.com
a=rtpmap:96 t140/1000
a=sendonly
m=audio 50000 RTP/AVP 0
c=IN IP4 B.domain.com
a=recvonly
```

(17) ACK SDP AT2+BT2

```
m=text 20002 RTP/AVP 96
c=IN IP4 A.domain.com
a=rtpmap:96 t140/1000
a=recvonly
m=audio 50002 RTP/AVP 0
c=IN IP4 B.domain.com
a=sendonly
```

Four media streams have been established at this point:

1. Text from A to T1.domain.com:30000
2. Audio from T1 to B.domain.com:50000
3. Audio from B to T2.domain.com:40002
4. Text from T2 to A.domain.com:20002

Note that B, the user agent server, needs to support two media streams; one sendonly and the other recvonly. At present, some user agents, although they support a single sendrecv media stream, they do not support a different media line per direction. Implementers are encouraged to build support for this feature.

3.6 Transcoding Services in Serial

In a distributed environment, a complex transcoding service (e.g., English text to Spanish speech) is often provided by several servers. For example, one server performs English text to Spanish text translation, and its output is feed into a server that performs text-to-speech conversion. The flow in Figure 5 shows how A invokes T1 and T2.

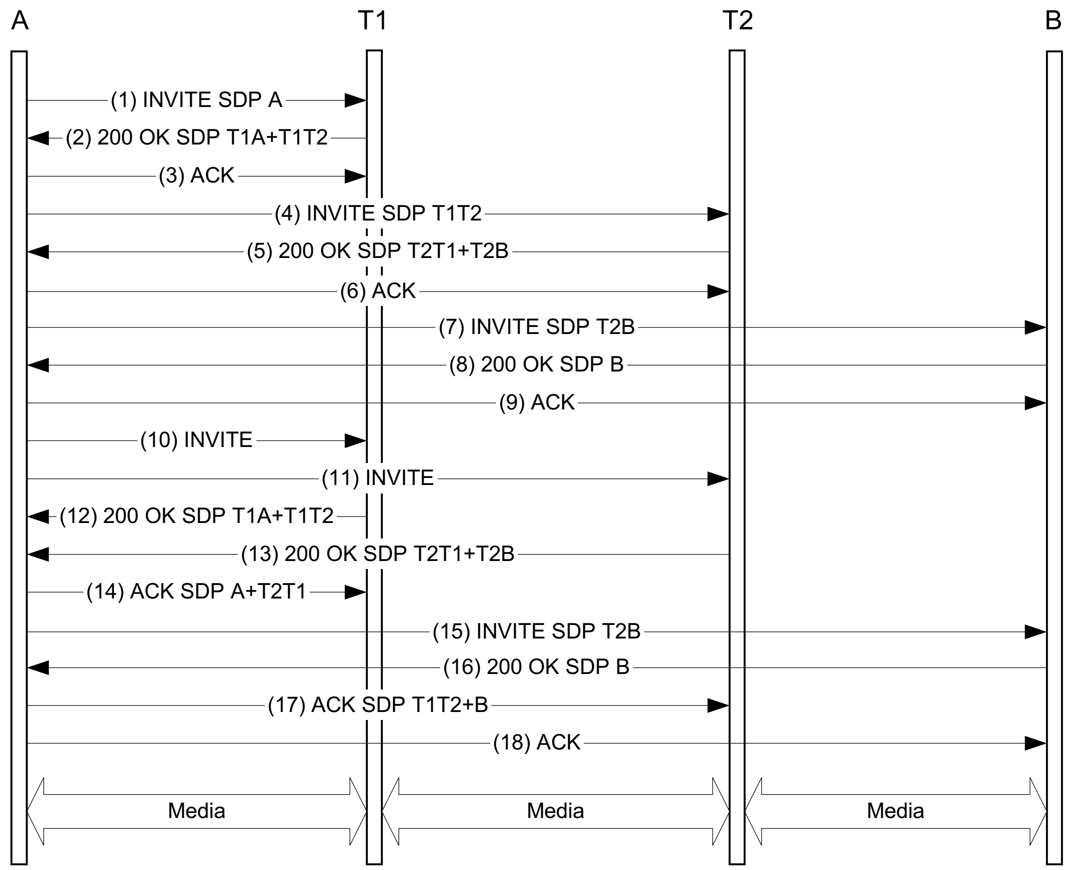

Figure 5: Transcoding services in serial

4 Security Considerations

This document describes how to use third party call control to invoke transcoding services. It does not introduce new security considerations besides the ones discussed in [2].

5 Authors' Addresses

Gonzalo Camarillo
Ericsson
Advanced Signalling Research Lab.
FIN-02420 Jorvas
Finland
electronic mail: Gonzalo.Camarillo@ericsson.com

Eric W. Burger
SnowShore Networks, Inc.
Chelmsford, MA
USA
electronic mail: eburger@snowshore.com

Henning Schulzrinne
Dept. of Computer Science
Columbia University 1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
electronic mail: schulzrinne@cs.columbia.edu

Arnoud van Wijk
Viataal
Research & Development
Afdeling RDS
Theerestraat 42
5271 GD Sint-Michielsgestel
The Netherlands
electronic mail: a.vwijk@viataal.nl

References

[1] G. Camarillo, "Framework for transcoding with the session initiation protocol," Internet-Draft draft-camarillo-sipping-transc-framework-00, Internet Engineering Task Force, Aug. 2003. Work in progress.

[2] J. Rosenberg, J. L. Peterson, H. Schulzrinne, and G. Camarillo, "Best current practices for third party call control in the session initiation protocol," Internet-Draft draft-ietf-sipping-3pcc-05, Internet Engineering Task Force, Oct. 2003. Work in progress.

[3] N. Charlton, M. Gasson, G. Gybels, M. Spanner, and A. van Wijk, “User requirements for the session initiation protocol (SIP) in support of deaf, hard of hearing and speech-impaired individuals,” RFC 3351, Internet Engineering Task Force, Aug. 2002.

[4] G. Camarillo, J. Holler, G. Eriksson, and H. Schulzrinne, “Grouping of m lines in SDP,” internet-draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

Full Copyright Statement