INTERNET-DRAFT
August 2001
Expires: February 2002

Vijay K. Gurbani
Lucent Technologies, Inc.
Vidhi Rastogi
Wipro Technologies

Accessing IN services from SIP networks

<draft-gurbani-iptel-sip-to-in-05.txt>

ABSTRACT

In Internet telephony, the call control functions of a traditional circuit switch are replaced by a IP-based call controller that must provide features normally provided by the traditional switch, including operating as a SSP for IN features. A traditional switch is armed with an IN call model that provides it a means to reach out and make service decisions based on intelligence stored elsewhere. Internet call controllers, by contrast, do not have an IN call model. Furthermore, since there are many Internet call models with varying number of states than the IN call model, there has to be a mapping from the IN call model states to the equivalent states of the Internet call model if existing services are to be accessed transparently. To leverage the existing IN services from the Internet domain, this draft proposes a mapping from the states of the IN call model to the states of SIP, an Internet call signaling protocol.

Accessing IN services from SIP networks

1. Introduction

In Internet telephony, the call control functions of a traditional circuit switch are replaced by a device referred to, in different contexts, as a call agent, a SIP server, a H.323 Gatekeeper, a feature server, or a soft- switch. This device (which we will refer to as an Internet call agent, or simply a call agent) is an IP entity that coordinates the calls. A call agent executes a finite number of state transitions as it processes the call; these state transitions constitute its call model. To be precise, the term "call model" when applied to an Internet call agent is a misnomer; a better term would be a "protocol state machine." Unlike a traditional switch armed with an IN call model, the protocol state machine on a call agent does not contain IN specific triggers and states. Also, the number of call-related states of an Internet call agent are much less than those of the IN call model. Currently, there are at least two major Internet call signaling protocols in use - H.323 and SIP - both with varying number of states than the IN call model.

In order to access IN services transparently using Internet telephony, the Internet protocol state machine must be mapped to the IN call model. This has the added benefit of accessing existing IN services using the same detection points (DPs) from the same well known point in call (PIC). From the viewpoint of other IN elements like the service control point (SCP), the fact that the request originated from a call agent versus a call processing function on a traditional switch is immaterial. Thus, it is important that the call agent be able to provide features normally provided by the traditional switch, including operating as a SSP for IN features. The call agent should also maintain call state and trigger queries to IN-based services, just as traditional switches do.

The IN call model consists of two halves: the Originating call model and the Terminating call model. If the called and calling party share the same switch, the originating call model is assigned to the calling party and the terminating call model is assigned to the called party. If the call has to go through multiple switches to get to the destination, each of the intervening switch will run the two halves of the call model, with the destination switch's terminating call model providing services to the called party. While this model has worked well for traditional circuit-based switching, it may not be desirable to implement it in an analogous manner on an Internet call model. A later section will discuss this issue in more detail.

The most expeditious manner for providing existing IN services in the Internet telephony domain would be to use the deployed IN

infrastructure as much as possible and leverage existing services. The logical point in the Internet telephony domain to tap into for accessing existing IN services is the call agent. However, the call agent, as we have discovered above, does not run an IN call model. Instead, the various Internet call agents run their respective native protocol state machines for call signaling - either Q.931 in H.323 or a SIP stack in SIP. The trick, then, is to overlay this state machine with an IN layer such that call acceptance and routing is performed by the native state machine and services are accessed through the IN layer using an IN call model. This draft proposes using SIP as the native state machine and accessing IN services by mapping SIP states to the IN call model states. Doing this enables Internet access to well known telephony services such as number translation, call screening, call routing and distribution services, which mostly occur before call setup is complete.

The rest of the paper is organized as follows: Section 2 briefly discusses the IN and SIP call models. Section 3 discusses some issues that necessarily arise from mapping call models. Section 4 outlines some possible SIP/IN architectures. Section 5 establishes a mapping between the IN call model and SIP. Section 6 discusses a few call flows; section 7 includes a report on the implementation status of this I-D, and section 8 touches on some security considerations.

2. Call models

2.1 Overview of IN calls

In a traditional switch environment, when the SSP recognizes a call that requires IN treatment, it temporarily suspends the call processing and sends a query to the SCP. The SCP analyzes the information it received from the SSP and makes a decision on how to continue processing the call. The decision is sent to the SSP, which now continues with further call processing. It is important to realize that IN treatment for a call is not limited to simple request-reply transactions. Including simple querying, the following are the major functions that are part of ITU-T CS-1 and CS-2 [1]:

2.1.1 Querying

The SSP sends a query to the SCP over SS7 in form of a INAP query message. The SCP analyzes the information in the INAP query and sends back a INAP response to the SSP which contains instructions on how to further handle this call.

2.1.2 Caller interaction

Instead of normal query-response, the SSP and SCP may enter an extended interaction. After receiving a query message from the SSP, the SCP may send back to the SSP a INAP conversation with permission message. This prompts the SSP to collect additional information from the caller, possibly by involving other IN devices like the Intelligent Peripheral or a Service Node. The caller may send back to the SSP dial-pulse digits or DTMF signals. Whatever the format of the response, the SSP returns this information back to the SCP in a INAP conversation package message.

2.1.3 Trigger activation/deactivation

Most DPs in a switch are armed by the SMF offline. However, it is possible for the SCP to inform a switch to arm a DP for the duration of a call. DPs armed in the former manner are said to be statically armed and those armed in the latter manner are said to be dynamically armed. Dynamically armed DPs remain in effect for the duration of that particular call [2].

2.1.4 Response processing

The SSP, upon receiving a INAP response message from the SCP, must take the appropriate actions such as routing the call, redirecting the call, disconnecting the call, playing announcements to the caller, and so on. The SCP may further control the SSP by requesting that it be notified when the call ends, or requesting it to monitor certain facilities.

2.2 IN call model

The IN generic basic call state model (BCSM), independent of any capability sets, is divided into two halves - an originating call model (O_BCSM) and a terminating call model (T_BCSM) [3]. There are a total of 19 PICs and 35 DPs between both the halves (11 PICs and 21 DPs for O_BCSM; 8 PICs and 14 DPs for T_BCSM) [2]. The SSPs, SCPs and other IN elements track a call's progress in terms of the basic call model. The basic call model provides a common context for communication about a call.

O_BCSM has 11 PICs. These are:

```
  O_NULL: starting state; call does not exist yet.
  AUTH_ORIG_ATTEMPT: switch detects a call setup request.
  COLLECT_INFO: switch collects the dial string from the calling
  party.
  ANALYZE_INFO: complete dial string is translated into a routing
```

address.
SELECT_ROUTE: physical route is selected, based on the routing address.
AUTH_CALL_SETUP: switch ensures the calling party is authorize to place call.
CALL_SENT: control of call send to terminating side.
O_ALERTING: switch waits for the called party to answer.
O_ACTIVE: connection established; communication ensue.
O_DISCONNECT: connection torn down.
O_EXCEPTION: switch detected an exceptional condition.

T_BCSM has 8 PICS. These are:

T_NULL: starting state; call does not exist yet.
AUTH_TERM_ATT: switch verifies whether call can be send to terminating party.
SELECT_FACILITY: switch picks a terminating resource to send the call on.
PRESENT_CALL: call is being presented to the called party.
T_ALERTING: switch alerts the called party, e.g. ringing the line.
T_ACTIVE: connection established; communications ensue.
T_DISCONNECT: connection torn down.
T_EXCEPTION: switch detected an exceptional condition.

The state machine for O_BCSM and T_BCSM is provided in [2] page 98 and 103 respectively. This state machine will be used for subsequent discussion when the IN call states are mapped into SIP.

It is beyond the scope of this document to explain all PICs and DPs in an IN call model. It is assumed that the reader has some familiarity with the PICs and DPs of the IN call model. More information can be found in [2].

2.3 SIP call model

SIP is a lightweight signaling protocol for Internet telephony. SIP has 6 methods -- INVITE, ACK, OPTIONS, BYE, CANCEL, and REGISTER -- and various response codes divided among the following 6 classes:

Class	Meaning
1xx	Informational
2xx	Success
3xx	Redirection
4xx	Request failure
5xx	Server failure

6xx	Global failure

Table 1: SIP response codes

To establish a mapping between IN call state and SIP, the SIP protocol state machine can be viewed as essentially consisting of an INVITE message, interim response codes for the invitation ("100 Trying" or "180 Ringing"), an acceptance (or a decline) of the INVITE message, and an acknowledgement for the acceptance (or decline). If the invitation was accepted, SIP provides a BYE message for signaling the end of the call.

It is beyond the scope of this document to cover SIP in a justifiable manner. It is assumed that the reader has some familiarity with SIP. More information can be found in [4]

3. Issues in IN call model mappings

One way in which IN services can be invoked transparently from a SIP server processing a telephony call is to overlay the SIP protocol state machine with the IN call model. Thus the call receives treatment from two call models, both working in synchrony; the SIP state machine handles the acceptance and final delivery of the call while the IN call model interfaces with the IN to provide services for the call. Figure 1 demonstrates this concept: a SIP server accepts a call, notifies the IN call handling layer of this event; the IN call handling layer interfaces with the IN elements to provide services for the call, ultimately informing the SIP server on how to deliver the call. The interface between the IN call handling layer and the SCP is not specified by this I-D and indeed, can be any one of the following depending on the interfaces supported by the SCP: INAP over IP, INAP over SIGTRAN, INAP over TALI or INAP over SS7.

```
                                          +-----+
                                          |  S  |
                                          |  C  |
                              +---------->|  P  |
                              |           +-----+
                              |
                              V
          +------------------------+
          | IN call handling       |
          +------------------------+
          |           ^ Process    |
          |          / \ call      |
          |           |            |
--------->| SIP call handling      |----------->
Accept    +------------------------+          Deliver call
Call
```

Figure 1: IN call model overlayed on SIP

The notion of feature interaction, i.e. the notion about where a call gets its features serviced from -- SIP or IN -- is not addressed here. SIP itself has a rich set of features that can be applied on a call by call basis, as does IN. While it is entirely possible that a SIP server applies certain features to an incoming call before handing it to the IN layer, this draft limits its discussion on services accessed from the IN by a SIP proxy server. The underlying assumption is that IN is servicing the call by providing it features, and SIP is simply routing the call based on the decisions made by the IN layer.

Another fundamental problem here lies in the notion of a call state. The IN call model is necessarily a stateful one. A SIP proxy can operate in either stateful or stateless mode, depending on the needs of the application. For speed, reliability and scalability, SIP proxies may be run in the stateless mode. The duration and amount of state maintained at a SIP proxy are small compared to the traditional telephone network, where the switch must maintain the call state for the entire duration of the call. For a SIP proxy to run in the call-stateful mode, it has to indicate a willingness to remain in the signaling path till the call is disconnected. This is accomplished using the Record-Route header field of a SIP message.

4. SIP/IN architecture

In order to apply the stateful IN call model to a SIP proxy, the originating and terminating SIP network servers must run in call-state aware mode and have the IN call model layer working in

conjunction with SIP as depicted in Figure 1. Other intervening SIP proxies may remain stateless and have no need to run the IN call model layer. The originating and terminating SIP network servers mimic the originating and terminating switches in a traditional phone network. IN services accessed through DPs on originating or terminating side can now be handled by the IN layer on the originating or terminating SIP proxy. Figure 2 demonstrates this:

```
          +---+                                                +---+
          | S |                                                | S |
          | C |<--+                                        +-->| C |
          | P |   |                                        |   | P |
          +---+   |                                        |   +---+
                  |                                        |
          +-------|----------------------------------------|-------+
          |       |           SIP network                  |       |
          |       V                                        V       |
          | +--------+                                 +--------+ |
          | | IN     |                                 | IN     | |
          | +--------+     +--------+     +--------+     +--------+ |
Calling   | | O SIP |---->| C SIP | ... | C SIP |---->| T SIP | |     Called
Party ----|>+--------+     +--------+     +--------+     +--------+-|---> Party
          |                                                        |
          +--------------------------------------------------------+

Legend:
O SIP: Originating SIP network server, running IN call model layer
T SIP: Terminating SIP network server, running IN call model layer
C SIP: Core SIP network servers (may be proxy or redirect)
IN:    IN layer
```

Figure 2: IN-controlled SIP network

There are three other points worth mentioning in Figure 2:

1) If the called party and the calling party are handled by the same SIP server, both halves of the IN call model will run on that server. This is analogous to the traditional telephone network.

2) In the traditional telephone network, the interexchange switches can run both halves of the call model. This can also be accomplished in the SIP network if desired. Figure 2 shows the IN call model running on originating and terminating SIP proxies. However, any of the core SIP proxies could also have hosted the IN call model if needed.

3) If the called party and calling party are handled by different SIP network servers, each with its own IN layer, the IN call state information has to be propagated between these servers. This draft does not include any information on such a transaction, except to note that there are other protocols like SIP+ which address such problems. Or in fact, the IN layers of the originating and terminating SIP proxies can communicate directly with each other using ISUP over IP to share call state between themselves.

Figure 2 showed an end-to-end SIP network, with SIP proxies running the IN call model reaching out to the SCP for service logic. Figure 3 shows a SIP network providing services from the SCP through the IN call model and routing the call to the GSTN. In this example, it is assumed that both halves of the IN call model are running on the same SIP proxy:

```
          +---+
          | S |
          | C |<--+
          | P |   |
          +---+   |
                  |
          +-------|-----------------+              +---------- ...
          |       |  SIP network    |              | GSTN network
          |       V                 |              |
          | +-------+               |              |
          | | IN    |               |              |
          | +-------+       +-------------+        |
Calling   | | O SIP |------>| SIP Gateway |------->|
Party ----|>+-------+       +-------------+        |
          |                         |              |
          +-------------------------+              +---------- ...
```

Figure 3: SIP network with GSTN gateway

5. Mapping call states

At first glance, it would appear that mapping a 19 PIC and 35 DP IN call model into SIP is a losing proposition. However, such is not the case. In fact, certain IN services like freephone, originating call screening, caller name identification, are very easy to implement. By and large, IN services that have a "query-response" nature are easily translated to SIP. On the other hand, services involving the media path (e.g. "prompt-and-collect", mid-call announcement capability) are comparitively harder to implement; and in fact these services may be implemented in a specialized

"application server" instead of a SIP proxy [5].

IN states (listed in Section 2.3) will be mapped to the appropriate SIP methods or response codes (also listed in Section 2.3). While mapping call states from SIP to IN, it is important to note that there will not be a 1-to-1 mapping between IN call states and SIP states.

5.1 SIP and O_BCSM

The 11 PICs of O_BCSM come into play when a call request (SIP INVITE message) arrives from an upstream SIP client to an originating SIP proxy running the IN call model. This SIP proxy will create a O_BCSM object and initialize it in the O_NULL PIC. The next five IN PICs -- AUTH_ORIG_ATT, COLLECT_INFO, ANALYZE_INFO, SELECT_ROUTE and AUTH_CALL_SETUP -- can all be mapped to the SIP INVITE message.

The SIP INVITE message has enough functionality to absorb these five PICs as described below:

AUTH_ORIG_ATT - In this PIC, an IN SSP has detected that someone wishes to make a call. Under some circumstances (e.g. the user is not allowed to make calls during certain hours), such a call cannot be placed. SIP has the ability to authorize the calling party using a set of policy directives configured by the SIP administrator. If the called party is authorized to place the call, the IN layer is instructed to enter the next PIC, COLLECT_INFO.

COLLECT_INFO - This PIC is responsible for collecting a dial string from the calling party. The SIP proxy can detect a malformed address and may send the calling party a "484 Address Incomplete" message and remain in this state until a valid "dial string" is received. Once it has obtained a valid dial string, the IN layer is instructed to enter the next PIC, ANALYZE_INFO.

ANALYZE_INFO - This PIC is responsible for translating the dial string to a routing number. Many IN service such as freephone, LNP, OCS, etc. occur during this PIC. The IN layer can use the Request-URI of the SIP INVITE request for analysis. If the analysis succeeds, the IN layer is instructed to enter the next PIC, SELECT_ROUTE.

SELECT_ROUTE - In the circuit-switched network, the actual physical route has to be selected at this point. The SIP analogue of this would be to determine the next hop SIP server. The next hop SIP server could be chosen by a variety of means. For instance, if the

Request URI in the incoming INVITE request is an E.164 number, the SIP proxy can use a protocol like TRIP [6] to find the best gateway to egress the request onto the GSTN.

AUTH_CALL_SETUP - Certain service features restrict the type of call that may originate on a given line or trunk. This PIC is the point at which relevant restrictions are examined.

If the above 5 PICs have been successfuly negotiated, the SIP proxy running the IN call model now sends the SIP INVITE message to the next hop server.   If the SIP proxy running the IN call layer gets back a "100 Trying" message for that call, it can instruct the IN layer to enter enter the next PIC, CALL_SENT. In IN terms, the control over the establishment of the call has been transferred to the T_BCSM object, and the O_BCSM object is waiting for a signal confirming that either the call has been presented to the called party or that the called party cannot be reached for a particular reason.

When the SIP proxy running the IN call layer gets back a "180 Ringing" for the call, it now instructs the IN layer to enter the next PIC, O_ALERTING. At this point, O_BCSM is waiting for the called party to answer. Assuming the called party answers, the SIP proxy running the IN layer receives a "200 OK". The receipt of this message is followed by the SIP proxy instructing the IN layer to enter the next PIC, O_ACTIVE. The call is now active.

When either of the party hangs up, the SIP proxy running the IN call layer receives a SIP BYE message. Since it is running in call-stateful mode, it can correlate the BYE message with the call that needs to be torn down. The SIP server instructs the IN layer to enter its next PIC, O_DISCONNECT and perform cleanup. Subsequently, the state of the call in the SIP proxy itself is also destroyed.

Figure 4 below provides a visual mapping from the SIP proxy protocol state machine to the originating half of the IN call model. Note that control of the call shuttles between the SIP protocol machine and the IN O_BCSM call model while it is being serviced.

```
      SIP                                         O_BCSM

 +-----------------+                       +------------------+
 | INVITE          |=====================>| O_NULL           |
 +--+--------------+                       +--------+---------+
    |         /\                                    |
    |         ||                                    |
    |         ||                           +--------V---------+
    |         ||                           | AUTH_ORIG_ATT    |
    |         ||                           +--------+---------+
    |         ||                                    |
    |         ||                                    |
    |         ||                           +--------V---------+
    |         ||<=======================>| COLLECT_INFO     |
    |         ||                           +--------+---------+
    |         ||                                    |
    |         ||                                    |
    |         ||                           +--------V---------+
    |         ||                           | ANALYSE_INFO     |
    |         ||                           +--------+---------+
    |         ||                                    |
    |         ||                                    |
    |         ||                           +--------V---------+
    |         ||                           | SELECT_ROUTE     |
    |         ||                           +--------+---------+
    |         ||                                    |
    |         ||                                    |
    |         ||                           +--------V---------+
    |         ||<======================== | AUTH_CALL_SETUP |
    |                                      +------------------+
    |
 +--V---------------+                      +------------------+
 | 100 Trying       |=====================>| CALL_SENT        |
 +--+---------------+                      +--||--------------+
    |         /\                              ||
    |         ||                              ||
    |         ||=============================||
    |
 +--V---------------+                      +------------------+
 | 180 Ringing      |=====================>| O_ALERTING       |
 +--+---------------+                      +--||--------------+
    |         /\                              ||
    |         ||                              ||
    |         ||=============================||
    |
 +--V---------------+                      +------------------+
```

```
| 200 OK          |=====================>| O_ACTIVE        |
+-----------------+                      +-----------------+


------------------------------------------------------------
Communication established; call active
------------------------------------------------------------

+-----------------+                      +-----------------+
| BYE             |=====================>| O_DISCONNECT    |
+-----------------+                      +--||-------------+
            /\                              ||
            ||                              ||
            ||==============================||
Legend:

| Communication between
| states in the same
V protocol

======> Communication between IN layer and SIP
```

Figure 4: Mapping from SIP to O_BCSM

5.2 SIP and T_BCSM

The T_BCSM object is created when a SIP INVITE message makes its way to the terminating SIP proxy running the IN layer. The SIP proxy creates the T_BCSM object and initializes it to the T_NULL PIC.

The IN layer is instructed to enter the next PIC, AUTH_TERM_ATT, during which the fact that the called party wishes to receive the present type of call is ascertained. IN services such as Caller Identification and Call Forwarding are normally triggered from DPs associated with this PIC. Once a positive indication is received from the AUTH_TERM_ATT PIC, the IN layer is instructed to enter the next PIC, SELECT_FACILITY.

The intent of this PIC, in traditional circuit networks, is to select a line or a trunk to reach the called party. In a hybrid (GSTN/IP) network, where the callee resides on the GSTN, a SIP proxy can use SELECT_FACILITY PIC to interface with a gateway and select a line/trunk to route the call. If a facility was thus successfully seized, the SIP INVITE request is deemed to be successfull.

In a traditional circuit-switched network, PRESENT_CALL presents (via

ISUP ACM or Q.931 Alerting messages) the call to the called party. Since the incoming INVITE succeeded, the SIP proxy server instructs the IN layer to enter PRESENT_CALL. The IN layer will remain in this state (the SIP proxy remains in the INVITE state) until the SIP proxy gets back a "180 Ringing", whereupon it instructs the IN layer to enter the next state, T_ALERTING.

T_ALERTING "alerts" the called party by ringing the phone. When the SIP proxy receives a "200 OK", it instructs the IN layer to enter the next PIC, T_ACTIVE. The call is now active.

When either of the party hangs up, the SIP proxy running the IN call layer receives a SIP BYE message. Since it is running in call-stateful mode, it can correlate the BYE message with the call that needs to be torn down. The SIP server instructs the IN layer to enter its next PIC, T_DISCONNECT and perform cleanup. Subsequently, the state of the call in the SIP proxy itself is also destroyed.

Figure 5 below provides a visual mapping from the SIP proxy protocol state machine to the terminating half of the IN call model:

```
       SIP                                      T_BCSM

 +----------------+                       +-----------------+
 | INVITE         |======================>| T_NULL          |
 +--+-------------+                       +--------+--------+
    |         /\                                   |
    |         ||                                   |
    |         ||                          +--------V--------+
    |         ||                          | AUTH_TERM_ATT   |
    |         ||                          +--------+--------+
    |         ||                                   |
    |         ||                                   |
    |         ||                          +--------V--------+
    |         ||                          | SELECT_FACILITY |
    |         ||                          +--------+--------+
    |         ||                                   |
    |         ||                                   |
    |         ||                          +--------V--------+
    |         ||========================= | PRESENT_CALL    |
    |                                     +-----------------+
    |
 +--V--------------+                      +-----------------+
 | 180 Ringing     |=====================>| T_ALERTING      |
 +--+--------------+                      +--||-------------+
    |         /\                             ||
    |         ||                             ||
    |         ||=============================||
    |
 +--V--------------+                      +-----------------+
 | 200 OK          |=====================>| T_ACTIVE        |
 +-----------------+                      +-----------------+


 ------------------------------------------------------------
 Communication established; call active
 ------------------------------------------------------------

 +-----------------+                      +-----------------+
 | BYE             |=====================>| T_DISCONNECT    |
 +-----------------+                      +--||-------------+
             /\                               ||
             ||                               ||
             ||==============================||
 Legend:

 | Communication between
```

```
         | states in the same
         V protocol

         ======> Communication between IN call model and SIP
                 protocol state machine
```

Figure 5: Mapping from SIP to T_BCSM

6. Call flows

Two examples are provided here to understand how SIP protocol state machine and the IN call model work synchronously with each other.

In the first example, a SIP UAC originates a call request destined to a 800 freephone number:

```
   INVITE sip:18005551212@lucent.com SIP/2.0
   From: sip:16309795218@il0015vkg1.ih.lucent.com
   To: sip:18005551212@lucent.com
   Via: SIP/2.0/UDP il0015vkg1.ih.lucent.com
   Call-ID: 67188121@lucent.com
   CSeq: 1 INVITE
```

The request makes its way to the originating SIP network server running an IN call model. The SIP network server hands, at the very least, the To: field and the From: field to the IN layer for freephone number translation. The IN layer proceeds through its PICs and in the ANALYSE_INFO PIC consults the SCP for freephone translation. The translated number is returned to the SIP network server, which forwards the message to the next hop SIP proxy, with the freephone number replaced by the translated number:

```
   INVITE sip:16302240216@lucent.com SIP/2.0
   From: sip:16309795218@il0015vkg1.ih.lucent.com
   Via: SIP/2.0/UDP il0015vkg1.ih.lucent.com
   Via: SIP/2.0/UDP sip-in1.ih.lucent.com
   To: sip:18005551212@lucent.com
   Call-ID: 67188121@lucent.com
   CSeq: 1 INVITE
```

In the next example, a SIP UAC originates a call request destined to a 900 number:

```
   INVITE sip:19005551212@lucent.com SIP/2.0
   From: sip:16302240216@lucent.com
   To: sip:19005551212@lucent.com
```

```
   Via: SIP/2.0/UDP il0015vkg1.ih.lucent.com
   Call-ID: 88112@lucent.com
   CSeq: 1 INVITE
```

The request makes its way to the originating SIP network server running an IN call model. The SIP network server hands, at the very least, the To: field and the From: field to the IN layer for 900 number translation. The IN layer proceeds through its PICs and in the ANALYSE_INFO PIC consults the SCP for the translation. During the translation, the SCP detects that the originating party is not allowed to make 900 calls. It passes this information to the originating SIP network server, which informs the SIP UAC using SIP "403 Forbidden" response status code:

```
   SIP/2.0 403 Forbidden
   From: sip:16302240216@lucent.com
   To: sip:19005551212@lucent.com
   Via: SIP/2.0/UDP il0015vkg1.ih.lucent.com
   Call-ID: 88112@lucent.com
   CSeq: 1 INVITE
```

7. Implementation Status

The work described in this I-D has been implemented. A SIP proxy server has been written to map the states of the IN call model to the SIP protocol state machine as described in this I-D. A portable Call Model object has also been implemented in C++ and used successfully with the SIP proxy server to provide the services mentioned in this I-D. Implementation experience and other details related to it are provided in [7].

8. Security Considerations

The work described in this I-D did not focus too deeply into the security aspects. However, that does not imply that security considerations do not exist. Listed below are some of the security implications inherent in implementing services that cross (Internet and GSTN) domains:

(1) Obviously, if a SIP proxy server is to access IN services in the GSTN domain, a trust relationship should exist between the provider of the SIP proxy and the provider of the IN-related data (if they are not the same entity).

(2) A SIP proxy that receives SIP requests from an upstream UAC must be able to authenticate the (user of the) UAC for services

that are triggered on the detection points in the originating BCSM.

(3) Services such as Calling Name Delivery (triggered on the detection points in the TBCSM) can be easily accomplished in SIP by querying the From general header field of a SIP request. However, some guarantee must be made that the name displayed in the From header field and the person who originated the call are indeed the same entities.

Overall, the security considerations tend to cluster around the well known axes: authentication, authorization, encryption, and trust. There are various proposals, both SIP-specific, as well as general proposals, to deal with these issues. Further work needs to be done to codify and quantify the security threats and how to address them.

9. Acknowledgements

Many thanks to Alec Brusilovksy, Janet Douglas, Igor Faynberg, Jonathan Rosenberg, John Stanaway, and Kumar Vemuri for their insights, inputs, and comments.

10. Changes from earlier versions

10.1 Changes from draft -04

- o Changed text in Section 3 to specify possible interfaces between the IN call handling layer and the SCP.

- o In Section 3, took out the phrase "...using SIP as a signaling transport" since that conveyed the idea that the body of the SIP message consisted of INAP (or TCAP) payload. Rephrased the latter part of the second paragraph in Section 3.

10.2 Changes from draft -03

- o Changed references of SIP "server" to SIP "proxy" wherever appropriate.

- o Mapped IN PICs SELECT ROUTE and SELECT_FACILITY to SIP states.

10.3 Changes from draft -02

- o Added section on Security.

o Updated section on Implementation Status.

11. Abbreviations:

DP	Detection Point
GSTN	Global Switched Telephone Network
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IP	Internet Protocol or Intelligent Peripheral
LNP	Local Number Portability
O_BCSM	Originating Basic Call State Model
OCS	Originating Call Screening
PIC	Point In Call
PRI	Primary Rate Interface
SCP	Service Control Point
SIP	Session Initiation Protocol
SMF	Service Management Function
SS7	Signaling System 7
SSP	Service Switching Point
T_BCSM	Terminating Basic Call State Model
UAC	(SIP) User Agent Client
UAS	(SIP) User Agent Server

12. References:

[1] Uyless Black, "The Intelligent Network: Customizing Telecommunications and Services," Prentice Hall, 1998.

[2] I. Faynberg, L. Gabuzda, M. Kaplan, and N.Shah, "The Intelligent Network Standards: Their Application to Services," McGraw-Hill, 1997.

[3] ITU-T Q.1204 1993: Recommendation Q.1204, "Intelligent Network Distributed Functional Plane Architecture," International Telecommunications Union Standardization Section, Geneva.

[4] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol", IETF Standards RFC 2543, March 1999.

[5] J. Rosenberg, J. Peterson, H. Schulzrinne, "Third Party Call Control in SIP", IETF Internet Draft <draft-rosenberg-sip-3pcc-00.txt>, Expires September 2000, Work in Progress.

[6] Jonathan Rosenberg, Henning Schulzrinne, "A Framework for Telephony Routing Over IP", IETF Informational RFC 2871, June 2000.

[7] Vijay Gurbani, "SIP enabled IN Services - an implementation report", IETF Internet Draft <draft-gurbani-iptel-sip-in-imp-01.txt>, Expires May, 2001, Work in Progress.

10. Authors' Address

Vijay K. Gurbani
E-mail: vkg@lucent.com
Telephone: +1-630-224-0216
Fax: +1-630-713-5840
Lucent Technologies
2000 Lucent Lane, Rm 6G-440
Naperville, IL 60566 USA

Vidhi Rastogi
E-mail:vidhi.rastogi@wipro.com
Telephone: 91-80-5539134
Fax: 91-80-5539701
Wipro Technologies
271, Sri Ganesha Complex
Hosur Main Road, Madiwala
Bangalore - 560 068, INDIA

INTERNET-DRAFT Expires February 2002