Internet Engineering Task Force
INTERNET DRAFT
<draft-huitema-sgcp-v1-02.txt>

Mauricio Arango
RSL COM
Christian Huitema
Bellcore
July 30, 1998

Simple Gateway Control Protocol (SGCP)

Mauricio Arango, Christian Huitema

Version 1.1 draft

July 30, 1998

Status of this document

Abstract

This document describes an application programming interface (SGCI) and a corresponding protocol (SGCP) for controlling Voice over IP (VoIP) Gateways from external call control elements. The SGCP assumes a call control architecture where the call control 'intelligence' is outside the gateways and handled by external call control elements.

The document is structured in 5 main sections:

*   The introduction presents the basic assumptions and the relation to other protocols such as H.323, RTSP, SAP or SIP.

- The interface section presents a conceptual overview of the SGCP, presenting the naming conventions, the usage of the session description protocol SDP, and the five procedurs that compose SGCP: Notifications Request, Notification, Create Connection, Modify Connection and Delete Connection.

- The protocol description section presents the SGCP encodings, which are based on simple text formats, and the transmission procedure over UDP.

- The security section presents the security requirement of SGCP, and its usage of IP security services (IPSEC).

- An example section presents two detailed examples of call set up procedures using SGCP.

- The description of the changes between version 1.0 and version 1.1

1. Introduction

This document describes an application programming interface (SGCI) and a corresponding protocol (SGCP) for controlling Telephony Gateways from external call control elements. A telephony gateway is a network element that provides conversion between the audio signals carried on telephone circuits and data packets carried over the Internet or over other packet networks. Example of gateways are:

- Trunking gateways, that interface between the telephone network and a Voice over IP network. Such gateways typically manage a large number of digital circuits.

- Residential gateways, that provide a traditional analog (RJ11) interface to a Voice over IP network.

- Network Access Servers, that can attach a "modem" to a telephone circuit and provide data access to the Internet. We expect that, in the future, the same gateways will combine Voice over IP services and Network Access services.

The SGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and handled by external call control elements. The SGCP assumes that these call control elements, or Call Agents, will synchronize with each other to send coherent commands to the gateways under their control. The SGC defined in this document does not define a mechanism for synchronizing Call Agents.

The SGCP assumes a connection model where the basic constructs are endpoints and connections. Endpoints are sources or sinks of data and could

be physical or virtual. Examples of physical endpoints are:

- An interface on a gateway that terminates a trunk connected to a PSTN switch (e.g., Class 5, Class 4, etc.). A gateway that terminates trunks is called a trunk gateway.

- An interface on a gateway that terminates an analog POTS connection to a phone, key system, PBX, etc. A gateway that terminates residential POTS lines (to phones) is called a residential gateway.

An example of a virtual endpoint is an audio source in an audio-content server. Creation of physical endpoints requires hardware installation, while creation of virtual endpoints can be done by software.

Connections may be either point to point or multipoint. A point to point connection is an association between two endpoints with the purpose of transmitting data between these endpoints. Once this association is established for both endpoints, data transfer between these endpoints can take place. A multipoint connection is established by connecting the endpoint to a multipoint session.

1.1. Relation with the H.323 standards

The SGCP is designed as an internal protocol within a distributed system that appears to the outside as a single VoIP gateway. This system is composed of a Call Agent, that may or may not be distributed over several computer platforms, and of a set of gateways. In a typical configuration, this distributed gateway system will interface on one side with one or more telephony (i.e. circuit) switches, and on the other side with H.323 conformant systems, as indicated in the following table:

Functional Plane	Phone switch	Terminating Entity	H.323 conformant systems
Signaling Plane	Signaling exchanges through SS7/ISUP	Call agent	Signaling exchanges with the call agent through H.225/RAS and H.225/Q.931.
			Possible negotiation of logical channels and transmission parameters through H.245 with the call agent.
		Internal synchronization through SGCP	
Bearer Data Transport Plane	Connection through high speed trunk groups	Telephony gateways	Optional negotiation of logical channels and transmission parameters through H.245 directly with the telephony gateway. data using RTP, directly between the H.323 station and the gateway.

In the SGCP model, the gateways focus on the audio signal translation function, while the Call Agent handles the signaling and call processing functions. As a consequence, the Call Agent implements the "signaling" layers of the H.323 standard, and presents itself as a "Gatekeeper" to the H.323 systems. Calls are established using the "Gatekeeper Routed" call model.

1.2. Relation with the IETF standards

While H.323 is the recognized standard for VoIP terminals, the IETF has also produced specifications for other types of multi-media applications. These other specifications include:

* the Session Description Protocol (SDP), RFC 2327,

* the Session Announcement Protocol (SAP),

* the Session Initiation Protocol (SIP),

* the Real Time Streaming Protocol (RTSP), RFC 2326.

The latter three specifications are in fact alternative signaling standards that allow for the transmission of a session description to an interested party. SAP is used by multicast session managers to distribute a multicast session description to a large group of recipients, SIP is used to invite an individual user to take part in a point-to-point or unicast session, RTSP is used to interface a server that provides real time data. In all three cases, the session description is described according to SDP; when audio is transmitted, it is transmitted through the Real-time Transport Protocol, RTP.

The distributed gateway systems and SGCP will enable PSTN telephony users to access sessions set up using SAP, SIP or RTSP. The Call Agent provides for signaling conversion, according to the following table:

Functional Plane	Phone switch	Terminating Entity	IETF conforming systems
Signaling Plane	Signaling exchanges through SS7/ISUP	Call agent	Signaling exchanges with the call agent through SAP, SIP or RTSP.
			Negotiation of session description parameters through SDP (telephony gateway terminated but passed via the call agent to and from the IETF conforming system)
Bearer Data Transport Plane	Connection through high speed trunk groups	Internal Telephony gateways	Transmission of VoIP data using RTP, directly between the remote IP end system and the gateway.

The SDP standard has a pivotal status in this architecture. We will see in the following description that we also use it to carry session descriptions in SGCP.

1.3. Definitions

Trunk: A communication channel between two switching systems. E.g., a DS0 on a T1 or E1 line.

2. Simple Gateway Control Interface

The interface functions provide for connection control and endpoint control. Both use the same system model and the same naming conventions.

2.1. Model and naming conventions.

The SGCP assumes a connection model where the basic constructs are endpoints and connections. Connections are grouped in calls. One or more connections can belong to one call. Connections and calls are set up at the initiative of one or several Call Agents.

2.1.1. Names of endpoints

Endpoints names have two components:

- the domain name of the gateway that is managing the endpoint ,
- a local name within that gateway.

In the case of trunking gateways, endpoints are trunk circuits linking a gateway to a telephone switch. These circuits are typically grouped into a digital multiplex, that is connected to the gateway by a physical interface. Such circuits are named in three contexts:

- In the ISUP protocol, trunks are grouped into trunk groups, identified by the SS7 point codes of the switches that the group connects. Circuits within a trunk group are identified by a circuit number (CIC in ISUP).
- In the gateway configuration files, physical interfaces are typically identified by the name of the interface, an arbitrary text string. When the interface multiplexes several circuits, individual circuits are typically identified by a circuit number.

*   In SGCP, the endpoints are identified by an endpoint

     name.

The Call Agents use configuration databases to map ranges of circuit numbers within an ISUP trunk group to corresponding ranges of circuits in a multiplex connected to a gateway through a physical interface. The gateway will be identified, in SGCP, by a domain name. The local name will be structured to encode both the name of the physical interface, for example X35V3+A4, and the circuit number within the multiplex connected to the interface, for example 13. The circuit number will be separated from the name of the interface by a fraction bar, as in:

```
        X35V3+A4/13
```

The circuit number can be omitted when the physical interface only includes one circuit, or if the Call Agent requests the gateway to choose one available circuit within a multiplex.

Other types of endpoints will use different conventions. For example, an endpoint that produces an "all lines busy" announcement could be named:

```
        all-lines-busy/loop@announce-23.whatever.net
```

The exact syntax of such names should be specified in the corresponding server specification.

2.1.2. Names of calls

Calls are identified by unique identifiers, independent of the underlying platforms or agents. These identifiers are created by the Call Agent. They are treated in SGCP as unstructured octet strings.

Call identifiers are expected to be unique within the system. When a Call Agent builds several connections that pertain to the same call, either on the same gateway or in different gateways, these connections will all be linked to the same call through the globally unique identifier. This identifier can then be used by accounting or management procedures, which are outside the scope of SGCP.

2.1.3. Names of connections

Connection identifiers are created by the gateway when it is requested to create a connection. They identify the connection within the context of an endpoint and a call. They are treated in SGCP as unstructured

octet strings. The gateway should make sure that a proper waiting period, at least 3 minutes, elapses between the end of a connection that used this identifier and its use in a new connection.

2.1.4. Names of Call Agents and other entities

The simple gateway control protocol has been designed to allow the implementation of redundant Call Agents, for enhanced network reliability. This means that there is no fixed binding between entities and hardware platforms or network interfaces.

Reliability can be improved by the following precautions:

* Entities such as endpoints or Call Agents are identified by their domain name, not their network addresses. Several addresses can be associated with a domain name. If a command or a response cannot be forwarded to one of the network addresses, implementations should retry the transmission using another address.

* Entities may move to another platform. The association between a logical name (domain name) and the actual platform are kept in the domain name service. Call Agents and Gateways should keep track of the time-to-live of the record they read from the DNS. They should query the DNS to refresh the information if the time to live has expired.

2.1.5. Digit maps

The Call Agent can ask the gateway to collect digits dialed by the user. This facility is intended to be used with residential gateways to collect the numbers that a user dials; it may also be used with trunking gateways and access gateways alike, to collect the access codes, credit card numbers and other numbers requested by call control services.

An alternative procedure is for the gateway to notify the Call Agent of the dialed digits, as soon as they are dialed. However, such a procedure generates a large number of interactions. It is preferable to accumulate the dialed numbers in a buffer, and to transmit them in a single message.

The problem with this accumulation approach, however, is that it is hard for the gateway to predict how many numbers it needs to accumulate before transmission. For example, using the phone on our desk, we can dial the following numbers:

0	Local operator
00	Long distance operator
xxxx	Local extension number
8xxxxxxx	Local number
#xxxxxxx	Shortcut to local number at other corporate sites
*xx	Star services
91xxxxxxxxxx	Long distance number
9011 + up to 14 digits	International number

The solution to this problem is to load the gateway with a digit map that correspond to the dial plan. This digit map is expressed using a syntax derived from the Unix system command, egrep. For example, the dial plan described above results in the following digit map:

```
         (0T| 00T|[1-7]xxx|8xxxxxxx|#xxxxxxx|*xx|91xxxxxxxxxx|9011x.T)
```

The formal syntax of the digit map is described by the following BNF notation:

```
        Digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" |
                  "7" | "8" | "9"

        Timer ::= "T" -- matches the detection of a timer

        Letter::= Digit | Timer | "#" | "*" | "A" | "B" | "C"
                  | "D"

        Range ::= "x" -- matches any digit
                 | "[" Letters "]" -- matches any of the specified letters

        Letters ::= Subrange | Subrange Letters

        Subrange ::= Letter -- matches the specified letter
          | Digit "-" Digit -- matches any digit between first and
                            -- last
        Position ::= Letter
                   | Range

        StringElement::= Position -- matches an occurrence of
                                  -- the position
                   | Position "." -- matches an arbitrary number of
                                  -- occurrences
                                  -- of the position, including 0
        String::= StringElement | StringElement String

        StringList::= String | String "|" StringList

        DigitMap::= String | "(" StringList ")"
```

A DigitMap, according to this syntax, is defined either by a "string" or by a list of strings. Each string in the list is an alternative numbering scheme. A gateway that detects digits, letters or timers will:

1. **Add the event parameter code as a token to the end of an internal state variable called the "current dial string"**

2. **Apply the current dial string to the digit map table, attempting a match to each regular expression in the Digit Map in lexical order**

3. **If the result is under-qualified (partially matches at least one entry in the digit map), do nothing further.**

If the result matches, or is over-qualified (i.e. no further digits could possibly produce a match), send the current digit string to the Call Agent.

Digit maps are provided to the gateway by the Call Agent, whenever the Call Agent instructs the gateway to listen for digits.

2.2. Usage of SDP

The Call Agent uses the SGCP to provision the gateways with the description of connection parameters such as IP addresses, UDP port and RTP profiles. These descriptions will follow the conventions delineated in the Session Description Protocol which is now an IETF proposed standard, documented in RFC 2327.

SDP allows for description of multimedia conferences. This version limits SDP usage to the setting of audio circuits and data access circuits. The initial session descriptions contain the description of exactly one media, of type "audio" for audio connections, "nas" for data access.

2.3. Gateway Control Functions

This section describes the commands of the SGCP. The service consists of connection handling and endpoint handling commands. There are five commands in the protocol:

*   The Call Agent can issue a NotificationRequest command to a gateway, instructing the gateway to watch for specific events such as hook actions or DTMF tones on a specified endpoint .

*   The gateway will then use the Notify command to inform the Call Agent when the requested events occur.

*   The Call Agent can use the CreateConnection command to create a connection that terminates in an "endpoint" inside the gateway.

*   The Call Agent can use the ModifyConnection command to change the parameters associated to a previously established connection.

*   The Call Agent can use the DeleteConnection command to delete an existing connection. The DeleteConnection command may also be used by a gateway to indicate that a connection can no longer be sustained.

These services allow a controller (normally, the Call Agent) to instruct a gateway on the creation of connections that terminate in an "endpoint" attached to the gateway, and to be informed about events occurring at the endpoint. An endpoint may be for example:

*   A specific trunk circuit, within a trunk group terminating in a

gateway,

- A specific announcement handled by an announcement server.

Connections are grouped into "calls". Several connections may belong to the same call, terminating in the same endpoint . Each connection is qualified by a "mode" parameter, which can be set to "send", "receive", "send/receive", data, "inactive", "loopback" or "continuity test."

The handling of the audio signals received on these connections is determined by the mode parameters:

- Audio signals received in data packets through connections in "receive" or "send/receive" mode are mixed and sent to the endpoint.
- Audio signals originating from the endpoint are transmitted over all the connections whose mode is "send" or "send/receive."

The "loopback" and "continuity test" modes are used during maintenance and continuity test operations. There are two flavors of continuity test, one specified by ITU and one used in the US. In the first case, the test is a loopback test. The originating switch will send a tone (the go tone) on the bearer circuit and expect the terminating switch to loopback the circuit. If the originating switch sees the same tone returned (the return tone), the COT has passed. If not, the COT has failed. In the second case, the go and return tones are different. The originating switch sends a certain go tone. The terminating switch detects the go tone, it asserts a different return tone in the backwards direction. When the originating switch detects the return tone, the COT is passed. If the originating switch never detects the return tone, the COT has failed.

If the mode is set to "loopback", the gateway is expected to return the incoming signal from the endpoint back into that same endpoint. This procedure will be used, typically, for testing the continuity of trunk circuits according to the ITU specifications.

If the mode is set to "continuity test", the gateway is informed that the other end of the circuit has initiated a continuity test procedure according to the GR specification. The gateway will place the circuit in the transponder mode required for dual-tone continuity tests.

2.3.1. NotificationRequest

The NotificationRequest commands are used to request the gateway to send notifications upon the occurrence of specified events in an endpoint. For example, a notification may be requested for when a gateway detects that an endpoint is receiving tones associated with fax communication. The entity receiving this notification may decide to use a different type of encoding method in the connections bound to this endpoint.

```
                NotificationRequest( EndpointId,
                                     NotifiedEntity,
                                     RequestedEvents,
                                     RequestIdentifier,
                                     DigitMap,
                                     SignalRequests)
```

EndpointId is the identifier for the endpoint in the gateway where NotificationRequest executes.

NotifiedEntity is an optional parameter that specifies where the notifications should be sent. When this parameter is absent, the notifications should be sent to the originator of the NotificationRequest.

RequestIdentifier is used to correlate this request with the notifications that it triggers.

RequestedEvents is a list of events that the gateway is requested to detect and report. Such events include, for example:

- fax tones,

- modem tones,

- continuity tone,

- continuity detection (as a result of a continuity test),

- on-hook transition (occurring in classic telephone sets when the user hangs-up the handset),

- off-hook transition (occurring in classic telephone sets when the user lifts the handset),

- flash hook (occurring in classic telephone sets when the user briefly presses the hook that holds the handset),

- wink,
- DTMF digits (or pulse digits).

To each event is associated an action, which can be:

- Notify the event immediately, together with the accumulated list of observed events,
- Swap audio,
- Accumulate according to Digit Map,
- Ignore the event.

Events that are not specified in the list will, by default, be ignored.

The Swap Audio action can be used when a gateway handles more than one active connection on an endpoint. This will be the case for three-way calling, call waiting, and possibly other feature scenarios. In order to avoid the round-trip to the Call Agent when just changing which connection is attached to the audio functions of the endpoint, the NotificationRequest can map an event (usually hook flash, but could be some other event) to a local function swap audio, which selects the "next" connection in a round robin fashion. If there is only one connection, this action is effectively a no-op.

Hook transition events are normally observed only by access gateways. Tone detection can be done by any gateway, including trunking gateways. The Call Agent can send a NotificationRequest whose RequestedEvent list is empty. It will do so, for example, to an access gateway when it does not want to collect any more DTMF digits.

DigitMap is an optional parameter that allows the Call Agent to provision the gateways with a digit map according to which digits will be accumulated. This parameter must be present if the RequestedEvent parameters contain an request to "accumulate according to the digit map." The collection of these digits will result in a digit string. The digit string is initialized to a null string upon reception of the NotificationRequest, so that a subsequent notification only returns the digits that were collected after this request.

SignalRequests is a parameter that contains the set of actions that the gateway is asked to perform on the endpoint , such as, for example:

* Ringing,

* Distinctive ringing, which can occur in 8 variants numbered 0 to 7,

* Ring back tone,

* Dial tones,

* Intercept tone,

* Network Congestion tone,

* Busy tone,

* Confirm tone,

* Answer tone,

* Call waiting tone,

* Off hook warning tone,

* Preemption tone,

* Continuity tones,

* Continuity test,

* Announcement (completed by an announcement name and a parameter),

* ASDI display (and string to be displayed).

The action triggered by the SignalRequests is synchronized with the collection of events specified in the RequestedEvents parameter. For example, if the NotificationRequest mandates "ringing" and the event request ask to look for an "off-hook" event, the ringing shall stop as soon as the gateway detect an off hook event.

The specific definition of actions that are requested via these SignalRequests, such as the duration of and frequency of a DTMF digit, is outside the scope of SGC. This definition may vary from location to location and hence from gateway to gateway.

The RequestedEvents and SignalRequests refer to the same events. In one case, the gateway is asked to detect the occurrence of the event, and in the other case it is asked to generate it. There are only a few exception to this rule, notably the ASDI display, which can only be signaled

but not detected, and the fax and modem tones, which can be detected but can not be signaled. However, we cannot necessarily expect all endpoint s to detect all events. For example, a digital trunk interface would normally not detect hook events, and may not be able to modulate DTMF digit; signals such as "wink" only make sense in some types of trunks. Gateways that receive a request to detect an event that they are not equipped to detect, or to generate a signal that they are not equipped to generate, should refuse the request and return an error code. Similarly, gateways that are equipped to generate announcements but cannot generate the specific announcement that the Call Agent requested, should return an appropriate error code.

The Call Agent can send a NotificationRequest whose requested signal list is empty. It will do so for example when tone generation should stop.

2.3.2. Notifications

Notifications are sent via the Notify command and are sent by the gateway when the observed events occur.

```
                   Notify( EndpointId,
                           NotifiedEntity,
                           RequestIdentifier,
                           ObservedEvents)
```

EndpointId is the identifier for the endpoint in the gateway which is issuing the Notify command.

NotifiedEntity is an optional parameter that identifies the entity to which the notifications is sent. This parameter is equal to the NotifiedEntity parameter of the NotificationRequest that triggered this notification. The parameter is absent if there was no such parameter in the triggering request. In this case, the notification is sent to the entity from which the request was received.

RequestIdentifier is parameter that repeats the RequestIdentifier parameter of the NotificationRequest that triggered this notification. It is used to correlate this notifications with the request that triggered it.

ObservedEvents is a list of events that the gateway detected. A single notification may report a list of events that will be reported in the order in which they were detected. The list may only contain the identification of events that were requested in the RequestedEvents parameter of the triggering NotificationRequest. It will contain the events that were either accumulated (but not notified) or treated according to digit map (but no match yet), and the final event that triggered the

detection or provided a final match in the digit map.

2.3.3. CreateConnection

This command is used to create a connection.

```
            ConnectionId,
            [SpecificEndPointId,]
            [LocalConnectionDescriptor]
                     <--- CreateConnection(CallId,
                                              EndpointId,
                                              NotifiedEntity,
                                              LocalConnectionOptions,
                                              Mode,
                                              RemoteConnectionDescriptor,
                                              RequestedEvents,
                                              RequestIdentifier,
                                              DigitMap,
                                              SignalRequests)
```

This function creates a connection between two endpoints. A connection is defined by its endpoints. The input parameters in CreateConnection provide the data necessary to build a gateway's "view" of a connection.

CallId is a globally unique parameter that identifies the call (or session) to which this connection belongs. This parameter is unique within the whole network of gateways; connections that belong to the same call share the same call-id. The call-id can be used to identify calls for reporting and accounting purposes.

EndpointId is the identifier for the connection endpoint in the gateway where CreateConnection executes. The EndpointId can be fully-specified by assigning a value to the parameter EndpointId in the function call or it may be under-specified and the full value will be assigned by the gateway and its complete value returned in the SpecificEndPointId parameter of the response.

The NotifiedEntity is an optional parameter that specifies where the Notify or DeleteConnection commands should be sent. If the parameter is absent, the Notify or DeleteConnection commands should be sent to the originator of the CreateConnection command.

LocalConnectionOptions is a structure that describes the characteristics of the data communications from the point of view of the gateway executing CreateConnection. The fields contained in LocalConnectionOptions are the following:

- Encoding Method
- Packetization period
- Bandwidth
- Type of Service,
- Usage of echo cancellation

The values of these fields are defined in the SDP standard. For each of the first three fields, the Call Agent has three options:

- It may state exactly one value, which the gateway will then use for the connection,
- It may provide a loose specification, such as a list of allowed encoding methods or a range of packetization periods,
- It may simply provide a bandwidth indication, leaving the choice of encoding method and packetization period to the gateway.

The bandwidth specification shall not contradict the specification of encoding methods and packetization period. If an encoding method is specified, then the gateway is authorized to use it, even if it results in the usage of a larger bandwidth than specified.

The LocalConnectionOptions parameter may be absent in the case of a data call.

The Type of Service specifies the class of service that will be used for the connection. When the connection is transmitted over an IP network, the parameters encodes the 8-bit type of service value parameter of the IP header. When the Type of Service is not specified, the gateway shall use a default or configured value.

By default, the telephony gateways always perform echo cancellation. However, it is necessary, for some calls, to turn off these operations. The echo cancellation parameter can have two values, "on" (when the echo cancellation is requested) and "off" (when it is turned off.)

RemoteConnectionDescriptor is the connection descriptor for the remote side of a connection, on the other side of the IP network. It includes the same fields as in the LocalConnectionDescriptor, i.e. the fields that describe a session according to the SDP standard. This parameter may have a null value when the information for the remote end is not

known yet. This occurs because the entity that builds a connection starts by sending a CreateConnection to one of the two gateways involved in it. For the first CreateConnection issued, there is no information available about the other side of the connection. This information may be provided later via a ModifyConnection call. In the case of data connections (mode=data), this parameter describes the characteristics of the data connection.

Mode indicates the mode of operation for this side of the connection. The options are: FullDuplex, ReceiveOnly, SendOnly, Inactive, Loopback, Continuity Test and Data

* FullDuplex indicates that this side of the connection sends data generated by the associated endpoint and can receive data from the remote end and feed it to the endpoint.

* ReceiveOnly indicates this connection end can only receive data form the remote end and feed it to the associated endpoint.

* SendOnly indicates that this connection end can only send data from its endpoint to the other end of the connection.

* Inactive indicates that this connection end does not send or receive any data. The connection exists but is not active. It has to be explicitly activated by a ModifyConnection command.

* Loopback indicates that the circuit to which the endpoint refers should be placed in loopback mode, so that audio signals received from the phone circuit are sent back on this same circuit.

* Continuity Test indicates that the other end of the circuit has initiated a continuity test procedure according to the GR specification. The gateway will place the circuit in the transponder mode required for dual-tone continuity tests.

* Data indicates that the circuit to which the endpoint refers will used for network access (such as Internet access) rather than for Internet telephony.

The gateway returns a ConnectionId, that uniquely identifies the connection within one endpoint , and a LocalConnectionDescriptor, which is a session description that contains information about addresses and RTP ports, as defined in SDP. The LocalConnectionDescriptor is not returned in the case of data connections. The SpecificEndPointId is an optional parameter that identifies the responding endpoint. It can be used when the EndpointId argument referred to a generic endpoint name. When a

SpecificEndPointId is returned, the Call Agent should use it as the EndpointId value is successive commands referring to this call.

After receiving a "CreateConnection" request that did not include a RemoteConnectionDescriptor parameter, a gateway is in an ambiguous situation. Because it has exported a LocalConnectionDescriptor parameter, it can potentially receive packets. Because it has not yet received the RemoteConnectionDescriptor parameter of the other gateway, it does not know whether the packets that it receives have been authorized by the Call Agent. It must thus navigate between two risks, i.e. clipping some important announcements or listening to insane data. The behavior of the gateway is determined by the value of the Mode parameter:

* If the mode was set to ReceiveOnly, the gateway should accept the voice signals and transmit them through the endpoint .

* If the mode was set to Inactive, Loopback or Continuity Test, the gateway should refuse the voice signals.

Note that the mode values FullDuplex and SendOnly don't make sense in this situation. They should be treated as ReceiveOnly and Inactive.

The RequestedEvents, RequestIdentifier, DigitMap, and SignalRequests parameters are optional. They can be used by the Call Agent to transmit a NotificationRequest that is executed simultaneously with the creation of the connection. For example, when the Call Agent wants to initiate a call to an residential gateway, it should:

* ask the residential gateway to prepare a connection, in order to be sure that the user can start speaking as soon as the phone goes off hook,

* ask the residential gateway to start ringing,

* ask the residential gateway to notify the Call Agent when the phone goes off-hook.

This can be accomplished in a single CreateConnection command, by also transmitting the RequestedEvent parameters for the off hook event, and the SignalRequest parameter for the ringing signal.

When these parameters are present, the creation and the NotificationRequests should be synchronized, which means that both should be accepted, or both refused. In our example, the CreateConnection may be refused if

the gateway does not have sufficient resources, or cannot get adequate resources from the local network access, and the off-hook Notification-Request can be refused in the glare condition, if the user is already off-hook. In this example, the phone should not ring if the connection cannot be established, and the connection should not be established if the user is already off hook.

2.3.4. ModifyConnection

This command is used to modify the characteristics of a gateway's "view" of a connection. This "view" of the call includes both the local connection descriptors as well as the remote connection descriptor.

```
           [LocalConnectionDescriptor]
                    <--- ModifyConnection(CallId,
                                          EndpointId,
                                          ConnectionId,
                                          NotifiedEntity,
                                          LocalConnectionOptions,
                                          Mode,
                                          RemoteConnectionDescriptor,
                                          RequestedEvents,
                                          RequestIdentifier,
                                          DigitMap,
                                          SignalRequests)
```

The parameters used are the same as in the CreateConnection command, with the addition of a ConnectionId that identifies the connection within the call. This parameter is returned by the CreateConnection function, as part of the local connection descriptor. It uniquely identifies the connection within the call.

The ModifyConnection command can be used to affect parameters of a connection in the following ways:

*   Provide information on the other end of the connection, through the RemoteConnectionDescriptor.

*   Activate or deactivate the connection, by changing the value of the Mode parameter. This can occur at any time during the connection, with arbitrary parameter values.

*   Change the sending parameters of the connection, for example by switching to a different coding scheme, changing the packetization period, or modifying the handling of echo cancellation.

Connections can only be activated if the RemoteConnectionDescriptor has been provided to the gateway.

The command will only return a LocalConnectionDescriptor if the local connection parameters, such as RTP ports, were modified. (Usage of this feature is actually for further study.)

The RequestedEvents, RequestIdentifier, DigitMap, and SignalRequests parameters are optional. They can be used by the Call Agent to transmit a NotificationRequest that is executed simultaneously with the modification of the connection. For example, when a connection is accepted, the calling gateway should be instructed to place the circuit in send-receive mode and to stop providing ringing tones.

This can be accomplished in a single ModifyConnection command, by also transmitting the RequestedEvent parameters, for the on hook event, and an empty SignalRequest parameter, to stop the provision of ringing tones.

When these parameters are present, the modification and the NotificationRequests should be synchronized, which means that both should be accepted, or both refused.

2.3.5. DeleteConnection (from the Call Agent)

This command is used to terminate a connection. As a side effect, it collects statistics on the execution of the connection.

```
           Connection-parameters <-- DeleteConnection(CallId,
                                                      EndpointId,
                                                      ConnectionId,
                                                      NotifiedEntity,
                                                      RequestedEvents,
                                                      RequestIdentifier,
                                                      DigitMap,
                                                      SignalRequests)
```

In the general case where a connection has two ends, this command has to be sent to both gateways involved in the connection. Some connections, however, may use IP multicast. In this case, they can be deleted individually.

After the connection has been deleted, the endpoint should be placed in inactive mode. Any loopback that has been requested for the connection should be cancelled.

In response to the DeleteConnection command, the gateway returns a list of parameters that describe the status of the connection. These parameters are: <DIR>

Number of packets sent:

The total number of RTP data packets transmitted by the sender since starting transmission on this connection. The count is not reset if the sender changes its synchronization source identifier (SSRC, as defined in RTP), for example as a result of a Modify command. The value is zero if the connection was set in "receive only" mode.

Number of octets sent:

The total number of payload octets (i.e., not including header or padding) transmitted in RTP data packets by the sender since starting transmission on this connection. The count is not reset if the sender changes its SSRC identifier, for example as a result of a ModifyConnection command. The value is zero if the connection was set in "receive only" mode.

Number of packets received:

The total number of RTP data packets received by the sender since starting reception on this connection. The count includes packets received from different SSRC, if the sender used several values. The value is zero if the connection was set in "send only" mode.

Number of octets received:

The total number of payload octets (i.e., not including header or padding) transmitted in RTP data packets by the sender since starting transmission on this connection. The count includes packets received from different SSRC, if the sender used several values. The value is zero if the connection was set in "send only" mode.

Number of packets lost:

The total number of RTP data packets that have been lost since the beginning of reception. This number is defined to be the number of packets expected less the number of packets actually received, where the number of packets received includes any which are late or duplicates. The count includes packets received from different SSRC, if the sender used several values. Thus packets that arrive late are not counted as lost, and the loss may be negative if there are duplicates. The count includes packets received from different SSRC, if the sender used several values. The number of packets expected is defined to be the extended last sequence number received, as defined next, less the

initial sequence number received. The count includes packets received from different SSRC, if the sender used several values. The value is zero if the connection was set in "send only" mode. This parameter is omitted if the connection was set in "data" mode.

Interarrival jitter:

An estimate of the statistical variance of the RTP data packet interarrival time measured in milliseconds and expressed as an unsigned integer. The interarrival jitter J is defined to be the mean deviation (smoothed absolute value) of the difference D in packet spacing at the receiver compared to the sender for a pair of packets. Detailed computation algorithms are found in RFC 1889. The count includes packets received from different SSRC, if the sender used several values. The value is zero if the connection was set in "send only" mode. This parameter is omitted if the connection was set in "data" mode.

Average transmission delay:

An estimate of the network latency, expressed in milliseconds. This is the average value of the difference between the NTP timestamp indicated by the senders of the RTCP messages and the NTP timestamp of the receivers, measured when this messages are received. The average is obtained by summing all the estimates, then dividing by the number of RTCP messages that have been received. This parameter is omitted if the connection was set in "data" mode. </DIR>

For a detailed definition of these variables, refer to RFC 1889.

The NotifiedEntity, RequestedEvents, RequestIdentifier, DigitMap, and SignalRequests parameters are optional. They can be used by the Call Agent to transmit a NotificationRequest that is executed simultaneously with the deletion of the connection. For example, when a user hangs up is accepted, the gateway should be instructed to delete the connection and to start looking for an off hook event.

This can be accomplished in a single DeleteConnection command, by also transmitting the RequestedEvent parameters, for the off hook event, and an empty SignalRequest parameter.

When these parameters are present, the DeleteConnection and the NotificationRequests should be synchronized, which means that both should be accepted, or both refused.

2.3.6. DeleteConnection (from the VoIP gateway)

In some circumstances, a gateway may have to clear a connection, for example because it has lost the resource associated with the connection, or because it has detected that the endpoint no longer is capable or willing to send or receive voice. The gateway terminates the connection by using a variant of the DeleteConnection command:

```
                DeleteConnection( CallId,
                                  EndpointId,
                                  ConnectionId,
                                  Reason-code,
                                  Connection-parameters)
```

In addition to the call, endpoint and connection identifiers, the gateway will also send the call's parameters that would have been returned to the Call Agent in response to a DeleteConnection command. The reason code indicates the cause of the disconnection.

2.3.7. DeleteConnection (multiple connections, from the Call Agent)

A variation of the DeleteConnection function can be used by the Call Agent to delete multiple connections at the same time. The command can be used to delete all connections that relate to a Call for an endpoint:

```
                DeleteConnection( CallId,
                                  EndpointId)
```

It can also be used to delete all connections that terminate in a given endpoint:

```
                DeleteConnection( EndpointId)
```

Finally, Call Agents can take advantage of the hierarchical naming structure of endoints to delete all the connections that belong to a group of endpoints. For example, if endpoints names are structured as the combination of a physical interface name and a circuit number, as in "X35V3+A4/13", the Call Agent may replace the circuit number by a wild card character "*", as in "X35V3+A4/*". This "wildcard" command instructs the gateway to delete all the connections that where attached to circuits connected to the physical interface "X35V3+A4".

After the connections have been deleted, the endpoint should be placed in inactive mode. Any loopback that has been requested for the connections should be cancelled.

This command does not return any individual statistics or call parameters.

2.4. Race conditions

In order to implement proper call signalling, the Call Agent must keep track of the state of the endpoint, and the gateway must make sure that events are properly notified to the call agent. SGCP deals with race conditions through the notion of a "quarantine list" and through explicit detections of desynchronizations.

2.4.1. Quarantine list

SGCP controlled gateways will receive "notification requests" that ask them to watch for a list of "events." The two protocol elements that determine the handling of these events are the "Requested Events" list and the "Digit Map."

When the endpoint is initialized, the requested events list and the digit map are empty. After reception of a command, the gateway starts observing the endpoint for occurences of the event mentioned in the list.

The events are examined as they occur. The action that follows is determined by the "action" parameter associated to the event in the list of requested events, and also by the digit map. The events that are defined as "accumulate" are accumulated in a list of events, the events that are marked as treated according to the digit map will be accumulated in the dialed string. This will go on until one event is encountered that triggers a Notification to the "notified entity."

The gateway, at this point, will transmit the notification command and will place the endpoint in a "notification" state. As long as the endpoint is in this notification state, the events that may occur on the endpoint are accumulated, but not transmitted, even if they had normally triggered a notification. The events are, in a sense, "quarantined." All events for which the request list specifies actions such as Notify, Accumulate or "Treat according to digit map" should be quarantined.

The endpoint exits the notification state when the acknowledgement of the notification command is received. During that period, the notification command may be retransmitted, as specified in section 3.5.

When the gateway receives the acknowledgement, it resets the "current dial string" of the endpoint to a null value and starts processing the list of quarantined events, using the already received list of requested events and digit map. When processing these events, the gateway may

encounter an event which requires a Notify command to be sent. If that is the case, the gateway can adopt one of the two following behaviors:

* it can immediately transmit a Notify command that will report all events that where accumulated until the triggering event, included, leaving the unprocessed events in the quarantine list,

* or it can attempt to empty the quarantined list and transmit a single Notify command reporting several sets of events and possibly several dial strings. The dial string is reset to a null value after each triggering event that satisfies the DigitMap. The events that follow the last triggering event are left in the quarantine list.

If the gateway transmits a Notify command, the end point will remain in the notification state until the acknowledgement is received. If the gateway does not find a quarantined event that requests a notification, it places the end point in a normal state. Events are then processed as they come, in exactly the same way as if a Notification Request command had just been received.

A gateway may receive at any time a new Notification Request for the end point. When a new notification request is received in the notification state, the list of quarantined events is emptied, and the endpoint is taken out of the notification state without waiting for the acknowledgement of the notification command. The pending notification command will not be repeated.

Whatever the state of the endpoint, the requested events list and digit map are replaced by the newly received parameters, the accumulated events are discarded, and the accumulated dial string is reset to a null value.

2.4.2. Explicit detection

A key element of this state is the position of the hook. A race condition may occur when the user decides to go off-hook before the Call Agent has the time to ask the gateway to notify an off hook event (the "glare" condition well known in telephony), or if the user goes on-hook before the Call Agent has the time to request the event's notification.

To avoid this race condition, the gateway should check the condition of the endpoint before acknowledging a NotificationRequest. It should return an error:

1- If the gateway is requested to notify an "off hook" transition while the phone is already off hook,

2- If the gateway is requested to notify an "on hook" or "flash hook" condition while the phone is already on hook.

The other state variables of the gateway, such as the list of RequestedEvent or list of requested signals, are entirely replaced after each successful NotificationRequest, which prevents any long term discrepancy between the Call Agent and the gateway.

When a NotificationRequest is unsuccessful, the list of RequestedEvents and requested signals are emptied. They must be reinstated by a new request.

Another race condition may occur when a notification is issued shortly before the reception by the gateway of a NotificationRequest. The RequestIdentifier is used to correlate Notify commands with NotificationRequest commands.

2.5. Return codes and error codes.

All SGCP commands are acknowledged. The acknowledgment carries a return code, which indicates the status of the command. The return code is an integer number, for which three ranges of values have been defined:

* values between 200 and 299 indicate a successful completion,

* values between 400 and 499 indicate a transient error,

* values between 500 and 599 indicate a permanent error.

The values that have been already defined are listed in the following table:

Code	Meaning
200	The requested transaction was executed normally.
250	The connection was deleted.
400	The transaction could not be executed, due to a transient error.
401	The phone is already off hook
402	The phone is already on hook
500	The transaction could not be executed, because the endpoint is unknown.
501	The transaction could not be executed, because the endpoint is not ready.
502	The transaction could not be executed, because the endpoint does not have sufficient resources
510	The transaction could not be executed, because a protocol error was detected.
511	The transaction could not be executed, because the command contained an unrecognized extension.
512	The transaction could not be executed, because the gateway is not equipped to detect one of the requested events.
513	The transaction could not be executed, because the gateway is not equipped to generate one of the requested signals.
514	The transaction could not be executed, because the gateway cannot send the specified announcement.
515	The transaction refers to an incorrect connection-id (may have been already deleted)

3. Simple Gateway Control Protocol

The SGCP implements the simple gateway control interface as a set of transactions. The transactions are composed of a command and a mandatory response. There are five types of command:

*   CreateConnection

*   ModifyConnection

*   DeleteConnection

- NotificationRequest

- Notify

The first four commands are sent by the Call Agent to a gateway. The Notify command is sent by the gateway to the Call Agent. The gateway may also send a DeleteConnection as defined in 2.3.6.

3.1. General description

All commands are composed of a Command header, optionally followed by a session description.

All responses are composed of a Response header, optionally followed by a session description.

Headers and session descriptions are encoded as a set of text lines, separated by a line feed character. The headers are separated from the session description by an empty line.

SGCP uses a transaction identifier to correlate commands and responses. The transaction identifier is encoded as a component of the command header and repeated as a component of the response header (see section 3.2.1, 3.2.1.2 and 3.3).

Transaction identifiers have values between 1 and 999999999. An SGCP entity shall not reuse an identifier sooner than 3 minutes after completion of the previous command in which the identifier was used.

3.2. Command Header

The command header is composed of:

- A command line, identifying the requested action or verb, the endpoint towards which the action is requested, and the SGC protocol version,

- A set of parameter lines, composed of a parameter name followed by a parameter value.

3.2.1. Command line

The command line is composed of:

- The name of the requested verb,

- The identification of the transaction,

- The name of the endpoint that should execute the command (in notifications, the name of the endpoint that is issuing the notification),

- The protocol version.

These four items are encoded as strings of printable ASCII characters, separated by white spaces, i.e. the ASCII space (0x20) or tabulation (0x09) characters. It is recommended to use exactly one ASCII space separator.

3.2.1.1. Coding of the requested verb

The five verbs that can be requested are encoded as four letter upper or lower case ASCII codes (comparisons should be case insensitive) as defined in the following table:

Verb	Code
CreateConnection	CRCX
ModifyConnection	MDCX
DeleteConnection	DLCX
NotificationRequest	RQNT
Notify	NTFY

The transaction identifier is encoded as a string of up to 9 decimal digits. In the command lines, it immediately follows the coding of the verb.

New verbs may be defined in further versions of the protocol. It may be necessary, for experimentation purposes, to use new verbs before they are sanctioned in a published version of this protocol. Experimental verbs should be identified by a four letter code starting with the letter X, such as for example XPER.

3.2.1.2. Coding of the endpoint names

The endpoint names are encoded as e-mail addresses, as defined in RFC 821. **In these addresses, the domain name identifies the system where the**

endpoint is attached, while the left side identifies a specific endpoint on that system.

Examples of such addresses can be:

123456@gw23.whatever.net	Circuit number 123456 in the Gateway 23 of the "Whatever" network
Call-agent@ca.whatever.net	Call Agent for the "whatever" network
Busy-signal@ann12.whatever.net	The "busy signal" virtual endpoint in the announcement server number 12.

The name of notified entities is expressed with the same syntax, with the possible addition of a port number as in:

```
 Call-agent@ca.whatever.net:5234
```

3.2.1.3. Coding of the protocol version

The protocol version is coded as the key word SGCP followed by a white space and the version number. The version number is composed of a major version, coded by a decimal number, a dot, and a minor version number, coded as a decimal number. The version described in this document is version 1.1.

In the initial messages, the version will be coded as:

```
        SGCP 1.1
```

Version 1.1 is a superset of version 1.0. Gateways that implement version 1.1 shall accept commands coded according to version 1.0. They should refrain from using version 1.1 extensions (See the section on compatibility with version 1.0).

3.2.2. Parameter lines

Parameter lines are composed of a parameter name, which in most cases is composed of a single upper case character, followed by a colon, a white space and the parameter value. The parameter that can be present in commands are defined in the following table:

Parameter name	Code	Parameter value
CallId	C	Hexadecimal string, at most 32 characters
ConnectionId	I	Hexadecimal string, at most 32 characters
NotifiedEntity	N	An identifier, in RFC 821 format, composed of an arbitrary string and of the domain name of the requesting entity, possibly completed by a port number, as in: Call-agent@ca.whatever.net:5234
RequestIdentifier	X	Hexadecimal string, at most 32 characters
LocalConnectionOptions description	L	See
Connection Mode	M	See description
RequestedEvents	R	See description
SignalRequests	S	See description
DigitMap	D	A text encoding of a digit map
ObservedEvents	O	See description
ConnectionParameters	P	See description
ReasonCode	E	An arbitrary character string
SpecificEndpointID	Z	An identifier, in RFC 821 format, composed of an arbitrary string, followed by an "@" followed by the domain name of the gateway to which this endpoint is attached.

The parameters are not necessarily present in all commands. The following table provides the association between parameters and commands. The letter M stands for mandatory, O for optional and F for forbidden.

Parameter name	CRCX	MDCX	DLCX	RQNT	NTFY
CallId	M	M	O	F	F
ConnectionId	F	M	O	F	F
RequestIdentifier	O	O	O	M	M
LocalConnectionOptions	O	O	F	F	F
Connection Mode	M	M	F	F	F
RequestedEvents	O	O	O	O*	F
SignalRequests	O	O	O	O*	F
NotifiedEntity	O	O	O	O	O
ReasonCode	F	F	O	F	F
ObservedEvents	F	F	F	F	M
DigitMap	O	O	O	O	F
Connection parameters	F	F	O	F	F
Specific Endpoint ID	F	F	F	F	F

Note (*) that the RequestedEvents and SignalRequests parameters are optional in the NotificationRequest. If these parameters are omitted, the corresponding lists will be considered empty.

If implementers need to experiment with new parameters, for example when developing a new application of SGCP, they should identify these parameters by names that start with the string "X-", such as for example:

```
X-FlowerOfTheDay: Daisy
```

The parameters will only be recognized by applications that have been upgraded to become part of the experiment. A gateway that receives an extension that it cannot understand should refuse to execute the command. It should respond with an error code 511 (Unrecognized extension.)

3.2.2.1. Local connection options

The local connection options describe the operational parameters that the Call Agent suggests to the gateway. These parameters are:

* The packetisation period in milliseconds, encoded as the keyword "p", followed by a colon and a decimal number. If the Call Agent specifies a range of values, the range will be specified as two decimal numbers separated by an hyphen.

* The preferred type of compression algorithm, encoded as the keyword "a", followed by a character string. If the Call Agent specifies a

list of values, these values will be separated by a semicolon.

- The bandwidth in kilobits per second (1000 bits per second), encoded as the keyword "b", followed by a colon and a decimal number. If the Call Agent specifies a range of values, the range will be specified as two decimal numbers separated by an hyphen.

- The echo cancellation parameter, encoded as the keyword "e", followed by a colon and the value "on" or "off".

Each of the parameters is optional. When several parameters are present, the values are separated by a comma.

Examples of connection descriptors are:

```
L: p:10, a:G.711
L: p:10, a:G.711;G.726-32
L: p:10-20, b: 64
L: b:32-64, e:off
```

3.2.2.2. **Connection parameters**

Connection parameters are encoded as a string of type and value pairs, where the type is a two letter identifier of the parameter, and the value a decimal integer. Types are separated from value by an `=' sign. Parameters are encoded from each other by a comma.

The connection parameter types are specified in the following table:

Connection parameter name	Code	Connection parameter value
Packets sent	PS	The number of packets that were sent on the connection.
Octets sent	OS	The number of octets that were sent on the connection.
Packets received	PR	The number of packets that were received on the connection.
Octets received	OR	The number of octets that were received on the connection.
Packets lost	PL	The number of packets that were not received on the connection, as deduced from gaps in the sequence number.
Jitter	JI	The average inter-packet arrival jitter, in milliseconds, expressed as an integer number.
Latency	LA	Average latency, in milliseconds, expressed as an integer number.

An example of connection parameter encoding is:

```
P: PS=1245, OS=62345, PR=0, OR=0, PL=0, JI=0, LA=48
```

3.2.2.3. Connection mode

The connection mode describes the mode of operation of the connection. The possible values are:

Mode	Meaning
M: sendonly	The gateway should only send packets
M: recvonly	The gateway should only receive packets
M: sendrecv	The gateway should send and receive packets
M: inactive	The gateway should neither send nor receive packets
M: loopback	The gateway should place the circuit in loopback mode.
M: conttest	The gateway should place the circuit in test mode.
M: data	The gateway should use the circuit for network access for data (e.g., PPP, SLIP, etc.).

3.2.2.4. Coding of event names

Event names are mentioned in the RequestedEvents, SignalRequest and Observed Events parameter. Each event is identified by a code, as indicated in the following table. These ASCII encodings are not case sensitive. Values such as "hu", "Hu", "HU" or "hU" should be considered equal.

The following codes are used to identify events:

Code	Event
A string of digits [0-9], hash and star marks [*,#], the letters A, B, C, D and the timer indication T	DTMF tones
ann	Announcement
asdi	ASDI display
aw	Answer tone
bz	Busy tone
cf	Confirm tone
cbk	Call back request
cg	Network Congestion tone
co	Default continuity tone
co1	Continuity tone (single tone)
co2	Continuity test (go tone, in dual tone procedures)
cl	Carrier lost
cv	Continuity verified (response tone, in dual tone procedures)
dl	Dial tone
ft	Fax tones
hd	Off-hook transition
hf	Flash hook
hu	On-hook transition
it	Intercept tone
ld	Long duration connection
mt	Modem tones
oc	Operation Complete (e.g., end of announcement)
ot	Off hook warning tone
pa	Packet arrival
pt	Preemption tone
r0, r1, r2, r3, r4, r5, r6 or r7 (0 .. 7)	Distinctive ringing
rg	Ringing
rt	Ring back tone
t	Timer
wk	Wink
wt	Call waiting tone

3.2.2.5. RequestedEvents

The RequestedEvent parameter provides the list of events that have been

requested. The event codes are described in the previous section. In addition to this coded events, the list may also include a parameter specifying digit collection. This parameter could take one of the following forms:

- individual digits, pound sign (octothorpe), star sign or letters A, B, C or D.
- timer mark T,
- ranges of digits, enclosed within square brackets, e.g. "[0-9]" or "[0-9*#T]"

Each event can be qualified by a requested action, or by a list of actions. The actions, when specified, are encoded as a list of keywords, enclosed in parenthesis and separated by commas. The codes for the various actions are:

Action	Code
Notify immediately	N
Accumulate	A
Treat according to digit map	D
Swap	S
Ignore	I

When no action is specified, the default action is to notify the event. This means that, for example, ft and ft(N) are equivalent. Events that are not listed are ignored.

The digit-map action can only be specified for the digits, letters and timers.

The requested list is encoded on a single line, with event/action groups separated by commas. Examples of RequestedEvents encoding are:

```
        R: hu(N), hf(S,N)
        R: hu(N), [0-9#T](D)
```

3.2.2.6. SignalRequests

The SignalRequests parameter provides the name of the signals that have been requested. Each signal is identified by a code, as indicated in the

previous section.

Two events, announcement and ASDI display, can be qualified by additional parameters:

- the name and parameters of the announcement,

- the string that should be displayed.

These parameters will be enclosed within parenthesis, as in:

```
        S: asdi(123456 Your friend)
        S: ann(no-such-number, 1234567)
```

When several signals are requested, their codes are separated by a comma, as in:

```
        S: asdi(123456 Your friend), rg
```

3.2.2.7. ObservedEvent

The observed event parameters provides the list of events that have been observed. The event codes are the same as those used in the NotificationRequest. Events that have been accumulated according to the digit map are grouped in a single string. Examples of observed actions are:

```
        O: hu
        O: 8295555T
        O: hf, hf, hu
```

The packet arrival event is used to notify that at least one packet was recently sent to an Internet address that is observed by an endpoint. The event report includes the Internet address, in standard ASCII encoding, between parenthesis:

```
        O: pa(192.96.41.1)
```

The call back event is used to notify that a call back has been requested during the initial phase of a data connection. The event report includes the identification of the user that should be called back, between parenthesis:

```
        O: cbk(user25)
```

3.3. Format of response headers

The response header is composed of a response line, optionally followed by headers that encode the response parameters.

The response line starts with the response code, which is a three digit numeric value. The code is followed by a white space, the transaction identifier, and an optional commentary.

In the case of a CreateConnection message, the response line is followed by a Connection-Id parameter. It may also be followed a Specific-Endpoint-Id parameter, if the creation request was sent to a generic Endpoint-Id.

In the case of a DeleteConnection message, the response line is followed by a Connection Parameters parameter, as defined in section 3.2.2.2.

A LocalConnectionDescriptor should be transmitted with a positive response (code 200) to a CreateConnection. It may be transmitted in response to a ModifyConnection command, if the modification resulted in a modification of the session parameters. The LocalConnectionDescriptor is encoded as a "session description," as defined in section 3.4. It is separated from the response header by an empty line.

3.4. Encoding of the session description

The session description is encoded in conformance with the session description protocol, SDP. SGCP implementations are expected to be fully capable of parsing any conformant SDP message, and should send session descriptions that strictly conform to the SDP standard. The usage of SDP actually depends on the type of session that is being, as specified in the "mode" parameter:

- if the mode is set to "data", the session description describes the configuration of a data access service.

- if the mode is set to any other value, the session description is for an audio service.

For an audio service, the gateway will consider the information provided in SDP for the "audio" media. For a data service, the gateway will consider the information provided for the "network-access" media.

3.4.1. Usage of SDP for an audio service

In a telephony gateway, we only have to describe sessions that use exactly one media, audio. The parameters of SDP that are relevant for the telephony application are:

At the session description level:

* The IP address of the remote gateway (in commands) or of the local gateway (in responses), or multicast address of the audio conference, encoded as an SDP "connection data" parameter. This parameter specifies the IP address that will be used to exchange RTP packets.

For the audio media:

* Media description field (m) specifying the audio media, the transport port used for receiving RTP packets by the remote gateway (commands) or by the local gateway (responses) , the RTP/AVP transport, and the list of formats that the gateway will accept. This list should normally always include the code 0 (reserved for G.711).

* Optionally, RTPMAP attributes that define the encoding of dynamic audio formats,

* Optionally, a packetization period (packet time) attribute (Ptime) defining the duration of the packet,

* Optionally, an attribute defining the type of connection (sendonly, recvonly, sendrecv, inactive)

* The IP address of the remote gateway (in commands) or of the local gateway (in responses), if it is not present at the session level.

There is a request, in some environments, to use the SGCP to negotiate connections that will use other transmission channels than RTP over UDP and IP. This will be detailed in an extension to this document.

3.4.2. Usage of SDP in a network access service

The parameters of SDP that are relevant for a data network access application are:

For the data media:

* Media description field (m) specifying the network access media, identified by the code "m=nas/xxxx", where "xxxx" describes the access control method that should be used for

parametrizing the network access, as specified below. The field may also specify the port that should be used for contacting the server, as specified in the SDP syntax.

- Connection address parameter (c=) specifying the address, or the domain name, of the server that implement the access control method. This parameter may also be specified at the session level.

- Optionally, a bearer type attribute (a=bearer:) describing the type of data connection to be used.

- Optionally, a framing type attribue (a=framing:) describing the type of framing that will be used on the channel.

- Optionally, attributes describing the called number (a=dialed:), the number to which the call was delivered (a=called:) and the calling number (a=dialing:).

- Optionally, attributes describing the range of addresses that could be used by the dialup client on its LAN (a=subnet:).

- Optionally, an encryption key, encoded as specified in the SDP protocol (k=).

The connection address shall be encoded as specified in the SDP standard. It will be used in conjunction with the port specified in the media line to access a server, whose type will one of:

Method name	Method description
radius	Authentication according to the Radius protocol.
tacacs	Authentication according to the TACACS+ protocol.
diameter	Authentication according to the Diameter protocol.
l2tp	Level 2 tunneling protocol. The address and port are those of the LNS.
login	Local login. (There is normally no server for that method.)
none	No authentication required. (The call was probably vetted by the Call Agent.)

If needed, the gateway may use the key specified in the announcement to access the service. That key, in particular, may be used for the establishment of an L2TP tunnel.

The valid values of the bearer attribute are defined in the following table:

Type of bearer description	Example of values
ITU modem standard	V.32, V.34, V.90.
ISDN transparent access, 64 kbps	ISDN64
ISDN64 + V.110	ISDN64/V.110
ISDN64 + V.120	ISDN64/V.120
ISDN transparent access, 56 kbps	ISDN56
Informal identification	(Requires coordination between the Call Agent and the gateway)

The valid values of the framing attribute are defined in the following table:

Type of framing description	Example of values
PPP, asynchronous framing	ppp-asynch
PPP, HDLC framing	ppp-hdlc
SLIP, asynchronous	slip
Asynchronous, no framing	asynch

The network access authentication parameter provides instructions on the access control that should be exercized for the data call. This optional attribute is encoded as:

```
        "a=subnet:" <network type> <address type>
           <connection address> "/" <prefix length>
```

Where the parameters "network type", "address type", and "connection address" are formatted as defined for the connection address parameter (c=) in SDP, and where the "prefix length" is a decimal representation of the number of bits in the prefix.

Examples of SDP announcement for the network access service could be:

```
        v=0
        m=nas/radius
        c=IN IP4 radius.example.net
        a=bearer:v.34
        a=framing:ppp-asynch
        a=dialed:18001234567
        a=called:12345678901
        a=dialing:12340567890

        v=0
        m=nas/none
        c=IN IP4 128.96.41.1
        a=subnet:IN IP4 123.45.67.64/26
        a=bearer:isdn64
        a=framing:ppp-sync
        a=dialed:18001234567
        a=dialing:2345678901

        v=0
        c=IN IP4 access.example.net
        m=nas/l2tp
        k=clear:some-shared-secret
        a=bearer:v.32
        a=framing:ppp-asynch
        a=dialed:18001234567
        a=dialing:2345678901
```

3.5. Transmission over UDP

SGCP messages are transmitted over UDP. Commands are sent to one of the IP addresses defined in the DNS for the specified endpoint . The responses are sent back to the source address of the commands.

When no port is specified for the endpoint, the commands should be sent to the default SGCP port, 2427.

SGCP messages, being carried over UDP, may be subject to losses. In the absence of a timely response, commands are repeated. SGCP entities are expected to keep in memory a list of the responses that they sent to recent transactions, i.e. a list of all the responses they sent over the last 30 seconds, and a list of the transactions that are currently being executed. The transaction identifiers of incoming commands are compared to the transaction identifiers of the recent responses. If a match is found, the SGCP entity does not execute the transaction, but simply repeats the response. The remaining commands will be compared to the

list of current transaction. If a match is found, the SGCP entity does not execute the transaction, which is simply ignored.

It is the responsibility of the requesting entity to provide suitable time outs for all outstanding commands, and to retry commands when time outs have been exceeded. Furthermore, when repeated commands fail to be acknowledged, it is the responsibility of the

requesting entity to seek redundant services and/or clear existing or pending connections.

The specification purposely avoids specifying any value for the retransmission timers. These values are typically network dependent. The retransmission timers should normally estimate the timer by measuring the time spent between the sending of a command and the return of a response. One possibility is to use the algorithm implemented in TCP-IP, which uses two variables:

- the average acknowledgement delay, AAD, estimated through an exponentially smoothed average of the observed delays,
- the average deviation, ADEV, estimated through an exponentially smoothed average of the absolute value of the difference between the observed delay and the current average

The retransmission timer, in TCP, is set to the sum of the average delay plus N times the average deviation.

After the any retransmission, the SGCP entity should do the following:

- It should double the estimated value of the average delay, AAD
- It should compute a random value, uniformly distributed between 0.5 AAD and AAD
- It should set the retransmission timer to the sum of that random value and N times the average deviation.

This procedure has two effects. Because it includes an exponentially increasing component, it will automatically slow down the stream of messages in case of congestion. Because it includes a random component, it will break the potential synchronization between notifications triggered by the same external event.

4. Security requirements

If unauthorized entities could use the SGCP, they would be able to set-up unauthorized calls, or to interfere with authorized calls. We expect that SGCP messages will always be carried over secure Internet connections, as defined in the IP security architecture as defined in RFC 1825, using either the IP Authentication Header, defined in RFC 1826, or the IP Encapsulating Security Payload, defined in RFC 1827. The complete SGCP protocol stack would thus include the following layers:

```
                    ________________________________
                   |             SGCP               |
                   |________________________________|
                   |             UDP                |
                   |________________________________|
                   |          IP security           |
                   | (authentication or encryption)|
                   |________________________________|
                   |             IP                 |
                   |________________________________|
                   |      transmission media        |
                   |________________________________|
```

Adequate protection of the connections will be achieved if the gateways and the Call Agents only accept messages for which IP security provided an authentication service. An encryption service will provide additional protection against eavesdropping, thus forbidding third parties from monitoring the connections set up by a given endpoint

The encryption service will also be requested if the session descriptions are used to carry session keys, as defined in SDP.

These procedure do not necessarily protect against denial of service attacks by misbehaving gateways or misbehaving call agents. However, they will provide an identification of these misbehaving entities, which should then be deprived of their authorization through maintenance procedures.

5. Example of call flows

In order to understand the way the SGCP interface will be used, we have described here two possible call flows between a TGW, which is a trunking gateway that implements SGCP, and an RGW, which is a residential gateway that implements SGCP, as well as four call flows describing how SGCP could be used to control a network access service.

The diagrams also show a Common Database (CDB) that can be queried for

authorization and routing information, and an Accounting Gateway (ACC) that collects accounting information at the start and the end of calls.

These diagrams are solely meant to exhibit the behavior of the SGCP, and to help understanding this protocol. They are not meant as a tutorial on the implementation of a Call Agent. They may very well include miscellaneous errors and imprecisions.

5.1. Basic call, RGW to TGW

```
 _____________________________________________________________________
|  Usr  |    RGW   |       CA     |  CDB|  ACC|    TGW   |  SS7/|  CO |
|       |          |              |     |     |          |  ISUP|     |
|_______|__________|______________|_____|_____|__________|______|_____|
|       |     <-   |  Notification|     |     |          |      |     |
|       |          |    Request   |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |      |     |
|  Off  |   Notify |       ->     |     |     |          |      |     |
| -hook |     <-   |      Ack     |     |     |          |      |     |
| (Dial |     <-   |  Notification|     |     |          |      |     |
| -tone)|          |    Request   |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |      |     |
| Digit |   Notify |       ->     |     |     |          |      |     |
|       |     <-   |      Ack     |     |     |          |      |     |
| (pro- |     <-   |  Notification|     |     |          |      |     |
| gress)|          |    Request   |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |      |     |
|       |     <-   |     Create   |     |     |          |      |     |
|       |          |   Connection |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |      |     |
|       |          |     Query    |     |     |          |      |     |
|       |          |  (E.164 S,D) |  -> |     |          |      |     |
|       |          |       <-     |  IP |     |          |      |     |
|       |          |     Create   |     |     |          |      |     |
|       |          |   Connection |  - -|  - -|     ->   |      |     |
|       |          |              |     |     |  (cut in)|      |     |
|       |          |       <-     |  - -|  - -|    ack   |      |     |
|       |          |      IAM     |  - -|  - -|    - -   |   -> |     |
|       |     <-   |    Modify    |     |     |          |  IAM |  -> |
|       |          |   Connection |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |   <- |  ACM|
|       |          |       <-     |  - -|  - -|    - -   |  ACM |     |
|       |     <-   |  Notification|     |     |          |      |     |
|       |          |    Request   |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |      |     |
|       |          |              |     |     |          |   <- |  ANM|
|       |          |       <-     |  - -|  - -|    - -   |  ANM |     |
|       |     <-   |  Notification|     |     |          |      |     |
|       |          |    Request   |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |      |     |
|       |     <-   |     Modify   |     |     |          |      |     |
|       |          |   Connection |     |     |          |      |     |
|       |    Ack   |       ->     |     |     |          |      |     |
|       |  (cut in)|   Call start |  - -|  -> |          |      |     |
|_______|__________|______________|_____|_____|__________|______|_____|
```

```
   __________________________________________________________________
  |   Usr  |   RGW  |       CA     |  CDB|  ACC|   TGW |  SS7/|  CO |
  |        |        |              |     |     |       |  ISUP|     |
  |________|________|______________|_____|_____|_______|______|_____|
  |        |        |              |     |     |       |   <- |  REL|
  |        |        |       <-     |  - -|  - -|   - - |  REL |     |
  |        |    <-  |     Delete   |     |     |       |      |     |
  |        |        |   Connection |     |     |       |      |     |
  |        |        |     Delete   |     |     |       |      |     |
  |        |        |   Connection |  - -|  - -|   ->  |      |     |
  |        |   Perf |              |     |     |       |      |     |
  |  Data  |    ->  |              |     |     |       |      |     |
  |        |        |       <-     |  - -|  - -|  perf |      |     |
  |        |        |              |     |     |  data |      |     |
  |        |        |    Call end  |  - -|  -> |       |      |     |
  | On-hook|  Notify|       ->     |     |     |       |      |     |
  |        |    <-  |      Ack     |     |     |       |      |     |
  |        |    <-  |  Notification|     |     |       |      |     |
  |        |        |    Request   |     |     |       |      |     |
  |        |   Ack  |       ->     |     |     |       |      |     |
  |________|________|______________|_____|_____|_______|______|_____|
```

During these exchanges the SGCP is used by the Call Agent to control both the TGW and the residential gateway. The exchanges occur on two sides.

The first command is a NotificationRequest, sent by the Call Agent to the residential gateway. The request will consist of the following lines:

```
        RQNT 1201 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        N: ca@ca1.whatever.net:5678
        X: 0123456789AB
        R: hd
```

The gateway, at that point, is instructed to look for an off-hook event, and to report it. It will first acknowledge the command, repeating in the acknowledgement message the transaction id that the Call Agent attached to the query.

```
        200 1201 OK
```

When the off hook event is noticed, the gateway initiates a

NotificationRequest to the Call Agent:

```
        NTFY 2001 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        N: ca@ca1.whatever.net:5678
        X: 0123456789AB
        O: hd
```

The Call Agent immediately acknowledges that notification.

```
        200 2001 OK
```

The Call Agent examines the services associated to an off hook action (it could take special actions in the case of a direct line). In most cases, it will send a NotificationRequest, asking for more digits. The current example provides the gateway with a permanent digit map, and request the gateway to play a dialtone:

```
        RQNT 1202 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        N: ca@ca1.whatever.net:5678
        X: 0123456789AC
        R: hu, [0-9#*T](D)
        D: (0T|00T|[1-7]xxx|8xxxxxxx|#xxxxxxx|*xx|91xxxxxxxxxx|9011x.T)
        S:
        dt
```

The gateway immediately acknowledges that command.

```
        200 1202 OK
```

The gateway will start accumulating digits according to that digit map. When it has noticed a sufficient set of values, it will notify the observed string to the Call Agent:

```
        NTFY 2002 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        N: ca@ca1.whatever.net:5678
        X: 0123456789AC
        O: 912018294266
```

The Call Agent immediately acknowledges that notification.

```
        200 2002 OK
```

At this stage, the Call Agent will send a NotificationRequest, to stop collecting digits yet continue watch for an on-hook transition:

```
        RQNT 1203 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789AD
        R: hu
```

The Call Agent immediately acknowledges that command.

```
        200 1203 OK
```

The Call Agent will then seize the incoming circuit, creating a connection:

```
        CRCX 1204 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        L: p:10, a:G.711;G.726-32
        M: recvonly
```

The gateway immediately acknowledges the creation, sending back the identification of the newly created connection and the session description used to receive audio data:

```
        200 1204 OK
        I:FDE234C8

        v=0
        c=IN IP4 128.96.41.1
        m=audio 3456 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The SDP announcement, in our example, specifies the address at which the gateway is ready to receive audio data (128.96.41.1), the transport protocol (RTP), the RTP port (3456) and the audio profile (AVP). The audio profile refers to RFC 1890, which defines that the payload type 0 has been assigned for G.711 transmission. The gateway is also ready to use ADPCM encoding at 32 kbps (G.726 4). There is no standard payload type associated to ADPCM, so the gateway mentions its readiness to use a non

standard payload associated to the dynamic type 96. The "rtpmap" attribute entry associates the payload type 96 to G726-32/4.

The Call Agent, having seized the incoming trunk and completed a routing look up to identify the outgoing gateway, must now seize the outgoing trunk. It does so by sending a connection command to the e-gress gateway:

```
        CRCX 1205 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        L: p:10, a:G.711;G.726-32
        M: sendrecv

        v=0
        c=IN IP4 128.96.41.1
        m=audio 3456 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The CreateConnection command has the same parameters as the command sent to the ingress gateway, with two differences:

*   The EndpointId points towards the outgoing trunk,

*   The message carries the session description returned by the ingress gateway,

*   Because the session description is present, the "mode" parameter is set to "send/receive".

We observe that the call identifier is identical for the two connections. This is normal: the two connections belong to the same call, which has a global identifier in our system.

The trunking gateway will acknowledge the connection command, sending in the session description its own parameters such as address, ports and RTP profile:

```
        200 1205 OK
        I:32F345E2

        v=0
        c=IN IP4 128.96.63.25
        m=audio 1297 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The Call Agent will relay the information to the ingress gateway, using a ModifyConnection command:

```
        MDCX 1206 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:FDE234C8
        M: recvonly

        v=0
        c=IN IP4 128.96.63.25
        m=audio 1297 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The residential gateway immediately acknowledges the modification:

```
        200 1206 OK
```

At this stage, the Call Agent has established a half duplex transmission path. The phone attached to the residential gateway will be able to receive the signals, such as tones or announcements, that the remote switch may send through the trunking gateway.

When the call progresses, the Call Agent will receive from the remote switch progress messages, for example an "address complete" message (ACM). The Call Agent will analyze the message to determine whether signal are transmitted in band. If this is not the case, the Call Agent will instruct the RGW to generate ringing tones by sending a NotificationRequest:

```
        RQNT 1207 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789AE
        R: hu
        S: rt
```

The gateway immediately acknowledges the command:

```
        200 1207 OK
```

After the called user answers the call, the Call Agent will receive an answering message (ANM) from the CO switch. At that point, it will send a NotificationRequest to the residential gateway, to stop the ringing tones, and a ModifyConnection command, to place the connection in full

duplex mode:

```
        RQNT 1208 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789AF
        R: hu

        MDCX 1209 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:FDE234C8
        M: sendrecv
```

The residential gateway will acknowledge these two commands:

```
        200 1208 OK

        200 1209 OK
```

At this point, the connection is established.

When the Call Agent receives the REL message from the CO switch, it will have to tear down the call. It will do so by sending to both gateways a DeleteConnection command:

```
        DLCX
        1210 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:FDE234C8

        DLCX 1211 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:32F345E2
```

The gateways will respond with acknowledgements that should include a "call parameters" header fields:

```
        250 1210 OK
        P: PS=1245, OS=62345, PR=780, OR=45123, PL=10, JI=27,
        LA=48

        250 1211 OK
        P: PS=790, OS=45700, PR=1230, OR=61875, PL=15, JI=27,
        LA=48
```

At this point, phone attached to the residential gateway, in our scenario, goes on-hook. This event is notified to the Call Agent, according to the policy received in the last NotificationRequest by sending a Notify command:

```
        NTFY 2005 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789AF
        O: hu
```

After this notification, the Call Agent should send an acknowledgement:

```
        200 2005 OK
```

It should then issue a new NotificationRequest, to be ready to receive the next off-hook detected by the residential gateway:

```
        RQNT 1212 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789B0
        R: hd
```

The gateway will acknowledge this message:

```
        200 1212 OK
```

Both gateways, at this point, are ready for the next

call.

5.2. Basic call, TGW to RGW

```
  ___________________________________________________________________
 | CO |  SS7/|    TGW   |       CA      |  CDB|  ACC|   RGW  |  Usr |
 |    |  ISUP|          |               |     |     |        |      |
 |____|______|__________|_______________|_____|_____|________|______|
 | IAM|   -> |          |               |     |     |        |      |
 |    |  IAM |    - -   |       ->      |     |     |        |      |
 |    |      |          |      Check    |  -> |     |        |      |
 |    |      |          |       <-      |  IP |     |        |      |
 |    |      |     <-   |     Create    |     |     |        |      |
 |    |      |          |   Connection  |     |     |        |      |
 |    |      |    Ack   |       ->      |     |     |        |      |
 |    |      |          |     Create    |     |     |        |      |
 |    |      |          |   Connection  |  - -|  - -|    ->  |      |
 |    |      |          |       <-      |  - -|  - -|   Ack  |      |
 |    |      |     <-   |     Modify    |     |     |        |      |
 |    |      |          |   Connection  |     |     |        |      |
 |    |      |    Ack   |       ->      |     |     |        |      |
 |    |      |          |  Notification |     |     |        |      |
 |    |      |          |     Request   |  - -|  - -|    ->  |  ring|
 |    |      |          |       <-      |  - -|  - -|   Ack  |      |
 |    |   <- |    - -   |       ACM     |     |     |        |      |
 | <- |  ACM |          |               |     |     |        |      |
 |    |      |          |               |     |     |        |  off |
 |    |      |          |               |     |     |        |  hook|
 |    |      |          |       <-      |  - -|  - -|  Notify|      |
 |    |      |          |       Ack     |  - -|  - -|    ->  |      |
 |    |      |          |  Notification |     |     |        |      |
 |    |      |          |     Request   |  - -|  - -|    ->  |      |
 |    |      |          |       <-      |  - -|  - -|   Ack  |      |
 |    |      |     <-   |    Modify     |     |     |        |      |
 |    |      |          |   Connection  |     |     |        |      |
 |    |      |    Ack   |       ->      |     |     |        |      |
 |    |      |  (cut-in)|   Call start  |  - -|  -> |        |      |
 |    |   <- |    - -   |       ANM     |     |     |        |      |
 | <- |  ANM |          |               |     |     |        |      |
 |____|______|__________|_______________|_____|_____|________|______|
```

```
     _____________________________________________________________
    | CO|  SS7/|  TGW |       CA     |  CDB|  ACC|   RGW  |  Usr |
    |   |  ISUP|      |              |     |     |        |      |
    |___|______|______|______________|_____|_____|________|______|
    |   |      |      |              |     |     |        |  on  |
    |   |      |      |              |     |     |        |  hook|
    |   |      |      |       <-     |  - -|  - -|  Notify|      |
    |   |      |      |      Ack     |  - -|  - -|    ->  |      |
    |   |      |      |    Delete    |     |     |        |      |
    |   |      |      |   Connection |  - -|  - -|    ->  |      |
    |   |      |   <- |    Delete    |     |     |        |      |
    |   |      |      |   Connection |     |     |        |      |
    |   |   <- |  - - |      REL     |     |     |        |      |
    | <-|  REL |      |              |     |     |        |      |
    |   |      |  Perf|              |     |     |        |      |
    |   |      |  data|       ->     |     |     |        |      |
    |   |      |      |       <-     |  - -|  - -|  perf  |      |
    |   |      |      |              |     |     |   data |      |
    |   |      |      |    Call end  |  - -|  -> |        |      |
    |   |      |      |  Notification|     |     |        |      |
    |   |      |      |    Request   |  - -|  - -|    ->  |      |
    |   |      |      |       <-     |  - -|  - -|   Ack  |      |
    |___|______|______|______________|_____|_____|________|______|
```

This diagram shows the various exchange of messages during a call from a telephone user on the circuit-switched PSTN to a residential user connected to a residential gateway. During these exchanges the Call Agent uses SGCP to control both the TGW and the residential gateway. The exchanges occur on two sides.

Upon reception of the IAM message, the Call Agent immediately sends a CreateConnection request to the trunking gateway to connect to the incoming trunk, creating a connection:

```
        CRCX
        1237 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        L: p:10, a:G.711;G.726-32
        M: recvonly
```

The trunking gateway immediately acknowledges the creation, sending back the identification of the newly created connection and the session description used to receive audio data:

```
        200 1237 OK
        I: FDE234C8

        v=0
        c=IN IP4 128.96.41.1
        m=audio 3456 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The SDP announcement, in our example, specifies the address at which the gateway is ready to receive audio data (128.96.41.1), the transport protocol (RTP), the RTP port (3456) and the audio profile (AVP). The audio profile refers to RFC 1890, which defines that the payload type 0 has been assigned for G.711 transmission. The gateway is also ready to use ADPCM encoding at 32 kbps (G.726 4). There is no standard payload type associated to ADPCM, so the gateway mentions its readiness to use a non standard payload associated to the dynamic type 96. The "rtpmap" attribute entry associates the payload type 96 to G726/4.

The Call Agent, having seized the incoming trunk, must now reserve the outgoing circuit. It does so by sending a connection command to the residential gateway:

```
        CRCX 1238 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        L: p:10, a:G.711;G.726-32
        M: sendrecv

        v=0
        c=IN IP4 128.96.41.1
        m=audio 3456 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The CreateConnection command has the same parameters as the command sent to the ingress gateway, with two differences:

*   The EndpointId points towards the outgoing trunk,

*   The message carries the session description returned by the ingress gateway,

*   Because the session description is present, the "mode" parameter is set to "send/receive".

We observe that the call identifier is identical for the two

connections. This is normal: the two connection belong to the same call, which has a global identifier in our system.

The trunking gateway will acknowledge the connection command, sending in the session description its own parameters such as address, ports and RTP profile:

```
        200 1238 OK
        I:32F345E2

        v=0
        c=IN IP4 128.96.63.25
        m=audio 1297 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The Call Agent will relay the information to the ingress gateway, using a ModifyConnection command:

```
        MDCX 1239 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:FDE234C8
        M: recvonly

        v=0
        c=IN IP4 128.96.63.25
        m=audio 1297 RTP/AVP 0 96
        a=rtpmap:96 G726-32/8000
```

The trunking gateway immediately acknowledges the modification:

```
        200 1239 OK
```

At this stage, the Call Agent has established a half-duplex transmission path. The Call Agent must now tells the residential gateway to ring the called line. It will send a NotificationRequest, consisting of the following lines:

```
        RQNT
        1240 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789B1
        R: hd
        S: rg
```

The residential gateway, at that point, is instructed to look for an off-hook event, and to report it. It will first acknowledge the command, repeating in the acknowledgement message the transaction id that the Call Agent attached to the query.

```
        200 1240 OK
```

Upon reception of this message, the Call Agent sends an address complete message (ACM) to the calling switch, which will generate ringing tones for the calling user.

When the gateway notices the off hook event, it sends a Notify command to the Call Agent:

```
        NTFY 2001 endpoint-1@rgw-2567.whatever.net SGCP 1.1X:
        0123456789B0
        O: hd
```

The Call Agent immediately acknowledges that notification.

```
        200 2001 OK
```

The Call Agent now asks the residential gateway to send a Notify command on the occurrence of an on-hook event. It does so by sending a NotificationRequest to the residential gateway:

```
        RQNT 1241 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789B1
        R: hu
```

The gateway acknowledges that command:

200 1241 OK

In parallel, the Call Agent will send a ModifyConnection command to the trunking gateway, to place the connection in full duplex mode:

```
        MDCX 1242 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:FDE234C8
        M: sendrecv
```

The trunking gateway will acknowledge that command:

```
        200 1242 OK
```

The Call Agent can now send an answer message (ANM) to the calling switch.

After some time, the Call Agent will have to tear down the call. In our example, this is triggered by the residential user, who hangs up. The Notify command is sent to the Call Agent:

```
        NTFY 2005 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789B1
        O: hu
```

The Call Agent acknowledges the notification.

```
        200 2005 OK
```

It will then send to both gateways a DeleteConnection command:

```
        DLCX
        1243 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:FDE234C8

        DLCX 1244 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I:32F345E2
```

The gateways will respond with a message that should include a "call parameters" header fields:

```
        250 1243 OK
        P: PS=1245, OS=62345, PR=780, OR=45123, PL=10, JI=27,
        LA=48

        250 1244 OK
        P: PS=790, OS=45700, PR=1230, OR=61875, PL=15, JI=27,
        LA=48
```

The Call Agent should now issue a new NotificationRequest to the

residential gateway to detect the next off-hook event:

```
        RQNT 1245 endpoint-1@rgw-2567.whatever.net SGCP 1.1
        X: 0123456789B2
        R: hd
```

The residential gateway will acknowledge this command:

```
        200 1245 OK
```

Both gateways, at this point, are ready for the next call.

5.3. Data call to a TGW

```
_______________________________________________________________________
|    PC    |  CO |  SS7/|      TGW    |        CA      |  ACC|  Radius|
|          |     |  ISUP|             |                |     |        |
|__________|_____|______|_____________|________________|_____|________|
| dials in |     |      |             |                |     |        |
|          |  IAM|   -> |             |                |     |        |
|          |     |  IAM |      - -    |        ->      |     |        |
|          |     |      |             |  Check called  |     |        |
|          |     |      |             |     number.    |     |        |
|          |     |      |             |     Notices    |     |        |
|          |     |      |             |    data call.  |     |        |
|          |     |      |             |    Call start  |  -> |        |
|          |     |      |      <-     |      Create    |     |        |
|          |     |      |             |    Connection  |     |        |
|          |     |      |             |      (data)    |     |        |
|          |     |      |      Ack    |        ->      |     |        |
|          |     |      |             |   Connection   |     |        |
|          |     |      |             |  is completed. |     |        |
|          |     |      |             |      Call      |     |        |
|          |     |      |             |   established  |  -> |        |
|          |     |   <- |      - -    |       ANM      |     |        |
|          |  <- |  ANM |             |                |     |        |
|  modem   |  - -|  - - |      ->     |                |     |        |
|    <-    |  - -|  - - |   handshake |                |     |        |
|   PPP    |  - -|  - - |      ->     |                |     |        |
|          |     |      |    obtain   |                |     |        |
|          |     |      |   user-id,  |                |     |        |
|          |     |      |   password  |                |     |        |
|          |     |      |     Check   |       - -      |  - -|    ->  |
|          |     |      |      <-     |       - -      |  - -|   Ack  |
|    <-    |  - -|  - - |  Validates  |                |     |        |
|          |     |      |    call,    |                |     |        |
|    <-    |  - -|  - - |   procures  |                |     |        |
|          |     |      |      IP     |                |     |        |
|          |     |      |    address  |                |     |        |
|Connected |     |      |             |                |     |        |
|   to     |     |      |             |                |     |        |
|   the    |     |      |             |                |     |        |
| Internet |     |      |             |                |     |        |
|__________|_____|______|_____________|________________|_____|________|
```

```
     ______________________________________________________________
    |     PC     |  CO |  SS7/|   TGW |      CA    |  ACC|  Radius|
    |            |     |  ISUP|       |            |     |        |
    |____________|_____|______|_______|____________|_____|________|
    |   Closes   |     |      |       |            |     |        |
    | connection.|     |      |       |            |     |        |
    |            |  REL|   -> |       |            |     |        |
    |            |     |  REL |   - - |      ->    |     |        |
    |            |     |      |   <-  |   Delete   |     |        |
    |            |     |      |       |  Connection|     |        |
    |            |     |      |  Perf |            |     |        |
    |            |     |      |  data |      ->    |     |        |
    |            |     |   <- |  - -  |     RLC    |     |        |
    |            |  <- |  RLC |       |            |     |        |
    |            |     |      |       |   Call end |  -> |        |
    |____________|_____|______|_______|____________|_____|________|
```

This diagram shows the exchange of messages during a call from a modem user to an Internet Service Provider, using a trunking gateway that doubles as a Network Access Server. During these exchanges the SGCP is used by the Call Agent to control both the trunking gateway. Since there is no "other end" of the call, only the trunk gateway is involved in the call.

Upon reception of the IAM message, the Call Agent determines that the call is a data call (e.g., by bearer capability, the called number, etc.). Using configuration databases, the Call Agent selects the type of modem parameters and authentication parameters that correspond to the called number and to the calling number. It uses this knowledge to send a CreateConnection command to the TGW, programming the incoming trunk:

```
        CRCX 1237 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        M: data
        X: 0123456789B1
        R: cl

        v=0
        m=nas/radius
        c=IN IP4 radius.example.net
        a=bearer:v.32
        a=framing:ppp-asynch
        a=dialed:18001234567
        a=dialing:2345678901
```

The trunking gateway checks that it has adequate resources for the call.

If the trunking gateway did not have adequate resources, for example if it could not support the requested modem type, it should refuse the creation and send an error response to the Call Agent. If the gateway has sufficient resources, it immediately acknowledges the creation, sending back the identification of the newly created connection. (There is no need to transmit a session description in the case of a data call.)

```
        200 1237 OK
        I: FDE234C8
```

The Call Agent, knowing that this is a data call, can immediately acknowledge the establishment of the connection, sending an ANM message back to the calling switch.

The trunk gateway connects the incoming trunk to a DSP loaded with the specified modem code. Once the call is established, the modem of the calling PC will start a training sequence with the modem associated to the trunk in the trunk gateway. The caller will then proceed to a normal PPP synchronization, which probably implies a PPP login. The authentication parameters, in our example, are checked using Radius. The Radius server that will be used is typically chosen as a function of the called number, which identifies the data service that the calling modem requested. In fact, the number can also be used to identify the specific form of authentication that is requested (but not usually).

In our example, the call is completed when the calling modem hangs up. This triggers an ISUP release message, which is forwarded to the Call Agent. The Call Agent will request the TGW to delete the connection:

```
        DLCX 1244 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I: FDE234C8
```

The gateways will respond with a message that should include a "call parameters" header fields:

```
        250 1244 OK
        P: PS=1245, OS=62345, PR=780, OR=45123
```

We should note that, because this is a data call, the call parameters only include a count of the packets and octets that were sent and received.

5.4. Outgoing data call through a TGW

```
 _____________________________________________________________________
|     PC    |  CO |  SS7/|     TGW    |      CA     |  ACC|   Router |
|           |     |  ISUP|            |             |     |          |
|___________|_____|______|____________|_____________|_____|__________|
|           |     |      |            |             |     |  notices |
|           |     |      |            |             |     |  packet  |
|           |     |      |            |             |     |   to PC  |
|           |     |      |            |      <-     |  - -|    NTFY  |
|           |     |      |            |      Ack    |  - -|    ->    |
|           |     |      |            |  Decides to |     |          |
|           |     |      |            |   place an  |     |          |
|           |     |      |            |   outgoing  |     |          |
|           |     |      |            |     call.   |     |          |
|           |     |      |            |  Call start |  -> |          |
|           |     |      |      <-    |    Create   |     |          |
|           |     |      |            |  Connection |     |          |
|           |     |      |            |    (data)   |     |          |
|           |     |      |     Ack    |      ->     |     |          |
|           |     |   <- |     - -    |      IAM    |     |          |
|  (rings)  |  <- |  IAM |            |             |     |          |
|           |  ACM|   -> |            |             |     |          |
|           |     |  ACM |     - -    |      ->     |     |          |
|  (answer) |  ANM|   -> |            |             |     |          |
|           |     |  ANM |     - -    |      ->     |     |          |
|           |     |      |            |  Connection |     |          |
|           |     |      |            |  complete.  |     |          |
|           |     |      |            |    Call     |     |          |
|           |     |      |            |  established|  -> |          |
|    PPP    |  - -|  - - |      ->    |             |     |          |
|     <-    |  - -|  - - |  Validates |             |     |          |
|           |     |      |    call,   |             |     |          |
|           |     |      |  announces |             |     |          |
|           |     |      |  IP address|      - -    |  - -|    ->    |
| Connected |     |      |            |             |     |          |
|  to the   |     |      |            |             |     |          |
|  Internet |     |      |            |             |     |          |
|           |     |      |            |             |     |          |
|___________|_____|______|____________|_____________|_____|__________|
```

```
 __________________________________________________________________
|     PC     |  CO |  SS7/|     TGW    |      CA    |  ACC|  Router|
|            |     |  ISUP|            |            |     |        |
|____________|_____|______|____________|____________|_____|________|
|   Closes   |     |      |            |            |     |        |
| connection.|     |      |            |            |     |        |
|            |  REL|   -> |            |            |     |        |
|            |     |  REL |    - -     |     ->     |     |        |
|            |     |      |     <-     |   Delete   |     |        |
|            |     |      |            |  Connection|     |        |
|            |     |      |   ceases   |            |     |        |
|            |     |      |  announcing|            |     |        |
|            |     |      |  IP address|     - -    |  - -|   ->   |
|            |     |      |    Perf    |            |     |        |
|    data    |  -> |      |            |            |     |        |
|            |     |   <- |    - -     |     RLC    |     |        |
|            |  <- |  RLC |            |            |     |        |
|            |     |      |            |  Call end  |  -> |        |
|____________|_____|______|____________|____________|_____|________|
```

This diagram shows the exchange of messages during a call from an an Internet Service Provider to a modem, using a trunking gateway that doubles as a Network Access Server. During these exchanges the SGCP is used by the Call Agent to control both the TGW, and will also be used between the Call Agent and a default router of the ISP.

In the example configuration, the calls are set on demand, when data have to actually be sent from the Internet to the dial-up user. When no connection is established, the local routing is configured to send the packets towards a default router which may or may not be the same machine as the TGW. In redundant configurations, there could be many default routers. Each of these default routers has been programmed (through a notification request) to send a notification to the Call Agent when it receives a packet on the default route:

```
        NTFY 2005 default-route@router25.whatever.net SGCP 1.1
        X:
        0123456789AF
        O: pa(192.96.41.1)
```

After this notification, the Call Agent should send an acknowledgement:

```
        200 2005 OK
```

(We should note here that using SGCP for this function is a stretch.

There are other protocols, notably RMON, that already provide an adequate service. These protocols could be used instead of SGCP without affecting the discussion that follows.)

The Call Agent deduces from the notification that a circuit should be established towards the dial-up user, or towards the dial-up router. Using configuration databases, the Call Agent selects the number that should be called, and also the type of modem parameters and authentication parameters that correspond to the called number. The Call Agent uses its routing table to select an adequate TGW, with an available outgoing trunk. It uses a create connection command to seize this outgoing trunk:

```
        CRCX 1237 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        M: data
        X: 0123456789B1
        R: cl

        v=0
        m=nas/none
        c=IN IP4 128.96.41.1
        a=subnet:IN IP4 123.45.67.64/26
        a=bearer:isdn64
        a=framing:ppp-hdlc
        a=dialed:18001234567
        a=dialing:2345678901
```

The gateway immediately acknowledges the creation, sending back the identification of the newly created connection. (There is no session description in the case of a data call.)

```
        200 1237 OK
        I: FDE234C8
```

Once the trunk has been seized, the Call Agent will send an IAM message to the switch that controls the trunk. The dialed PC will "ring" and eventually take the call, triggering the arrival of progress messages and then an answer message (ANM). At that point, the Call Agent knows that the call is established.

The DSP associated to the incoming trunk has been loaded with the specified modem code a simple HDLC framing in our example. Once the call is established, the calling PC will train with the modem associated with the trunk. In our example, no authentication is requested: the Call Agent has identified the dialed user through its called number.

Once the association is established and the IP service is validated, the gateway announces that it serves the local user. In our example, there is no address configuration performed through PPP: the dialed user has a permanent address, which has been programmed when it subscribed to the service. However, one the circuit is validated, the gateway should start announcing its access to this permanent address in the routing tables. In our example, the dialed station is in fact an access point to a local network, and the TGW should start announcing accessibility of that local network (123.45.67.64/26) through the local routing procedures (an IGP such as RIP, OSPF or EIGRP).

Note that the current design makes the hypothesis that the Call Agent "tells" the address of the LAN to the TGW. This is a very debatable design. If a secure IGP is used (e.g. using embedded keyed MD5 authentication, or using IPSEC) then the routing prefix will be naturally exchanged through this IGP. On the other hand, some form of configuration can provide a "double check" against user errors.

In our example, the call is completed when the called modem hangs up. This triggers an ISUP release message, which is forwarded to the Call Agent. The Call Agent will request the TGW to delete the connection:

```
        DLCX 1244 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I: FDE234C8
```

The gateways will respond with a message that should include a "call parameters" header fields:

```
        250 1244 OK
        P: PS=1245, OS=62345, PR=780, OR=45123
```

We should note that, because this is a data call, the call parameters only include a count of the packets and octets that were sent and received.

5.5. Call back, using a TGW

There are three classic forms of call-back:

1-   ANI-based Callback

2-   PPP Callback (Microsoft Callback is a variant of this)

3-   Login-based callback

The ANI based call-back can be implemented entirely in the Call Agent, as indicated in the following diagram:

```
     ______________________________________________________________
    |   PC   |  CO |  SS7/|  TGW|             CA            |  ACC|
    |        |     |  ISUP|     |                           |     |
    |________|_____|______|_____|___________________________|_____|
    |  dials |  IAM|   -> |     |                           |     |
    |        |     |  IAM |  - -|             ->            |     |
    |        |     |      |     |      Notices that the     |     |
    |        |     |      |     |  called number corresponds|     |
    |        |     |      |     |   to a call back service, |     |
    |        |     |      |     |    and that the calling   |     |
    |        |     |      |     |   number has subscribed   |     |
    |        |     |      |     |      to that service.     |     |
    |        |     |      |     |       Terminates the      |     |
    |        |     |      |     |       incoming call.      |     |
    |        |     |   <- |  - -|             REL           |     |
    |        |  <- |  REL |     |                           |     |
    |        |  RLC|   -> |     |                           |     |
    | hangup |     |  RLC |  - -|             ->            |     |
    |        |     |      |     |      Decides to place     |     |
    |        |     |      |     |      an outgoing call.    |     |
    |        |     |      |     |         Call start        |  -> |
    |        |     |      |  <- |           Create          |     |
    |        |     |      |     |      Connection (data)    |     |
    |        |     |      |  Ack|             ->            |     |
    |        |     |   <- |  - -|             ANM           |     |
    | (rings)|  <- |  ANM |     |                           |     |
    |________|_____|______|_____|___________________________|_____|
```

The PPP callback suppose that the modem first establishes an incoming connection, and go through the authentication exchange. The following diagram provides an example of these exchanges:

```
 ___________________________________________________________________
|    PC   |  CO |  SS7/|     TGW    |        CA      |  ACC|  Radius|
|         |     |  ISUP|            |                |     |        |
|_________|_____|______|____________|________________|_____|________|
| dials in|     |      |            |                |     |        |
|         |  IAM|   -> |            |                |     |        |
|         |     |  IAM |     - -    |        ->      |     |        |
|         |     |      |            |  Checks called |     |        |
|         |     |      |            |     number.    |     |        |
|         |     |      |            |     Notices    |     |        |
|         |     |      |            |    data call.  |     |        |
|         |     |      |            |    Call start  |  -> |        |
|         |     |      |      <-    |      Create    |     |        |
|         |     |      |            |    Connection  |     |        |
|         |     |      |            |      (data)    |     |        |
|         |     |      |     Ack    |        ->      |     |        |
|         |     |      |            |    Connection  |     |        |
|         |     |      |            |    completed.  |     |        |
|         |     |      |            |      Call      |     |        |
|         |     |      |            |   established  |  -> |        |
|         |     |   <- |     - -    |       ANM      |     |        |
|         |  <- |  ANM |            |                |     |        |
|  modem  |  - -|  - - |      ->    |                |     |        |
|    <-   |  - -|  - - |  handshake |                |     |        |
|   PPP   |  - -|  - - |      ->    |                |     |        |
|         |     |      |   obtain   |                |     |        |
|         |     |      |  user-id,  |                |     |        |
|         |     |      |   password |                |     |        |
|         |     |      |    Check   |       - -      |  - -|    ->  |
|         |     |      |      <-    |       - -      |  - -|   Ack  |
|         |     |      |   reports  |                |     |        |
|         |     |      |  call back |                |     |        |
|         |     |      |  condition |                |     |        |
|         |     |      |     NTFY   |        ->      |     |        |
|         |     |      |      <-    |       ACK      |     |        |
|         |     |      |            |     Decides    |     |        |
|         |     |      |            |   to place an  |     |        |
|         |     |      |            |  outgoing call.|     |        |
|         |     |      |      <-    |      Delete    |     |        |
|         |     |      |            |    Connection  |     |        |
|         |     |      |    Perf    |                |     |        |
|         |     |      |     data   |        ->      |     |        |
|         |     |   <- |     - -    |       REL      |     |        |
|         |  <- |  REL |            |                |     |        |
|         |  REL|   -> |            |                |     |        |
|  hangup |     |  REL |     - -    |        ->      |     |        |
|_________|_____|______|____________|________________|_____|________|
```

```
__________________________________________________________________________
|    PC     |  CO |  SS7/|        TGW       |      CA     |  ACC|  Radius|
|           |     |  ISUP|                  |             |     |        |
|___________|_____|______|__________________|_____________|_____|________|
|           |     |      |                  |  Call start |  -> |        |
|           |     |      |         <-       |    Create   |     |        |
|           |     |      |                  |  Connection |     |        |
|           |     |      |                  |    (data)   |     |        |
|           |     |      |        Ack       |      ->     |     |        |
|           |     |   <- |        - -       |      IAM    |     |        |
|  (rings)  |  <- |  IAM |                  |             |     |        |
|           |  ACM|   -> |                  |             |     |        |
|           |     |  ACM |        - -       |      ->     |     |        |
| (answer)  |  ANM|   -> |                  |             |     |        |
|           |     |  ANM |        - -       |      ->     |     |        |
|           |     |      |                  |  Connection |     |        |
|           |     |      |                  |  complete.  |     |        |
|           |     |      |                  |    Call     |     |        |
|           |     |      |                  |  established|  -> |        |
|    PPP    |  - -|  - - |         ->       |             |     |        |
|    <-     |  - -|  - - |  Validates call, |             |     |        |
|Connected  |     |      |                  |             |     |        |
|  to the   |     |      |                  |             |     |        |
| Internet  |     |      |                  |             |     |        |
|           |     |      |                  |             |     |        |
|  Closes   |     |      |                  |             |     |        |
|connection.|     |      |                  |             |     |        |
|           |  REL|   -> |                  |             |     |        |
|           |     |  REL |        - -       |      ->     |     |        |
|           |     |      |         <-       |    Delete   |     |        |
|           |     |      |                  |  Connection |     |        |
|           |     |      |     Perf data    |      ->     |     |        |
|           |     |   <- |        - -       |      RLC    |     |        |
|           |  <- |  RLC |                  |             |     |        |
|           |     |      |                  |   Call end  |  -> |        |
|___________|_____|______|__________________|_____________|_____|________|
```

This diagram shows the exchange of messages during a call from a modem user to an Internet Service Provider, using a trunking gateway that doubles as a Network Access Server. During these exchanges the SGCP is used by the Call Agent to control the TGW.

Upon reception of the IAM message, the Call Agent notices that the called number corresponds to a data service. Using configuration databases, the Call Agent selects the type of modem parameters and authentication parameters that correspond to the called number and to the

calling number. It uses this knowledge to send a connection command to the TGW, programming the incoming trunk:

```
        CRCX 1237 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        M: data
        X: 0123456789B1
        R: cl, cbk

        v=0
        m=nas/radius
        c=radius.example.net
        a=bearer:v.32
        a=framing:ppp-asynch
        a=dialed:18001234567
        a=dialing:2345678901
```

The gateway immediately acknowledges the creation, sending back the identification of the newly created connection. (There is no session description in the case of a data call.)

```
        200 1237 OK
        I: FDE234C8
```

The Call Agent, knowing that this is a data call, can immediately acknowledge the establishment of the connection, sending an ANM message back to the calling switch.

The DSP associated to the incoming trunk has been loaded with the specified modem code. Once the call is established, the modem of the calling PC will be synchronized with the modem associated to the trunk. The caller will then proceed to a normal PPP synchronization, which probably implies a PPP login. The login parameters, in our example, are checked using Radius. The Radius server that will be used is typically chosen as a function of the called number, which identifies the data service that the calling modem requested. In fact, the number can also be used to identify the specific form of authentication that is requested.

In the call back example, the Radius server will indicate that the call cannot be completed as such, and that the user should be called back (for example, using a "Callback Framed" service type in its access-accept response.) The TGW will thus send a Notify message to the Call Agent, indicating that a call-back is requested:

```
        NTFY 2005 card23/21@trgw-7.whatever.net SGCP 1.1
        X: 0123456789B1
```

```
        O: cbk(user-id)
```

After this notification, the Call Agent should send an acknowledgement:

```
        200 2005 OK
```

The Call Agent will check that the call back request can be followed though. In its databases, it will find the regular address associated to the "user-id," and prepare to set up a call to that address. It will first clear the incoming call, sending a DeleteConnection command to the TGW:

In our example, the call is completed when the calling modem hangs up. This triggers an ISUP release message, which is forwarded to the Call Agent. The Call Agent will request the TGW to delete the connection, and reset the list of observed events:

```
        DLCX 1244 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I: FDE234C8
        X: 0123456789B2
        R:
```

The gateways will respond with a message that should include a "call parameters" header fields:

```
        250 1244 OK
        P: PS=2, OS=345, PR=1, OR=123
```

We should note that, because this is a data call, the call parameters only include a count of the packets and octets that were sent and received.

The Call Agent will then proceed to set up an outgoing data call. This call may be routed through the same TGW that received the incoming call, but can also be routed through an entirely different endpoint , for example if the calling user has moved out of its normal region.

5.6. Data call to a TGW, using L2TP

```
 _______________________________________________________________________
|    PC    |  CO |  SS7/|       TGW     |       CA     |  ACC|    LNS  |
|          |     |  ISUP|               |              |     |         |
|__________|_____|______|_______________|______________|_____|_________|
| dials in |     |      |               |              |     |         |
|          |  IAM|   -> |               |              |     |         |
|          |     |  IAM |       - -     |       ->     |     |         |
|          |     |      |               |  Check called|     |         |
|          |     |      |               |    number.   |     |         |
|          |     |      |               |    Notices   |     |         |
|          |     |      |               |   data call. |     |         |
|          |     |      |               |   Call start |  -> |         |
|          |     |      |       <-      |     Create   |     |         |
|          |     |      |               |   Connection |     |         |
|          |     |      |               |     (data)   |     |         |
|          |     |      |       Ack     |       ->     |     |         |
|          |     |      |               |   Connection |     |         |
|          |     |      |               |   complete.  |     |         |
|          |     |      |               |      Call    |     |         |
|          |     |      |               |  established |  -> |         |
|          |     |   <- |       - -     |      ANM     |     |         |
|          |  <- |  ANM |               |              |     |         |
|  modem   |  - -|  - - |        ->     |              |     |         |
|    <-    |  - -|  - - |    handshake  |              |     |         |
|   PPP    |  - -|  - - |        ->     |              |     |         |
|          |     |      |     obtain    |              |     |         |
|          |     |      |    user-id,   |              |     |         |
|          |     |      |    password   |              |     |         |
|          |     |      |    Establish  |              |     |         |
|          |     |      |     Tunnel    |              |     |         |
|          |     |      |     SCC-REQ   |      - -     |  - -|    ->   |
|          |     |      |       <-      |      - -     |  - -|  SCC-REP|
|          |     |      |       <-      |      - -     |  - -|  SCC-CON|
|          |     |      |     IC-REQ    |      - -     |  - -|    ->   |
|          |     |      |       <-      |      - -     |  - -|  IC-REP |
|          |     |      |       <-      |      - -     |  - -|  IC-CON |
|          |     |      |  Spoof PPP/LCP|      - -     |  - -|    ->   |
|    <-    |  - -|  - - |   Relays PPP  |      - -     |  - -|    ->   |
|Connected |     |      |               |              |     |         |
| to the   |     |      |               |              |     |         |
| Internet |     |      |               |              |     |         |
|__________|_____|______|_______________|______________|_____|_________|
```

```
    _______________________________________________________________
   |     PC     |  CO |  SS7/|     TGW   |      CA    |  ACC|  LNS|
   |            |     |  ISUP|           |            |     |     |
   |____________|_____|______|___________|____________|_____|_____|
   |   Closes   |     |      |           |            |     |     |
   | connection.|     |      |           |            |     |     |
   |            |  REL|   -> |           |            |     |     |
   |            |     |  REL |     - -   |      ->    |     |     |
   |            |     |      |     <-    |    Delete  |     |     |
   |            |     |      |           |  Connection|     |     |
   |            |     |      |    Perf   |            |     |     |
   |            |     |      |    data   |      ->    |     |     |
   |            |     |   <- |     - -   |     RLC    |     |     |
   |            |  <- |  RLC |           |            |     |     |
   |            |     |      |     CDN   |     - -    |  - -|  -> |
   |            |     |      |  Stop-CC-N|     - -    |  - -|  -> |
   |            |     |      |           |   Call end |  -> |     |
   |____________|_____|______|___________|____________|_____|_____|
```

This diagram shows the exchange of messages during a call from a modem user to an Internet Service Provider, using a trunking gateway that doubles as a Network Access Server. During these exchanges the SGCP is used by the Call Agent to control the TGW. The PPP information is relayed to a network server (LNS) using L2TP.

Upon reception of the IAM message, the Call Agent notices that the called number corresponds to a data service. Using configuration databases, the Call Agent selects the type of modem parameters and authentication parameters that correspond to the called number and to the calling number. It uses this knowledge to send a connection command to the TGW, programming the incoming trunk:

```
        CRCX 1237 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        M: data
        X: 0123456789B1
        R: cl

        v=0
        c=IN IP4 access.example.net
        m=nas/l2tp
        k=clear:some-shared-secret
        a=bearer:v.32
        a=framing:ppp-asynch
        a=dialed:18001234567
        a=dialing:2345678901
```

The gateway immediately acknowledges the creation, sending back the identification of the newly created connection. (There is no need to transmit a session description in the case of a data call.)

```
        200 1237 OK
        I: FDE234C8
```

The Call Agent, knowing that this is a data call, can immediately acknowledge the establishment of the connection, sending an ANM message back to the calling switch.

The DSP associated to the incoming trunk has been loaded with the specified modem code. Once the call is established, the modem of the calling PC will be synchronized with the modem associated to the trunk. The caller will then proceed to a normal PPP synchronization, which probably implies a PPP login.

Once PPP has been properly synchronized, the TGW establishes a tunnel towards the LNS. Because L2TP is a two-layer protocol, the TGW must first establish an L2TP control connection between itself and the LNS. This connection may or may not have been established prior to the call set-up.

Tunnel establishment requires a shared secret between the LNS and the TGW; in our example, that secret is passed by the Call Agent, along with the name of the LNS. Once the supporting tunnel is installed, the TGW has to establish an L2TP tunnel, to relay the "incoming call." Once the call is established, the PPP packets received on the trunk will be relayed over the L2TP tunnel, and vice-versa.

In our example, the call is completed when the calling modem hangs up. This triggers an ISUP release message, which is forwarded to the Call Agent. The Call Agent will request the TGW to delete the connection:

```
        DLCX 1244 card23/21@trgw-7.whatever.net SGCP 1.1
        C: A3C47F21456789F0
        I: FDE234C8
```

The gateways will respond with a message that should include a "call parameters" header fields:

```
        250 1244 OK
        P: PS=1245, OS=62345, PR=780, OR=45123
```

We should note that, because this is a data call, the call parameters only include a count of the packets and octets that were sent and received.

6. Versions and compatibility

Version 1.1 introduces the following changes from version 1.0:

* Extension parameters (X-??:)

* Error Code 511 (Unrecognized extension).

* All event codes can be used in RequestEvent, SignalRequest and ObservedEvent parameters.

* Error Code 512 (Not equipped to detect requested event).

* Error Code 513 (Not equipped to generate requested signal).

* Error Code 514 (Unrecognized announcement).

* Specific Endpoint-ID can be returned in creation commands.

* Changed the code for the ASDI display from "ad" to "asdi" to avoid conflict with the digits A and D.

* Changed the code for the answer tone from "at" to "aw" to avoid conflict with the digit A and the timer mark T

* Changed the code for the busy tone from "bt" to "bz" to avoid conflict with the digit B and the timer mark T

* Specified that the continuity tone value is "co" (CT was incorrectly used in several instances; CT conflicts with .)

* Changed the code for the dial tone from "dt" to "dl" to avoid conflict with the digit D and the timer mark T

* Added a code point for announcement requests.

* Added a code point for the "wink" event.

* Set the "octet received" code in the "Connection Parameters" to "OR" (was set to RO, but then "OR" was used throughout all examples.)

* Added a "data" mode.

- Added a description of SDP parameters for the network access mode (NAS).

- Added four flow diagrams for the network access mode.

- Incorporated numerous editing suggestions to make the description easier to understand. In particular, cleared the confusion between requests, queries, functions and commands.

- Defined the continuity test mode as specifying a dual-tone transponder, while the loopback mode can be used for a single tone test.

- Added event code "OC", operation completed.

- Added the specification of the "quarantine list", which clarifies the expected handling of events and notifications.

- Added the specification of a "wildcard delete" operation.

7. Acknowledgements

We want to thank here the many reviewers who provided us with advice on the design of SGCP, notably Flemming Andreasen, David Auerbach, Bob Biskner, Barry Hoffner, Dave Oran, John Pickens, Lou Rubin, Chip Sharp, Kurt Steinbrenner and Joe Stone.

8. References

- Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

- Schulzrinne, H., "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 1890, January 1996

- Handley, M, Jacobson, V., "SDP: Session Description Protocol", RFC 2327, April 1998.

- Handley, M., "SAP - Session Announcement Protocol", Work in Progress.

- Handley, M., Schooler, E., and H. Schulzrinne, "Session Initiation Protocol (SIP)", Work in Progress.

- Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

*   ITU-T, Recommendation Q.761, "FUNCTIONAL DESCRIPTION OF THE ISDN USER PART OF SIGNALLING SYSTEM No. 7", (Malaga-Torremolinos, 1984; modified at Helsinki, 1993)

*   ITU-T, Recommendation Q.762, "GENERAL FUNCTION OF MESSAGES AND SIGNALS OF THE ISDN USER PART OF SIGNALLING SYSTEM No. 7", (Malaga-Torremolinos, 1984; modified at Helsinki, 1993)

*   ITU-T, Recommendation H.323, "VISUAL TELEPHONE SYSTEMS AND EQUIPMENT FOR LOCAL AREA NETWORKS WHICH PROVIDE A NON-GUARANTEED QUALITY OF SERVICE."

*   ITU-T, Recommendation H.225, "Call Signaling Protocols and Media Stream Packetization for Packet Based Multimedia Communications Systems."

*   ITU-T, Recommendation H.245, "LINE TRANSMISSION OF NON-TELEPHONE SIGNALS."

*   Atkinson, R., "Security Architecture for the Internet Protocol." RFC 1825, August 1995.

*   Atkinson, R., "IP Authentication Header." RFC 1826, August 1995.

*   Atkinson, R., "IP Encapsulating Security Payload (ESP)." RFC 1827, August 1995.

9. Authors' Addresses

Mauricio Arango
RSL COM Latin America
6300 N.W. 5th Way, Suite 100
Ft. Lauderdale, FL 33309

Phone: (954) 489-3971
Email: marango@rslcom.com

Christian Huitema
Bellcore
MCC 1J236B
445 South Street
Morristown, NJ 07960
U.S.A.

Phone: +1 973-829-4266
EMail: huitema@bellcore.com

Further information is available on the SGCP web site:

http://sgcp.bellcore.com