Network Working Group
Request for Comments: 3910
Category: Standards Track

V. Gurbani, Ed.
A. Brusilovsky
I. Faynberg
Lucent Technologies, Inc.
J. Gato
Vodafone Espana
H. Lu
Bell Labs/Lucent Technologies
M. Unmehopa
Lucent Technologies, Inc.

October 2004

The SPIRITS (Services in PSTN requesting Internet Services) Protocol

Status of this Memo

Copyright Notice

Abstract

This document describes the Services in PSTN (Public Switched Telephone Network) requesting Internet Services (SPIRITS) protocol. The purpose of the SPIRITS protocol is to support services that originate in the cellular or wireline PSTN and necessitate interactions between the PSTN and the Internet. On the PSTN side, the SPIRITS services are most often initiated from the Intelligent Network (IN) entities. Internet Call Waiting and Internet Caller-ID Delivery are examples of SPIRITS services, as are location-based services on the cellular network. The protocol defines the building blocks from which many other services can be built.

Table of Contents

1. Introduction

SPIRITS (Services in the PSTN Requesting Internet Services) is an IETF architecture and an associated protocol that enables call processing elements in the telephone network to make service requests that are then processed on Internet hosted servers. The term Public Switched Telephone Network (PSTN) is used here to include the wireline circuit-switched network, as well as the wireless circuit-switched network.

The earlier IETF work on the PSTN/Internet Interworking (PINT) resulted in the protocol (RFC 2848) in support of the services initiated in the reverse direction - from the Internet to PSTN.

This document has been written in response to the SPIRITS WG chairs call for SPIRITS Protocol proposals. Among other contributions, this document is based on:

- Informational "Pre-SPIRITS implementations" [10]
- Informational "The SPIRITS Architecture" [1]
- Informational "SPIRITS Protocol Requirements" [4]

1.1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [2].

2. Overview of operations

The purpose of the SPIRITS protocol is to enable the execution of services in the Internet based on certain events occurring in the PSTN. The term PSTN is used here to include all manner of switching; i.e. wireline circuit-switched network, as well as the wireless circuit-switched network.

In general terms, an Internet host is interested in getting notifications of certain events occurring in the PSTN. When the event of interest occurs, the PSTN notifies the Internet host. The Internet host can execute appropriate services based on these notifications.

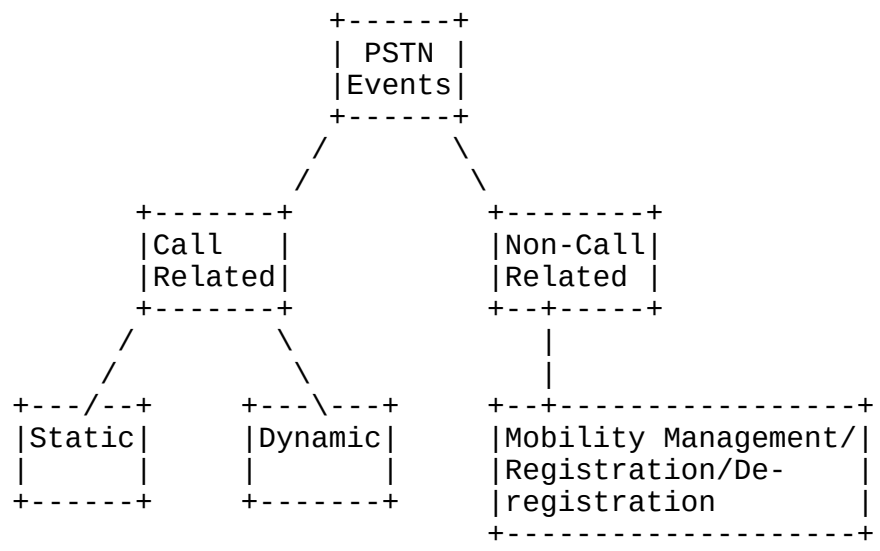

Figure 1: The SPIRITS Hierarchy.

Figure 1 contains the SPIRITS events hierarchy, including their subdivision in two discrete classes for service execution: events related to the setup, teardown and maintenance of a call and events un-related to call setup, teardown or maintenance. Example of the latter class of events are geo-location mobility events that are tracked by the cellular PSTN. SPIRITS will specify the framework to provide services for both of these types of events.

Call-related events, its further subdivisions, and how they enable services in the Internet is contained in Section 5. Services enabled from events not related to call setup, teardown, or maintenance are covered in detail in Section 6.

For reference, the SPIRITS architecture from [1] is reproduced below. This document is focused on interfaces B and C only. Interface D is a matter of local policy; the PSTN operator may have a functional interface between the SPIRITS client or a message passing interface. This document does not discuss interface D in any detail.

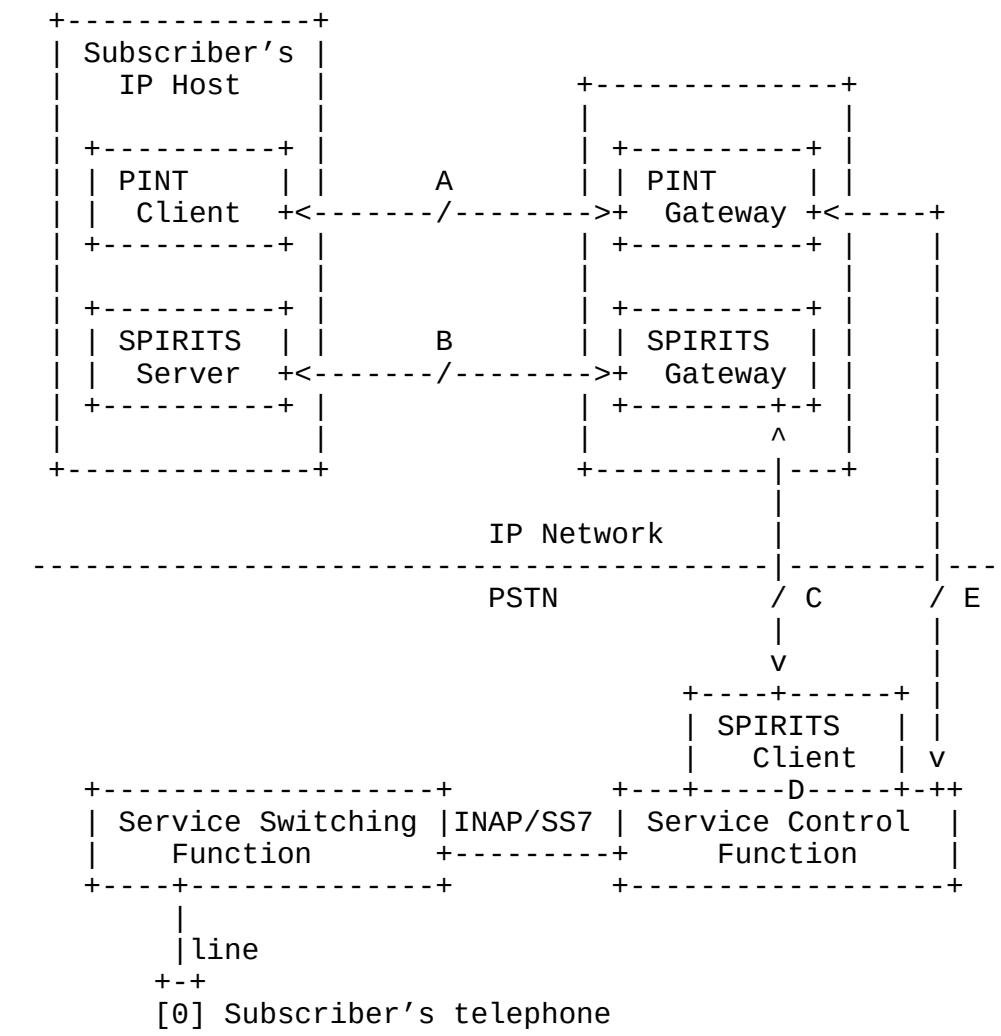

Figure 2: The SPIRITS Architecture.

(Note: The interfaces A-E are described in detail in the SPIRITS Architecture document [1].)

The PSTN today supports service models such as the Intelligent Network (IN), whereby some features are executed locally on switching elements called Service Switching Points (SSPs). Other features are executed on service elements called Service Control Points (SCPs). The SPIRITS architecture [1] permits these SCP elements to act as intelligent entities to leverage and use Internet hosts and capabilities to further enhance the telephone end-user's experience.

The protocol used on interfaces B and C consists of the SPIRITS protocol, and is based on SIP and SIP event notification [3]. The requirements of a SPIRITS protocol and the choice of using SIP as an enabler are detailed in [4].

The SPIRITS protocol is a set of two "event packages" [3]. It contains the procedural rules and semantic context that must be applied to these rules for processing SIP transactions. The SPIRITS protocol has to carry subscriptions for events from the SPIRITS server to the SPIRITS client and notifications of these events from the SPIRITS client to the SPIRITS server. Extensible Markup Language (XML) [12] is used to codify the subscriptions and notifications.

Finally, in the context of ensuing discussion, the terms "SPIRITS server" and "SPIRITS client" are somewhat confusing since the roles appear reversed; to wit, the "SPIRITS server" issues a subscription which is accepted by a "SPIRITS client". To mitigate such ambiguity, from now on, we will refer to the "SPIRITS server" as a "SPIRITS subscriber" and to the "SPIRITS client" as a "SPIRITS notifier". This convention adheres to the nomenclature outlined in [3]; the SPIRITS server in Figure 2 is a subscriber (issues subscriptions to events), and the SPIRITS client in Figure 2 is a notifier (issues notifications whenever the event of interest occurs).

2.1. Terminology

For ease of reference, we provide a terminology of the SPIRITS actors discussed in the preceding above:

Service Control Function (SCF): A PSTN entity that executes service logic. It provides capabilities to influence the call processing occurring in the Service Switching Function (SSF). For more information on how a SCF participates in the SPIRITS architecture, please see Sections 5 and 5.1.

SPIRITS client: see SPIRITS notifier.

SPIRITS server: see SPIRITS subscriber.

SPIRITS notifier: A User Agent (UA) in the PSTN that accepts subscriptions from SPIRITS subscribers. These subscriptions contain events that the SPIRITS subscribers are interested in receiving a notification for. The SPIRITS notifier interfaces with the Service Control Function such that when the said event occurs, a notification will be sent to the relevant SPIRITS subscriber.

SPIRITS subscriber: A UA in the Internet that issues a subscription containing events in the PSTN that it is interested in receiving a notification for.

3. Using XML for subscription and notification

The SPIRITS protocol requirements mandate that "SPIRITS-related parameters be carried in a manner consistent with SIP practices" (RFC3298:Section 3). SIP already provides payload description capabilities through the use of headers (Content-Type, Content-Length). This document defines a new MIME type -- "application/spirits-event+xml" -- and registers it with IANA (Section 7). This MIME type MUST be present in the "Content-Type" header of SPIRITS requests and responses, and it describes an XML document that contains SPIRITS-related information.

This document defines a base XML schema for subscriptions to PSTN events. The list of events that can be subscribed to is defined in the SPIRITS protocol requirements document [4] and this document provides an XML schema for it. All SPIRITS subscribers (any SPIRITS entity capable of issuing a SUBSCRIBE, REGISTER, or INVITE request) MUST support this schema. All SPIRITS notifiers (any SPIRITS entity capable of receiving and processing a SUBSCRIBE, REGISTER, or INVITE request) MUST support this schema. The schema is defined in Section 9.

The support for the SIP REGISTER request is included for PINT compatibility (RFC3298:Section 6).

The support for the SIP INVITE request is mandated because pre-existing SPIRITS implementations did not use the SIP event notification scheme. Instead, the initial PSTN detection point always arrived via the SIP INVITE request.

This document also defines a base XML schema for notifications of events (Section 9). All SPIRITS notifiers MUST generate XML documents that correspond to the base notification schema. All SPIRITS subscribers MUST support XML documents that correspond to this schema.

The set of events that can be subscribed to and the amount of notification that is returned by the PSTN entity may vary among different PSTN operators. Some PSTN operators may have a rich set of events that can be subscribed to, while others have only the primitive set of events outlined in the SPIRITS protocol requirements document [4]. This document defines a base XML schema (in Section 9) which MUST be used for the subscription and notification of the primitive set of events. In order to support a richer set of event

subscription and notification, implementations MAY use additional XML namespaces corresponding to alternate schemas in a SPIRITS XML document. However, all implementations MUST support the base XML schema defined in Section 9 of this document. Use of the base schema ensures interoperability across implementations, and the inclusion of additional XML namespaces allows for customization.

A logical flow of the SPIRITS protocol is depicted below (note: this example shows a temporal flow; XML documents and related SPIRITS protocol syntax is specified in later sections of this document). In the flow below, S is the SPIRITS subscriber and N is the SPIRITS notifier. The SPIRIT Gateway is presumed to have a pure proxying functionality and thus is omitted for simplicity:

```
1  S->N Subscribe (events of interest in an XML document instance
                   using base subscription schema)
2  N->S 200 OK (Subscribe)
3  N->S Notify
4  S->N 200 OK (Notify communicating current resource state)
5  ...
6  N->S Notify (Notify communicating change in resource state;
                payload is an XML document instance using
                XML extensions to the base notification schema)
7  S->N 200 OK (Notify)
```

In line 1, the SPIRITS subscriber subscribes to certain events using an XML document based on the base schema defined in this document. In line 6, the SPIRITS notifier notifies the SPIRITS subscriber of the occurrence of the event using extensions to the base notification schema. Note that this document defines a base schema for event notification as well; the SPIRITS notifier could have availed itself of these. Instead, it chooses to pass to the SPIRITS subscriber an XML document composed of extensions to the base notification schema. The SPIRITS subscriber, if it understands the extensions, can interpret the XML document accordingly. However, in the event that the SPIRITS subscriber is not programmed to understand the extensions, it MUST search the XML document for the mandatory elements. These elements MUST be present in all notification schemas and are detailed in Section 9.

4. XML format definition

This section defines the XML-encoded SPIRITS payload format. Such a payload is a well formed XML document and is produced by SPIRITS notifiers and SPIRITS subscribers.

The namespace URI for elements defined in this document is a Uniform Resource Name (URN) [14], using the namespace identifier 'ietf' defined in [15] and extended by [16]:

urn:ietf:params:xml:ns:spirits-1.0

SPIRITS XML documents may have a default namespace, or they may be associated with a namespace prefix following the convention established in XML namespaces [17]. Regardless, the elements and attributes of SPIRITS XML documents MUST conform to the SPIRITS XML schema specified in Section 9.

The <spirits-event> element

The root of a SPIRITS XML document (characterized by a Content-Type header of "application/spirits-event+xml">) is the <spirits-event> element. This element MUST contain a namespace declaration ('xmlns') to indicate the namespace on which the XML document is based. XML documents compliant to the SPIRITS protocol MUST contain the URN "urn:ietf:params:xml:ns:spirits-1.0" in the namespace declaration. Other namespaces may be specified as needed.

<spirits-event> element MUST contain at least one <Event> element, and MAY contain more than one.

The <Event> element

The <Event> element contains three attributes, two of which are mandatory. The first mandatory attribute is a 'type' attribute whose value is either "INDPs" or "userprof".

These types correspond, respectively, to call-related events described in Section 5 and non-call related events described in Section 6.

The second mandatory attribute is a 'name' attribute. Values for this attribute MUST be limited to the SPIRITS mnemonics defined in Section 5.2.1, Section 5.2.2, and Section 6.1.

The third attribute, which is optional, is a 'mode' attribute. The value of 'mode' is either "N" or "R", corresponding respectively to (N)otification or (R)equest (RFC3298:Section 4). The default value of this attribute is "N".

If the 'type' attribute of the <Event> element is "INDPs", then it MUST contain at least one or more of the following elements (unknown elements MAY be ignored): <CallingPartyNumber>, <CalledPartyNumber>, <DialledDigits>, or <Cause>. These elements

are defined in Section 5.2; they MUST not contain any attributes and MUST not be used further as parent elements. These elements contain a string value as described in Section 5.2.1 and 5.2.2.

If the 'type' attribute of the <Event> element is "userprof", then it MUST contain a <CalledPartyNumber> element and it MAY contain a <Cell-ID> element. None of these elements contain any attributes and neither must be used further as a parent element. These elements contain a string value as described in Section 6.1. All other elements MAY be ignored if not understood.

A SPIRITS-compliant XML document using the XML namespace defined in this document might look like the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<spirits-event xmlns="urn:ietf:params:xml:ns:spirits-1.0">
   <Event type="INDPs" name="OD" mode="N">
      <CallingPartyNumber>5551212</CallingPartyNumber>
   </Event>
   <Event type="INDPs" name="OAB" mode="N">
      <CallingPartyNumber>5551212</CallingPartyNumber>
   </Event>
</spirits-event>
```

5. Call-related events

For readers who may not be familiar with the service execution aspects of PSTN/IN, we provide a brief tutorial next. Interested readers are urged to consult [19] for a detailed treatment of this subject.

Services in the PSTN/IN are executed based on a call model. A call model is a finite state machine used in SSPs and other call processing elements that accurately and concisely reflects the current state of a call at any given point in time. Call models consist of states called PICs (Points In Call) and transitions between states. Inter-state transitions pass through elements called Detection Points or DPs. DPs house one or more triggers. Every trigger has a firing criteria associated with it. When a trigger is armed (made active), and its associated firing criteria are satisfied, it fires. The particulars of firing criteria may vary based on the call model being supported.

When a trigger fires, a message is formatted with call state information and transmitted by the SSP to the SCP. The SCP then reads this call related data and generates a response which the SSP then uses in further call processing.

Detection Points are of two types: TDPs (or Trigger Detection Points), and EDPs (or Event Detection Points). TDPs are provisioned with statically armed triggers (armed through Service Management Tools). EDPs are dynamically armed triggers (armed by the SCP as call processing proceeds). DPs may also be classified as "Request" or "Notification" DPs. Thus, one can have TDP-R's, TDP-N's, EDP-R's and EDP-N's.

The "-R" type of DPs require the SSP to suspend call processing when communication with the SCP is initiated. Call processing resumes when a response is received. The "-N" type of DPs enable the SSP to continue with call processing when the trigger fires, after it sends out the message to the SCP, notifying it that a certain event has occurred.

Call models typically support different types of detection points. Note that while INAP and the IN Capability Set (CS)-2 [7] call model are used in this document as examples, and for ease of explanation, other call models possess similar properties. For example, the Wireless Intelligent Network (WIN) call model also supports the dynamic arming of triggers. Thus, the essence of this discussion applies not just to the wireline domain, but applies equally well to the wireless domain as well.

When the SCP receives the INAP formatted message from the SSP, if the SCP supports the SPIRITS architecture, it can encode the INAP message contents into a SPIRITS protocol message which is then transmitted to SPIRITS-capable elements in the IP network. Similarly, when it receives responses back from said SPIRITS capable elements, it can reformat the response content into the INAP format and forward these messages back to SSPs. Thus the process of inter-conversion and/or encoding between the INAP parameters and the SPIRITS protocol is of primary interest.

An SCP is a physical manifestation of the Service Control Function. An SSP is a physical manifestation of the Service Switching Function (and the Call Control Function). To support uniformity of nomenclature between the various SPIRITS drafts, we shall use the terms SCP and SCF, and SSP and SSF interchangeably in this document.

5.1. IN-specific requirements

Section 4 of [4] outlines the IN-related requirements on the SPIRITS protocol. The SUBSCRIBE request arriving at the SPIRITS notifier MUST contain the events to be monitored (in the form of a DP list), the mode (request or a notification, the difference being that for a

request, the SPIRITS subscriber can influence subsequent call processing and for a notification, no further influence is needed), and any DP-related parameters.

Section 4 of [4] also enumerates a list of Capability Set 3 (CS-3) DPs for SPIRITS services. It is a requirement (RFC3298:Section 4) that the SPIRITS protocol specify the relevant parameters of the DPs. These DPs and their relevant parameters to be carried in a SUBSCRIBE request are codified in an XML schema. All SPIRITS subscribers MUST understand this schema for subscribing to the DPs in the PSTN. The schema is defined in Section 9.

When a DP fires, a notification -- using a SIP NOTIFY request -- is transmitted from the SPIRITS notifier to the SPIRITS subscriber. The NOTIFY request contains an XML document which describes the DP that fired and any relevant parameters. The DPs and their relevant parameters to be carried in a NOTIFY request are codified in an XML schema. All SPIRITS notifiers MUST understand this schema; this schema MAY be extended. The schema is defined in Section 9.

In addition, Appendices A and B of [6] contain a select subset of CS-2 DPs that may be of interest to the reader. However, this document will only refer to CS-3 DPs outlined in [4].

5.2. Detection points and required parameters

The IN CS-3 DPs envisioned for SPIRITS services (RFC3298:Section 4) are described next. IN DPs are characterized by many parameters, however, not all such parameters are required -- or even needed -- by SPIRITS. This section, thus, serves to list the mandatory parameters for each DP that MUST be specified in subscriptions and notifications. Implementations can specify additional parameters as XML extensions associated with a private (or public and standardized) namespace.

The exhaustive list of IN CS-3 DPs and their parameters can be found in reference [13].

Each DP is given a SPIRITS-specific mnemonic for use in the subscriptions and notifications.

5.2.1. Originating-side DPs

Origination Attempt Authorized
SPIRITS mnemonic: OAA
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameters in NOTIFY: CallingPartyNumber, CalledPartyNumber

CallingPartyNumber: A string used to identify the calling party for the call. The actual length and encoding of this parameter depend on the particulars of the dialing plan used.

CalledPartyNumber: A string containing the number (e.g., called directory number) used to identify the called party. The actual length and encoding of this parameter depend on the particulars of the dialing plan used.

Collected Information
SPIRITS mnemonic: OCI
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameters in NOTIFY: CallingPartyNumber, DialledDigits

DialledDigits: This parameter contains non-translated address information collected/received from the originating user/line/trunk

Analyzed Information
SPIRITS mnemonic: OAI
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameters in NOTIFY: CallingPartyNumber, DialledDigits

Origination Answer
SPIRITS mnemonic: OA
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameters in NOTIFY: CallingPartyNumber, CalledPartyNumber

Origination Term Seized
SPIRITS mnemonic: OTS
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameter in NOTIFY: CallingPartyNumber, CalledPartyNumber

Origination No Answer
SPIRITS mnemonic: ONA
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameter in NOTIFY: CallingPartyNumber, CalledPartyNumber

Origination Called Party Busy
SPIRITS mnemonic: OCPB
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameters in NOTIFY: CallingPartyNumber, CalledPartyNumber

Route Select Failure
SPIRITS mnemonic: ORSF
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameter in NOTIFY: CallingPartyNumber, CalledPartyNumber

Origination Mid Call
SPIRITS mnemonic: OMC
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameter in NOTIFY: CallingPartyNumber

Origination Abandon
SPIRITS mnemonic: OAB

Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameter in NOTIFY: CallingPartyNumber

Origination Disconnect
SPIRITS mnemonic: OD
Mandatory parameter in SUBSCRIBE: CallingPartyNumber
Mandatory parameter in NOTIFY: CallingPartyNumber, CalledPartyNumber

5.2.2. Terminating-side DPs

Termination Answer
SPIRITS mnemonic: TA
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameters in NOTIFY: CallingPartyNumber, CalledPartyNumber

Termination No Answer
SPIRITS mnemonic: TNA Mandatory parameter in SUBSCRIBE:
CalledPartyNumber
Mandatory parameters in NOTIFY: CallingPartyNumber, CalledPartyNumber

Termination Mid-Call
SPIRITS mnemonic: TMC
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber

Termination Abandon
SPIRITS mnemonic: TAB
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber

Termination Disconnect
SPIRITS mnemonic: TD
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameters in NOTIFY: CalledPartyNumber, CallingPartyNumber

Termination Attempt Authorized
SPIRITS mnemonic: TAA
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameters in NOTIFY: CalledPartyNumber, CallingPartyNumber

Termination Facility Selected and Available
SPIRITS mnemonic: TFSA
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber

Termination Busy
SPIRITS mnemonic: TB
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameters in NOTIFY: CalledPartyNumber, CallingPartyNumber, Cause

Cause: This parameter contains a string value of either "Busy" or "Unreachable". The difference between these is translated as a requirement (RFC3298:Section 5) to aid in the SPIRITS subscriber in determining if the called party is indeed busy (engaged), or if the called party is unavailable (as it would be if it were on the cellular PSTN and the mobile subscriber was not registered with the network).

5.3. Services through dynamic DPs

Triggers in the PSTN can be armed dynamically, often outside the context of a call. The SIP event notification mechanism [3] is, therefore, a convenient means to exploit in those cases where triggers housed in EDPs fire (see section 3 of [4]). Note that [4] uses the term "persistent" to refer to call-related DP arming and associated interactions.

The SIP Events Package enables IP endpoints (or hosts) to subscribe to and receive subsequent notification of events occurring in the PSTN. With reference to Figure 2, this includes communication on the interfaces marked "B" and "C".

5.3.1. Normative usage

A subscriber will issue a SUBSCRIBE request which identifies a set of events (DPs) it is interested in getting the notification of. This set MUST contain at least one DP, it MAY contain more than one. The SUBSCRIBE request is routed to the notifier, where it is accepted, pending a successful authentication.

When any of the DPs identified in the set of events fires, the notifier will format a NOTIFY request and direct it towards the subscriber. The NOTIFY request will contain information pertinent to the event that was triggered. The un-encountered DPs MUST be subsequently dis-armed by the SPIRITS notifier and/or the SCF.

The dialog established by the SUBSCRIBE terminates when the event of interest occurs and this notification is passed to the subscriber through a NOTIFY request. If the subscriber is interested in the future occurrence of the same event, it MUST issue a new SUBSCRIBE request, establishing a new dialog.

When the subscriber receives a NOTIFY request, it can subsequently choose to act in a manner appropriate to the notification.

The remaining sections fill in the specific package responsibilities raised in RFC3265 [3], Section 4.4.

5.3.2. Event package name

This document defines two event packages; the first of these is defined in this section and is called "spirits-INDPs". This package MUST be used for events corresponding to IN detection points in the cellular or wireline PSTN. All entities that implement the SPIRITS protocol and support IN detection points MUST set the "Event" request header [3] to "spirits-INDPs." The "Allow-Events" general header [3] MUST include the token "spirits-INDPs" if the entity implements the SPIRITS protocol and supports IN detection points.

```
Event: spirits-INDPs
Allow-Events: spirits-INDPs
```

The second event package is defined and discussed in Section 6.

5.3.3. Event package parameters

The "spirits-INDPs" event package does not support any additional parameters to the Event header.

5.3.4. SUBSCRIBE bodies

SUBSCRIBE requests that serve to terminate the subscription MAY contain an empty body; however, SUBSCRIBE requests that establish a dialog MUST contain a body which encodes three pieces of information:

(1) The set of events (DPs) that is being subscribed to. A subscriber MAY subscribe to multiple DPs in one SUBSCRIBE request, or MAY issue a different SUBSCRIBE request for each DP it is interested in receiving a notification for. The protocol allows for both forms of representation, however, it recommends the former manner of subscribing to DPs if the service depends on any of the DPs being triggered.

(2) Because of the requirement [4] that IN be informed whether the detection point is set as the request or notification, all events in the "spirits-INDPs" package (but not in the "spirits-user-prof" package) are required to provide a "mode" parameter, whose values are "R" (for Request) and "N" for notification.

(3) A list of the values of the parameters associated with the event detection point (Note: the term "event" here refers to the IN usage -- a dynamically armed DP is called an Event Detection Point). Please see Section 5.2.1 and Section 5.2.2 for a list of parameters associated with each DP.

The default body type for SUBSCRIBEs in SPIRITS is denoted by the MIME type "application/spirits-event+xml". The "Accept" header, if present, MUST include this MIME type.

5.3.5. Subscription duration

For package "spirits-INDPs", the purpose of the SUBSCRIBE request is to arm the DP, since as far as IN is concerned, being armed is the first essential pre-requisite. A DP maybe armed either statically (for instance, through service provisioning), or dynamically (by the SCF). A statically armed DP remains armed until it is disarmed proactively. A dynamically armed DP remains armed for the duration of a call (or more appropriately, no longer than the duration of a particular SSF-SCF relationship).

Dynamically armed DPs are automatically disarmed when the event of interest occurs in the notifier. It is up to the subscriber to re-arm the DPs within the context of a call, if it so desires.

Statically armed DPs are considered outside the scope of the SPIRITS protocol requirements [4] and thus will not be considered any further.

5.3.6. NOTIFY bodies

Bodies in NOTIFY requests for the "spirits-INDPs" package are optional. If present, they MUST be of the MIME type "application/spirits-event+xml". The body in a NOTIFY request encapsulates the following pieces of information which can be used by the subscriber:

(1) The event that resulted in the NOTIFY being generated (typically, but not always, this will be the same event present in the corresponding SUBSCRIBE request).

(2) The "mode" parameter; it is simply reflected back from the corresponding SUBSCRIBE request.

(3) A list of values of the parameters associated with the event that the NOTIFY is being generated for. Depending on the actual event, the list of the parameters will vary.

If the subscriber armed multiple DPs as part of a single SUBSCRIBE request, all the un-encountered DPs that were part of the same SUBSCRIBE dialog MUST be dis-armed by the SPIRITS notifier and/or the SCF/SCP.

5.3.7. Notifier processing of SUBSCRIBE requests

When the notifier receives a SUBSCRIBE request, it MUST authenticate the request and ensure that the subscriber is authorized to access the resource being subscribed to, in this case, PSTN/IN events on a certain PSTN line.

Once the SUBSCRIBE request has been authenticated and authorized, the notifier interfaces with the SCF over interface D to arm the detection points corresponding to the PSTN line contained in the SUBSCRIBE body. The particulars about interface D is out of scope for this document; here we will simply assume that the notifier can affect the arming (and disarming) of triggers in the PSTN through interface D.

5.3.8. Notifier generation of NOTIFY requests

If the notifier expects the arming of triggers to take more than 200 ms, it MUST send a 202 response to the SUBSCRIBE request immediately, accepting the subscription. It should then send a NOTIFY request with an empty body. This NOTIFY request MUST have a "Subscription-State" header with a value of "pending".

This immediate NOTIFY with an empty body is needed since the resource identified in the SUBSCRIBE request does not have as yet a meaningful state.

Once the notifier has successfully interfaced with the SCF, it MUST send a subsequent NOTIFY request with an empty body and a "Subscription-State" header with a value of "active."

When the event of interest identified in the SUBSCRIBE request occurs, the notifier sends out a new NOTIFY request which MUST contain a body (see Section 5.3.6). The NOTIFY request MUST have a "Subscription-State" header and its value MUST be set to "terminated" with a reason parameter of "fired".

5.3.9. Subscriber processing of NOTIFY requests

The exact steps executed at the subscriber when it gets a NOTIFY request will depend on the service being implemented. As a generality, the UA associated with the subscriber should somehow impart this information to the user by visual or auditory means, if at all possible.

If the NOTIFY request contained a "Subscription-State" header with a value of "terminated" and a reason parameter of "fired", the UA associated with the subscriber MAY initiate a new subscription for the event that was just reported through the NOTIFY request.

> Whether or not to initiate a new subscription when an existing one expires is up to the context of the service that is being implemented. For instance, a user may configure her UA to always re-subscribe to the same event when it fires, but this is not necessarily the normative case.

5.3.10. Handling of forked requests

Forking of SUBSCRIBE requests is prohibited. Since the SUBSCRIBE request is targeted towards the PSTN, highly irregular behaviors occur if the request is allowed to fork. The normal SIP DNS lookup and routing rules [11] should result in a target set with exactly one element: the notifier.

5.3.11. Rate of notifications

For reasons of security more than network traffic, it is RECOMMENDED that the notifier issue two or, at most three NOTIFY requests for a subscription. If the subscription was accepted with a 202 response, a NOTIFY will be sent immediately towards the subscriber. This NOTIFY serves to inform the subscriber that the request has been accepted and is being acted on.

Once the resource (detection points) identified in the SUBSCRIBE request have been initialized, the notifier MUST send a second NOTIFY request. This request contains the base state of the resource.

When an event of interest occurs which leads to the firing of the trigger associated with the detection points identified in the SUBSCRIBE request, a final NOTIFY is sent to the subscriber. This NOTIFY request contains more information about the event of interest.

If the subscription was accepted with a 200 response, the notifier simply sends two NOTIFY requests: one containing the base state of the resource, and the other containing information that lead to the firing of the detection point.

5.3.12. State agents

State agents are not used in SPIRITS.

5.3.13. Examples

This section contains example call flows for a SPIRITS service called Internet Caller-ID Delivery (ICID). One of the benchmark SPIRITS service, as described in section 2.2 of [1] is Internet Caller-ID delivery:

> This service allows the subscriber to see the caller's number or name or both while being connected to the Internet. If the subscriber has only one telephone line and is using the very line for the Internet connection, the service is a subset of the ICW service and follows the relevant description in Section 2.1. Otherwise, the subscriber's IP host serves as an auxiliary device of the telephone to which the call is first sent.

We present an example of a SPIRITS call flow to realize this service. Note that this is an example only, not a normative description of the Internet Caller-ID service.

Further text and details of SIP messages below refer to the call flow provided in Figure 3. Figure 3 depicts the 4 entities that are an integral part of any SPIRITS service (the headings of the entities refer to the names established in Figure 1 in [1]) -- the SPIRITS subscriber, the SPIRITS notifier and the SCF. Note that the SPIRITS gateway is not included in this figure; logically, SPIRITS messages flow between the SPIRITS server and the SPIRITS client. A gateway, if present, may act as a proxy.

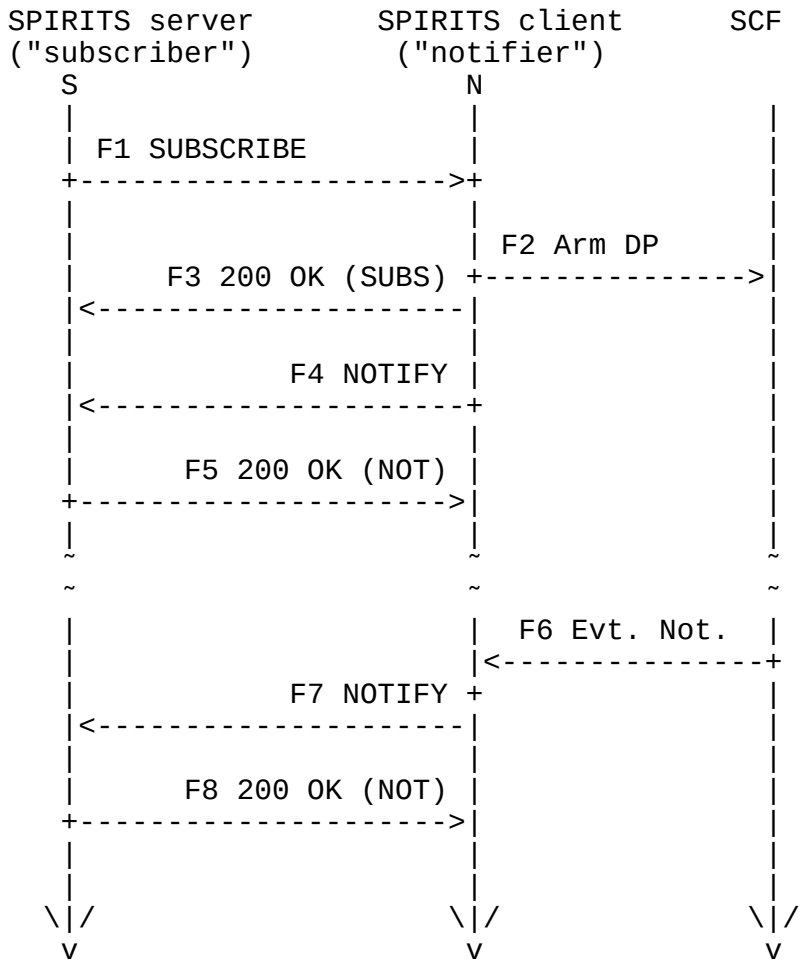

Figure 3: Sample call flow

This call flow depicts an overall operation of a "subscriber" successfully subscribing to the IN Termination_Attempt_Authorized DP (the "subscriber" is assumed to be a user, possibly at work, who is interested in knowing when he/she gets a phone call to his/her home phone number) -- this interaction is captured in messages F1 through F8 in Figure 3. The user sends (F1) a SIP SUBSCRIBE request identifying the DP it is interested in along with zero or more parameters relevant to that DP (in this example, the Termination_Attempt_DP will be employed). The SPIRITS notifier in turns interacts with the SCF to arm the Termination_Attempt_DP for the service (F2). An immediate NOTIFY with the current state information is send to the subscriber (F4, F5).

At some point after the above sequence of events has transpired, the PSTN gets a call to the users phone. The SSF informs the SCF of this event when it encounters an armed Termination_Attempt_DP (not shown in Figure 3). The SCF informs the SPIRITS notifier of this event (F6).

When the SPIRITS notifier receives this event, it forms a SIP NOTIFY request and directs it to the SPIRITS subscriber (F7). This NOTIFY will contain all the information elements necessary to identify the caller to the subscriber. The subscriber, upon receiving the notification (F8) may pop open a window with the date/time and the number of the caller.

The rest of this section contains the details of the SIP messages in Figure 3. The call flow details below assume that the SPIRITS gateway is, for the purpose of this example, a SIP proxy that serves as the default outbound proxy for the notifier and an ingress host of the myprovider.com domain for the subscriber. The subscriber and notifier may be in separate administrative domains.

F1: S->N

```
SUBSCRIBE sip:myprovider.com SIP/2.0
From: <sip:vkg@example.com>;tag=8177-afd-991
To: <sip:16302240216@myprovider.com>
CSeq: 18992 SUBSCRIBE
Call-ID: 3329as77@host.example.com
Contact: <sip:vkg@host.example.com>
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK776asdhds
Expires: 3600
Event: spirits-INDPs
Allow-Events: spirits-INDPs, spirits-user-prof
Accept: application/spirits-event+xml
Content-Type: application/spirits-event+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<spirits-event xmlns="urn:ietf:params:xml:ns:spirits-1.0">
   <Event type="INDPs" name="TAA" mode="N">
         <CalledPartyNumber>6302240216</CalledPartyNumber>
   </Event>
</spirits-event>
```

The subscriber forms a SIP SUBSCRIBE request which identifies the DP that it wants to subscribe to (in this case, the TAA DP) and the actual line it wants that DP armed for (in this case, the line

associated with the phone number 6302240216). This request eventually arrives at the SIPRITS notifier, N, which authenticates it (not shown) and sends a successful response to the subscriber:

F3: N->S

```
SIP/2.0 200 OK
From: <sip:vkg@example.com>;tag=8177-afd-991
To: <sip:16302240216@myprovider.com>;tag=SPIRITS-TAA-6302240216
CSeq: 18992 SUBSCRIBE
Call-ID: 3329as77@host.example.com
Contact: <sip:notifier.myprovider.com>
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK776asdhds
Expires: 3600
Accept: application/spirits-event+xml
Content-Length: 0
```

The notifier interacts with the SCF to arm the DP and also sends an immediate NOTIFY towards the subscriber informing the subscriber of the current state of the notification:

F4: N->S

```
NOTIFY sip:vkg@host.example.com SIP/2.0
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-TAA-6302240216
To: <sip:vkg@example.com>;tag=8177-afd-991
Via: SIP/2.0/UDP gateway.myprovider.com;branch=z9hG4bK-9$0-1
Via: SIP/2.0/UDP notifier.myprovider.com;branch=z9hG4bKqo--9
Call-ID: 3329as77@host.example.com
Contact: <sip:notifier.myprovider.com>
Subscription-State: active
CSeq: 3299 NOTIFY
Accept: application/spirits-event+xml
Content-Length: 0
```

F5: S->N

```
SIP/2.0 200 OK
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-TAA-6302240216
To: <sip:vkg@example.com>;tag=8177-afd-991
Via: SIP/2.0/UDP gateway.myprovider.com;branch=z9hG4bK-9$0-1
Via: SIP/2.0/UDP notifier.myprovider.com;branch=z9hG4bKqo--9
Call-ID: 3329as77@host.example.com
Contact: <sip:vkg@host.example.com>
CSeq: 3299 NOTIFY
Accept: application/spirits-event+xml
Content-Length: 0
```

At some later point in time (before the subscription established in F1 expires at the notifier), a call arrives at the number identified in XML-encoded body of F1 -- 6302240216. The SCF notifies the notifier (F6). Included in this notification is the relevant information from the PSTN, namely, the phone number of the party attempting to call 6302240216. The notifier uses this information to create a SIP NOTIFY request and sends it to the subscriber. The SIP NOTIFY request has a XML-encoded body with the relevant information from the PSTN:

F7: N->S

```
NOTIFY sip:vkg@host.example.com SIP/2.0
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-TAA-6302240216
To: <sip:vkg@example.com>;tag=8177-afd-991
Via: SIP/2.0/UDP notifier.myprovider.com;branch=z9hG4bK9inn-=u7
Call-ID: 3329as77@host.example.com
Contact: <sip:notifier.myprovider.com>
CSeq: 3300 NOTIFY
Subscription-State: terminated;reason=fired
Accept: application/spirits-event+xml
Event: spirits-INDPs
Allow-Events: spirits-INDPs, spirits-user-prof
Content-Type: application/spirits-event+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<spirits-event xmlns="urn:ietf:params:xml:ns:spirits-1.0">
   <Event type="INDPs" name="TAA" mode="N">
         <CalledPartyNumber>6302240216</CalledPartyNumber>
         <CallingPartyNumber>3125551212</CallingPartyNumber>
   </Event>
</spirits-event>
```

There are two important issues to note in the call flows for F7:

(1) The body of the NOTIFY request contains the information passed to the SPIRITS notifier from the SCF. In this particular example, this is the phone number of the party (3125551212) that attempted to call 6302240216.

(2) Since the notification occurred, the subscription established in F1 terminated (as evident by the Subscription-State header). The subscription terminated normally due to the DP associated with TAA firing (hence the reason code of "fired" in the Subscription-State header). If the subscriber

wants to get notified of another attempt to call the number 6302240216, he/she should send a new SUBSCRIBE request to the notifier.

The subscriber can take any appropriate action upon the receipt of the NOTIFY in F7. A reasonable implementation may pop up a window populated with the information contained in the body of F12, along with a button asking the subscriber if they would like to re-subscribe to the same event. Alternatively, a re-subscription could be generated automatically by the subscriber's UA based on his/her preferences.

To complete the protocol, the subscriber also sends a 200 OK message towards the notifier:

F8: S->N

```
200 OK SIP/2.0
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-TAA-6302240216
To: <sip:vkg@example.com>;tag=8177-afd-991
Via: SIP/2.0/UDP notifier.myprovider.com;z9hG4bK9inn-=u7
Call-ID: 3329as77@host.example.com
CSeq: 3300 NOTIFY
Content-Length: 0
```

5.3.14. Use of URIs to retrieve state

The "spirits-INDPs" package MUST NOT use URIs to retrieve state. It is expected that most state information for this package is compact enough to fit in a SIP message. However, to err on the side of caution, implementations MUST follow the convention outlined in Section 18.1.1 of [5] and use a congestion controlled transport if the size of the request is within 200 bytes of the path MTU if known, or if the request size is larger than 1300 bytes and the path MTU is unknown.

5.4. Services through static DPs

We mentioned in Section 5.1 that the first trigger that fires during call processing is typically a TDP since there isn't any pre-existing control relationship between the SSF and the SCF. Some Internet hosts may have expressed an interest in executing services based on TDPs (through an a-priori arrangement, which is not a part of this specification). Thus, the PSTN will notify such hosts. To do so, it will send a SIP request (typically an INVITE) towards the Internet host. The body of the SIP request MUST contain multi-part MIME with two MIME components: the first part corresponding to the normal payload, if any, of the request; and the second part will contain

SPIRITS-specific information (e.g., the DP that fired). Responses to the INVITE request, or subsequent SUBSCRIBE messages from the Internet host to the PSTN within a current call context may result in EDPs being armed.

5.4.1. Internet Call Waiting (ICW)

ICW as a benchmark SPIRITS service actually predates SPIRITS itself. Pre-SPIRITS implementations of ICW are detailed in [10]. However, as the document notes, while a diversity of implementations exists, these implementations are not interoperable. At the time [10] was published, the industry did not have the depth of experience with SIP as is the case now. The use of SIP in [10] does not constitute normative usage of SIP as described in [5]; for instance, no mention is made of the SDP (if any) in the initial INVITE (especially since this pertains to "accept the call using VoIP" case). Thus this section serves to provide a normative description of ICW in SPIRITS.

The description of ICW is deceptively simple: it is a service most useful for single line phone subscribers that use the line to establish an Internet session. In a nutshell, the service enables a subscriber engaged in an Internet dial-up session to

- be notified of an incoming call to the very same telephone line that is being used for the Internet connection,
- specify the desirable treatment of the call, and
- have the call handled as specified.

5.4.2. Call disposition choices

Section 2 of [10] details the call disposition outcome of a ICW session. They are reproduced here as a numbered list for further discussion:

1. Accepting the call over the PSTN line, thus terminating the Internet (modem) connection

2. Accepting the call over the Internet using Voice over IP (VoIP)

3. Rejecting the call

4. Playing a pre-recorded message to the calling party and disconnecting the call

5. Forwarding the call to voice mail

6. Forwarding the call to another number

7. Rejecting (or Forwarding) on no Response - If the subscriber fails to respond within a certain period of time after the dialog box has been displayed, the incoming call can be either rejected or handled based on the treatment pre-defined by the subscriber.

It should be pointed out for the sake of completeness that ICW as a SPIRITS service is not possible without making the SCP aware of the fact that the subscriber line is being used for an Internet session. That awareness, however, is not a part of the ICW service, but solely a pre-requisite. One of the following three methods MUST be utilized to impart this information to the SCP:

A. ICW subscriber based method: the ICW client on the subscriber's PC notifies the SCP of the Internet session by issuing a SIP REGISTER request.

B. IN based method: SCP maintains a list of Internet Service Provider (ISP) access numbers for a geographical area; when one of these numbers is dialed and connected to, it (the SCP) assumes that the calling party is engaged in an Internet session.

C. Any combination of methods A and B.

ICW depends on a TDP to be provisioned in the SSP. When the said TDP is encountered, the SSP suspends processing of the call and sends a request to the SPIRITS-capable SCP. The SCP determines that the subscriber line is being used for an Internet session. It instructs the SPIRITS notifier on the SCP to create a SIP INVITE request and send it to the SPIRITS subscriber running on the subscriber's IP host.

The SPIRITS subscriber MUST return one of the possible call disposition outcomes catalogued in Section 5.4.2. Note that outcomes 1 and 4 through 7 can all be coalesced into one case, namely redirecting (using the SIP 3xx response code) the call to an alternative SIP URI. In case of 1, the URI of the redirected call MUST match the very same number being used by the customer to get online. Rejecting the call implies sending a non-2xx and non-3xx final response; the remaining outcomes result in the call being redirected to an alternate URI which provides the desired service (i.e., play a pre-recorded announcement, or record a voice message).

Further processing of a SPIRITS notifier when it receives a final response can be summarized by the following steps:

1. If the response is a 4xx, 5xx, or 6xx class of response, generate and transmit an ACK request and instruct the SSP to play a busy tone to the caller.

2. Else, for all 3xx responses, generate and transmit an ACK request, and compare the redirected URI to the subscriber's line number:

   2a. If the comparison indicates a match, instruct the SSP to hold onto the call for just enough time to allow the SPIRITS subscriber to disconnect the modem, thus freeing up the line; and then continue with normal call processing, which will result in the subscriber's phone to ring.

   2b. If the comparison fails, instruct the SSP to route the call to the redirected URI.

3. Else, for a 2xx response, follow the steps in section 5.4.3.

5.4.3. Accepting an ICW session using VoIP

One call handling option in ICW is to "accept an incoming call using VoIP". The SPIRITS notifier has no way of knowing a-priori if the subscriber (callee) will be choosing this option; nonetheless, it has to account for such a choice by adding a SDP in the body of the INVITE request. A possible way of accomplishing this is to have the SPIRITS notifier control a PSTN gateway and allocate appropriate resources on it. Once this is done, the SPIRITS notifier adds network information (IP address of the gateway and port numbers where media will be received) and codec information as the SDP portion of the body in the INVITE request. SPIRITS requires the DP information to be carried in the request body as well. To that extent, the SPIRITS notifier MUST also add the information associated with the TDP that triggered the service. Thus, the body of the INVITE MUST contain multi-part MIME, with two components.

The SPIRITS notifier transmits the INVITE request to the subscriber and now waits for a final response. Further processing when the SPIRITS subscriber returns a 200 OK MUST be handled as follows:

On the receipt of a 200 OK containing the SDP of the subscriber's UA, the SPIRITS notifier will instruct the SSP to terminate the call on a pre-allocated port on the gateway. This port MUST be correlated by the gateway to the SDP that was sent in the earlier INVITE.

The end result is that the caller and callee hold a voice session with part of the session occurring over VoIP.

6. Non-call related events

There are network events that are not related to setting up, maintaining, or tearing down voice calls. Such events occur on the cellular wireless network and can be used by SPIRITS to provide services. The SPIRITS protocol requirement explicitly includes the following events for which SPIRITS notification is needed (RFC3298:Section 5(b)):

1. Location update in the same Visitor Location Register (VLR) service area

2. Location update in another VLR service area

3. International Mobile Subscriber Identity (IMSI) attach

4. Mobile Subscriber (MS) initiated IMSI detach

5. Network initiated IMSI detach

6.1. Non-call events and their required parameters

Each of the five non-call related event is given a SPIRITS-specific mnemonic for use in subscriptions and notifications.

Location update in the same VLR area
SPIRITS mnemonic: LUSV
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber, Cell-ID

Cell-ID: A string used to identify the serving Cell-ID. The actual length and representation of this parameter depend on the particulars of the cellular provider’s network.

Location update in different VLR area
SPIRITS mnemonic: LUDV
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber, Cell-ID

IMSI attach
SPIRITS mnemonic: REG
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber, Cell-ID

MS initiated IMSI detach
SPIRITS mnemonic: UNREGMS
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber

Network initiated IMSI detach
SPIRITS mnemonic: UNREGNTWK
Mandatory parameter in SUBSCRIBE: CalledPartyNumber
Mandatory parameter in NOTIFY: CalledPartyNumber

6.2. Normative usage

A subscriber will issue a SUBSCRIBE request which identifies a set of non-call related PSTN events it is interested in getting the notification of. This set MAY contain exactly one event, or it MAY contain multiple events. The SUBSCRIBE request is routed to the notifier where it is accepted, pending a successful authentication.

When any of the events identified in the set occurs, the notifier will format a NOTIFY request and direct it towards the subscriber. The NOTIFY request will contain information pertinent to the one of the event whose notification was requested.

The dialog established by the SUBSCRIBE persists until it expires normally, or is explicitly expired by the subscriber. This behavior is different than the behavior for subscriptions associated with the "spirits-INDPs" package. In the cellular network, the events subscribed for may occur at a far greater frequency than those compared to the wireline network (consider location updates as a cellular user moves around). Thus it is far more expedient to allow the subscription to expire normally.

When a subscriber receives a NOTIFY request, it can subsequently choose to act in a manner appropriate to the notification.

The remaining sections fill in the specific package responsibilities raised in RFC3265 [3], Section 4.4.

6.3. Event package name

This document defines two event packages; the first was defined in Section 5.3. The second package, defined in this section is called "spirits-user-prof". This package MUST be used for events corresponding to non-call related events in the cellular network. All entities that implement the SPIRITS protocol and support the non-call related events outlined in the SPIRITS protocol requirements

(RFC3298:Section 5(b)) MUST set the "Event" header request header[3] to "spirits-user-prof." The "Allow-Events" general header [3] MUST include the token "spirits-user-prof" as well.

Example:

```
Event: spirits-user-prof
Allow-Events: spirits-user-prof, spirits-INDPs
```

6.4. Event package parameters

The "spirits-user-prof" event package does not support any additional parameters to the Event header

6.5. SUBSCRIBE bodies

SUBSCRIBE requests that serve to terminate the subscriptions MAY contain an empty body; however, SUBSCRIBE requests that establish a dialog MUST contain a body which encodes two pieces of information:

(1) The set of events that is being subscribed to. A subscriber MAY subscribe to multiple events in one SUBSCRIBE request, or MAY issue a different SUBSCRIBE request for each event it is interested in receiving a notification for. The protocol allows for both forms of representation. However, note that if one SUBSCRIBE is used to subscribe to multiple events, then an expiry for the dialog associated with that subscription affects all such events.

(2) A list of values of the parameters associated with the event. Please see Section 6.1 for a list of parameters associated with each event.

The default body type for SUBSCRIBEs in SPIRITS is denoted by the MIME type "application/spirits-event+xml". The "Accept" header, if present, MUST include this MIME type.

6.6. Subscription duration

The duration of a dialog established by a SUBSCRIBE request is limited to the expiration time negotiated between the subscriber and notifier when the dialog was established. The subscriber MUST send a new SUBSCRIBE to refresh the dialog if it is interested in keeping it alive. A dialog can be terminated by sending a new SUBSCRIBE request with an "Expires" header value of 0, as outlined in [3].

6.7. NOTIFY bodies

Bodies in NOTIFY requests for the "spirits-user-prof" package are optional. If present, they MUST be of the MIME type "application/spirits-event+xml". The body in a NOTIFY request encapsulates the following pieces of information which can be used by the subscriber:

(1) The event that resulted in the NOTIFY being generated (typically, but not always, this will be the same event present in the corresponding SUBSCRIBE request).

(2) A list of values of the parameters associated with the event that the NOTIFY is being generated for. Depending on the actual event, the list of the parameters will vary.

6.8. Notifier processing of SUBSCRIBE requests

When the notifier receives a SUBSCRIBE request, it MUST authenticate the request and ensure that the subscriber is authorized to access the resource being subscribed to, in this case, non-call related cellular events for a mobile phone.

Once the SUBSCRIBE request has been authenticated and authorized, the notifier interfaces with the SCF over interface D to set marks in the HLR corresponding to the mobile phone number contained in the SUBSCRIBE body. The particulars of interface D are outside the scope of this document; here we simply assume that the notifier is able to set the appropriate marks in the HLR.

6.9. Notifier generation of NOTIFY requests

If the notifier expects the setting of marks in the HLR to take more than 200 ms, it MUST send a 202 response to the SUBSCRIBE request immediately, accepting the subscription. It should then send a NOTIFY request with an empty body. This NOTIFY request MUST have a "Subscription-State" header with a value of "pending".

This immediate NOTIFY with an empty body is needed since the resource identified in the SUBSCRIBE request does not have as yet a meaningful state.

Once the notifier has successfully interfaced with the SCF, it MUST send a subsequent NOTIFY request with an empty body and a "Subscription-State" header with a value of "active."

When the event of interest identified in the SUBSCRIBE request occurs, the notifier sends out a new NOTIFY request which MUST contain a body as described in Section 6.7.

6.10. Subscriber processing of NOTIFY requests

The exact steps executed at the subscriber when it receives a NOTIFY request depend on the nature of the service that is being implemented. As a generality, the UA associated with the subscriber should somehow impart this information to the user by visual or auditory means, if at all possible.

6.11. Handling of forked requests

Forking of SUBSCRIBE requests is prohibited. Since the SUBSCRIBE request is targeted towards the PSTN, highly irregular behaviors occur if the request is allowed to fork. The normal SIP DNS lookup and routing rules [11] should result in a target set with exactly one element: the notifier.

6.12. Rate of notifications

For reasons of congestion control, it is important that the rate of notifications not become excessive. For instance, if a subscriber subscribes to the location update event for a notifier moving through the cellular network at a high enough velocity, it is entirely conceivable that the notifier may generate many NOTIFY requests in a small time frame. Thus, within this package, the location update event needs an appropriate throttling mechanism.

Whenever a SPIRITS notifier sends a location update NOTIFY, it MUST start a timer (Tn) with a value of 15 seconds. If a subsequent location update NOTIFY request needs to be sent out before the timer has expired, it MUST be discarded. Any future location update NOTIFY requests MUST be transmitted only if Tn has expired (i.e. 15 seconds have passed since the last NOTIFY request was send out). If a location update NOTIFY is send out, Tn should be reset to go off again in 15 seconds.

6.13. State agents

State agents are not used in SPIRITS.

6.14. Examples

This section contains an example of a SPIRITS service that may be used to update the presence status of a mobile user. The call flow is depicted in Figure 4 below.

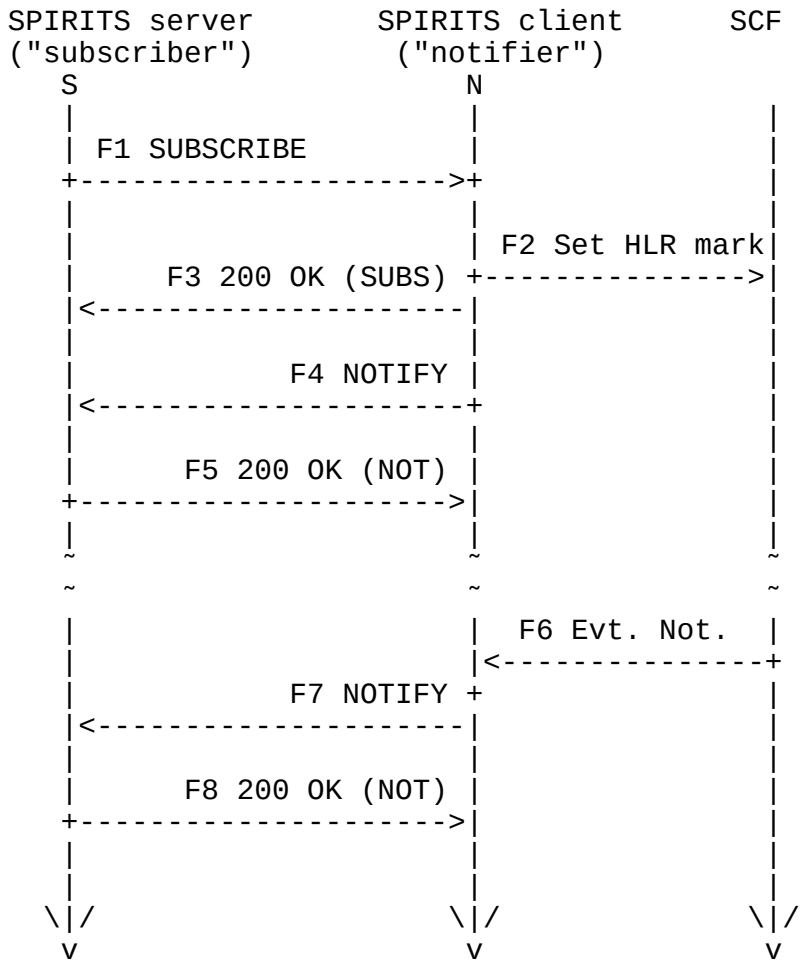

Figure 4: Sample call flow

In F1 of Figure 4, the subscriber indicates an interest in receiving a notification when a mobile user registers with the cellular network. The cellular network notes this event (F2) and confirms the subscription (F3-F5). When the mobile user turns on her cell phone and registers with the network, this event is detected (F6). The cellular network then sends out a notification to the subscriber informing it of this event (F7-F8).

We present the details of the call flow next.

In F1, the subscriber subscribes to the registration event (REG) of a cellular phone number.

```
F1: S->N
SUBSCRIBE sip:myprovider.com SIP/2.0
From: <sip:vkg@example.com>;tag=8177-afd-991
To: <sip:16302240216@myprovider.com>
CSeq: 18992 SUBSCRIBE
Call-ID: 3329as77@host.example.com
Contact: <sip:vkg@host.example.com>
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK776asdhdsa8
Expires: 3600
Event: spirits-user-prof
Allow-Events: spirits-INDPs, spirits-user-prof
Accept: application/spirits-event+xml
Content-Type: application/spirits-event+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<spirits-event xmlns="urn:ietf:params:xml:ns:spirits-1.0">
   <Event type="userprof" name="REG">
         <CalledPartyNumber>6302240216</CalledPartyNumber>
   </Event>
</spirits-event>
```

The subscription reaches the notifier which authenticates the request (not shown) and interacts with the SCF to update the subscribers database for this event. The notifier sends out a successful response to the subscription:

```
F3: N->S
SIP/2.0 200 OK
From: <sip:vkg@example.com>;tag=8177-afd-991
To: <sip:16302240216@myprovider.com>;tag=SPIRITS-REG-16302240216
CSeq: 18992 SUBSCRIBE
Call-ID: 3329as77@host.example.com
Contact: <sip:notifier.myprovider.com>
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK776asdhdsa8
Expires: 3600
Allow-Events: spirits-INDPs, spirits-user-prof
Accept: application/spirits-event+xml
Content-Length: 0
```

The notifier also sends out a NOTIFY request confirming the subscription:

```
F4: N->S
NOTIFY sip:vkg@host.example.com SIP/2.0
To: <sip:vkg@example.com>;tag=8177-afd-991
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-REG-16302240216
CSeq: 9121 NOTIFY
```

```
Call-ID: 3329as77@host.example.com
Contact: <sip:notifier.myprovider.com>
Subscription-State: active
Via: SIP/2.0/UDP notifier.myprovider.com;branch=z9hG4bK7007-091a
Allow-Events: spirits-INDPs, spirits-user-prof
Accept: application/spirits-event+xml
Content-Length: 0
```

The subscriber confirms the receipt of the NOTIFY request:

```
F5: S->N
SIP/2.0 200 OK
To: <sip:vkg@example.com>;tag=8177-afd-991
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-REG-16302240216
CSeq: 9121 NOTIFY
Call-ID: 3329as77@host.example.com
Contact: <sip:vkg@host.example.com>
Via: SIP/2.0/UDP notifier.myprovider.com;branch=z9hG4bK7007-091a
Content-Length: 0
```

In F6, the mobile user identified by the PSTN number "6302240216" turns the mobile phone on, thus causing it to register with the cellular network. The cellular network detects this event, and since a subscriber has indicated an interest in receiving a notification of this event, a SIP NOTIFY request is transmitted towards the subscriber:

```
F7: N->S
NOTIFY sip:vkg@host.example.com SIP/2.0
To: <sip:vkg@example.com>;tag=8177-afd-991
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-REG-16302240216
CSeq: 9122 NOTIFY
Call-ID: 3329as77@host.example.com
Contact: <sip:notifier.myprovider.com>
Subscription-State: terminated;reason=fired
Via: SIP/2.0/UDP notifier.myprovider.com;branch=z9hG4bK7yi-p12
Event: spirits-user-prof
Allow-Events: spirits-INDPs, spirits-user-prof
Accept: application/spirits-event+xml
Content-Type: application/spirits-event+xml
Content-Length: ...
```

```
<?xml version="1.0" encoding="UTF-8"?>
<spirits-event xmlns="urn:ietf:params:xml:ns:spirits-1.0">
   <Event type="userprof" name="REG">
         <CalledPartyNumber>6302240216</CalledPartyNumber>
         <Cell-ID>45987</Cell-ID>
   </Event>
</spirits-event>
```

The subscriber receives the notification and acknowledges it by sending a response:

F8: S->N

```
SIP/2.0 200 OK
To: <sip:vkg@example.com>;tag=8177-afd-991
From: <sip:16302240216@myprovider.com>;tag=SPIRITS-REG-16302240216
CSeq: 9122 NOTIFY
Call-ID: 3329as77@host.example.com
Via: SIP/2.0/UDP notifier.myprovider.com;branch=z9hG4bK7yi-p12
Content-Length: 0
```

Note that once the subscriber has received this notification, it can execute appropriate services. In this particular instance, an appropriate service may consist of the subscriber acting as a composer of a presence service and turning the presence status of the user associated with the phone number "6302240216" to "on". Also note in F7 that the notifier included a Cell ID in the notification.

The Cell ID can be used as a basis for location specific services; however, a discussion of such services is out of the scope of this document.

6.15. Use of URIs to retrieve state

The "spirits-user-prof" package MUST NOT use URIs to retrieve state. It is expected that most state information for this package is compact enough to fit in a SIP message. However, to err on the side of caution, implementations MUST follow the convention outlined in Section 18.1.1 of [5] and use a congestion controlled transport if the size of the request is within 200 bytes of the path MTU if known, or if the request size is larger than 1300 bytes and the path MTU is unknown.

7. IANA Considerations

This document calls for IANA to:

- o register two new SIP Event Packages per [3].
- o register a new MIME type per [20].
- o register a new namespace URN per [16].
- o register a new XML schema per [16].

7.1. Registering event packages

Package Name: spirits-INDPs

Type: package

Contact: Vijay K. Gurbani, vkg@lucent.com

Reference: RFC 3910

Package Name: spirits-user-prof

Type: package

Contact: Vijay K. Gurbani, vkg@lucent.com

Reference: RFC 3910

7.2. Registering MIME type

MIME media type name: application

MIME subtype name: spirits-event+xml

Mandatory parameters: none

Optional parameters: charset (same semantics of charset parameter in application/xml [9])

Encoding considerations: same as considerations outlined for application/xml in [9].

Security considerations: Section 10 of [9] and Section 8 of this document.

Interoperability considerations: none.

Published specifications: this document.

Applications which use this media type: SPIRITS aware entities which adhere to this document.

Additional information:

Magic number(s): none.

File extension(s): none.

Macintosh file type code(s): none.

Object Identifier(s) or OID(s): none.

Person and email address for further information: Vijay K. Gurbani, <vkg@lucent.com>

Intended usage: Common

Author/Change controller: The IETF

7.3. Registering URN

URI
urn:ietf:params:xml:ns:spirits-1.0

Description
This is the XML namespace URI for XML elements defined by this document. Such elements describe the SPIRITS information in the "application/ spirits-event+xml" content type.

Registrant Contact
IESG.

XML
BEGIN

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=utf-8"/>
  <title>Namespace for SPIRITS-related information</title>
</head>
<body>
  <h1>Namespace for SPIRITS-related information</h1>
```

```
        <h2>application/spirits-event+xml</h2>
        <p>See <a href="[[[URL of published RFC]]]">RFC3910</a>.</p>
      </body>
      </html>
    END
```

7.4. Registering XML schema

URI
urn:ietf:params:xml:schema:spirits-1.0

Description
XML base schema for SPIRITS entities.

Registrant Contact
IESG.

XML
Please see XML schema definition in Section 9 of this document.

8. Security Considerations

This section focuses on security considerations which are unique to SPIRITS. SIP security mechanisms are discussed in detail in the core SIP specification [5] and are outside the scope of this document. SPIRITS security mechanisms are based on and strengthen SIP security [5], for example, SPIRITS mandates the support of S/MIME. Beyond that, any other security solutions specified in [5], i.e., TLS or HTTP Digest authentication, may be utilized by SPIRITS operators.

As outlined in Chapter 9 (Security Consideration) of RFC3298 [4], the following security aspects are applicable to the SPIRITS protocol:

Authentication

Integrity

Confidentiality

Non-repudiation

The SPIRITS architecture in Figure 1 contains 5 interfaces -- A, B, C, D, and E. Of these, only two -- B and C -- are of interest to SPIRITS. Interfaces A and E are PINT interfaces and are thus assumed secured by the PINT RFC [8]. Security for interface D is out of scope in this document since it deals primarily with the PSTN infrastructure. We will discuss security aspects on interfaces B and C predicated on certain assumptions.

A driving assumption for SPIRITS security is that the SPIRITS gateway is owned by the same PSTN operator that owns the SPIRITS notifier. Thus, it is attractive to simply relegate security of interface C to the PSTN operator, and in fact, there are merits to doing just that since interface C crosses the IP Network and PSTN boundaries. However, a closer inspection reveals that both interfaces B and C transmit the SPIRITS protocol; thus, any security arrangement we arrive at for interface B can be suitably applied to interface C as well. This makes it possible to secure interface C in case the SPIRITS gateway is not owned by the same PSTN operator that owns the SPIRITS notifier.

The ensuing security discussion assumes that the SPIRITS subscriber is communicating directly to the SPIRITS notifier (and vice-versa) and specifies a security apparatus for this arrangement. However, the same apparatus can be used to secure the communication between a SPIRITS subscriber and an intermediary (like the SPIRITS gateway), and the same intermediary and the SPIRITS notifier.

Confidentiality of the SPIRITS protocol is essential since the information carried in the protocol data units is of a sensitive nature and may lead to privacy concerns if revealed to non-authorized parties. The communication path between the SPIRITS notifier and the SPIRITS subscriber should be secured through S/MIME [18] to alleviate privacy concerns, as is described in the Security Consideration section of the core SIP specification [5].

> S/MIME is an end-to-end security mechanism which encrypts the SPIRITS bodies for transit across an open network. Intermediaries need not be cognizant of S/MIME in order to route the messages (routing headers travel in the clear).

S/MIME provides all the security aspects for SPIRITS outlined at the beginning of this section: authentication, message integrity, confidentiality, and non-repudiation. Authentication properties provided by S/MIME would allow the recipient of a SPIRITS message to ensure that the SPIRITS payload was generated by an authorized entity. Encryption would ensure that only those SPIRITS entities possessing a particular decryption key are capable of inspecting encapsulated SPIRITS bodies in a SIP request.

All SPIRITS endpoints MUST support S/MIME signatures (CMS SignedData) and MUST support encryption (CMS EnvelopedData).

If the B and C interfaces are owned by the same PSTN operator, it is possible that public keys will be installed in the SPIRITS endpoints.

S/MIME supports two methods -- issuerAndSerialNumber and subjectKeyIdentifier -- of naming the public key needed to validate a signature. Between these, subjectKeyIdentifier works with X.509 certificates and other schemes as well, while issuerAndSerialNumber works with X.509 certificates only. If the administrator configures the necessary public keys, providing integrity through procedural means, then S/MIME can be used without X.509 certificates.

All requests (and responses) between SPIRITS entities MUST be encrypted.

When a request arrives at a SPIRITS notifier from a SPIRITS subscriber, the SPIRITS notifier MUST authenticate the request. The subscription (or registration) from a SPIRITS subscriber MUST be rejected if the authentication fails. If the SPIRITS subscriber successfully authenticated itself to the SPIRITS notifier, the SPIRITS notifier MUST, at the very least, ensure that the SPIRITS subscriber is indeed allowed to receive notifications of the events it is subscribing to.

> Note that this document does not proscribe how the SPIRITS notifier achieves this. In practice, it could be through access control lists (ACL) that are populated by a service management system in the PSTN, or through a web interface of some sort.

Requests from the SPIRITS notifier to the SPIRITS subscribers MUST also be authenticated, lest a malicious party attempts to fraudulently pose as a SPIRITS notifier to hijack a session.

9. XML schema definition

The SPIRITS payload is specified in XML; this section defines the base XML schema for documents that make up the SPIRITS payload. All SPIRITS entities that transport a payload characterized by the MIME type "application/spirits-event+xml" MUST support documents corresponding to the base schema below.

Multiple versions of the base schema are not expected; rather, any additional functionality (e.g., conveying new PSTN events) must be accomplished through the definition of a new XML namespace and a corresponding schema. Elements from the new XML namespace will then co-exist with elements from the base schema in a document.

```
<xs:schema targetNamespace="urn:ietf:params:xml:ns:spirits-1.0"
       xmlns:tns="urn:ietf:params:xml:ns:spirits-1.0"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">

     <!-- This import brings in the XML language attribute xml:lang-->
     <xs:import namespace="http://www.w3.org/XML/1998/namespace"
                schemaLocation="http://www.w3.org/2001/xml.xsd"/>

     <xs:annotation>
        <xs:documentation xml:lang="en">
              Describes SPIRITS events.
        </xs:documentation>
     </xs:annotation>

     <xs:element name="spirits-event" type="tns:SpiritsEventType"/>

     <xs:complexType name="SpiritsEventType">
        <xs:sequence>
           <xs:element name="Event" type="tns:EventType" minOccurs="1"
               maxOccurs="unbounded"/>
           <xs:any namespace="##other" processContents="lax"
               maxOccurs="unbounded"/>
        </xs:sequence>
     </xs:complexType>

     <xs:complexType name="EventType">
        <xs:sequence>
           <xs:element name="CalledPartyNumber" type="xs:token"
               minOccurs="0" maxOccurs="1"/>
           <xs:element name="CallingPartyNumber" type="xs:token"
               minOccurs="0" maxOccurs="1"/>
           <xs:element name="DialledDigits" type="xs:token"
               minOccurs="0" maxOccurs="1"/>
           <xs:element name="Cell-ID" type="xs:token"
               minOccurs="0" maxOccurs="1"/>
           <xs:element name="Cause" type="tns:CauseType"
               minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="type" type="tns:PayloadType"
            use="required"/>
        <xs:attribute name="name" type="tns:EventNameType"
            use="required"/>
        <xs:attribute name="mode" type="tns:ModeType"
            use="optional" default="N"/>
     </xs:complexType>
```

```
<xs:simpleType name="PayloadType">
   <!-- The <spirits-event> will contain either a list of -->
   <!-- INDPs events or a list of userprof events -->
   <xs:restriction base="xs:string">
      <xs:enumeration value="INDPs"/>
      <xs:enumeration value="userprof"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="EventNameType">
   <xs:restriction base="xs:string">
      <!-- These are the call related events (DPs).  If the -->
      <!-- PaylaodType is "INDPs", then the value of the "name" -->
      <!-- attribute is one of these; example -->
      <!-- <spirits-event type="INDPs" name="OCI"> -->
      <xs:enumeration value="OAA"/>
      <xs:enumeration value="OCI"/>
      <xs:enumeration value="OAI"/>
      <xs:enumeration value="OA"/>
      <xs:enumeration value="OTS"/>
      <xs:enumeration value="ONA"/>
      <xs:enumeration value="OCPB"/>
      <xs:enumeration value="ORSF"/>
      <xs:enumeration value="OMC"/>
      <xs:enumeration value="OAB"/>
      <xs:enumeration value="OD"/>
      <xs:enumeration value="TA"/>
      <xs:enumeration value="TMC"/>
      <xs:enumeration value="TAB"/>
      <xs:enumeration value="TD"/>
      <xs:enumeration value="TAA"/>
      <xs:enumeration value="TFSA"/>
      <xs:enumeration value="TB"/>
      <!-- These are the non-call related events.  If the -->
      <!-- PayloadType is "user-prof", then the value of the -->
      <!-- "name" attribute is one of these; example -->
      <!-- <spirits-event type="userprof" name="LUDV"> -->
      <xs:enumeration value="LUSV"/>
      <xs:enumeration value="LUDV"/>
      <xs:enumeration value="REG"/>
      <xs:enumeration value="UNREGMS"/>
      <xs:enumeration value="UNREGNTWK"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ModeType">
   <!-- One of two values: "N"otification or "R"equest -->
   <xs:restriction base="xs:string">
```

```
            <xs:enumeration value="N"/>
            <xs:enumeration value="R"/>
         </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="CauseType">
         <xs:restriction base="xs:string">
            <xs:enumeration value="Busy"/>
            <xs:enumeration value="Unreachable"/>
         </xs:restriction>
      </xs:simpleType>
</xs:schema>
```

10. Acknowledgements

The authors are grateful to participants in the SPIRITS WG for the discussion that contributed to this work. These include J-L. Bakker, J. Bjorkner, J. Buller, J-E. Chapron, B. Chatras, O. Cleuziou, L. Conroy, R. Forbes, F. Haerens, J. Humphrey, J. Kozik, W. Montgomery, S. Nyckelgard, M. O'Doherty, A. Roach, J. Rosenberg, H. Sinnreich, L. Slutsman, D. Varney, and W. Zeuch. The authors also acknowledge Steve Bellovin, Allison Mankin and Jon Peterson for help provided on the Security section.

11. Acronyms

ACL	Access Control List
CS	Capability Set
DP	Detection Point
DTD	Document Type Definition
EDP	Event Detection Point
EDP-N	Event Detection Point "Notification"
EDP-R	Event Detection Point "Request"
IANA	Internet Assigned Numbers Authority
ICW	Internet Call Waiting
IMSI	International Mobile Subscriber Identity
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunications Union
MIME	Multipurpose Internet Mail Extensions
MS	Mobile Station (or Mobile Subscriber)
OBCSM	Originating Basic Call State Model
PIC	Point In Call
PINT	PSTN/Internet Interworking
PSTN	Public Switched Telephone Network
SCF	Service Control Function

SCP	Service Control Point
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SIP-T	SIP for Telephones
SPIRITS	Services in the PSTN/IN Requesting InTernet Services
SSF	Service Switching Function
SSP	Service Switching Point
STD	State Transition Diagram
TBCSM	Terminating Basic Call State Model
TDP	Trigger Detection Point
TDP-N	Trigger Detection Point "Notification"
TDP-R	Trigger Detection Point "Request"
TLS	Transport Layer Security
UA	User Agent
VLR	Visitor Location Register
WIN	Wireless Intelligent Network
XML	Extensible Markup Language

12. References

12.1. Normative References

[1] Slutsman, L., Faynberg, I., Lu, H., and M. Weissman, "The SPIRITS Architecture", RFC 3136, June 2001.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[4] Faynberg, I., Gato, J., Lu, H., and L. Slutsman, "Service in the Public Switched Telephone Network/Intelligent Network (PSTN/IN) Requesting InTernet Service (SPIRITS) Protocol Requirements", RFC 3298, August 2002.

[5] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

12.2. Informative References

[6] M. Unmehopa, K. Vemuri, A. Brusilovsky, E. Dacloush, A. Zaki, F. Haerens, J-L. Bakker, B. Chatras, and J. Dobrowolski, "On selection of IN parameters to be carried by the SPIRITS Protocol", Work In Progress, January 2003.

[7] Intelligent Network Capability Set 2. ITU-T, Recommendation Q.1228.

[8] Petrack, S. and L. Conroy, "The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services", RFC 2848, June 2000.

[9] Murata, M., St.Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[10] Lu, H., Faynberg, I., Voelker, J., Weissman, M., Zhang, W., Rhim, S., Hwang, J., Ago, S., Moeenuddin, S., Hadvani, S., Nyckelgard, S., Yoakum, J., and L. Robart, "Pre-Spirits Implementations of PSTN-initiated Services", RFC 2995, November 2000.

[11] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[12] Thompson, H., Beech, D., Maloney, M. and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC REC-xmlschema-1-20010502, May 2001. <http://www.w3c.org/XML/>.

[13] "Interface recommendations for intelligent network capability set 3: SCF-SSF interface", ITU-T Recommendation Q.1238.2, June 2000.

[14] Moats, R., "URN Syntax", RFC 2141, May 1997.

[15] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[16] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[17] Tim Bray, Dave Hollander, and Andrew Layman, "Namespaces in XML", W3C recommendation: xml-names, 14th January 1999, <http://www.w3.org/ TR/REC-xml-names>.

[18] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

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

[20] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text ", RFC 2047, November 1996.

Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 2048, November 1996.

Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, November 1996.

13. Contributors

Kumar Vemuri
Lucent Technologies, Inc.
2000 Naperville Rd.
Naperville, IL 60566
USA

EMail: vvkumar@lucent.com

14. Authors' Addresses

Vijay K. Gurbani, Editor
2000 Lucent Lane
Rm 6G-440
Naperville, IL 60566
USA

EMail: vkg@lucent.com

Alec Brusilovsky
2601 Lucent Lane
Lisle, IL 60532-3640
USA

EMail: abrusilovsky@lucent.com

Igor Faynberg
Lucent Technologies, Inc.
101 Crawfords Corner Rd.
Holmdel, NJ 07733
USA

EMail: faynberg@lucent.com

Jorge Gato
Vodafone Espana
Isabel Colbrand, 22
28050 Madrid
Spain

EMail: jorge.gato@vodafone.com

Hui-Lan Lu
Bell Labs/Lucent Technologies
Room 4C-607A, 101 Crawfords Corner Road
Holmdel, New Jersey, 07733

Phone: (732) 949-0321
EMail: huilanlu@lucent.com

Musa Unmehopa
Lucent Technologies, Inc.
Larenseweg 50,
Postbus 1168
1200 BD, Hilversum,
The Netherlands

EMail: unmehopa@lucent.com

15. Full Copyright Statement

Intellectual Property

Acknowledgement

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