Abstract

This document describes a framework for an Internet Intelligent Network (IIN) architecture which accommodates a service logic execution environment on a session initiation protocol (SIP) proxy, but at the same time is able to support the traditional PSTN IN-based architecture which uses a service control point (SCP) as a service execution environment. A large number of services, including traditional advanced 800 and other legacy services, as well as IN specific services (Internet Call Waiting (ICW) for ex ample) must be made available for the Internet. Implementation of such services requires a software support architecture which addresses the following issues: 1) where and how to create the service logic; 2) what kind of software instrumentation should be used to write the service logic (e.g., CPL scripts, Parlay Group API’s, or JAIN SIP API); and 3) where to run and how to trigger the service logic. Traditional PSTN architectures address these issues under the umbrella of IN. The traditional IN approach
is subject to the parameters of the PSTN architectural and traditional legacy services constraints. Though it would be preferable to avoid being locked into these constraints, it is not possible to bypass them completely (this is especially true in instances when PSTN and IN networks are inter-working). Hence the draft proposes to support both a) "IP-network-based IN" (e.g., use of script-based-methods such as CPL or CGI), and b) integration of traditional IN-based service logic (e.g., SCP-based) and SIP call control protocols.

TABLE OF CONTENTS

1. Introduction
2. Transparent Access to Traditional PSTN IN Services from SIP-Based Networks
   2.1 Direct Matching of SIP Transitions with the Standard BCMS FSMs
   2.2 Software Switch Approach
3. SPIRITS Architecture and its Relationship with IN
4. Generalized IIN model
   4.1 IIN Architecture Framework
   4.2 How Service Logic is Invoked and Connected to the Call
      4.2.1 How to Invoke a Script
      4.2.2 Protocol and Global Variables
      4.2.3 Scripts and Global Variables
5. Service Example
6. Service Creation Environment
7. Acknowledgments
8. Authors’ Addresses
9. Full Copyright Statement
10. Bibliography
11. Figures

<draft-lslutsman-sip-iin-framework-00.txt> March 2000

Framework and Requirements for the Internet Intelligent Networks (IIN) [Page 3]
1. Introduction

H.323, DOSA-SIP, and, especially, SIP protocols [1,7,9] are gathering momentum in the VoIP sector of the telecommunications industry. The need to support intelligent (software assisted) services is a must for the longevity and applicability of these protocols. Contributions towards IN support within IP-based networks have been submitted to both ITU-T and IETF [5,6]. These initial approaches mimic the traditional IN model and standards, published by the International Telecommunication Union (ITU) in its Q12 00 series of standards [8,10]. In order to make this draft self-contained we provide a short explanation of basic IN principles below.

The traditional IN relies on the Service Control Point to both 1) serve as a service logic depository, and 2) run service programs in response to stimuli from network switches. The central piece of the traditional IN is a Basic Call State Model (BCSM) which specifies two finite state machines representing call processing flows: one for the originating, and one for the terminating part of the call. In addition to a call state, called Point in Call [PIC], each part also specifies special states called Detection Points (DPs) which are prospective service entry points associated with the transitions between PICs. The PICs are best viewed as "primary" states in that DPs are directly associated with the transitions from PIC to PIC. Thus, the basic construct of the BCSM is PIC--DP--PIC, although non-DP-associated transitions (i.e., PIC--PIC) also exist. If a transition passes through a DP, the switch pauses and examines a) whether a DP is armed (i.e., active) and b) whether it meets the criteria for launching an IN query or sending a notification. Thus, depending on the type of the active DP, the switch can either a) issue a notification and transit to the next PIC, or b) suspend call processing, issue a request to the SCP, and wait for the response. When the response comes back, it may contain an instruction for how to continue with the call.

The constraints of the traditional IN model have their roots in both hardware limitations of switches and the traditional service philosophy. PSTN switches have limited "smartness" and computational capacity and therefore are not suitable to serve as a service execution environment and this creates a need for a centralized SCP. The traditional PSTN services, such as 800 or SDN-based services, have been created by service providers and provided little room for customization or enhancement by the end users. At best, end users may have limited ability to customize services such as 800-routing, by using and combining standard building blocks (e.g. branch on area code, time of day routing, percentage of call allocation to call centers). There are number of graphical-oriented tools which offer a user-friendly interface for using the building blocks, such as the Telcordia "Space" tool.

In the IP environment, however, the situation can be quite different.
because both end-systems and network-servers may have significant computational capabilities. Combined with the network-wide IP connectivity and open standardized protocols (SIP, H.323), these capabilities allow both end users and third parties to provision, design and implement new services, in a decentralized and distributive manner, with minimum interaction with a service provider.

This document describes a framework for the IIN architecture in which network nodes (end-points and servers) respond, if necessary, to call processing events by invoking service programs that may be either in the centralized SCP or within an end-point/server. The draft proposes to support both a) "IP-network-based IN" (e.g., use of script-based-methods such as CPL or CGI), and b) integration of traditional IN-based service logic (e.g., SCP-based) and SIP call control protocols.

2. Transparent Access to Traditional PSTN IN Services from SIP-Based Networks
There are a number of reasons to access traditional IN services from SIP networks which support IP telephony, including: a) embedded investments in the development of the IN and SCP applications, b) intellectual property investments in the IN architecture, and c) expediency and time-to-market in delivering VoIP services. A general architecture of such an IIN system is depicted in Figure 1. As a call progresses in the call-state model to an armed DP, the trigger database is queried, using the conditions encountered in the DP (such as the Calling and Called party addresses), to index an appropriate application. To implement this architecture one has to make the SCP ‘believes’ that the underlying SIP network nodes behave pretty much as do PSTN switches. This may be done in two slightly different ways, which are discussed in subsequent sections.

2.1 Direct Matching of SIP Transitions with the Standard BCMS FSMs
To illustrate this concept, let us consider a successful SIP call scenario, which is constituted by two SIP transactions: INVITE and ACK. The transition diagrams for both client and server are provided in [1, Figures 12,13]. They depict the same kind of functionality as the BCSM originating and terminating FSMs, but the PICs are different. Gurbani [6] shows that by adding "pass-through" states along with DPs, SIP transition diagrams may be mapped to the traditional BCSM.

2.2 Software Switch Approach
Haerens [5] suggests a slightly different approach. The concept of the ‘softSSF’ (see Figure-2) is introduced and acts as an overlay between the IP telephony call control and the Intelligent Network layer provided by the intelligent network application part (INAP)
The difference between the two approaches is in their implementation. The approach in Section 2.1 assumes that additional states and appropriate DPs are explicitly added into the SIP protocol, while the approach in Section 2.2 lets the 'softSSF' interpret the SIP states and translate them into the appropriate PICs of the BCSM. None of the above approaches treat the case of a 'stateful' SIP proxy.

3. SPIRITS Architecture and its Relationship with IN

The Internet support for services that originated in the PSTN is the subject of the SPIRITS WG. The SPIRITS charter relates to the IP/IN issues discussed in this draft and to some degree illustrates the limitations of the approach we described in Section 2. A simplified SPIRITS architecture is given in Figure-3. SPIRITS services are invoked when a request message from a SPIRITS Client (located in the IN Service Control Point [SCP] or Service Node [SN]) arrives on interface A to the SPIRITS Server over the IP network. The request from the SPIRITS client, in turn, is caused by a query from the Central Office, when the call processing in the switch progresses to a SPIRITS "significant" trigger (such as a Termination Attempt Trigger (TAT)). The information discovered at the trigger will be passed to the SPIRITS server. While the SPIRITS server will carry out the PSTN request (acting as a virtual service execution environment), the call processing in the Central Office would be postponed until the SPIRITS client (SCP/SN) grants permission to continue. The SPIRITS client, in turn, has to wait for instructions from the SPIRITS server (which arrive on Interface B). The messages the SPIRITS server may send to its client include the following: a) upon completion of the IP portions of the service, the SPIRITS server may have to return control to the SCP to resume the execution of the PSTN service logic; and b) upon receiving the initial request, the SPIRITS server may want to ask the SCP to arm certain DPs in the switch to complete the service. One way to understand the end-to-end SPIRITS architecture is to view the SPIRITS client as an IP-based SCP, which also provides the service execution environment for the IP portion of the service. The end-user PC is treated as a subordinate "switch" while the SPIRITS Client is treated as a "peer".

In an internet call waiting (ICW) example of Figure 3, the following sequence of steps will occur:

a) the end-user is logged on to the internet over a PPP connection,

b) a call arrives trying to access the end-user’s telephone over the
c) the central office recognizes this condition with a TAT trigger in its call processing logic and makes an ICW TCAP query to the SCP/SPRITS client,
d) the SCP, which also functions as the SPRITS client provides necessary information on the condition to the SPRITS server (information B), which instructs the end-user PC (information C) to display ICW information and options (e.g., accept call, reject call, advance call to voice-mail) on the end-user’s PC,
e) the end-user selects an option, which is reported to the SPRITS server,
f) the SPRITS server informs the SPRITS client (information B) which informs the central office (via TCAP).

4. Generalized IIN model

In general, the IIN should provide programming support for VoIP services for both "pure" end-to-end SIP networks and "mixed" networks (e.g. interworking with PSTN, H.323, etc.). In this draft we focus initially on pure SIP networks. The major elements of the IIN model are a) flexible architecture; b) how service logic is invoked and connected to the call process; c) the service execution environment (where services are run) and how call control elements (SIP client/server or SIP proxy) communicate with the running service programs; d) service logic imposed call routing; e) the service creation environment.

4.1 IIN Architecture Framework

The suggested IIN architecture (Figure 4) provides both the transparent support of the traditional IN services from SIP networks and the script based service logic. Access to the SCP services may be accomplished with either TCAP/INAP messaging or via an API (Parlay Group, SIP [11] or Java Telephony APIs). When APIs are used, a transaction layer (TCAP-like capability) is needed to correlate requests and responses and link together related requests. The script approach is gaining momentum in the VoIP industry. The main idea of this approach is to run scripts directly on a call control element (CCE) for both simplicity and performance reasons. Within the script approach there are at least two possibilities: Call Processing Language [2] and Common Gateway Interface (CGI)[3]. CPL is a conditions-actions pair type of language (very similar to AWK). Typically, a CPL script is associated with a particular telephony address(s) (normally called the party address). CPL scripts will be typically used to replace the user location functionality of a SIP proxy. SIP CGI provides an interface and set of primitives for implementing services on SIP servers. CGI is programming language independent, thus CGI scripts may be written in a variety of programming languages including Perl, C++, Visual Basic, etc. CGI is more powerful but less
efficient than CPL.

4.2 How Service Logic is Invoked and Connected to the Call

In order to run a script we must deal with three things: 1) identify and invoke an appropriate script; 2) make sure that the protocol keeps updating global variables that are used by the script (specifically for CPL these variable are used when conditions are evaluated); 3) make sure that the protocol makes appropriate decisions based on the global variables updated by the script.

4.2.1 How to Invoke a Script:

The Triggers database may be used to invoke an appropriate script(s). For performance reasons, we expect this database to reside on the CCE whenever feasible. Each record in the database has the following format (Trigger,S1, S2,..Sn), where S1, S2,..Sn are scripts listed in the desirable order of invocation and "Trigger" = ("Call-Leg", "State"). "Call-Leg" is a SIP call leg as defined in [1]. The use of wild-card notations is allowed in "Trigger", thus (*,<sip:catalogordert@sears.org>,*) is a valid trigger that matches any call directed to catalogordert@sears.org. "State" is a state from the appropriate SIP transition diagram (in other words a SIP PIC). For example, if the CCE in question is a SIP Client, "State" may take any one of the following values: Initial, Calling, Ringing, and Completed [1].

4.2.2 Protocol and Global Variables

The global variables mentioned above should be defined and provisioned according to the current capability set of the IIN. This is true for both global variables modifiable by the protocol and to global variables modifiable by scripts. It is a responsibility of the protocol to set call related global variables (such as the call arrival time). These variables are intended to evaluate conditions used by scripts.

4.2.3 Scripts and Global Variables

Certain global variables are allowed to be modified by scripts. These variables may be used for two purposes: 1) to pass information from one script to another (for example a variable modified by one script may be used to evaluate a condition in another one); 2) to provide feedback to the protocol, for example, a script running on a "forking" proxy may instruct this proxy to forward a response upstream or act as a virtual client and terminate it.
5. Service Example

The service design for the IIN model may be different from the design for a traditional SCP-based IN. Let us illustrate that by a simple example. In this example, 800-calls directed to catalogorder@sears.org should be delivered to one of the following two destinations, warehouse1@sears.org, and warehouse2@sears.org, in a ratio of 6:4. One way to implement this is with two scripts (see Figure 5a): the first script should be run by any SIP-client (to reduce a number of routing hops) that receives a call to catalogorder@sears.org. The purpose of this script is to direct these calls to a dedicated sears-proxy. This proxy will run a second script which performs the final routing by analyzing two global variables, cnt1 and cnt2, which are equal to the number of calls that had been sent to warehouse1 and warehouse2, respectively. Upon routing the call, the script will modify either cnt1 or cnt2.

In Figure 5b, we illustrate the same functionality, but in this case the SSF in the SIP CCE’s sends the call requests to the SCP, which determines the routing address to send the call based on the cnt1 and cnt2 information held in the SCP.

6. Service Creation Environment

The model does not impose any specific requirements on the SCE, only that it should be capable of producing scripts and delivering them to CCEs. The end user can write scripts directly using CPL and CGI or use a software tool which provides a user-friendly graphical interface and automatically generate scripts.

In summary, as depicted in Figure 4, the draft proposes to support both a) "IP-network-based IN" (e.g., use of script-based-methods such as CPL or CGI), and b) integration of traditional IN-based service logic (e.g., with SCP-, SCE-, and SEE-based capabilities) and SIP call control protocols.

7. Acknowledgments

We would like to thank Igor Faynberg for his insights into IN standards and Doris Lebovits, for her comments.

8. Authors’ Addresses

Gerald Ash

<draft-lslutsman-sip-iin-framework-00.txt> March 2000

Framework and Requirements for the Internet Intelligent Networks (IIN) [Page 9]
9. Full Copyright Statement

Copyright (C) The Internet Society (1998). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works.

However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
10. Bibliography


11. Figures
Figure 1. General Architecture for Accessing IN Services

Legend:
O_CCE: Originating Call Control Element (such as SIP Client) that runs/emulates O_BCSM
T_CCE: Terminating Call Control Element (Such as SIP Server) that runs/emulate T_BCSM
P_CCE: Intermediate Call Control Element (such as SIP "stateless proxy or H.323 Gateway)

Figure 2. Functional Architecture Based on Softswitch
Figure-3. SPIRITS Interface Architecture

SPIRITS End-User (PC) <-> SPIRITS SERVER B --> SPIRITS CLIENT-SCP A

END-User Telephone ---- Central OFFICE ---- TCAP

Figure-4. Suggested IIN Architecture

Service Creation Environment

SCP
Service Execution Environment

TCAP/INAP | API
SoftSwitch LAYER | Transaction Layer

Figure-5a. Service Example Using Scripts

<draft-lslutsman-sip-iin-framework-00.txt> March 2000

Framework and Requirements for the Internet Intelligent Networks (IIN) [Page 3]
Figure-5b. Service Example Using SCP

SCP: updates numbers of calls to warehouses and determine the destination

Query+ * Destination

SIP Client    SIPProxy    SIP Server

User End System    Warehouse