A Model for Service Invocation in SIP using ServiceCodes

STATUS OF THIS MEMO

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as work in progress.

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

Although the Session Initiation Protocol (SIP) needs no special mechanism to enable network services and applications, in complex architectures, feature composition and service invocation become unmanageable without a mechanism for specifying the services that are associated with routes. This draft introduces distributed application architectures that span multiple administrative domains and proposes the use of ServiceCodes in SIP to allow service providers to compose chains of features across application servers.

1. Introduction

The Session Initiation Protocol (SIP, [2543]) is a protocol that allows network elements to discover one another in order to exchange contextual information about a session (usually real-time media transmission) they would like to share. As SIP is frequently used for
Internet telephony service, service providers are grappling with the problem of offering familiar telephone features to end users. Recent work has demonstrated that while some features are best implemented within SIP endpoints themselves, others are more suited to centralized implementation at nodes owned by service providers; this has given rise to a class of SIP network elements known as application servers (or AppServers for short).

Application servers (which are described in detail below) are repositories of service logic to which an Internet telephony service provider can route calls. In simple network architectures, SIP accommodates this model of service invocation without modification. However, as the architecture of the networks involved grows more complex, certain problems arise which require a clarification within signaling of which services should be executed.

To that purpose, this draft illustrates how ServiceCodes (described in [SIP-SC]) can be used within the Request-URI of a SIP message to denote which service an application server should execute. It also recommends design methodologies for network implementers and authors of applications that will facilitate feature composition and thereby enhance user experience.

This draft is agnostic to the means by which ServiceCodes are provisioned in the service provider network; proxy servers can be provisioned manually with ServiceCodes associated with specific routes. However, the full power of ServiceCodes is only realized when dynamic route provisioning through a protocol like TRIP (see [TRIP] and [TRIP-L]) is utilized.

2. Application Servers

The concept of an application server is a fairly common one in SIP. Service logic is commonly understood to be implemented by applications that execute on these application servers. AppServers are not formally defined in the SIP standard; however, their behavior often replicates that of a redirect server, a proxy/location server, or a user agent.

An AppServer usually resides on a general purpose computer. It may be composed of a SIP interface coupled tightly with some particular logic, or the logic may be a modular component programmatically segregated from the SIP interface itself. In this latter case, an AppServer would support one or more authoring systems that provide a framework in which modular logic can be written.

An authoring system constitutes a set of tools that allow a service provider to create an instance of service logic that will operate on
an AppServer. These tools range from graphical Service Creation Environments to scripting languages to bare-bones Application Programming Interfaces. Common authoring systems include SIP CGI (see [CGI]) and CPL (see [CPL]).

Deployment of AppServers is becoming common today in Internet Telephony Service Provider (ITSP) networks; however, more commonly SIP Application Service Providers (ASPs), who not only host AppServers but also author applications, offer network access to their AppServers to ITSPs. This decomposition is desirable for ITSPs in so far as application development and management can be outsourced, allowing for more rapid introduction of services to the network.

3. Two Service Invocation Problems

SIP needs no special mechanism to invoke service logic stored in an AppServer. Commonly, a SIP network element elects to route a session to an AppServer just as it might route a session to a conventional endpoint - the AppServer is then responsible for dealing with the call as its services dictate. This model is adequate for instantiating features in a simple network, but, as network architectures become more complicated, some problems arise. Notably, once ASPs offer an ITSP access to AppServers where features are instantiated, service invocation becomes problematic, especially when multiple features (a 'feature chain') may be required in the course of routing a particular session.

Consider a case in which a proxy server in an ITSPs network has been provisioned with a route, which we'll call R1, that points to an AppServer in an ASP network. By 'route' we signify here a mapping from an element extracted from a SIP INVITE message that is used to route the call (or 'routable element' for short) to a SIP endpoint that is the proper destination for the INVITE. So, to take a simple example, a prefix matching route for telephone numbers (which appear in the Request-URI of an INVITE as tel URLs) to a particular AppServer might be represented as follows:

+17208881 -> appserver.asp.net:5060

When call arrives at the proxy server which meets R1's criteria (i.e. it matches the prefix '+17208881'), the proxy server will forward the INVITE to the AppServer.

A problem arises, however, when the AppServer in question manages more than one application. The AppServer must use some criteria (probably something similar to the routes described above) to
determine which applications should service any given call. What if
the ITSP wanted only one particular application operated by the ASP
to be applied to the call? One ASP might have favorable rates and
policies for, say, Local Number Portability (LNP) dips, but another
ASP is a better provider of freephone translation; if both services
need to be invoked for a single call, how can the ITSP prevent either
of the ASPs from performing both services when their respective
AppServers are handed control of the call?

Two issues arise from this example.

First, when an ITSP provisions a route in its network, it is not
possible for the ITSP to express, when applicable, which service it
is attempting to invoke. Therefore, the ASP must attempt to analyze
the routable elements within the INVITE itself, and to choose
services appropriately. This draft proposes the use of ServiceCodes
in SIP to allow the network elements of an ITSP to identify one or
more particular services which should be executed by an ASP.

Second, if multiple services are to be enacted for a single session,
there needs to be a single point of feature composition which
dictates the order in which services will be executed and identifies
the AppServers that are to be awarded the right to
perform a service for a particular session. This draft proposes a
logical 'feature proxy' function of an ITSP network that provides
such a single point of composition. In order to allow such
composition to take place, applications implemented by ASPs should be
designed with composition in mind, when possible.

4. Using ServiceCodes in SIP

No extensions to SIP are required to make use of ServiceCodes.
Rather, an optional URI parameter is proposed for use within the
Request-URI of INVITEs that bear ServiceCodes.

The parameter will use the token 'sc=', followed by the ServiceCode
in question. The format and semantics of ServiceCodes are
authoritatively described in [TRIP-SC]; in brief, ServiceCodes are
IANA-registered two-byte numbers. In SIP, ServiceCodes should be
represented in ASCII (e.g. '0' to '65535').

For example:

INVITE tel:+18775550000;sc=1 SIP/2.0

ServiceCodes can appear multiple times within the Request-URI if the
ITSP desires the invocation of more than one service.
5. Centralizing Feature Composition

An ITSP has a set of subscribers to whom it provides 'Internet telephony' service. Generally, such service implies that the ITSP is responsible for tracking the presence of subscribers (who might, for example, be sending REGISTER messages to ITSP registrar proxies) and subsequently routing appropriate calls to available subscribers. 'Internet telephony' service might also entail the presence of local outbound proxies to which subscribers send their outbound SIP calls.

Services might target SIP messages associated, from the subscriber's perspective, with either inbound or outbound calls. Therefore, any single point of feature composition must serve as both a location server (to which inbound messages for a subscriber are sent) and a local outbound proxy (the target of calls from subscribers). In fact, these two resources need only be consistent in provisioning to serve as a point of feature composition.

The term "feature proxy" will be used in this document to denote the logical function, in the ITSP network, of a location server and local outbound proxy that can be provisioned with routes accompanied by ServiceCodes.

In order for a feature proxy to compose multiple features together effectively, the feature proxy, rather than any other element in the ITSP or ASP network, must be solely responsible for selecting the services that will execute and the order in which these services resolve. This control ultimately must derive from the implementations developed by ASPs for applications.

6. Designing Services for Service Composition

Many (if not most) services operate on session signaling; that is, the application analyzes certain elements of an INVITE and subsequently either sends a response message back to the originator or, potentially, modifies the signaling and sends it to a new destination. Some services may also require access to session media, in which case the service must effectively terminate the call, although it may re-originate the call if necessary.

In order for services to be composed at the feature proxy, the general agreement between ASP and ITSP must specify that if an AppServer does not intend to complete the call with any of the services specified by the ITSP in the INVITE, it must return control of the call to ITSP. An AppServer's options, then, for handling a call are as follows:
o Redirect - Modifies only the Request-URI (via 3xx message)

o Spiral - Modifies arbitrary elements of the INVITE (retains idempotency)

o Terminate - Responds with either an OK or >300 code (may re-originate)

While these three alternatives may initially seem limiting, really, these are the tools commonly used to implement applications today. The primary distinction is that AppServers often choose to proxy the call to a destination based on its own internal routing logic rather than spiraling the call back to the server from which it received the call.

It could be argued that nothing guarantees, in this sort of architecture, that an ASP will play by the rules, as it were, and allow features to be composed at the feature proxy. The underlying incentive for an ASP to play by the rules is that the end user will expect feature composition and will not put up with services that prohibit it. If an end user subscribes to a new call-waiting service, and because of that subscription their voicemail suddenly doesn't work any more, then (provided there are alternatives) the end user will most likely shop for call-waiting services elsewhere.

6.1 Redirect

Some services may need only to modify the Request-URI of an INVITE. In these cases, a service may want to use a redirection (3xx) SIP message to express the new Request-URI for this session.

Once an AppServer has returned a redirection for a call, it will no longer be in the path of signaling, and no further state needs to be kept for the call once an acknowledgment to the redirection has been received. Redirection services thus tend to be fast and scalable.

When a redirection is sent, the new Request-URI for the INVITE is contained in the Contact header. A service can send multiple Contact headers, if desired; usually these Contacts will be searched in parallel, although this is highly dependent on the ITSP network architecture.

6.2 Spiral

If modification of headers (rather than the Request-URI) is required for a service, then the ASP must return an INVITE to the ITSP network, with modifications, rather than redirecting with a 302; this practice is known as 'spiraling' the request.

If the spiraling is employed, more or less any aspect of the INVITE
can be modified that does interfere with idempotency (i.e. the To, From, Call-ID, and CSeq headers) and which does not give rise to message routing failures.

Unlike redirection, spiraling forces the AppServer to be in the path of responses for the signaling (since the AppServer must add a Via header). If this is desirable for the implementation of the service, the AppServer may add itself to the Record-Route in the INVITE in order to be in the path of new requests for the call.

Of course, when a request is spiraled, it is not possible to express multiple route choices in a single INVITE; if the service requires multiple choices to be explored, each choice must be sent in its own INVITE, and the AppServer will field all responses to one such INVITE. If a rejection is received by the AppServer it may send a new INVITE that explores any alternative route choice for the call. Caching of route choices is not possible when a request is spiraled.

6.3 Terminate

For a variety of services, an ASP may find it useful to terminate a call provisionally at an AppServer which has an associated media server. While the call is 'parked' in this fashion, the media server may play announcements to the caller, collect tones, or perform similar actions.

To park a call, an AppServer must return a 200 OK, providing an ordinary termination for the call. The media characteristics in the SDP within the 200 should reflect an available resource in the media server which the AppServer controls.

Once the call has been parked, the service may decide to send the call to a final destination (possibly after some interaction with the caller through the audio channel). At this time, the service behaves as a B2BUA (back-to-back user agent).

For the purposes of feature composition no services can follow the invocation of a service that results in call completion. If a complete is actually a B2BUA, and the B2BUA spins a new leg of the call back to the feature proxy, it will be treated as a new call with a new feature chain.

7. Associating ServiceCodes with Routes

When a feature proxy sends an INVITE with a ServiceCode, it will do so on the basis of provisioning. As was illustrated above, an ITSP's feature proxies will be provisioned with routes that direct calls matching certain criteria to the ASP's AppServers. Those routes may
be provisioned either manually or, through a protocol like TRIP, dynamically. If the use of ServiceCodes is desired, then a ServiceCode corresponding to each such route must be supplied.

In the manual case, an ASP will presumably provide the ITSP with information about available services which ITSP operations personnel will use to configure feature proxies appropriately, including ServiceCodes. As networks grow large and diverse, this business practice may result in an undue amount of overhead when new services are created. Therefore, in large networks dynamic registration of services with ServiceCodes is recommended.

Following the example given above, the AppServer (in an ASP network) propagates three routes associated with different ServiceCodes to the location server (in ITSP network, which is the 'owner' of the end user).

The location server is merely a front-end to provision the logical feature proxy; it uses an internal policy matrix (see [TRIP-SC]) to prioritize the routes among the total state of the feature server, and subsequently provisions its PS with appropriate data for these three routes.

Note that in this model, particular applications associated with an ASP network can dynamically register and de-register at the ASP's discretion - this ultimately is a very powerful tool for the ASP in so far as it facilitates load balancing, maintenance, and similar operational functions.

7. Conclusion

The interplay of ServiceCodes in SIP and TRIP ultimately allows an extremely powerful model for service invocation. Consider that a subscriber belonging to a particular ITSP could, through navigating a web page or similar construct operated by an ASP, electronically order a service that would automatically register itself with the ITSP network via TRIP. This tools benefit all of the players in the Internet telephony community (ITSP, ASP and subscriber) equally.

8. To Do

- Edge policies associated with ServiceCodes - ITSPs may wish to implement particular policies at their network edges in order to ensure that only permitted services are invoked in remote networks, and moreover that remote networks behave in the expected fashion when they are executing a service.

- Add some diagrams
9. References


10. Author's Address

Jon Peterson
1025 Eldorado Blvd
Broomfield, CO
jon.peterson@level3.com

Full Copyright Statement

Copyright (c) The Internet Society (2000). 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.