SIPPING
Internet-Draft
Intended status: Best Current Practice
Expires: November 8, 2007
J. Rosenberg
Cisco
May 7, 2007

Identification of Communications Services in the Session Initiation Protocol (SIP)

draft-rosenberg-sipping-service-identification-02

Status of this Memo

Copyright Notice

Abstract

This document considers the problem of service identification in the Session Initiation Protocol (SIP). Service identification is the process of determining the user-level use case that is driving the signaling being utilized by the user agent. While seemingly simple, this process is quite complex, and when not addressed properly, can lead to fraud, interoperability problems, and stifling of innovation.

This document discusses these problems and makes recommendations on how to address them.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [2] defines mechanisms for initiating and managing communications sessions between agents. SIP allows for a broad array of session types between agents. It can manage audio sessions, ranging from low bitrate voice-only up to multi-channel hi fidelity music. It can manage video sessions, ranging from small, "talking-head" style video chat, up to high definition multipoint video conferencing, to low bandwidth user-generated content, up to high definition movie and TV content. SIP endpoints can be anything - adaptors that convert an old analog telephone to Voice over IP (VoIP), dedicated hardphones, fancy hardphones with rich displays and user entry capabilities, softphones on a PC, buddylist and presence applications on a PC, dedicated videoconferencing peripherals, and speakerphones.

This breadth of applicability is SIPs greatest asset, but it also introduces numerous challenges. One of these is that, when an endpoint generates a SIP INVITE for a session, or receives one, that session can potentially be within the context of any number of different use cases and endpoint types. For example, a SIP INVITE with a single audio stream could represent a Push-To-Talk session between mobile devices, a VoIP session between softphones, or audio-based access to stored content on a server.

These differing use cases have driven implementors and system designers to seek techniques for service identification. Service identification is the process of determining and/or signaling the specific use case that is driving the signaling being generated by a user agent. At first glance, this seems harmless and easy enough. It is tempting to define a new header, "Service-ID", for example, and have a user agent populate it with any number of well-known tokens which define what the service is. This information could then be consumed for any number of purposes.

However, as this document will demonstrate, service identification is a very complex and difficult process, and can very easily lead to fraud, systemic interoperability failures, and a complete stifling of the innovation that SIP was meant to achieve.

Section 3 begins by defining a service and the service identification problem. Section 4 gives some concrete examples of services and why they can be challenging to identify. Section 5 explores the ways in which a service identification can be utilized within a network. Next, Section 6 discusses the key architectural principles of service identification, and how explicit service identifiers can lead to fraud, interoperability failures, and stifling of service innovation.

2. Terminology

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 RFC 2119 [1].

3. Services and Service Identification

The problem of identifying services within SIP is not a new one. The problem has been considered extensively in the context of presence. In particular, the presence data model for SIP [3] defines the concept of a service as one of the core notions that presence describes. Services are described in Section 3.3 of RFC 4479, which has this to say on the topic:

> 3.3. Service
>
> Each presentity has access to a number of services. Each of these represents a point of reachability for communications that can be used to interact with the user. Examples of services are telephony (that is, traditional circuit-based telephone service), push-to-talk, instant messaging, Short Message Service (SMS), and Multimedia Message Service (MMS).
>
> It is difficult to give a precise definition for service. One reasonable approach is to model each software or hardware agent in the system as a service. If a user starts a softphone application on their PC, then that represents a service. If a user has a videophone device, then that represents another service. This is effectively a physical view of services. This definition, however, starts to fall apart when a service is spread across multiple software agents or devices. For example, a SIP URI representing an address-of-record can be routed to a softphone or a videophone, or both. In that case, one might attempt instead to define a service based on its address on the network. This definition also falls apart when modeling devices or applications that receive calls and dispatch them to different "helpers" based on potentially complex logic. For example, a cellular telephone might house multiple SIP applications, each of which can "register" different handlers based on the method or even body type of the request. Each of those applications or handlers can rightfully be considered a service, but it doesn't have an address on the network distinct from the others.
>
> Because of this inherent difficulty in precisely defining a service, the data model doesn't try to constrain what can be considered a service. Rather, anything can be considered a service so long as it

exhibits a set of key properties defined by this model. In particular, each service is associated with characteristics that identify the nature and capabilities of that service, with reach information that indicates how to connect to the service, with status information representing the state of that service, and relative information that describes the ways in which that service relates to others associated with the presentity.

As a consequence, in this model, services are not explicitly enumerated. There is no central registry where one finds identifiers for each service. Consequently, each service does not have a single "service" attribute with values such as "ptt" or "telephony". That doesn't mean that these consolidated monikers aren't useful; indeed, they represent an essential summary of what the service is. Such summarization is useful in creating icons that allow a user to choose one service over another. A watcher is free to create such summarization information from any of the information associated with a service. The reach information often provides valuable information for creating such a summarization. Oftentimes, the scheme of the URI is synonymous with the view of what a service is. An "sms" URI [14] clearly indicates SMS, for example. For some URIs, there may be many services available, for example, SIP or tel [15], in which case the scheme is less meaningful as a way of creating a summary. The reach information could also indicate that certain application software has to be invoked (such as a videogame), in which case that aspect of the reach information would be useful for generating an iconic representation of the game.

Essentially, the service is the user-visible use case that is driving the behavior of the user-agents and servers in the SIP network. Being user-visible means that there is a difference in user experience between two services that are different. This user experience can be based on different media types (an audio call vs. a video chat), different content within a particular media type (stored content, such as a movie or TV session), different devices (a wireless device for "telephony" vs. a PC application for "voice-chat"), different user interfaces (a buddy list view of voice on a PC application vs. a software emulation of a hard phone), different communities that can be accessed (voice chat with other users that have the same voice chat client, vs. voice communications with any endpoint on the PSTN), or different applications that are invoked by the user (manually selecting a push-to-talk application from a wireless phone vs. a telephony application).

In some cases, there is very little difference in the underlying technology that will support two different services, and in other cases, there are big differences. However, for purposes of this

discussion, the key definition is that two services are distinct when there is a perceived difference by the user in the two services.

This leads naturally to the desire to perform service identification. Service identification is defined as the process of (1) determination of the underlying service which is driving a particular signaling exchange, (2) associating that service with some kind of moniker, and (3) attaching that moniker to a signaling message (typically a SIP INVITE), and then utilizing it for various purposes within the network. Service identification can be done in the endpoints, in which case the UA would insert the moniker directly into the signaling message based on its awareness of the service. Or, it can be done within a proxy in the network, based on inspection of the SIP message, or based on hints placed into the message by the user.

4. Example Services

It is very useful to consider several example services, especially ones that appear difficult to differentiate from each other.

4.1. IPTV vs. Multimedia

IP Television (IPTV) is the usage of IP networks to access traditional television content, such as movies and shows. SIP can be utilized to establish a session to a media server in a network, which then serves up multimedia content and streams it as an audio and video stream towards the client. Whether SIP is ideal for IPTV is, in itself, a good question. However, such a discussion is outside the scope of this document.

Consider multimedia conferencing. The user accesses a voice and video conference at a conference server. The user might join in listen-only mode, in which case the user receives audio and video streams, but does not send.

These two services - IPTV and multimedia conferencing, clearly appear as different services. They have different user experiences and applications. A user is unlikely to ever be confused about whether a session is IPTV or multimedia conferencing. Indeed, they are likely to have different software applications or endpoints for the two services.

However, these two services look remarkably alike based on the signaling. Both utilize audio and video. Both could utilize the same codecs. Both are unidirectional streams (from a server in the network to the client). Thus, it would appear on the surface that there is no way to differentiate them, based on inspection of the

signaling alone.

4.2. Gaming vs. Voice Chat

Consider an interactive game, played between two users from their mobile devices. The game involves the users sending each other game moves, using a messaging channel, in addition to voice. In another service, users have a voice and IM chat conversation using a buddy list application on their PC.

In both services, there are two media streams - audio and messaging. The audio uses the same codecs. Both use the Message Session Relay Protocol (MSRP) [5]. In both cases, the caller would send an INVITE to the AOR of the target user. However, these represent fairly different services, in terms of user experience.

4.3. Configuration vs. Pager Messaging

The SIP MESSAGE method [8] provides a way to send one-shot messages to a particular AOR. This specification is primarily aimed at Short Message Service (SMS) style messaging, commonly found in wireless phones. Receipt of a MESSAGE request would cause the messaging application on a phone to launch, allowing the user to browse message history and respond.

However, MESSAGE is sometimes used for the delivery of content to a device for other purposes. For example, some providers use it to deliver configuration updates, such as new phone settings or parameters, or to indicate that a new version of firmware is available. Though not designed for this purpose, MESSAGE gets used since, in existing wireless networks, SMS are used for this purpose, and MESSAGE is the SIP equivalent of SMS.

Consequently, the MESSAGE request sent to a phone can be for two different services. One would require invocation of a messaging app, whereas the other would be consumed by the software in the phone, without any user interaction at all.

5. Using Service Identification

It is important to understand what the service identity would be utilized for, if known. The discussions in Section 4 give some hints to the possible usages. Here, we explicitly discuss them.

5.1. Application Invocation in the User Agent

In some of the examples above, there were multiple software applications running within a single user agent. When an incoming INVITE or MESSAGE arrives, it must be delivered to the appropriate application software. When each service is bound to a distinct software application, it would seem that the service identity is needed to dispatch the message to the appropriate piece of software. This is shown in Figure 2.

```
                         +---------------------------------+
                         |                                 |
                         | +-------------+ +-------------+ |
                         | |     UI      | |     UI      | |
                         | +-------------+ +-------------+ |
                         | +-------------+ +-------------+ |
                         | |             | |             | |
                         | |  Service 1  | |  Service 2  | |
                         | |             | |             | |
                         | +-------------+ +-------------+ |
                         | +-----------------------------+ |
                         | |                             | |
                         | |             SIP             | |
                         | |            Layer            | |
                         | |                             | |
                         | +-----------------------------+ |
                         |                                 |
                         +---------------------------------+

                                    Physical Device
```

Figure 2

The role of the SIP layer is to parse incoming messages, handle the SIP state machinery for transactions and dialogs, and then dispatch request to the appropriate service. For the example services in Section 4.2, an incoming INVITE for the gaming service would be delivered to the gaming application software. An incoming INVITE for the voice chat service would be delivered to the voice chat application software. For the examples in Section 4.3, a MESSAGE request for user to user messaging would be delivered to the messaging or SMS app, and a MESSAGE request containing configuration data would be delivered to a configuration update application.

5.2. Application Invocation in the Network

Another usage of a service identifier would be to cause servers in the SIP network to provide additional processing, based on the

service. For example, an INVITE issued by a user agent for IPTV would pass through a server that does some kind of content rights management, authorizing whether the user is allowed to access that content. On the other hand, an INVITE issued by a user for multimedia conferencing would pass through a server providing "traditional" telephony features, such as outbound call screening and call recording. It would make no sense for the INVITE associated with IPTV to have outbound call screening and call recording applied, and it would make no sense for the multimedia conferencing INVITE to be processed by the content rights management server. Indeed, in these cases, its not just an efficiency issue (invoking servers when not needed), but rather, truly incorrect behavior can occur. For example, if an outbound call screening application is set to block outbound calls to everything except for the phone numbers of friends and family, an IPTV request that gets processed by such a server would be blocked (as its not targeted to the AOR of a friend or family member). This would block a user's attempt to access IPTV services, when that was not the goal at all.

Similarly, a MESSAGE request from Section 4.3 might need to pass through a message server for filtering when it is associated with chat, but not when it is associated with config. Consider a filter which gets applied to MESSAGE requests, and that filter runs in a server in the network. The filter operation prevents user Joe from sending messages to user Bob that contain the words "stock" or "purchase", due to some regulations that disallow Joe and Bob from discussing stock trading. However, a MESSAGE for configuration purposes might contain an XML document that uses the token "stock" as some kind of attribute. This configuration update would be discarded by the filtering server, when it should not have been.

5.3. Network Quality of Service Authorization

The IP network can provide differing levels of Quality of Service (QoS) to IP packets. This service can include guaranteed throughput, latency, or loss characteristics. Typically, the user agent will make some kind of QoS request, either using explicit signaling protocols (such as RSVP) or through marking of Diffserv value in packets. The network will need to make a policy decision based on whether these QoS treatments are authorized or not. One common authorization policy is to check if the user has invoked a service using SIP that they are authorized to invoke, and that this service requires the level of QoS treatment the user has requested.

For example, consider IPTV and multimedia conferencing as described in Section 4.1. IPTV is a non-real time service. Consequently, media traffic for IPTV would be authorized for bandwidth guarantees, but not for latency or loss guarantees. On the other hand,

multimedia conferencing is real time. Its traffic would require bandwidth, loss and latency guarantees from the network.

Consequently, if a user should make an RSVP reservation for a media stream, and ask for latency guarantees for that stream, the network would like to be able to authorize it if the service was multimedia conferencing, but not if it was IPTV. This would require the server performing the QoS authorization to know the service associated with the INVITE that set up the session.

5.4. Service Authorization

Frequently, a network administrator will want to authorize whether a user is allowed to invoke a particular service. Not all users will be authorized to use all services that are provided. For example, a user may not be authorized to access IPTV services, whereas they are authorized to utilize multimedia processing. A user might not be able to utilize a multiplayer gaming service, whereas they are authorized to utilize voice chat services.

Consequently, when an INVITE arrives at a proxy in the network, the proxy will need to determine what the requested service is, so that the proxy can make an authorization decision.

5.5. Accounting and Billing

Service authorization and accounting/billing go hand in hand. Presumably, one of the primary reasons for authorizing that a user can utilize a service is that they are being billed differently based on the type of service. Consequently, one of the goals of a service identity is to be able to include it in accounting records, so that the appropriate billing model can be applied.

For example, in the case of IPTV, a service provider can bill based on the content (US $5 per movie, perhaps), whereas for multimedia conferencing, they can bill by the minute. This requires the accounting streams to indicate which service was invoked for the particular session.

5.6. Negotiation of Service

In some cases, when the caller initiates a session, they don't actually know which service will be utilized. Rather, they might like to offer up all of the services they have available to the called party, and then let the called party decide, or let the system make a decision based on overlapping service capabilities.

As an example, s user can do both the game and the voice chat service

of Section 4.2. They initiate a session to a target AOR, but the devices used by that user can only support voice chat. Consequently, voice chat gets utilized for the session.

5.7. Dispatch to Devices

When a user has multiple devices, each with varying capabilities in terms of service, it is useful to dispatch an incoming request to the right device based on whether the device can support the service that has been requested.

For example, if a user initiates a gaming session with voice chat, and the target user has two devices - one that can support the gaming service, and the other that cannot, the INVITE should be dispatched to the device which supports the gaming session.

6. Key Principles of Service Identification

In this section, we describe some of the key principles of performing service identification.

6.1. Services are a By-Product of Signaling

Almost always, the first solution that people consider is to add some kind of field to the signaling messages which indicates what the service is. This field would then be inserted by the user agent, and then can be used by the proxies and other user agent as a service identifier.

This approach, however, misses a key point, which cannot be stressed enough, and which represents the core architectural principle to be understood here:

> A service is the by-product of the signaling - the effects of the signaling message once launched into the network. The service identity is therefore always derivable from the signaling without additional identifiers.

When a user sends an INVITE request to the network, and targets that request at an IPTV server, and includes SDP for audio and video streaming, the *result* of sending such an INVITE is that an IPTV session occurs. The entire purpose of the INVITE is to establish such a session, and therefore, invoke the service. Thus, a service is not something that is different from the rest of the signaling message. A service is what the user gets after the network and other user agents have processed a signaling message.

This principle leads to another important conclusion:

> If two services are different, but their signaling appears to be the same, it is because there is in fact something different that has been overlooked, or something has been implied from the signaling which should have been signaled explicitly.

This makes sense; if a service is the byproduct of signaling, how can a user have different experiences and different services when the signaling message is the same? There has to be something different in the messages, if the user experience was in fact different.

To illustrate this, let us take each of the example services in Section 4 and investigate whether there is, or should be, something different in the signaling in each case.

IPTV vs. Multimedia Conferencing: The two services in Section 4.1 appear to have identical signaling. They both involve audio and video streams, both of which are unidirectional. Both might utilize the same codecs. However, there is another important difference in the signaling - the target URI. In the case of IPTV, the request is targeted at a media server or to a particular piece of content to be viewed. In the case of multimedia conferencing, the target is a conference server. The administrator of the domain can therefore examine the two Request-URI, and figure out whether it is targeted for a conference server or a content server, and use that to derive the service associated with the request.

Gaming vs. Voice Chat: Though both sessions involve MSRP and voice, and both are targeted to the same AOR of the called user, there is a difference. The MSRP messages for the gaming session carry content which is game specific, whereas the MSRP messages for the voice chat are just regular text, meant for rendering to a user. Thus, the MSRP session in the SDP will indicate the specific content type that MSRP is carrying, and this type will differ in both cases. Even if the game moves look like text, since they are being consumed by an automata there is an underlying schema that dictates their content, and therefore, this schema represents the actual content type that should be signaled.

Configuration vs. Pager Messaging: Just as in the case of gaming vs. voice chat, the content type of the messages differentiates the service that occurs as a consequence of the messages.

This is ultimately an expression of the principle of DWIM vs. DWIS (Do-What-I-Mean vs. Do-What-I-Say). Explicit signaling is DWIS - the user is asking for a service by invoking the signaling that results

in the desired effect. A service identifier is DWIM - an unspecific request for something that is ill-defined and non-interoperable.

6.2. Perils of Explicit Identifiers

Given that the information in the signaling message always conveys enough information to identify the service, another important conclusion can be drawn:

> Inclusion of an explicit service identifier within a message is, at best, redundant, and at worst, an avenue for fraud, loss of interoperability, and stifling of service innovation.

By "explicit service identifier", we mean a field included in the signaling message that contains a token whose value indicates the specific service invoked by the calling user. This would be "IPTV" or "voice chat" or "shoot-em game" or "short message service". This explicit identifier would typically be inserted by the originating user agent, and carried in the signaling message.

Clearly, if the signaling message itself contains enough information to identify the service, inclusion of an extra field to say the same thing is going to be redundant. Redundancy by itself is not a big deal. However, redundancy can lead to other,more significant problems.

6.2.1. Fraud

First and foremost, it can lead to fraud. If a provider uses the service identifier for billing and accounting purposes, or for authorization purposes, it opens an avenue for attack. The user can construct the signaling message so that its actual effect (which is the service the user will receive), is what the user desires, but the service identity (which is what is used for billing and authorization) doesn't match, and indicates a cheaper service, or one that the user is authorized to receive. If, however, the service identity used by the domain admistrator is derived from the signaling itself, the user cannot lie. If they did lie, they wouldn't get the desired service.

Consider the example of IPTV vs. multimedia conferencing. If multimedia conferencing is cheaper, the user could send an INVITE for an IPTV session, but include a service identifier which indicates multimedia conferencing. They get the service associated with IPTV, but at the cost of multimedia conferencing.

This same principle shows up in other places. For example, in the identification of an emergency services call [6]. It is desirable to

give emergency services calls special treatment, such as being free, authorized even when the user cannot otherwise make calls, and to give them priority. If emergency calls where indicated through something other than the target of the call being an emergency services URN [7], it would open an avenue for fraud. The user could place any desired URI in the request-URI, and indicate that the call is an emergency services call. This could would then get special treatment, but of course get routed to the target URI. The only way to prevent this fraud is to consider an emergency call as any call whose target is an emergency services URN. Thus, the service identification here is based on the target of the request. When the target is an emergency services URN, the request can get special treatment. The user cannot lie, since there is no way to separately indicate this is an emergency call, besides targeting it to an emergency URN.

6.2.2. Systematic Interoperability Failures

How can inclusion of an explicit service identifier cause loss of interoperability? When such an identifier is used to drive functionality - such as dispatch on the phones, in the network, or QoS authorization, it means that the wrong thing can happen when this field is not set properly. Consider a user in domain 1, calling a user in domain 2. Domain 1 provides the user with a service they call "voice chat", which utilizes voice and IM for real time conversation, driven off of a buddy list application on a PC. Domain 2 provides their users with a service they call, "text telephony", which is a voice service on a wireless device that also allows the user to send text messages. Consider the case where domain 1 and domain 2 both have their user agents insert a service identifiers into the request, and then use that to derive QoS authorization, accounting, and invocation of applications in the network and in the device. The user in domain 1 calls the user in domain 2, and inserts the identifier "Voice Chat" into the INVITE. When this arrives at the proxy in domain 2, the service is unknown. Consequently, the request does not get the proper QoS treatment. When it gets delivered to the User Agent of the user in domain 2, the user agent does not see a service it understands, and so consequently, does not know to dispatch the request to the right application software. Thus, this call has completely failed, even when it could have succeeded. This illustrates the following key point:

> Explicit service identifiers, used between domains, cause interoperability failures unless all interconnected domains agree on exactly the same set of services and how to name them.

Of course, lack of service identifiers does not guarantee service interoperability. However, SIP was built with rich tools for

negotiation of capabilities at a finely granular level. One user agent can make a call using audio and video, but if the receiving UA only supports audio, SIP allows both sides to negotiate down to the lowest common denominator. Thus, communications is still provided. As another example, if one agent initiates a Push-To-Talk session (which is audio with a companion floor control mechanism), and the other side only did regular audio, SIP would be able to negotiate back down to a regular voice call. As another example, if a calling user agent is running a high-definition video conferencing endpoint, and the called user agent supports just a regular video endpoint, the codecs themselves can negotiate downward to a lower rate, picture size, and so on. Thus, interoperability is achieved. Interestingly, the final "service" may no longer be well characterized by the service identifier that would have been placed in the original INVITE. For example, in this case, of the original INVITE from the caller had contained the service identifier, "hi-fi video", but the video gets negotiated down to a lower rate and picture size, the service identifier is no longer really appropriate.

This illustrates another key aspect of the interoperability problem:

> Usage of explicit service identifiers in the request will result in inconsistencies with results of any SIP negotiation that might otherwise be applied in the session.

Of course, there are cases where negotiating to a common baseline is not what is desired. SIP provides tools (such as Require), to force the call to fail unless the desired capabilities are supported. However, this is not recommended as a general rule [4].

When a service identifier becomes something that both proxies and the user agent need to understand in order to properly treat a request, it becomes equivalent to including a token in the Proxy-Require and Require header fields of every single SIP request. The very reason that RFC 4485 frowns upon usage of Require and certainly Proxy-Require is the huge impact on interoperability it causes. It is for this same reason that explicit service identifiers need to be avoided:

> The usage of explicit service identifiers is equivalent to the usage of Require and Proxy-Require in the request, and has the same negative impact on interoperability as those headers have.

6.2.3. Stifling of Service Innovation

The probability that any two pair of service providers end up with the same set of services, and give them the same names, becomes decreasingly small as the number of providers grow. Indeed, it would

almost certainly require a centralized authority to identify what the services are, how they work, and what they are named. This, in turn, leads to a requirement for complete homogeneity in order to facilitate interconnection. Two providers cannot usefully interconnect unless they agree on the set of services they are offering to their customers, and each do the same thing. This is, in a very real sense, anathema to the entire notion of SIP, which is built on the idea that heterogeneous domains can interconnect and still get interoperability:

> Explicit service identifiers lead to a requirement for homogeneity in service definitions across providers that interconnect, ruining the very service heterogeneity that SIP was meant to bring.

Indeed, Metcalfe's law says that the value of a network grows with the square of the number of participants. As a consequence of this, once a bunch of large domains did get together, agree on a set of services, and then a set of well-known identifiers for those services, it would force other providers to also deploy the same services, in order to obtain the value that interconnection brings. This, in turn, will stifle innovation, and quickly force the set of services in SIP to become fixed and never expand beyond the ones initially agreed upon. This, too, is anathema to the very framework on which SIP is built, and defeats much of the purpose of why providers have chosen to deploy SIP in their own networks:

> Metcalfe's law, when combined with explicit service identifiers, will stifle the ability of providers to develop new SIP services, since they have no hope of interconnecting them with anyone else.

Consider the following example. Several providers get together, and standardize on a bunch of service identifiers. One of these uses audio and video (say, "multimedia conversation"). This service is successful, and is widely utilized. Endpoints look for this identifier to dispatch calls to the right software applications, and the network looks for it to invoke features, perform accouting, and QoS. A new provider gets the idea for a new service, say, avatar-enhanced multimedia conversation. In this service, there is audio and video, but there is a third stream, which renders an avatar. A caller can press buttons on their phone, to cause the avatar on the other person's device to show emotion, make noise, and so on. This is similar to the way emoticons are used today in IM. This service is enabled by adding a third media stream (and consequently, third m-line) to the SDP.

Normally, this service would be backwards compatible with a regular audio-video endpoint, which would just reject the third media stream. However, because a large network has been deployed that is expecting

to see the token, "multimedia conversation" and its associated audio+ video service, it is nearly impossible for the new provider to roll out this new service. If they did, it would fail completely, or partially fail, when their users call users in other provider domains.

7. Recommendations

From these principles, several recommendations can be made:

- o Systems needing to perform service identification must examine existing signaling constructs to identify the service based on fields that exist within the signaling message already.

- o If it appears that the signaling currently defined in standards is not sufficient to identify the service, it may be due to lack of sufficient signaling to convey what is needed, and new standards work should be undertaken to fill this gap.

- o The usage of an explicit service identifier does make sense as a way to cache a decision made by a network element, for usage by another network element within the same domain. However, service identifiers are fundamentally useful within a particular domain, and any such header must be stripped at a network boundary.

8. Security Considerations

Oftentimes, the service associated with a request is utilized for purposes such as authorization, accounting, and billing. When service identification is not done properly, the possibility of network fraud is introduced. It is for this reason, discussed extensively in Section 6.2.1, that the usage of explicit service identifiers inserted by a UA is NOT RECOMMENDED.

9. IANA Considerations

There are no IANA considerations associated with this specification.

10. Acknowledgements

This document is based on discussions with Paul Kyzivat and Andrew Allen, who contributed significantly to the ideas here.

11. References

11.1. Normative References

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

[2] 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.

11.2. Informational References

[3] Rosenberg, J., "A Data Model for Presence", RFC 4479, July 2006.

[4] Rosenberg, J. and H. Schulzrinne, "Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP)", RFC 4485, May 2006.

[5] Campbell, B., "The Message Session Relay Protocol", draft-ietf-simple-message-sessions-19 (work in progress), February 2007.

[6] Rosen, B., "Framework for Emergency Calling in Internet Multimedia", draft-ietf-ecrit-framework-00 (work in progress), October 2006.

[7] Schulzrinne, H., "A Uniform Resource Name (URN) for Services", draft-ietf-ecrit-service-urn-05 (work in progress), August 2006.

[8] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

Author's Address

Jonathan Rosenberg
Cisco
Edison, NJ
US

Email: jdrosen@cisco.com
URI: http://www.jdrosen.net

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).