Network Working Group
Request for Comments: 4240
Category: Informational

E. Burger, Ed.
J. Van Dyke
A. Spitzer
Brooktrout Technology, Inc.

December 2005

Basic Network Media Services with SIP

Status of This Memo

Copyright Notice

Abstract

In SIP-based networks, there is a need to provide basic network media services. Such services include network announcements, user interaction, and conferencing services. These services are basic building blocks, from which one can construct interesting applications. In order to have interoperability between servers offering these building blocks (also known as Media Servers) and application developers, one needs to be able to locate and invoke such services in a well defined manner.

This document describes a mechanism for providing an interoperable interface between Application Servers, which provide application services to SIP-based networks, and Media Servers, which provide the basic media processing building blocks.

Table of Contents

1. Overview

In SIP-based media networks (RFC 3261 [10]), there is a need to provide basic network media services. Such services include playing announcements, initiating a media mixing session (conference), and prompting and collecting information with a user.

These services are basic in nature, are few in number, and fundamentally have not changed in 25 years of enhanced telephony services. Moreover, given their elemental nature, one would not expect them to change in the future.

Multifunction media servers provide network media services to clients using server protocols such as SIP, often in conjunction with markup languages such as VoiceXML [20] and MSCML [21]. This document describes how to identify to a multifunction media server what sort of session the client is requesting, without modifying the SIP protocol.

It is critically important to note that the mechanism described here in no way modifies the SIP protocol, the meaning, or definition of a SIP Request URI, or does it put any restrictions, in any way, on devices that do not implement this convention.

Announcements are media played to the user. Announcements can be static media files, media files generated in real-time, media streams generated in real-time, multimedia objects, or combinations of the above.

Media mixing is the act of mixing different RTP streams, as described in RFC 3550 [13]. Note that the service described here suffices for simple mixing of media for a basic conferencing service. This service does not address enhanced conferencing services, such as floor control, gain control, muting, subconferences, etc. MSCML [21] addresses enhanced conferencing. However, that is beyond the scope of this document. Interested readers should read conferencing-framework [22] for details on the IETF SIP conferencing framework.

Prompt and collect is where the server prompts the user for some information, as in an announcement, and then collects the user's response. This can be a one-step interaction, for example by playing an announcement, "Please enter your pass code", followed by collecting a string of digits. It can also be a more complex interaction, specified, for example, by VoiceXML [20] or MSCML [21].

1.1. Conventions Used in This Document

RFC 2119 [6] the interpretations for the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" found in this document.

2. Mechanism

In the context of SIP control of media servers, we take advantage of the fact that the standard SIP URI has a user part. Multifunction media servers do not have users. Thus we use the user address, or the left-hand-side of the URI, as a service indicator.

The use of the user part of the SIP Request URI has a number of useful properties:

- There is no change to core SIP.
- Only devices that choose to conform to this standard have to implement it.
- This document only applies to multifunction SIP-controlled media servers.
- This document has no impact on non-multifunction SIP-controlled media servers.
- The mechanism described in this document has absolutely no impact on SIP devices other than media servers.

The last bullet point is crucial. In particular, the user part convention described here places absolutely no restrictions on any SIP user agent, proxy, back-to-back user agent (B2BUA), or any future device. The user parts defined here only apply to multifunction media servers that chose to implement the convention. With the exception of a conforming media server, these user names and conventions have no impact on the user part namespace. They do not restrict the use of these user names at devices other than a multifunction media server.

Note that the set of services is small, well defined, and well contained. The section The User Part (Section 7) discusses the issues with using a fixed set of user-space names.

For per-service security, the media server SHOULD use the security protocols described in RFC 3261 [10].

The media server MAY issue 401 challenges for authentication. The media server SHOULD support the sips: scheme for the announcement service. The media server MUST support the sips: scheme for the dialog and conference services. The level of authentication to require for each service is a matter of local policy.

The media server, upon receiving an INVITE, notes the service indicator. Depending on the service indicator, the media server will either honor the request or return a failure response code.

The service indicator is the concatenation of the service name and an optional service instance identifier, separated by an equal sign.

Per RFC 3261 [10], the service indicator is case insensitive. The service name MUST be from the set alphanumeric characters plus dash (US-ASCII %2C). The service name MUST NOT include an equal sign (US-ASCII %3D).

The service name MAY have long- and short-forms, as SIP does for headers.

A given service indicator MAY have an associated set of parameters. Such parameters MUST follow the convention set out for SIP URI parameters. That is, a semi-colon separated list of keyword=value pairs.

Certain services may have an association with a unique service instance on the media server. For example, a given media server can host multiple, separate conference sessions. To identify unique service instances, a unique identifier modifies the service name.

The unique identifier MUST meet the rules for a legal user part of a SIP URI. An equal sign, US-ASCII %3D, MUST separate the service indicator from the unique identifier.

Note that since the service indicator is case insensitive, the service instance identifier is also case insensitive.

The requesting client issues a SIP INVITE to the media server, specifying the requested service and any appropriate parameters.

If the media server can perform the requested service, it does so, following the processing steps described in the service definition document.

If the media server cannot perform the requested service or does not recognize the service indicator, it MUST respond with the response code 488 NOT ACCEPTABLE HERE. This is appropriate, as 488 refers to a problem with the user part of the URI. Moreover, 606 is not appropriate, as some other media server may be able to satisfy the request. RFC 3261 [10] describes the 488 and 606 response codes.

Some services require a unique identifier. Most services automatically create a service instance upon the first INVITE with the given identifier. However, if a service requires an existing service instance, and no such service instance exists on the media server, the media server MUST respond with the response code 404 NOT FOUND. This is appropriate as the service itself exists on the media server, but the particular service instance does not. It is as if the user was not home.

3. Announcement Service

A network announcement is the delivery of a multimedia resource, such as a prompt file, to a terminal device. Note the multimedia resource may be any multimedia object that the media server supports. This service can play a single object with multiple streams, such as a video and audio prompt. However, this service cannot play multiple objects on the same SIP dialog.

There are two types of network announcements. The differentiating characteristic between the two types is whether the network fully sets up the SIP dialog before playing the announcement. The analog in the Public Switched Telephone Network (PSTN) is whether answer supervision is supplied (i.e., does the announcement server answer the call prior to delivering the announcement?).

Playing an announcement after call setup is straightforward. First, the requesting device issues an INVITE to the media server requesting the announcement service. The media server negotiates the SDP and responds with a 200 OK. After receiving the ACK from the requesting device, the media server plays the requested object and issues a BYE to the requesting device.

If the media server supports announcements, but it cannot find the referenced URI, it MUST respond with the 404 response code and SHOULD send the reason phrase "Announcement content not found".

If the media server receives an INVITE for the announcement service without a "play=" parameter, it MUST respond with the response code 400 and SHOULD send the reason phrase "Mandatory play parameter missing".

If there is an error retrieving the announcement, the media server MUST respond with a 400 response code and SHOULD send the reason phrase "Announcement content could not be retrieved". In addition the media server SHOULD include a Warning header with appropriate explanatory text explaining what failed.

The Request URI fully describes the announcement service through the use of the user part of the address and additional URI parameters. The user portion of the address, "annc", specifies the announcement service on the media server. The service has several associated URI parameters that control the content and delivery of the announcement.

These parameters are described below:

play
: Specifies the resource or announcement sequence to be played.

repeat
: Specifies how many times the media server should repeat the announcement or sequence named by the "play=" parameter. The value "forever" means the repeat should be effectively unbounded. In this case, it is RECOMMENDED the media server implements some local policy, such as limiting what "forever" means, to ensure errant clients do not create a denial of service attack.

delay
: Specifies a delay interval between announcement repetitions. The delay is measured in milliseconds.

duration
: Specifies the maximum duration of the announcement. The media server will discontinue the announcement and end the call if the

maximum duration has been reached. The duration is measured in milliseconds.

locale
Specifies the language and optionally country variant of the announcement sequence named in the "play=" parameter. RFC 3066 [9] specifies the locale tag. The locale tag is usually a two- or three-letter code per ISO 639-1 [11]. The country variant is also often a two-letter code per ISO 3166-1 [12]. These elements are concatenated with a single under bar (%x5F) character, such as "en_CA". If only the language is specified, such as locale=en, the choice of country variant is an implementation matter. Implementations SHOULD provide the best possible match between the requested locale and the available languages in the event the media server cannot honor the locale request precisely. For example, if the request has locale=ca_FR, but the media server only has fr_FR available, the media server should use the fr_FR variant. Implementations SHOULD provide a default locale to use if no language variants are available.

param[n]
Provides a mechanism for passing values that are to be substituted into an announcement sequence. Up to 9 parameters ("param1=" through "param9=") may be specified. The mechanics of announcement sequences are beyond the scope of this document.

extension
Provides a mechanism for extending the parameter set. If the media server receives an extension it does not understand, it MUST silently ignore the extension parameter and value.

The "play=" parameter is mandatory and MUST be present. All other parameters are OPTIONAL.

NOTE: Some encodings are not self-describing. Thus, the implementation relies on filename extension conventions for determining the media type.

Note that RFC 3261 [10] implies that proxies are supposed to pass parameters through unchanged. However, be aware that non-conforming proxies may strip Request-URI parameters. That said, given the likely scenarios for the mechanisms presented in this document, this should not be an issue. Most likely, the proxy inserting the parameters is the last proxy before the media server. If the service provider deploys a proxy for load balancing or service location purposes, the service provider should ensure that its choice of proxy preserves parameters.

The form of the SIP Request URI for announcements is as follows. Note that the backslash, CRLF, and spacing before the "play=" in the example is for readability purposes only.

```
sip:annc@ms2.example.net; \
    play=http://audio.example.net/allcircuitsbusy.g711

sip:annc@ms2.example.net; \
    play=file://fileserver.example.net//geminii/yourHoroscope.wav
```

3.1. Operation

The scenarios below assume there is a SIP Proxy, application server, or media gateway controller between the caller and the media server. However, the announcement service works as described below even if the caller invokes the service directly. We chose to discuss the proxy case, as it will be the most common case.

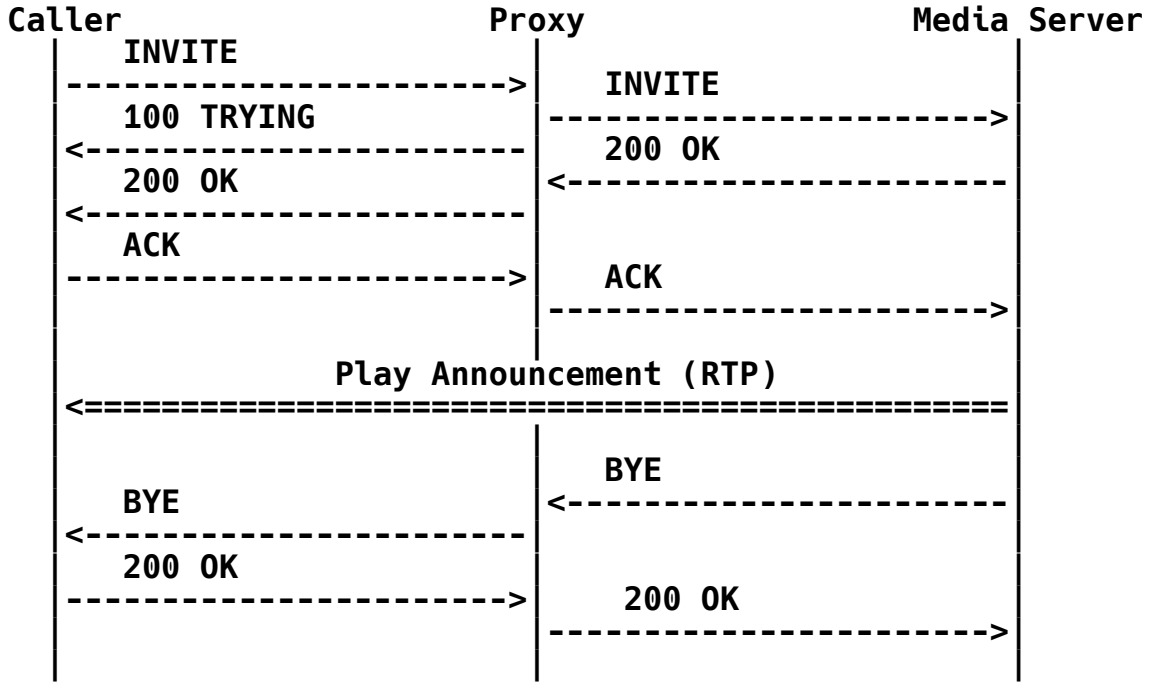
The caller issues an INVITE to the serving SIP Proxy. The SIP Proxy determines what audio prompt to play to the caller. The proxy responds to the caller with 100 TRYING.

It is important to note that the mechanism described here in no way modifies the behavior of SIP [10]. In particular, this convention does not modify SDP negotiation [18].

The proxy issues an INVITE to the media server, requesting the appropriate prompt to play coded in the play= parameter. The media server responds with 200 OK. The proxy relays the 200 OK to the caller. The caller then issues an ACK. The proxy then relays the ACK to the media server.

With the call established, the media server plays the requested prompt. When the media server completes the play of the prompt, it issues a BYE to the proxy. The proxy then issues a BYE to the caller.

3.2. Protocol Diagram



3.3. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC 4234 [7].

```
ANNC-URL        = sip-ind annc-ind "@" hostport
                    annc-parameters uri-parameters

sip-ind         = "sip:" / "sips:"
annc-ind        = "annc"

annc-parameters = ";" play-param [ ";" content-param ]
                                 [ ";" delay-param]
                                 [ ";" duration-param ]
                                 [ ";" repeat-param ]
                                 [ ";" locale-param ]
                                 [ ";" variable-params ]
                                 [ ";" extension-params ]

play-param      = "play=" prompt-url

content-param   = "content-type=" MIME-type

delay-param     = "delay=" delay-value
```

```
   delay-value     = 1*DIGIT

   duration-param  = "duration=" duration-value

   duration-value  = 1*DIGIT

   repeat-param    = "repeat=" repeat-value

   repeat-value    = 1*DIGIT / "forever"

   locale-param    = "locale=" token
                        ; per RFC 3066, usually
                        ; ISO639-1_ISO3166-1
                        ; e.g., en, en_US, en_UK, etc.

   variable-params = param-name "=" variable-value

   param-name      = "param" DIGIT ; e.g., "param1"

   variable-value  = 1*(ALPHA / DIGIT)

   extension-params = extension-param [ ";" extension-params ]

   extension-param  = token "=" token
```

"uri-parameters" is the SIP Request-URI parameter list as described in RFC 3261 [10]. All parameters of the Request URI are part of the URI matching algorithm.

The MIME-type is the MIME [1] [2] [3] [4] [5] content type for the announcement, such as audio/basic, audio/G729, audio/mpeg, video/mpeg, and so on.

A number of MIME registrations, which could be used here, have parameters, for instance, video/DV. To accommodate this, and retain compatibility with the SIP URI structure, the MIME-type parameter separator (semicolon, %3b) and value separator (equal, %d3) MUST be escaped. For example:

```
sip:annc@ms.example.net; \
    play=file://fs.example.net//clips/my-intro.dvi; \
    content-type=video/mpeg%3bencode%d3314M-25/625-50
```

The locale-value consists of a tag as specified in RFC 3066 [9].

The definition of hostport is as specified by RFC 3261 [10].

The syntax of prompt-url consists of a URL scheme as specified by RFC 3986 [8] or a special token indicating a provisioned announcement sequence. For example, the URL scheme MAY include any of the following.

- http/https
- ftp
- file (referencing a local or NFS (RFC 3530 [16]) object)
- nfs (RFC 2224 [14])

If a provisioned announcement sequence is to be played, the value of prompt-url will have the following form:

```
prompt-url      = "/provisioned/" announcement-id

announcement-id = 1*(ALPHA / DIGIT)
```

Note that the scheme "/provisioned/" was chosen because of a hesitation to register a "provisioned:" URI scheme.

This document is strictly focused on the SIP interface for the announcement service and, as such, does not detail how announcement sequences are provisioned or defined.

Note that the media type of the object the prompt-url refers to can be most anything, including audio file formats, text file formats, or URI lists. See the Prompt and Collect Service (Section 4) section for more on this topic.

4. Prompt and Collect Service

This service is also known as a voice dialog. It establishes an aural dialog with the user.

The dialog service follows the model of the announcement service. However, the service indicator is "dialog". The dialog service takes a parameter, voicexml=, indicating the URI of the VoiceXML script to execute.

```
sip:dialog@mediaserver.example.net; \
    voicexml=http://vxmlserver.example.net/cgi-bin/script.vxml
```

A Media Server MAY accept additional SIP request URI parameters and deliver them to the VoiceXML interpreter session as session variables.

Although not good VoiceXML programming practice, VoiceXML scripts might contain sensitive information, such as a user's pass code in a

DTMF grammar. Thus, the media server MUST support the https scheme for the voicexml parameter for secure fetching of scripts. Likewise, dynamic grammars often do have user-identifying information. As such, the VoiceXML browser implementation on the media server MUST support https fetching of grammars and subsequent documents.

Returned information often is sensitive. For example, the information could be financial information or instructions. Thus, the media server MUST support https posting of results.

4.1. Formal Syntax for Prompt and Collect Service

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC 4234 [7].

```
DIALOG-URL        = sip-ind dialog-ind "@" hostport
                        dialog-parameters

sip-ind           = "sip:" / "sips:"
dialog-ind        = "dialog"

dialog-parameters = ";" dialog-param [ vxml-parameters ]
                                     [ uri-parameters ]

dialog-param      = "voicexml=" vxml-url

vxml-parameters   = vxml-param [ vxml-parameters ]

vxml-param        = ";" vxml-keyword "=" vxml-value

vxml-keyword      = token

vxml-value        = token
```

The vxml-url is the URI of the VoiceXML script. If present, other parameters get passed to the VoiceXML interpreter session with the assigned vxml-keyword vxml-value pairs. Note that all vxml-keywords MUST have values.

If there is a vxml-keyword without a corresponding vxml-value, the media server MUST reject the request with a 400 BAD REQUEST response code. In addition, the media server MUST state "Missing VXML Value" in the reason phrase.

The media server presents the parameters as environment variables in the connection object. Specifically, the parameter appears in the connection.sip tree.

If the Media Server does not support the passing of keyword-value pairs to the VoiceXML interpreter session, it MUST ignore the parameters.

"uri-parameters" is the SIP Request-URI parameter list as described in RFC 3261 [10]. All parameters in the parameter list, whether they come from uri-parameters or from vxml-keyworks, are part of the URI matching algorithm.

5. Conference Service

One identifies mixing sessions through their SIP request URIs. To create a mixing session, one sends an INVITE to a request URI that represents the session. If the URI does not already exist on the media server and the requested resources are available, the media server creates a new mixing session. If there is an existing URI for the session, then the media server interprets it as a request for the new session to join the existing session. The form of the SIP request URI for conferencing is:

sip:conf=uniqueIdentifier@mediaserver.example.net

The left-hand side of the request URI is actually the username of the request in the request URI and the To header. The host portion of the URI identifies a particular media server. The "conf" user name conveys to the media server that this is a request for the mixing service. The uniqueIdentifier can be any value that is compliant with the SIP URI specification. It is the responsibility of the conference control application to ensure the identifier is unique within the scope of any potential conflict.

In the terminology of the conferencing framework [22], this URI convention tells the media server that the application server is requesting it to act as a Focus. The conf-id value identifies the particular focus instance.

As a focus in the conferencing framework, the media server MUST support the ";isfocus" parameter in the Request URI. Note, however, that the presence or absence of the ";isfocus" parameter has no protocol impact at the media server.

It is worth noting that the conference URI shared between the application and media servers provides enhanced security, as the SIP control interface does not have to be exposed to participants. It also allows the assignment of a specific media server to be delayed as long as possible, thereby simplifying resource management.

One can add additional legs to the conference by INVITEing them to the above-mentioned request URI. Per the matching rules of RFC 3261 [10], the conf-id parameter is part of the matching string.

Conversely, one can remove legs by issuing a BYE in the corresponding dialog. The mixing session, and thus the conference-specific request URI, remains active so long as there is at least one SIP dialog associated with the given request URI.

If the Request-URI has "conf" as the user part, but does not have a conf-id parameter, the media server MUST respond with a 404 NOT FOUND.

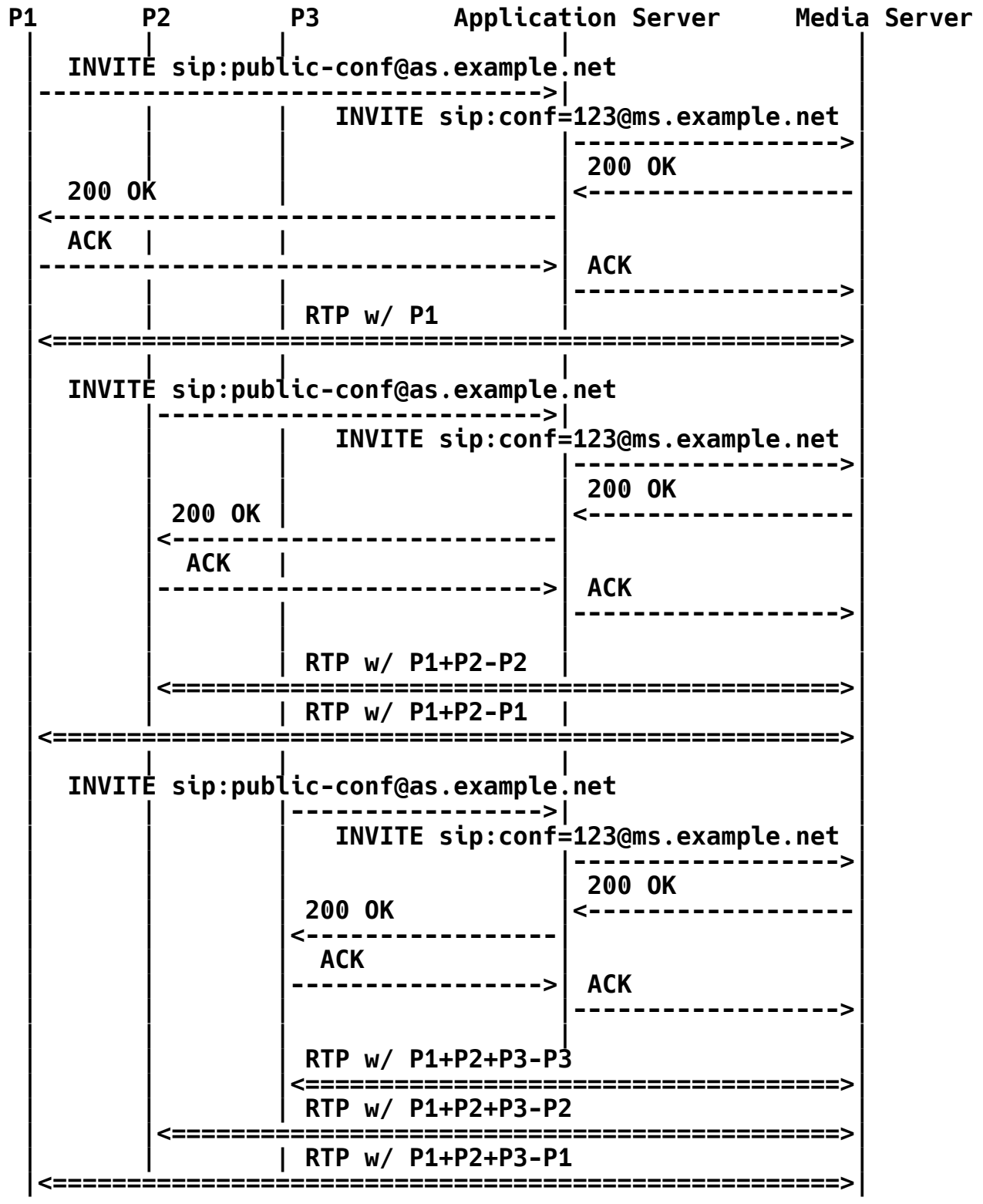
> NOTE: The media server could create a unique conference instance and return the conf-id string to the User Agent Clinet (UAC) if there is no conf-id present. However, such an operation may have other operational issues, such as permissions and billing. Thus an application server or proxy is a better place to do such an operation. Moreover, such action would make the media server into a Conference Factory in the terminology of conference-framework [22]. That is not the appropriate behavior for a media server.

Since some conference use cases, such as business conferencing, have billing implications, the media server SHOULD authenticate the application server or proxy. At a minimum, the media server MUST implement sips:.

5.1. Protocol Diagram

This diagram shows the establishment of a three-way conference. This section is informative. It is only one method of establishing a conference. This example shows a simple back-to-back user agent.

The conference-framework [22] describes additional parameters and behaviors of the Application Server. For example, the first INVITE from P1 to the Application Server would include the ";isfocus" parameter; the Application Server would act as a Conference Factory; and so on. However, none of that protocol machinery has an impact on the operation of the Application Server to Media Server interface, which is the focus of this protocol document.


P1
P2
P3
Application Server
Media Server
INVITE sip:public-conf@as.example.net
INVITE sip:conf=123@ms.example.net
200 OK
200 OK
ACK
ACK
RTP w/ P1
INVITE sip:public-conf@as.example.net
INVITE sip:conf=123@ms.example.net
200 OK
200 OK
ACK
ACK
RTP w/ P1+P2-P2
RTP w/ P1+P2-P1
INVITE sip:public-conf@as.example.net
INVITE sip:conf=123@ms.example.net
200 OK
200 OK
ACK
ACK
RTP w/ P1+P2+P3-P3
RTP w/ P1+P2+P3-P2
RTP w/ P1+P2+P3-P1

Using the terminology of conference-framework [22], the Application Server is the Conference Factory, and the Media Server is the Conference Focus.

Note that the above call flow does not show any 100 TRYING messages that would typically flow from the Application Server to the UACs; nor does it show the ACKs from the UACs to the Application Server or from the Application Server to the Media Server.

Each leg can drop out either under the supervision of the UAC, by the UAC sending a BYE, or under the supervision of the Application Server, by the Application Server issuing a BYE. In either case, the Application Server will either issue a BYE on behalf of the UAC or issue it directly to the Media Server, corresponding to the respective disconnect case.

It is left as a trivial exercise to the reader for how the Application Server can mute legs, create side conferences, and so forth.

Note that the Application Server is a server to the participants (UACs). However, the Application Server is a client for mixing services to the Media Server.

5.2. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC 4234 [7].

```
CONF-URL        = sip-ind conf-ind "=" instance-id "@" hostport
                  [ uri-parameters ]

sip-ind         = "sip:" / "sips:"

conf-ind        = "conf"

instance-id     = token
```

"uri-parameters" is the SIP Request-URI parameter list as described in RFC 3261 [10]. All parameters in the parameter list are part of the URI matching algorithm.

6. IANA Considerations

The IANA has registered the following parameters in the SIP/SIPS URI Parameters registry, following the specification required policy of RFC 3969 [19]:

Parameter Name	Predefined Values	Reference
play	no	RFC 4240
repeat	no	RFC 4240
delay	no	RFC 4240
duration	no	RFC 4240
locale	no	RFC 4240
param[n]	no	RFC 4240
extension	no	RFC 4240

7. The User Part

There has been considerable discussion about the wisdom of using fixed user parts in a request URI. The most common objection is that the user part should be opaque and a local matter. The other objection is that using a fixed user part removes those specified user addresses from the user address space.

We address the latter issue first. The common example is the Postmaster address defined by RFC 2821 [15]. The objection is that by using the Postmaster token for something special, one removes that token for anyone. Thus, the Postmaster General of the United States, for example, cannot have the mail address Postmaster@usps.gov. However, one may debate whether this is a significant limitation.

This document explicitly addresses this issue. The user names described in the text (namely annc, ivr, dialog, and conf) are available for whatever local use a given SIP user agent or proxy wishes for them. What this document does is give special meaning for these user names at media servers that implement this specification. If a media server chooses not to implement this specification, nothing breaks. If a user wishes to use one of the user names described in this document at their SIP user agent, nothing breaks and their user agent will work as expected.

The key point is, one cannot confuse the namespace at a Media Server with the namespace for an organization. For example, let us take the case where a network offers services for "Ann Charles". She likes to use the name "annc", and thus she would like to use "sip:annc@example.net". We offer there is ABSOLUTELY NO NAME COLLISION WHATSOEVER. Why is this so? This is so because sip:annc@example.net will resolve to the specific user at a specific

device for Ann. As an example, example.net's SIP Proxy Server resolves sip:annc@example.net to annc@anns-phone.example.net. Conversely, one directs requests for the media service annc directly to the Media Server, e.g., sip:annc@ms21.ap.example.net. Moreover, by definition, requests for Ann Charles, or anything other than the announcement service, will NEVER be directly sent to the Media Server. If that were not true, no phone in the world could use the user part "eburger", as eburger is a reserved user part in the Brooktrout domain. Clearly, this is not the case.

If one wishes to make their media server accessible to the global Internet, but retain one of the Media Server-specific user names in the domain, a SIP Proxy can easily translate whatever opaque name one chooses to the Media Server-specific user name. For example, if a domain wishes to offer services for the above mentioned Ann Charles at sip:annc@example.com, they can offer the announcement service at sip:my-special-announcement-service@example.com. The former address, sip:annc@example.com, would resolve to the actual device where annc resides. The latter would resolve to the media server announcement server address, sip:annc@mediaserver.example.com, as an example. Note that this convention makes it easier to provision this service. With a fixed mapping at the multifunction media server, there are less provisioning data elements to get wrong.

Here is another way of looking at this issue. Unix reserves the special user "root". Just about all Unix machines have a user root, who has an address "root@a-specific-machine.example.com", where "a-specific-machine" is the fully-qualified domain name (FQDN) of a particular instance of a machine. There are very well-defined semantics for the "root" user.

Even though most every Unix machine has a "root" user, often there is no mapping for a "root" user in a domain, such as "root@example.com". Conversely, there is no restriction on creating an MX record for "root@example.com". That choice is fully up to the administrative authority for the domain.

The "users" proposed by this document, "annc", "conf", and "dialog" are all users at a Media Server, just as the "root", "bin", and "nobody" users are "users" at a Unix host.

After much discussion, with input from the W3C URI work group, we considered obfuscating the user name by prepending "__sip-" to the user name. However, as explained above, this obfuscation is not necessary. There is a fundamental difference between a user name at a device and a user name at an MX record (SMTP) or Address-of-Record (SIP). Again, there is no possibility that the name on the device may "leak out" into the SIP routing network.

The most important thing to note about this convention is that the left-hand side of the request URI is opaque to the network. The only network elements that need to know about the convention are the Media Server and client. Even proxies doing mapping resolution, as in the example above for public announcement services, do not need to be aware of the convention. The convention is purely a matter of provisioning.

Some have proposed that such naming be a pure matter of local convention. For example, the thesis of the informational RFC RFC 3087 [17] is that you can address services using a request URI. However, some have taken the examples in the document to an extreme. Namely, that the only way to address services is via arbitrary, opaque, long user parts. Clearly, it is possible to provision the service names, rather than fixed names. While this can work in a closed network, where the Application Servers and Media Servers are in the same administrative domain, this does not work across domains, such as in the Internet. This is because the client of the media service has to know the local name for each service / domain pair. This is particularly onerous for situations where there is an ad hoc relationship between the application and the media service. Without a well-known relationship between service and service address, how would the client locate the service?

One very important result of using the user part as the service descriptor is that we can use all of the standard SIP machinery, without modification. For example, Media Servers with different capabilities can SIP Register their capabilities as users. For example, a VoiceXML-only device will register the "dialog" user, while a multi-purpose Media Server will register all of the users. Note that this is why the URI to play is a parameter. Doing otherwise would overburden a normal SIP proxy or redirect server. Conversely, having the conference ID be part of the user part gives an indication that requests get routed similarly (as opposed to requiring a Globally Routable User Agent URI (GRUU), which would restrict routing to the same device).

Likewise, this scheme lets us leverage the standard SIP proxy behavior of using an intelligent redirect server or proxy server to provide high-available services. For example, two Media Servers can register with a SIP redirect server for the annc user. If one of the Media Servers fails, the registration will expire and all requests for the announcement service ("calls to the annc user") will get sent to the surviving Media Server.

8. Security Considerations

Exposing network services with well-known addresses may not be desirable. The Media Server SHOULD authenticate and authorize requesting endpoints per local policy.

Some interactions in this document result in the transfer of confidential information. Moreover, many of the interactions require integrity protection. Thus, the Media Server MUST implement the sips: scheme. In addition, application developers are RECOMMENDED to use the security services offered by the Media Server to ensure the integrity and confidentiality of their user's data, as appropriate.

Untrusted network elements could use the convention described here for providing information services. Many extant billing arrangements are for completed calls. Successful call completion occurs with a 2xx result code. This can be an issue for the early media announcement service. This is one of the reasons why the early media announcement service is deprecated.

Services such as repeating an announcement forever create the possibility for denial of service attacks. The media server SHOULD have local policies to deal with this, such as time-limiting how long "forever" is, analyzing where multiple requests come from, implementing white-lists for such a service, and so on.

9. Contributors

Jeff Van Dyke and Andy Spitzer of SnowShore did just about all of the work developing netann, in conjunction with many application developers, media server manufacturers, and service providers, some of whom are listed in the Acknowledgements section. All I did was do the theory and write it up. That also means all of the mistakes are mine, as well.

10. Acknowledgements

We would like to thank Kevin Summers and Ravindra Kabre of Sonus Networks for their constructive comments, as well as Jonathan Rosenberg of Dynamicsoft and Tim Melanchuk of Convedia for their encouragement. In addition, the discussion at the Las Vegas Interim Workgroup Meeting in 2002 was invaluable for clearing up the issues surrounding the left-hand-side of the request URI. Christer Holmberg helped tune the language of the multimedia announcement service. Orit Levin from Radvision gave a close read on the most recent version of the document. Pete Danielsen from Lucent has consistently provided excellent reviews of the many different versions of this document.

Pascal Jalet provided the theoretical underpinning and David Rio provided the experimental evidence for why the conference identifier belongs in the user part of the request-URI.

I am particularly indebted to Alan Johnston for his review of this document and ensuring its conformance with the SIP conference control work in the IETF.

Mary Barnes, as usual, found the holes and showed how to fix them.

The authors would like to give a special thanks to Walter O'Connor for doing much of the initial implementation.

Note that at the time of this writing, there are 7 known independent server implementations that are interoperable with 23 known client implementations. Our apologies if we did not count your implementation.

11. References

11.1. Normative References

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

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

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

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

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

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

[7] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[8] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[9] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[10] 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] International Organization for Standardization, "Codes for the representation of names of languages -- Part 1: Alpha-2 code", ISO Standard 639-1, July 2002.

[12] International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes", ISO Standard 3166-1, October 1997.

11.2. Informative References

[13] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[14] Callaghan, B., "NFS URL Scheme", RFC 2224, October 1997.

[15] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[16] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[17] Campbell, B. and R. Sparks, "Control of Service Context using SIP Request-URI", RFC 3087, April 2001.

[18] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[19] Camarillo, G., "The Internet Assigned Number Authority (IANA) Uniform Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", BCP 99, RFC 3969, December 2004.

[20] Burnett, D., Hunt, A., McGlashan, S., Porter, B., Lucas, B., Ferrans, J., Rehor, K., Carter, J., Danielsen, P., and S. Tryphonas, "Voice Extensible Markup Language (VoiceXML) Version 2.0", W3C REC REC-voicexml20-20040316, March 2004.

[21] Van Dyke, J., Burger, E., Ed., and A. Spitzer, "Media Server Control Markup Language (MSCML) and Protocol", Work in Progress, December 2004.

[22] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", Work in Progress, October 2004.

Authors' Addresses

Eric Burger
Brooktrout Technology, Inc.
18 Keewaydin Dr.
Salem, NH 03079
USA

EMail: eburger@brooktrout.com

Jeff Van Dyke
Brooktrout Technology, Inc.
18 Keewaydin Dr.
Salem, NH 03079
USA

EMail: jvandyke@brooktrout.com

Andy Spitzer
Brooktrout Technology, Inc.
18 Keewaydin Dr.
Salem, NH 03079
USA

EMail: woof@brooktrout.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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