Internet Engineering Task Force MMUSIC WG
Internet Draft Schulzrinne/Rosenberg
draft-ietf-mmusic-sip-caller-00.txt Columbia U./Bell Laboratories
February 26, 1999
Expires: August 26, 1999

SIP Caller Preferences and Callee Capabilities

STATUS OF THIS MEMO

Abstract

This document describes a set of extensions to SIP which allow a caller to express preferences about request handling in servers. It also extends the SIP Contact header to allow users to describe their communications capabilities and characteristics.

1 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant SIP implementations.

2 Introduction

When a SIP server receives a request, there are a number of decisions

it can make regarding processing of the request. These include

- o whether to proxy or redirect the request;
- o which URIs to proxy or redirect to;
- o whether to fork or not;
- o whether to search recursively or not;
- o whether to search in parallel or sequentially;
- o whether to return just the first 200-class response, or all 2xx responses.

The server can base these decisions on a large number of factors, such as time of day, caller or callee identity, call urgency, caller preferences, network status, and the content of external databases.

There are three parties which have an interest in influencing the call processing logic at the server: (1) the administrator of the server, (2) the callee, and (3) the caller. The directives of the administrator are usually embedded in the configuration of the server, and can be expressed, for example, in the form of SIP CGI scripts [2]. The preferences of the callee can be expressed most easily through a script written in the call processing language (CPL) [3]. This document specifies SIP extensions which express caller preferences and describe callee capabilities by adding three new header fields (Request-Disposition, Accept-Contact, Reject-Contact) and extending the parameter list for the Contact header field.

> This draft is the result of partitioning the SIP call control extensions draft [4] into two components, this one and another on third-party call control, to be issued.

If the client wants to be sure that the server understands the headers described in this specification, it MUST include a Require option of org.ietf.sip.caller-preferences.

Since the Accept-Contact and Request-Disposition header are advisory only, a client can save an extra round-trip time if it only includes the Require option if the Reject-Contact header is present.

3 Design Alternatives

There are a number of alternatives for expressing caller preferences. Caller preferences and callee preferences must "meet" at the proxy or

redirect server responsible for the callee, so that the appropriate forwarding decision can be made. This server will always be under the control of the callee or an entity trusted by the callee, so that installing a call processing language script expressing caller preferences is not appropriate (unless, of course, the callee would like to accomodate the known preferences of a certain caller). Uploading a caller preference script to every possible server is also clearly not feasible. As another alternative, one could embed a script in the request, to be executed by proxy or redirect servers when making forwarding decisions. This would be an application-layer version of active networks. However, the generality of a script does not seem to be needed. It also makes combining caller and callee preferences a rather difficult problem and raises possible performance and security issues. Unlike the callee script, which needs to handle unknown callers, with a wide range of call properties, at unknown times in the future, a caller knows all but the set of communications capabilities of the callee. The caller can present the servers with its preferences on a call-by-call basis. Callers can thus place their preferences for this particular call in the request message. We propose a simple ordered list of preferences to make it possible to reconcile caller and callee preferences algorithmically.

In summary, there is a strong asymmetry in how preferences for callers and callees can be presented to the network. While a caller takes an active role by initiating the call, the callee takes a passive role in waiting for calls. This motivates the use of callee-supplied scripts and caller preferences included in the call request.

This asymmetry is also reflected in the appropriate relationship between caller and callee preferences. A server for a callee SHOULD respect the wishes of the caller to avoid certain locations, while the preferences among locations has to be the callee's choice, as it determines where, for example, the phone rings and whether the callee incurs mobile telephone charges for incoming calls.

The problem of feature negotation has also been approached in a more general way by [5]. However, that proposal is far more complicated than appears to be needed here, with syntax that does not fit into the current SIP syntax structure.

4 Overview

The extension specified in this document consists of three new header fields: Reject-Contact, Accept-Contact, and Request-Disposition. The first two express preferences about which URIs the client would like the request to reach. The Reject-Contact header field (Section 6.3) contains a list of URIs the user does not wish the request to be

proxied (or redirected) to. The Accept-Contact header field (Section 6.2) is a list of addresses that the caller would like to be proxied (or redirected) to, with the strength of preferences expressed through a q parameter, ranging in value from zero to one, with one indicating the highest preference. The Request-Disposition header field (Section 6.4) contains a set of tags which request particular processing for this request, such as whether or not the caller prefers proxy or redirection.

5 Determining Request Forwarding

The Accept-Contact header field indicates preferences for certain destinations among those chosen by a proxy server. The Reject-Contact header field describes those locations that the proxy server should not direct the request to.

Listing only URIs is of limited use, since the caller often will not know where the callee is located. For this reason, this specification proposes a number of URI parameters which describe characteristics of the user. These parameters include whether the location is a home or work address, whether it is fixed or mobile, and what media types are available. A server may know about a number of URIs for a user, along with parameters describing each one, for example, through user registration (REGISTER request). The combination of a URI along with a set of parameters is called a contact entry; a set of contact entries is called the contact list. When a request arrives with either Reject-Contact or Accept-Contact header field, the server performs a matching operation, described below, to create an ordered list of contact addresses that reflect the joint caller's and callee's preferences.

> There is some overlap between the indication of receiver capabilities in the session description message body and the Accept-Contact and Reject-Contact header fields. However, current session description formats cannot express the preferences described here. Also, the capabilities described here are fundamental to call-routing and thus should not depend on the particulars of the various session description formats that might be used.

5.1 Matching Rules

The server matches rules in the Accept-Contact and Reject-Contact request headers to contact entries in the contact list according to the following rules.

- If rule does not contain URI, only parameters are compared. If

the request header rule contains a URI, both the URI and parameters must match.

- o A URI in a rule matches a SIP URI in a contact entry if the user and host parts are equal, where equality is based on string equivalence. Either part may be omitted from the rule, and in this case it matches any value. For example, the rule " @acme.com " matches " foo@acme.com " or " bar@acme.com ", while the rule " joe@ " matches " joe@acme.com " and " joe@example.com ". Matches are case-insensitive. Other URIs are matched according to the equality rules for that URI scheme.

- o The parameters in a rule match the parameters in a contact entry if all parameters in the rule either have a matching value in the contact entry, or the parameter is not present in the contact entry.

  Ignoring the parameter if it does not exist in the contact list avoids that a parameter that the server does not know about causes the match to fail.

The pseudo-code below describes the matching procedure between the rule in the request header and the contact entry. For simplicity, we assume that the list of parameters in each rule is stored as an associative array, so that rule.para[duplex] yields the value of the attribute duplex, and is undefined if duplex is not specified.

```
struct {
  uri_t URI;           /* URI */
  parameter_t para[];  /* list of parameters */
} rule, entry;

boolean MATCH(rule r, entry e) {
  boolean match;

  match = TRUE;
  if (r.URI != "") {
    if (r.URI.scheme == e.URI.scheme) {
      if (r.URI.scheme == "sip") {
        match = (r.URI.host == "" || r.URI.host == e.URI.host) &&
                (r.URI.user == "" || r.URI.user == e.URI.user)
      } else {
        match = scheme-appropriate comparison;
      }
    } else {
      return FALSE;
```

```
      }
    }

    if(match == FALSE) return FALSE;

    /* compare parameters */
    foreach parameter p in r.para[] {
      if (isdefined(e.para[p])) {
        if (r.para[p][0] == "!")
          if (r.para[p] intersect e.para[p] != "") {
            return FALSE;
          }
        } else {
          if (r.para[p] is not subset of e.para[p]) {
            return FALSE;
          }
        }
      }
    }
    return TRUE;
  }
```

Parameter names are matched by case-insensitive string comparison. Parameter values are compared by set-comparisons. Parameter values in quoted strings are interpreted as sets, with elements separated by commas. The names of elements are case-insensitive. Parameter values that are tokens are interpreted as sets with one element. There are two cases: if the quoted-string parameter value in a rule starts with an exclamation mark (!), the rule matches if the intersection of the set in the rule and in the contact entry is empty. Otherwise, the rule matches if the intersection is the rule set itself, i.e., if the rule set is a subset of the contact entry parameter.

> The syntax for empty intersection is ugly. Using operators instead of equal may be preferable, but breaks the basic SIP parser model.

For example, the rule

```
;language="!en,de"
```

matches the contact entry containing

```
;language="es,nl"
```

but not any of

```
;language="en"
;language="de,en"
;language="en,es,fi"
```

As another example, the rule

```
;duplex="full,half"
```

matches the contact entry

```
;duplex="full"
```

but not

```
;duplex="send-only"
```

A server SHOULD NOT be aware of the particular semantics of any of the parameters. This allows for the definition of new parameters and values without explicitly programming them into the servers.

5.2 Contact List Processing

It is assumed that the server has a contact list for the callee. In this case, the server MAY elect to follow the procedure below for merging caller preferences and callee preferences. If the server has a call processing language for the callee, or some other form of complex logic, the way in which the caller preferences are merged is not defined, but left to the implementor. In general, the server SHOULD NOT proxy or redirect to a contact entry which matches a rule in the Reject-Contact request header. If the server was operating as a proxy, it SHOULD behave as if it had proxied, and had received a 404 "Not Found" in response. If the server was acting as a redirect server, if SHOULD NOT place the contact entry in the Contact header in the response. The rules in the Accept-Contact header SHOULD be treated as a hint, and the preferences honored when all else is equal.

In the case when the server has a simple, preference ordered contact list for the callee, the procedure is as follows.

The server first removes any contact entry from the contact list that

matches a rule in the Reject-Contact header field.

A contact entry may contain a priority parameter. This means that a request should not be proxied or redirected to that contact entry unless the request is of equal or higher priority. The priority value of the request is derived from the Priority header field. If the request does not contain a Priority header field, the request priority is set to "non-urgent". Priorities are ordered from "non-urgent" (lowest), "normal", "urgent" to "emergency" (highest). Priority values not known to the server are mapped to "non-urgent". The server then removes any contact entry whose priority value is higher than that of the request.

Each rule in the Accept-Contact header field is then processed. If the rule matches a contact entry, the q value of that entry is updated, in order to incorporate the caller's preferences. If the rule does not match a contact entry, nothing is done. This document does not prescribe a certain algorithm for updating. Among many possibilities, a server MAY set the q value to the average of the original value specified by the callee, and the average q value of the caller's rules that match the contact entry. This gives equal weight to caller and callee preferences. If a rule or contact entry does not have a q value, it is taken to be one (this is in agreement with the HTTP defaults).

> Note that this preference computation only determines the ordering of call attempts, so that the properties of the preference computation are of secondary importance. The q-value ordering provides only limited flexibility to indicate, for example, that a particular parameter is more important than another one or that combinations of two parameters should be weighed heavily.

If the server proxies, the contact list is then sorted according to the q value. The server first attempts to contact those with the highest q value in parallel. If these contact entries do not respond with a 2xx or 6xx response, the server tries the entries with the next-highest q value.

> Due to round-off errors and the computation of joint preferences, there may be an excessive bias here towards serialization rather than parallel attempts. Maybe a server should group all q values within, say, 0.1 into a single parallel-search group.

If the server receives a redirection, and elects to recurse

(depending on its configuration and the preference specified in the Request-Disposition header field), the contact addresses are added to the contact list and the algorithm continues. Note that this may cause addresses that were added by redirection to be tried before contact entries in the original contact list.

5.3 IANA Registration

New URI parameters and values can be defined at any time and registered with IANA. When registering new parameters and values, the following information MUST be provided:

Contact: Name, organization, email address, and phone number of person registering the attributes.

Attributes: A list of the new attributes being registered. For each, the meaning of the attribute must be described, in sufficient detail so that a user agent would be able to ascertain whether the parameter applies to it, and if so, which value to use. The attributes MUST also be associated with a finite set of values, each of which is a valid unicode string. For each value, a description of the value must be provided.

5.4 Use with REGISTER

A user agent normally registers with one or more servers, providing each with a list of Contact addresses. The user agent MAY add the parameters cp-param described in Section 6.2 to the Contact header field.

Furthermore, the REGISTER request MAY contain a Require header with the name of this extension if the client wants to be sure that the server honors callee preferences.

6 Header Field Definitions

The table below specifies which requests can contain which headers.

Since all three headers specify call routing logic, they can apply to any request which can normally be proxied.

6.1 Contact, Accept-Contact and Reject-Contact Parameters

This specification adds the following extension parameters to the Contact header field and defines the same parameters for the Accept-Contact and Reject-Contact header fields.

	type	ACK	BYE	INV	OPT	REG	CAN
Accept-Contact	R	-	o	o	o	o	-
Reject-Contact	R	-	o	o	o	o	-
Request-Disposition	R	-	o	o	o	o	-

Table 1: Summary of header fields. "o": optional "-": not applicable, "R': request header, "r": response header, "g": general header, "*": needed if message body is not empty. A numeric value in the "type" column indicates the status code the header field is used with.

```
cp-params       =  class-param | duplex-param |
                   features-param | language-param | media-param |
                   mobility-param | priority-param | service-param
class-param     =  "class" "=" <"> [<!>] class-tag <">
duplex-param    =  "duplex" "=" <"> [<!>] duplex-tag <">
feature-param   =  "feature" "=" <"> [<!>] 1#feature-tag <">
language-param  =  "language" "=" <"> [<!>] 1#language-tag <">
media-param     =  "media" "=" <"> [<!>] 1#media-tag <">
mobility-param  =  "mobility" "=" <"> [<!>] mobility-tag <">
service-param   =  "service" "=" <"> [<!>] service-tag <">
language-tag    =  primary-tag *( "-" subtag )
primary-tag     =  1*8ALPHA
subtag          =  1*8ALPHA
mobility-tag    =  "fixed" | "mobile"
class-tag       =  "personal" | "business"
duplex-tag      =  "full" | "half" | "receive-only" | "send-only"
service-tag     =  "fax" | "IP" | "PSTN" | "ISDN" | "text"
media-tag       =  ( "*/*" | (type "/" "*") |
                   (type "/" subtype) )
feature-tag     =  "voice-mail" | "attendant"
```

The exclamation mark in the parameter value MUST NOT be included if the cp-params are included in a Contact header.

class: The class parameter indicates whether this terminal is found in a residential or business setting. (A caller may defer a personal call if only a business line is available, for example.)

duplex: The duplex parameter lists whether the terminal can simultaneously send and receive ("full"), alternate between sending and receiving ("half"), can only receive ("receive-only") or only send ("send-only"). Typically, a caller will prefer a full-duplex terminal over a half-duplex terminal and these over receive-only or send-only terminals.

features: The feature list enumerates additional features. It is assumed that these features are orthogonal, and could occur in any combination. "Voice-mail" includes the recording of any multimedia stream, as appropriate.

language: The language parameter lists the languages spoken by the caller or callee. This feature may, for example, be used to have a caller automatically be directed to the appropriate attendant or customer service representative. Note that this parameter has a different functionality than the Accept-Language and Content-Language header fields, which describe the acceptable languages and languages used in the request and the media description, not the actual communications.

media: The media tag lists the media types supported by the terminal. Media types can be the standard Internet media types ("audio", "video", "text", "application"), optionally followed by a subtype (e.g., "text/html"). In addition, the type "application/email" is defined.

mobility: The mobility parameter indicates if the terminal is fixed or mobile. In some locales, this may affect audio quality or charges.

service: The service tag describes what service is being provided by the terminal. The "text" service refers to a device that can send or receive unformatted ASCII text, such as a pager or a TTY. The "IP" tag indicates a device, such as a personal computer, with Internet connectivity. The "fax" tag indicates a fax machine. The "PSTN" tag indicates a telephone connected to the public switched telephone network. The "ISDN" tag indicates a telephone connected to the Integrated Services Digital Network.

The service tags were chosen to maximize the orthogonality of the mobility and service parameters.

In addition, the Contact header field may contain the description-param and priority-param parameters. The description parameter further describes, as text, the terminal. The user agent MAY present this text when it is contained in a Contact header field in a 3xx response. The description parameter MUST NOT be used in the matching operation described above. The priority parameter indicates the minimum priority level this terminal is to be used for. It can be used for automatically restricting the choice of terminals available to the user, as described in the procedure above.

```
priority-param     =  "priority" "=" <"> priority-tag <">
description-param  =  "description" "=" quoted-string
priority-tag       =  "emergency" | "urgent" | "normal" | "non-urgent"
```

The first example below describes a SIP terminal whose owner speaks English, Spanish and German. The terminal is capable of sending and receiving audio and video and can participate in a chat session. However, the owner only wants callers to use the terminal if the call is of priority "urgent" or higher. This Contact header would normally be included in a REGISTER message.

```
Contact: Carol <sip:carol@example.com> ;language="en,es,de"
  ;media="audio,video,application/chat"
  ;duplex="full"
  ;priority="urgent"
```

6.2 Accept-Contact

The syntax for the Accept-Contact header is defined below:

```
Accept-Contact   =  "Accept-Contact" ":" 1# rule
rule             =  ( name-addr | addr-spec )
                    [ *( ";" (cp-params | extension-param | q-param) ) ]
q-param          =  "q" "=" qvalue
extension-param  =  extension-name "=" extension-value
extension-name   =  token
extension-value  =  token | quoted-string
```

The header field specifies contact addresses which are acceptable to the caller.

In the following example, the caller would prefer not to talk to sales@acme.com She has a slight preference for fixed as opposed to mobile phones.

```
Accept-Contact: sip:sales@acme.com ;q=0,
  ;media="!video" ;q=0.1,
  ;mobility="fixed"  ;q=0.6,
  ;mobility="!fixed" ;q=0.4
```

6.3 Reject-Contact

The Reject-Contact header field specifies a list of URIs that the caller does not wish to communicate with. The BNF for the header is:

```
Reject-Contact  =  "Reject-Contact" ":"
                   1# ( ( name-addr | addr-spec )
                   [ *( ";" new-params ) ] )
```

6.4 Request-Disposition

The Request-Disposition header field specifies caller preferences for how a proxy or user agent server should process a request. Its value is a list of tokens, each of which specifies a particular feature.

When the caller specifies a feature, the server SHOULD treat it as a hint, not as a requirement and MAY ignore the feature request.

The header field has the following syntax:

```
Request-Disposition  =  "Request-Disposition" ":"
                        1# (proxy-feature | cancel-feature |
                        fork-feature | recurse-feature |
                        parallel-feature | queue-feature)
proxy-feature        =  "proxy" | "redirect"
cancel-feature       =  "cancel" | "no-cancel"
fork-feature         =  "fork" | "no-fork"
recurse-feature      =  "recurse" | "no-recurse"
parallel-feature     =  "parallel" | "sequential"
queue-feature        =  "queue" | "no-queue"
```

proxy-feature: This feature indicates whether the caller would like each server to proxy or redirect.

cancel-feature: This feature indicates whether the caller would like each proxy server to send a CANCEL request downstream in response to a 200 OK from the downstream server, or whether this function should be left to the caller.

fork-feature: This feature indicates whether a proxy should fork a request, or proxy to only a single address. If the server is requested not to fork, the server should proxy the request to the "best" address. The feature is ignored if "redirect" has been requested.

recurse-feature: This feature indicates whether a proxy server receiving a 300-class response should send requests to the addresses listed in the response (i.e., recurse), or forward the list of addresses upstream towards the caller. The feature is ignored if "redirect" has been requested.

parallel-feature: For a forking proxy server, this feature indicates whether the caller would like the proxy server to proxy the request to all known addresses at once, or go through them sequentially, contacting the next address only after it has received a non-200 or non-600 final response for the previous one. The feature is ignored if "redirect" has been requested.

queue-feature: If the called party is temporarily unreachable, e.g., because it is in another call, the caller can indicate that it wants to have its call queued rather than rejected immediately. If the call is queued, the server returns "182 Queued". A pending call be terminated by a SIP CANCEL or BYE request.

The normal Proxy-Require/Require/Unsupported mechanism is used to indicate to the caller and/or downstream proxies that a particular service is required to complete the request. Otherwise, the service indication is to be taken as a hint.

Example:

```
Request-Disposition: proxy, recurse, parallel
```

7 Acknowledgements

Parameters of the terminal negotiation mechanism in Section 6.1 were influenced by Scott Petrack's CMA design. Jonathan Lennox provided helpful comments.

8 Bibliography

[1] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[2] J. Lennox, J. Rosenberg, and H. Schulzrinne, "Common gateway interface for SIP," Internet Draft, Internet Engineering Task Force, Nov. 1998. Work in progress.

[3] H. Schulzrinne and J. Lennox, "Call processing language requirements," Internet Draft, Internet Engineering Task Force, Aug. 1998. Work in progress.

[4] H. Schulzrinne and J. Rosenberg, "SIP call control services," Internet Draft, Internet Engineering Task Force, Feb. 1998. Work in progress.

[5] G. Klyne, "A syntax for describing media feature sets," Internet Draft, Internet Engineering Task Force, Dec. 1998. Work in progress.

Full Copyright Statement

Table of Contents

1 Terminology 1
2 Introduction 1
3 Design Alternatives 2
4 Overview 3
5 Determining Request Forwarding 4

5.1 Matching Rules 4
5.2 Contact List Processing 7
5.3 IANA Registration 9
5.4 Use with REGISTER 9
6 Header Field Definitions 9
6.1 Contact, Accept-Contact and Reject-Contact Parameters 9
6.2 Accept-Contact 12
6.3 Reject-Contact 13
6.4 Request-Disposition 13
7 Acknowledgements 14
8 Bibliography 14