SIP Caller Preferences and Callee Capabilities

STATUS OF THIS MEMO

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

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

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

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

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

Abstract

This document describes a set of extensions to SIP which allow a caller to express preferences about request handling in servers. These preferences include the ability to select which URIs a call gets proxied or redirected to, and to specify certain request handling directives in proxies and redirect servers. It does so by defining three new request headers, Accept-Contact, Reject-Contact and Request-Disposition, which specify the callers preferences. The extension also defines new parameters for the Contact header. These extra parameters are present in the Contact header in REGISTER requests, and are used to associated attributes with particular addresses.

1 Introduction

When a SIP [1] server receives a request, there are a number of
decisions it can make regarding processing of the request. These include

- whether to proxy or redirect the request;
- which URIs to proxy or redirect to;
- whether to fork or not;
- whether to search recursively or not;
- whether to search in parallel or sequentially;

The server can base these decisions on any local policy. This policy can be statically configured, or can be based on programmatic execution or database access.

However, the administrator of the server is not the only entity with an interest in call processing. There are at least three parties which have an interest: (1) the administrator of the server, (2) the callee, and (3) the caller. The directives of the administrator are embedded in the policy of the server. The preferences of the callee can be expressed most easily through a script written in the call processing language (CPL) \[2\]. However, no mechanism exists to incorporate the preferences of the caller. This extension fills that gap by specifying mechanisms by which a caller can provide preferences on processing of a call. These preferences include the ability to select which URIs a call gets proxied or redirected to, and to specify certain request handling directives in proxies and redirect servers. It does so by defining three new request headers, Accept-Contact, Reject-Contact and Request-Disposition, which specify the callers preferences. The extension also defines new parameters for the Contact header. These extra parameters are present in the Contact header in REGISTER requests, and are used to associated attributes with particular addresses.

2 Overview of Operation

This extension defines a set of additional parameters to the Contact header. These parameters specify attributes that define the characteristics of the UA at the address in the header. For example, there is a mobility parameter which indicates whether the UA is fixed or mobile. When a UA registers, it places these parameters in the Contact headers to characterize the URIs it is registering. This allows the proxy to have information about the contact addresses for a user.

When a caller sends an INVITE, it can optionally include new headers
which request certain handling at a proxy. These preferences fall into two categories. The first category, carried in the Request-Disposition header, describe desired server behavior. This includes whether the caller wishes the server to proxy or redirect, and whether sequential or parallel search is desired. These preferences can be applied at every proxy or redirect server on the call signaling path.

The second category of preferences are carried in both the Accept-Contact and Reject-Contact headers. These preferences contain rules that describe the set of desired URIs that the caller would like the server to proxy or redirect to. These rules are matched against the Contact headers sent in a registration (or through some other configuration means). If a rule in a Reject-Contact header matches a Contact header, that address is not proxied or redirected to. If a rule in an Accept-Contact header matches a Contact header, the q values in the rule are combined with the q values in the Contact header, resulting in a "merged" q value. This merged q value is then used by the proxy to determine the ordering of addresses to proxy or redirect to.

Note that this second category of preferences can only be applied at a proxy which accesses a registration database.

3 Design Alternatives

There are a number of alternatives for expressing caller preferences. Ideally, caller preferences, callee preferences, and administrator preferences "meet" at each server which makes processing decisions. In practicality, a callee cannot install logic at each server in the network. It can only do so (using the CPL, for example), at those servers with which it has some kind of established trust relationship. These servers are those whose main goal is to provide services for the callee.

One might try to place caller logic at these "callee servers" in much the same way the callee places logic there - through the CPL or some other programmatic directives. However, this is also infeasible. A caller cannot apriori install logic in every server for every individual he might call.

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 negotiation has also been approached in a more general way by [3]. 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 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 [4] and indicate requirement levels for compliant SIP caller preferences implementations.

5 Header Field Definitions

Table 5 specifies an extension of Table 5 in RFC 2543 [1] for the three new headers defined here.

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

5.1 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. These parameters apply to a
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.

single URI. When used in a Contact header, they specify characteristics of that URI. When used in the Accept-Contact or Reject-Contact headers, they specify rules to apply for matching URIs.

cp-params = class-param | duplex-param |
            features-param | language-param | media-param |
            mobility-param | priority-param | scheme-param |
            other-param

class-param = "class" "=" "<" [<!>] 1#class-value ">

duplex-param = "duplex" "=" "<" [<!>] 1#duplex-value ">

feature-param = "feature" "=" "<" [<!>] 1#feature-value ">

language-param = "language" "=" "<" [<!>] 1#language-tag ">

media-param = "media" "=" "<" [<!>] 1#media-value ">

mobility-param = "mobility" "=" "<" [<!>] 1#mobility-value ">

scheme-param = "scheme" "=" "<" [<!>] 1#scheme ">

other-param = other-name "=" "<" [<!>] 1#other-value ">

mobility-value = "fixed" | "mobile" | other-value

class-value = "personal" | "business" | other-value

media-value = ( "/*/*" | (type "/" "*")) |
              (type "/" subtype) 

feature-value = "voice-mail" | "attendant" | other-value

other-name = UTF8-TOKEN

other-value = UTF8-TOKEN

UTF8-TOKEN = <any UTF-8 character encoding
              except separator, CTL, and LWS>

The BNF and semantics of the language-tag are defined in Section 3.10 of RFC 2616 [5]. Note, however, that in their usage here they are case sensitive, and MUST appear as all lowercase. Also note that there MUST NOT be any linear white space between the tokens and...
quoted strings of the media-value. This is to align with HTTP 1.1 [5].

The exclamation mark in the parameter value MUST NOT be included if the cp-params are included in a Contact header. The scheme-param parameter MUST NOT be present if cp-params is included in a Contact header. Most importantly, there MUST NOT be more than one class-value, duplex-value, or mobility-value when cp-params is included in a Contact header. These parameters refer to attributes which are mutually exclusive. As a result, a URI can only have one as a characteristic, whereas a rule in the Accept-Contact or Reject-Contact can specify more than one.

The parameters and their values have the following meanings:

class: The class parameter indicates whether the UA 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 UA can simultaneously send and receive media ("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 UA over a half-duplex UA and these over receive-only or send-only UAs.

features: The feature parameter enumerates additional features of the UA. It is assumed that these features are orthogonal, and could occur in any combination. "voice-mail" means that an automated system exists at this UA, which is capable of recording messages. "attendant" means that a human operator is available to take messages.

language: The language parameter lists the languages spoken by user or system behind the UA. This parameter 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 parameter lists the media types supported by the UA. In this context, supported means that the media type is acceptable as part of the media session established by SIP (and usually described by SDP [6]). It does not refer to the media types which can be supported within the bodies of SIP messages. Media types can be the standard Internet media types ("audio", "video", "text", "application"), optionally followed by a subtype (e.g., "text/html").

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

scheme: The scheme parameter describes the set of URI schemes which the caller is willing to accept redirects to or communicate with. The BNF for scheme is given in RFC 2396 [7], and can be any valid URI scheme. The scheme parameter MUST NOT appear as a Contact parameter. It is only permitted as an Accept-Contact or Reject-Contact parameter.

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 UAC 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 in Section 6.3.1.

The priority parameter indicates the minimum priority level this UA is to be used for. It can be used for automatically restricting the choice of terminals available to the caller. The priority parameter is not used in the matching operation described in Section 6.3.1. Its application is described in the procedure in Section 6.3.2.

```
priority-param        =  "priority"  "="  priority-value  ""
description-param    =  "description" "="  quoted-string
```

Note that priority-value is defined in section 6.25 of [1].

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.

5.2 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 | q-param ) ]
q-param         =  "q" "=" qvalue
```

The header field specifies contact addresses which are acceptable to
the caller. If a "*" is specified instead of a name-addr or addr-
spec, it means the UAC doesn’t care about the URI of the user
eventually reached. Only the parameters of the Contact header are
important. If the name-addr or addr-spec is present, and the userinfo
field of the SIP URL is not present, it means the UAC doesn’t care
about the username of the user eventually reached. If the host
portion of the SIP URL is a hostname, and has the value "x", it means
the UAC doesn’t care about the host portion of the URI eventually
reached. If the name-addr or addr-spec is present, and contains URI
parameters, if means the UAC wishes to be connected to an address
that has been registered with these parameters.

We use "x" as the wildcard domain because of the URI
formatting constraints. The domain must be present in a SIP
URL, and cannot be the "*" character. The "x" character is
allowed and looks kind of similar.

In the following example, the caller would prefer not to talk to
sales@acme.com later. 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

In the next example, the caller would prefer to speak to someone from
sales.org that supports video:

Accept-Contact: sip:sales.org;media="video"

5.3 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 | "**") [*(";" cp-params)])

If name-addr or addr-spec is not present (the "**" is present), it means the UAC does not care about the particular user or domain the request is routed to. The cp-params are used to filter out contact addresses based on their parameters alone. This process is described in Section 6.3.1. If either name-addr or addr-spec is present, and the URI does not contain a userinfo field, it means the UAC does not have a preference regarding the user name and/or password of the UA eventually reached. If domain of the URI is equal to "x", it means the UAC does not have a preference regarding the domain of the UA eventually reached.

5.4 Contact Header

The cp-params parameter is allowed as an extension attribute to the Contact header, along with the priority-param and description-param. This effectively means that the BNF for extension-attribute, defined in Section 6.13 of RFC 2543 [1] can be redefined as:

extension-attribute = (cp-params | priority-param | description-param | (extension-name ["=" extension-value]))

The 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.
5.5 Request-Disposition

The Request-Disposition header field specifies caller preferences for how a 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 | \\
  ring-feature)
```

- **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 (generally the one with the highest q value). 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.

ring-feature: In certain cases, the caller is an administrator who wishes to convey an important announcement. In this case, the UAS should not alert the user and should automatically accept the call. If the UAC does not wish the user to be alerted, the "no-ring" token is included. Requests containing this token SHOULD be authenticated, and only privileged users should be allowed to cause calls to be accepted without alerting the called party.

Example:

Request-Disposition: proxy, recurse, parallel

6 Protocol Semantics

6.1 UAS Behavior

User agent servers MAY include cp-params, priority-param or
description-param parameters as part of each Contact addresses they register. These parameters can be set through configuration, user input, or any means the implementor seeks to use. They SHOULD reflect actual characteristics of the URLs being registered.

Furthermore, the REGISTER request MAY contain a Require header with the option tag "pref" if the client wants to be sure that the registration server honors caller preferences.

When a UAS receives a request with the Accept-Contact, Reject-Contact and Request-Disposition, it MAY ignore these headers so long as it does not redirect the request. If the request is redirected, the UAS SHOULD follow the rules described in Section 6.3 for a proxy/redirect server.

6.2 UAC Behavior

A UAC wishing to express preferences for a request includes the Accept-Contact, Reject-Contact, or Request-Disposition headers in the request, depending on its particular preferences. No additional behavior is required after the request is sent.

If the client wants to be sure that servers understand the headers described in this specification, it MAY include a Proxy-Require and Require option tag of "pref". However, this is NOT RECOMMENDED, as it leads to interoperability problems. In any case, client preferences can only be considered as preferences - there is no guarantee that the requested service or capability is executed. As such, inclusion of Proxy-Require and Require does not mean the preferences will be executed.

6.3 Proxy Behavior

The behavior described here assumes a server (proxy or redirect) has received a valid request with either the Accept-Contact or Reject-Contact headers, and that this proxy has a list of Contact headers obtained from looking up the Request-URI in the location service. The location service may have obtained this data through registrations, as described in Section 6.1, but other means may exist.

The processing depends heavily on a rule matching operation. This operation takes a rule (defined as a single element from the comma separate list of elements in the Accept-Contact or Reject-Contact headers), and matches it against the contact list obtained from the location service.

6.3.1 Rule Matching Procedures
The contact list is composed of a set of contact entries. Each contact entry consists of a URI along with a set of parameters. A rule, like a contact entry, consists of a URI (or the "*" character), and a set of parameters. To determine if a rule matches a contact entry, the following steps are taken:

- If the rule does not contain a URI, only parameters are compared. If the rule contains a URI, both the URI and parameters must match.

- If the URIs in the rule and contact entry are not SIP URLs, matching is based on the URL matching rules for the particular scheme. If they are both SIP URLs, the following procedure is followed to determine a match:

  - If the rule contains a userinfo field, that userinfo field must match the userinfo field in the URI in the contact entry. Matching is based on case sensitive string comparison. Note that if the rule contains a userinfo field, but the URI in the contact address does not, the rule does not match the contact entry. If the rule does not contain a userinfo field, no comparison is made on this portion of the URI.

  - If the rule contains a domain not equal to "x", the domain must match the domain the URI in the contact entry. Matching is based on case insensitive string comparison. If the rule has a domain equal to "x", no comparison is made on this portion of the URI.

  - If the rule contains URI parameters (port is considered a URI parameter for purposes of this discussion), each parameter in the rule must match a parameter in the URI in the contact entry. Matching is based on case sensitive string comparison of both parameter names and values. Note, however, if the rule contains a parameter with a default value, this matches a contact entry which does not contain this parameter. Similarly, a rule without a parameter matches a contact entry which contains a default value for that parameter.

- The parameters in the rule must match the parameters in the contact entry. Parameter names are matched by case-sensitive comparison. Parameter values are compared by set-comparisons. Parameter values in quoted strings are interpreted as sets, with elements separated by commas. Two elements match if they are equal based on a case sensitive string comparison. 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 of the rule set with the contact set is non-empty. Note that this process does not apply to the priority-param, description-param or scheme-param.

Case sensitive comparisons are necessary because of internationalization. Case insensitive matching in UTF-8 depends on regional rules, and overly complicates the procedure.

- If there is a scheme-param in the rule, and the rule starts with an exclamation mark, the scheme in the URI in the contact entry must not match any of the schemes listed in the rule. If the rule in the scheme-param doesn't start with an exclamation mark, the scheme in the URI in the contact entry must match one of the schemes listed in the rule. Matching of schemes is done by case insensitive string comparison [7].

The pseudo-code below describes the matching procedure between a rule and a contact entry. The function intersect() takes two arrays of strings, and returns true if there are any values common to both arrays, false otherwise. The function getparameterbyname() takes a rule and a string defining a parameter name. It returns a parameter from the rule with that name.

```c
struct uri_t {
    userinfo_t userinfo;
    host_t host;
    parameters_t params;
}

typedef struct {
    char *name;           /* parameter name */
    boolean exclamation   /* whether ! was present in value */
    char *values[];        /* list of elements in the value */
} parameter_t;

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

boolean MATCH(rule r, entry e) {
    boolean match;
```
parameter_t p, q;

match = TRUE;
if (r.URI != "*") {
    if (r.URI.scheme == e.URI.scheme) {
        if (r.URI.scheme == "sip") {
            match = (r.URI.host == "x" || (strcmp(r.URI.host, e.URI.host) == 0)) &&
                    (r.URI.user == "" || (strcmp(r.URI.user, e.URI.user) == 0))
        } else {
            match = scheme-appropriate comparison;
        }
    } else {
        return FALSE;
    }
}

if(match == FALSE) return FALSE;

/* compare parameters */
for(i = 0; r.para[i] != NULL; i++) {
    p = r.para[i];

    /* is this parameter defined in the contact entry */
    if ((q = getparameterbyname(e, p.name)) != NULL) {
        /* is this an empty set match */
        if (p.exclamation == TRUE) {
            if (intersect(p.values, q.values) == TRUE) {
                return FALSE;
            }
        } else {
            /* not an empty set case */
            if (intersect(p.values, q.values) == FALSE) {
                return FALSE;
            }
        }
    }
    return TRUE;
}

For example, the rule:
sip:dynamicoft.com;language="!en,de"

matches the contact entry:
sip:jdrosen@dynamicsoft.com;language="es,nl"

but not any of:
sip:jdrosen@dynamicsoft.com;language="en"
sip:phuang@dynamicsoft.com;language="de,en"
sip:islepchin@dynamicsoft.com;language="en,es,fi"

As another example, the rule
*;duplex="full,half"

matches the contact entry
sip:user@host;duplex="full"

but not
sip:jtoto@dynamicsoft.com;duplex="send-only"

A server need 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.

6.3.2 Contact List Processing

Given the matching rule above, the formal processing rules at the proxy proceeds as follows. The server begins with a contact list for the callee, and a set of rules in the Accept-Contact and Reject-Contact headers.

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 from the contact list 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 specific 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). The only requirement for the updating process is that if a contact entry has a q value of q1, and the q values among the matching rules are q2,q3,..qn, the merged q value, qm, must satisfy:

\[
\min(q_1, q_2, q_3, \ldots, q_n) \leq q_m \leq \max(q_1, q_2, q_3, \ldots, q_n)
\]

For those contact entries which did not match any rule in the Accept-Contact header, their final q value is set to zero.

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. Processing from this point depends on the configuration and policy of the server. If the server elects to do a sequential proxy, it SHOULD try the highest q value contact entry first, trying addresses with decreasing q values as each attempt fails. If the server elects to do a forking proxy, it SHOULD group contact entries with "close" q values together, and try the group with the highest q
value first, then the group with the next lowest q values, and so on. The precise method of the grouping is left to the implementor. A reasonable choice is to round each q value to the nearest tenth, and group those with the same rounded value.

If the server is recursing, it SHOULD apply the caller preferences to the Contact headers returned in the redirect responses. Any contact entries remaining after the application of caller preferences should be added to the list of untried addresses. This list is then resorted based on q values. The server uses this list for subsequent proxy operations.

If the proxy is redirecting, it SHOULD return all entries in the contact list, including those with a zero q value.

If the proxy is executing any other type of policy, as a general guideline, it SHOULD prefer contact entries with higher q values than those with lower q values.

6.3.3 Request-Disposition Processing

If the request contains a Request-Disposition header, the server SHOULD execute the behaviors described by the tokens, unless it has local policy configured to direct it otherwise.

7 Interactions with CPL

When the called party has a Call Processing Language (CPL) [8] script present, feature interactions are introduced. CPL addresses this by allowing the CPL script to control whether caller preferences are applied to the location list or not. CPL also allows the called party to discard certain rules from the caller preferences before their application. For more information, see [8].

8 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 UTF8-TOKEN. For each value, a description of the value must be provided. The registration MUST indicate whether the parameter values are mutually exclusive or not.

9 Changes since -01

- Discussion of procedures for redirects
- Addition of scheme-parameter for redirects
- Changed option tag in Require header to "pref"
- Added extension capabilities to BNF for cp-params and internationalized them
- Consistent use of terms "parameter", "tag", and "value"
- Changed case matching rules from insensitive to sensitive in order to enable internationalization
- Aligned URI matching rules with rfc2543bis
- Clarified that media parameter refers to acceptable media types for the RTP sessions, not to the media parameters acceptable in SIP bodies.
- Removed the service parameter. It was not clear how this could really be used. It contained values like ISDN and PSTN; however, SIP is an IP protocol. The notion it was trying to convey - of connecting to fax machines or phones through gateways, can be supported more easily through the scheme parameter and media type parameters.
- Updated discussion of CPL
- Updated code with more detail
- Reorganized for clarity

10 Security Considerations

The presence of caller preferences in a request has a significant way in which the request is handled at a server. As a result, is is especially important that requests with caller preferences be authenticated. The same holds true for registrations with contact
parameters.

Processing of caller preferences requires set operations and searches which can require some amount of computation. This enables a DOS attack whereby a user can send requests with substantial numbers of caller preferences, in the hopes of overloading the server. To counter this, servers SHOULD reject requests with too many rules. A reasonable number is around 20.

11 Acknowledgements

Parameters of the terminal negotiation mechanism in Section 5.1 were influenced by Scott Petrack's CMA design. Jonathan Lennox and John Hearty provided helpful comments.

12 Author's Addresses

Jonathan Rosenberg
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: jdrosen@dynamicsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: schulzrinne@cs.columbia.edu

13 Bibliography


Full Copyright Statement

Copyright (c) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

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

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
Table of Contents

1 Introduction ........................................ 1
2 Overview of Operation .................................. 2
3 Design Alternatives ................................... 3
4 Terminology ........................................... 4
5 Header Field Definitions ............................... 4
5.1 Contact, Accept-Contact and Reject-Contact Parameters .................................................... 4
5.2 Accept-Contact ...................................... 8
5.3 Reject-Contact ...................................... 9
5.4 Contact Header .................................... 9
5.5 Request-Disposition ................................. 10
6 Protocol Semantics .................................... 11
6.1 UAS Behavior ...................................... 11
6.2 UAC Behavior ...................................... 12
6.3 Proxy Behavior .................................... 12
6.3.1 Rule Matching Procedures ....................... 12
6.3.2 Contact List Processing ........................ 16
6.3.3 Request-Disposition Processing ............... 18
7 Interactions with CPL ................................ 18
8 IANA Registration .................................... 18
9 Changes since -01 .................................... 19
10 Security Considerations ............................. 19
11 Acknowledgements ................................... 20
12 Author’s Addresses .................................. 20
13 Bibliography ........................................ 20