

Network Working Group  
Request for Comments: 4596  
Category: Informational

J. Rosenberg  
P. Kyzivat  
Cisco Systems  
July 2006

## Guidelines for Usage of the Session Initiation Protocol (SIP) Caller Preferences Extension

### Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (2006).

### Abstract

This document contains guidelines for usage of the Caller Preferences Extension to the Session Initiation Protocol (SIP). It demonstrates the benefits of caller preferences with specific example applications, provides use cases to show proper operation, provides guidance on the applicability of the registered feature tags, and describes a straightforward implementation of the preference and capability matching algorithm specified in Section 7.2 of RFC 3841.

## Table of Contents

1. Introduction .....	4
2. Motivations for Caller Preferences .....	5
2.1. One-Number .....	7
2.2. Direct-to-Voicemail .....	7
3. Caller Preference Use Cases .....	8
3.1. Routing of INVITE and MESSAGE to Different UA .....	8
3.1.1. Desired Behavior .....	8
3.1.2. Solution .....	9
3.2. Single Contact Not Matching Implicit Preferences .....	10
3.2.1. Desired Behavior .....	10
3.2.2. Solution .....	10
3.3. Package-Based Routing .....	11
3.3.1. Desired Behavior .....	11
3.3.2. Solution .....	11
3.4. Package Routing II .....	12
3.4.1. Desired Behavior .....	12
3.4.2. Solution .....	13
3.5. Audio/Video vs. Audio Only .....	13
3.5.1. Desired Behavior .....	13
3.5.2. Solution .....	14
3.6. Forcing Audio/Video .....	15
3.6.1. Desired Behavior .....	15
3.6.2. Solution .....	15
3.7. Third-Party Call Control: Forcing Media .....	16
3.7.1. Desired Behavior .....	16
3.7.2. Solution .....	16
3.8. Maximizing Media Overlaps .....	17
3.8.1. Desired Behavior .....	17
3.8.2. Solution .....	17
3.9. Multilingual Lines .....	18
3.9.1. Desired Behavior .....	18
3.9.2. Solution .....	19
3.10. I Hate Voicemail! .....	20
3.10.1. Desired Behavior .....	20
3.10.2. Solution .....	20
3.11. I Hate People! .....	21
3.11.1. Desired Behavior .....	21
3.11.2. Solution .....	21
3.12. Prefer Voicemail .....	22
3.12.1. Desired Behavior .....	22
3.12.2. Solution .....	22
3.13. Routing to an Executive .....	22
3.13.1. Desired Behavior .....	22
3.13.2. Solution .....	22

3.14. Speak to the Executive .....	23
3.14.1. Desired Behavior .....	23
3.14.2. Solution .....	24
3.15. Mobile Phone Only .....	24
3.15.1. Desired Behavior .....	24
3.15.2. Solution .....	24
3.16. Simultaneous Languages .....	25
3.16.1. Desired Behavior .....	25
3.16.2. Solution .....	25
3.17. The Number You Have Called... ..	26
3.17.1. Desired Behavior .....	26
3.17.2. Solution .....	26
3.18. The Number You Have Called, Take Two .....	27
3.18.1. Desired Behavior .....	27
3.18.2. Solution .....	27
3.19. Forwarding to a Colleague .....	28
3.19.1. Desired Behavior .....	28
3.19.2. Solution .....	28
4. Capability Use Cases .....	30
4.1. Web Redirect .....	30
4.2. Voicemail Icon .....	30
5. Usage of the Feature Tags .....	31
5.1. Traditional Cell Phone .....	31
5.2. Traditional Work Phone .....	32
5.3. PC Messaging Application .....	32
5.4. Standalone Videophone .....	33
6. Example of Implementation of Preference and Capability Matching .....	33
6.1. Extracting a Feature Set from a Header .....	34
6.2. Extracting Values from a Feature Parameter .....	35
6.3. Comparing Two Value-Ranges .....	36
6.4. Feature Set to Feature Set Matching .....	36
6.5. Selecting and Ordering Contacts Based on Caller Preferences .....	37
6.5.1. Reject-Contact Processing .....	37
6.5.2. Accept-Contact Processing .....	37
7. Security Considerations .....	38
8. Acknowledgements .....	38
9. Informative References .....	38

## 1. Introduction

The Session Initiation Protocol (SIP) [1] extension for Callee Capabilities [2] describes mechanisms that allow a UA (User Agent) to register its capabilities in a REGISTER request. A caller can express preferences, either explicitly or implicitly, about how that request is to be handled. This is accomplished with the Accept-Contact and Reject-Contact header fields described in Caller Preferences for the Session Initiation Protocol[3].

The caller preferences extension can serve as a useful tool for supporting many applications. However, its generality makes it difficult to use correctly and effectively in any one situation. To remedy that, this document serves as a compendium of examples of the usage of the caller preferences extension.

NOTE: This document is intended to assist the reader in understanding RFCs 3840 and 3841. It is not intended to serve as a substitute for reading those documents. The examples presented in this document cannot be fully understood without awareness of the mechanisms defined in RFCs 3840 and 3841.

First, Section 2 demonstrates the benefits of using caller preferences by describing several concrete applications that are enabled by the extension. Section 3 describes a set of detailed use cases for expressing caller preferences. Each use case presents a situation, describes how caller preferences can be used to handle the requirements for the situation, and verifies that the desired behavior occurs by showing the results of the matching operation. These use cases validate that the caller preferences specification is complete and capable of meeting a specific set of requirements. Since the caller preferences specification predates the SIP change process [4], no requirements document was ever published for it. To some degree, this document "backfills" requirements. However, this is not an academic exercise only, since the use cases described here did result in changes in the caller preferences document as it evolved. These use cases also help implementors figure out how to use caller preferences in their own applications.

Section 4 discusses applications for the callee capabilities specification. Section 5 discusses the example registrations of the feature tags described in [2]. Proper usage of the caller preferences extension depends on proper interpretation of the semantics of these tags. More detail is provided on the tags, and example registrations are included that show typical usage.

Section 6 outlines an implementation approach to the matching algorithm that doesn't require RFC 2533 [6] to be implemented in all its generality.

## 2. Motivations for Caller Preferences

At its core, SIP is a protocol that facilitates rendezvous of users. The caller and callee need to meet up in order to exchange session information, so that they may communicate. The rendezvous process is complicated by the fact that a user has multiple points of attachment to the network. A called user (callee) can have a cell phone, a PDA, a work phone, a home phone, and one of several PC-based communications applications. When someone calls that user, to which of these devices is the call routed?

Certainly, the call can be routed to all of them at the same time, a process known as parallel forking. However, that is not always the desired behavior. Users may prefer that their registered devices be tried in a particular order. As an example, a user might prefer that his cell phone ring first, and if no one answers, that his work phone ring next. Another user might prefer that her cell phone ring first, and then her home and work phones ring at the same time, and then, if no one answers either of those, that the call be forwarded to voicemail. These variations are all referred to as find-me/follow-me features.

SIP supports find-me/follow-me features in many ways. The most basic is through the SIP registration process. Each device at which a user can be contacted registers to the network. This registration associates the device with the canonical name of the user, called the address-of-record (AOR), which is a SIP URI. Each registration can include a preference value, indicating the relative preference for receiving calls at that device, compared to other devices. When someone makes a call to the AOR, proxies compliant to RFC 3261 will try the registered devices in order of preference, unless administrative policy overrides user preferences.

Preference values in SIP registrations can only provide basic find-me/follow-me features. To support more complex features, the Call Processing Language (CPL) [5] has been specified. It is an XML script that provides specific call routing instructions. Users can upload these scripts to the network, instructing the servers how calls should be routed. As an example, a CPL script can instruct a proxy to route a call to the work phone during work hours (9 am - 5 pm) and then to the cell phone after hours, unless the call is from a family member, in which case it always goes to the cell phone.

It is important to note that both CPL scripts and preference values in registrations describe operation of a service from the perspective of the called party. That is, they describe how a call made to the called party should be routed by the network. However, the called party is not the only one with preferences. A caller will also have preferences for how they want their call to be routed. As an example, a caller will often want to reach a user on their cell phone. In the current telephone network, this is accomplished by requiring a user to have a separate number for each device. This way, when a caller wishes to reach the cell phone, they dial the number for the cell phone. This requires users to maintain lists of potential reach numbers for a user, and then select the appropriate one. A far better approach is for a user to maintain a single address-of-record. When someone wishes to reach them on their cell phone, they call the AOR, but indicate a preference for the call to be routed to the cell phone.

A caller may actually have a wide variety of preferences for how a call should be routed. They may prefer to go right to voicemail. They may prefer never to reach voicemail. They may prefer to reach the user on a device that supports video (because a video-conference is desired). They may wish to reach a device that has an attendant who can answer if the user is not there.

The SIP caller preferences extension allows a caller to express these preferences for the way in which their calls are handled. These preferences are expressed in terms of properties of the desired device. These properties are name-value pairs that convey some kind of information about a device. One example is the property "mobility", which can have the values "mobile" or "fixed". When a caller wishes to reach a cell phone, they include information in their call setup request (the INVITE method) which indicates that the call should be routed to a device that has the property "mobility" set to "mobile". When devices register to the network, they include their properties (also known as callee capabilities) as part of the registration. In this way, a proxy can match the caller's preferences against the capabilities of the various devices registered to the user and route the call appropriately.

While this document addresses the preferences of a caller, it does so from the perspective of a SIP User Agent representing the caller. Caller preferences are herein represented via syntactic elements placed in a SIP request. This document does not attempt to address how preferences might be conveyed by a human user to the User Agent. Thus this document is likely to be of most value to the developer of a User Agent.

The caller preferences extension can support a wide variety of call routing applications and features. Two particularly important examples are "one-number" and "direct-to-voicemail".

### 2.1. One-Number

In today's circuit-switched telephony networks, users have multiple devices, and each device is associated with its own phone number. A user will typically list all of these numbers on a business card: cell phone, work phone, home office phone, and so on. Other users need to store and manage all of these numbers. It is difficult to keep these numbers complete and up-to-date. Worse, when you want to call someone, you need to pick a number to try. Sometimes, you want a specific device (the cell phone); and other times, you just want to reach them wherever they are. In the latter case, a user is forced to try each number, one at a time. This is inefficient, and difficult to do while driving, for example.

As an alternative, a user can have a single address. This is the one and only address they give out to other users on their business cards. If a caller wishes to reach that user on their cell phone, they select that one address, and then access a pull-down menu of device types. This menu would include home phone, work phone, and cell phone. The caller can select cell-phone, and then the call is placed to the cell phone. There is no need to manage or maintain more than one number for the user -- a single number will suffice.

If, on the other hand, the caller wishes to reach the user wherever they are, they make a call to that one number without a selection of a preferred device. The network will ring all devices at the same time, and therefore reach the user as fast as possible.

This one-number service makes use of caller preferences. To express a preference for the cell phone, the caller's device would include a header in the SIP INVITE request, indicating a desire to reach a device with "mobility" equal to "mobile".

### 2.2. Direct-to-Voicemail

Frequently, a busy executive on the road wants to quickly pass a message to a colleague by voice. As an example, a boss might want to instruct an employee to call a specific customer and resolve a pending issue. In such a case, the user doesn't actually want to talk to the person; they just want to leave a voice message. Having a phone conversation may require too much time, whereas a voice message can be quick and to the point. The voice message can also serve as a record of exactly what is desired, whereas a fleeting voice conversation can be forgotten or misremembered.

In today's circuit-switched telephone networks, there is often no way to go directly to someone's voicemail and leave a message. Sometimes, you can dial the main number for the voicemail system, enter in the extension of the desired party, and leave a message by entering a specific prompt. This is time consuming, and requires the caller to know the main voicemail number.

Instead, an address book in a cell phone can have an option called "leave voice message", available for each entry in the address book. When this option is selected, a call is made directly to the voicemail for that user, which immediately picks up and prompts for a message. In fact, a rapid greeting is played, so that the caller can go directly to the recording procedure.

This saves time for the caller, making it very easy to quickly leave recorded messages for a large number of people.

This feature is possible using the caller preferences extension. When the user selects the "leave voice message" option, the phone sends a SIP INVITE request, and includes a caller preferences header field that indicates a preference for devices whose "msgserver" attribute has a value of "true". This will cause the proxy to route the call directly to a registered voicemail service. Furthermore, the voicemail server will see that the caller asked to go directly to voicemail, and can therefore play an abbreviated greeting explicitly designed for this case.

### 3. Caller Preference Use Cases

Each use case is described as a situation along with a desired behavior. Then, it demonstrates how the various caller preferences headers and the proxy processing logic would result in the appropriate decision.

#### 3.1. Routing of INVITE and MESSAGE to Different UA

##### 3.1.1. Desired Behavior

Address of Record (AOR) Y has two contacts, Y1 and Y2. Y1 is a phone and supports the standard operations INVITE, ACK, OPTIONS, BYE, and CANCEL but does not support MESSAGE, whereas Y2 is a pager and supports only OPTIONS and MESSAGE. Caller X wants to send pages to Y. There is a lot of traffic in the network of both calls and pages, so there is a goal not to unnecessarily fork messages to devices that can't support them. So, this is done by ensuring that INVITES of Y are delivered only to Y1, while MESSAGES to Y are delivered only to Y2.



### 3.1.2. Solution

Y1 will create a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y1@pc.example.com>
       ;methods="INVITE,ACK,OPTIONS,BYE,CANCEL"
       ;uri-user="<Y1>"
       ;uri-domain="example.com"
       ;audio
       ;schemes="sip"
       ;mobility="mobile"
```

Y2 will create a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y2@pc.example.com>
       ;methods="OPTIONS,MESSAGE"
       ;uri-user="<Y2>"
       ;uri-domain="example.com"
       ;+sip.message
       ;schemes="sip,im"
       ;mobility="mobile"
```

When a UAC (User Agent Client) sends an INVITE, it will arrive at the proxy for example.com. There are no caller preferences in the request. However, per Section 7.2.2 of [3], the proxy will construct an implicit require-flagged Accept-Contact preference that looks like:

```
(& (sip.methods="INVITE"))
```

Applying the matching algorithm of RFC 2533 [6] to this feature set and those registered by Y1 and Y2, the feature set of Y1 alone matches. Because the Accept-Contact predicate has its require flag set, Y2 is discarded, and the INVITE is routed to Y1.

If the request was MESSAGE, the proxy constructs an implicit Accept-Contact preference with its require flag set (require-flagged) that looks like:

```
(& (sip.methods="MESSAGE"))
```

which matches the feature set of Y2, but not Y1. Thus, Y1 is discarded, and the request is routed to Y2.

### 3.2. Single Contact Not Matching Implicit Preferences

#### 3.2.1. Desired Behavior

AOR Y has a single contact, Y1. It's a phone, and therefore supports the standard operations INVITE, ACK, OPTIONS, BYE, and CANCEL but does not support MESSAGE. A caller X sends a MESSAGE request. The desired behavior is that the request is still routed to the solitary contact so that it can generate a 405 response.

#### 3.2.2. Solution

The single contact Y1 will generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y1@pc.example.com>
        ;methods="INVITE,ACK,OPTIONS,BYE,CANCEL"
        ;uri-user="<Y1>"
        ;uri-domain="example.com"
        ;audio
        ;schemes="sip"
        ;mobility="fixed"
        ;class="personal"
```

X sends a MESSAGE request. There are no explicit caller preferences. This results in an implicit require-flagged Accept-Contact preference:

```
(& (sip.methods="MESSAGE"))
```

Since Y1 doesn't match and the Accept-Contact predicate is require-flagged, it is discarded. However, according to section 7.2.4 of RFC 3841, if there are no matching targets, the original target set is used. Thus, the request is sent to the one original target, Y1, as desired. Y1 then responds with a 405.

If there were multiple contacts, and none of them matched the Accept-Contact predicate, then the original target set including all of the contacts would be restored. Then all the contacts would be processed according to Section 16.6 of RFC 3261.

### 3.3. Package-Based Routing

#### 3.3.1. Desired Behavior

AOR Y has a number of contacts, Y1, Y2, ..., Yn, that can each support the standard operations INVITE, ACK, OPTIONS, BYE, and CANCEL and can also support SUBSCRIBE for the "dialog" event package [7]. Y also has another contact, Yp, that is a presence agent (PA) [8]: it can accept only SUBSCRIBE requests for the "presence" event package. The goal is for SUBSCRIBE requests for presence to be routed to Yp while INVITES and SUBSCRIBES for the dialog package are forked to Y1...Yn.

#### 3.3.2. Solution

Y1..Yn will generate REGISTER requests that look like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Yi@pc.example.com>
        ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL,SUBSCRIBE"
        ;events="dialog"
        ;uri-user="<Yi>"
        ;uri-domain="example.com"
        ;audio
        ;schemes="sip"
        ;mobility="fixed"
        ;class="personal"
```

and Yp will generate a REGISTER request that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Yp@pc.example.com>;methods="SUBSCRIBE"
        ;events="presence"
        ;uri-user="<Yp>"
        ;uri-domain="example.com"
        ;schemes="sip,pres"
        ;mobility="fixed"
        ;class="business"
```

A SUBSCRIBE request for presence will arrive at the proxy for example.com. Since there are no explicit preferences, it constructs an implicit require-flagged Accept-Contact preference from the request:

```
(& (sip.methods="SUBSCRIBE") (sip.events="presence"))
```

Following Section 7.2.4 of RFC 3841, this feature set only matches the one registered by Yp. Because the require flag is set, the contacts which do not match are removed from the target set. Therefore, Y1..Yn are discarded. The request is sent to the remaining contact, Yp, representing the PA.

An INVITE request without explicit preferences results in an implicit require-flagged Accept-Contact preference:

```
(& (sip.methods="INVITE"))
```

The implicit Accept-Contact feature set matches Y1..Yn, but does not match Yp. Using the scoring algorithm from Section 7.2.4 of RFC 3841, the score for Y1..Yn against this predicate is 1.0. As a result, the caller preference Qa for each contact is 1.0. The registrations did not contain q-values, so the default q-value of 1.0 is applied to each Contact URI. Since the caller and callee preferences are the same and all equal to 1.0, there is no reordering of contacts. The result is that the proxy will consider Y1..Yn each as equally good targets for the request and possibly fork the request to each.

A SUBSCRIBE request for the dialog event package without explicit preferences will result in an implicit require-flagged Accept-Contact preference:

```
(& (sip.methods="SUBSCRIBE") (sip.events="dialog"))
```

This only matches Y1..Yn, so Yp is discarded, and the request is routed to the remaining contacts just as the INVITE was.

### 3.4. Package Routing II

#### 3.4.1. Desired Behavior

This case is nearly identical to that of Section 3.3. However, Y1..Yn omit the "events" feature tag from their registration. Yp registers as in Section 3.3. A SUBSCRIBE for the presence event package should still preferentially route to Yp.

### 3.4.2. Solution

The registration from Y1..Yn will look like:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Yi@pc.example.com>
        ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL,SUBSCRIBE"
        ;uri-user="<Yi>"
        ;uri-domain="example.com"
        ;audio
        ;schemes="sip"
        ;mobility="fixed"
        ;class="personal"
```

When the caller sends a SUBSCRIBE for the presence event package (without explicit preferences), the proxy computes an implicit preference:

```
(& (sip.methods="SUBSCRIBE") (sip.events="presence"))
```

This predicate matches Y1..Yn and Yp. However, the score for Y1..Yn against this predicate is 0.5, and the score of Yp is 1.0. The result is a caller preference Qa of 0.5 for Y1..Yn, and a caller preference Qa of 1.0 for Yp. Since the callee provided no q-values, the proxy will assume a default of 1.0. Thus, all contacts are in the same equivalence class. They are then sorted by Qa, so that Yp is first, followed by Y1 through Yn. It will therefore route the request first to Yp, and if that should fail, to Y1..Yn.

## 3.5. Audio/Video vs. Audio Only

### 3.5.1. Desired Behavior

X sends an invitation to Y to initiate an audio/video call, including both m=audio and m=video lines in the SDP. AOR Y has two contacts, Y1 and Y2. Y1 represents a normal audio phone, where Y prefers to receive their calls. It will answer an audio/video call, refusing the video. Y2 represents an audio/video phone that should only be used when needed. The caller really wants the call answered by a device that supports video, but will accept an audio-only call as a second choice.

### 3.5.2. Solution

Y1 will generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y1@pc.example.com>;q=1.0
        ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
        ;uri-user="<Y1>"
        ;uri-domain="example.com"
        ;audio
        ;schemes="sip,tel"
        ;mobility="fixed"
        ;class="business"
```

Y2 will generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y2@pc.example.com>;q=0.6
        ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
        ;uri-user="<Y2>"
        ;uri-domain="example.com"
        ;audio
        ;video
        ;schemes="sip,tel"
        ;mobility="fixed"
        ;class="business"
```

Note the different q-values, allowing Y2 to be selected as a device of "last resort".

To have the call preferentially routed to a device that supports video, the caller X sends an INVITE that looks like, in part:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *
        ;methods="INVITE"
        ;video
```

The proxy will convert this to a feature set. This feature set matches Y2 and Y1. However, the score for Y2 is 1.0, and 0.5 for Y1. The two contacts are then ordered by q-value and broken into equivalence classes. There are two equivalence classes, each with one contact. As a result, the caller preference values have no impact on the ordering. The call will first try the higher priority Y1, which will answer the call and reject the video stream. Thus, the desired behavior is not achieved.

The desired behavior could be achieved by adding the "explicit" and "require" tags to the Accept-Contact header field in the INVITE, as is done in Section 3.6. However, doing so may result in calls failing when they could occur, but without video. As discussed in [3], both the "require" and "explicit" tags are generally used only when the request cannot be serviced in any way unless the preferences are met. That is not the case here.

### 3.6. Forcing Audio/Video

#### 3.6.1. Desired Behavior

This case is similar to that of Section 3.5. However, X requires an audio/video call and would like the call to fail if this is not possible, rather than succeed with audio only.

#### 3.6.2. Solution

The solution is similar to that of Section 3.5; however, the Accept-Contact header field now includes the "explicit" and "require" tags, guaranteeing that the call is never established to any UA that had not explicitly indicated support for video:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;video;require;explicit
```

This arrives at the example.com proxy. This explicit feature set matches the feature set for Y2 and Y1. However, the match for Y1 did not have a score of 1. Since the "explicit" and "require" tags are present, the contact is discarded. That leaves Y2 only. The call will therefore get routed to the videophone, and if the user is not there, the audio phone will never ring.

Because both the "require" and "explicit" flags are present, a contact will also be discarded if it does not include a feature tag indicating support for video. Thus, a UA that can do video, but neglected to indicate it, would not be reached in this case. This is why it is important for a UA to indicate all of its capabilities. Note that this is only true for a contact that indicated some capabilities but not the video capability. Contacts that don't indicate any capabilities are "immune" from caller preferences filtering and would not be discarded.

### 3.7. Third-Party Call Control: Forcing Media

#### 3.7.1. Desired Behavior

Z is a third-party call control controller (3pcc) [9] trying to establish an audio/video call from X to Y. X has contacts X1 and X2, and Y has contacts Y1 and Y2. X1 and X2 have capabilities identical to Y1 and Y2, respectively. Z needs to send an offerless invite to X and use the offer proposed by X to send an invite to Y. When sending the offerless invite to X, the 3pcc controller must ensure that an audio/video contact (X2) is chosen over an audio only contact (X1).

#### 3.7.2. Solution

X1 will generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:X@example.com
Contact: <sip:X1@pc.example.com>;q=1.0
        ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
        ;uri-user="<X1>"
        ;uri-domain="example.com"
        ;audio
        ;schemes="sip,tel"
        ;mobility="fixed"
        ;class="business"
```

X2 will generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:X@example.com
Contact: <sip:X2@pc.example.com>;q=0.6
        ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
        ;uri-user="<X2>"
        ;uri-domain="example.com"
        ;audio
        ;video
        ;schemes="sip,tel"
        ;mobility="fixed"
        ;class="business"
```

Z would include, in its INVITE, an Accept-Contact header field:

```
INVITE sip:X@example.com SIP/2.0
Accept-Contact: */audio;video;require;explicit
```



This caller preference matches both X1 and X2. However, it matches X1 with a score of .5 and X2 with a score of 1. Because of the "require" and "explicit" tags, X1 is discarded despite X's preference for it. Thus, the call is routed to X2.

The same caveats apply here as do in Section 3.6. Generally, it is not advisable to mandate support for features (such as video) that are not strictly necessary for the request to proceed.

### 3.8. Maximizing Media Overlaps

#### 3.8.1. Desired Behavior

AOR Y has two contacts: Y1, which is a regular audio phone, and Y2, which is a PC capable of supporting both audio and session-oriented IM [10]. X is a PC with capability to support audio, video, and session-oriented IM. X calls Y for the purpose of establishing a voice call. However, X wishes to connect to the device that has the maximal overlap with its media capabilities, in order to maximize the functionality available to the caller.

#### 3.8.2. Solution

Y1 will generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y1@phone.example.com>
;methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
;uri-user="<Y1>"
;uri-domain="example.com"
;audio
;schemes="sip,tel"
;mobility="fixed"
;class="business"
```

Y2 will generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y2@pc.example.com>
        ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL,MESSAGE"
        ;uri-user="<Y2>"
        ;uri-domain="example.com"
        ;audio
        ;+sip.message
        ;schemes="sip,tel"
        ;mobility="fixed"
        ;class="business"
```

The solution requires the caller to support caller preferences. The caller would include, in their INVITE, an Accept-Contact header field that lists all the media types they support. In this case:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;audio;video;+sip.message
```

Both Y1 and Y2 match the predicate. Y1 matches with a score of 0.33, and Y2 matches with a score of 0.66. Since there is only one Accept-Contact predicate, the Qa for each contact is equal to the score. The registered contacts are then sorted by q-value and broken into equivalence classes. There is a single equivalence class with q-value of 1.0. The two contacts in that class are then re-ordered based on the values of Qa. Y2 has a higher Qa, so it is used first, followed by Y1. The result is that the call is routed to the device with the maximum overlap in media capabilities, as desired.

Note that neither "require" nor "explicit" tags are used because there is no intent to exclude contacts, only to order them.

### 3.9. Multilingual Lines

#### 3.9.1. Desired Behavior

AOR Y represents a shared line in an office. Several employees in the office have phones registered for Y. Some of the employees speak only English, some speak Spanish fluently and have some limited capability for English, and some speak both English and Spanish fluently. Calls from callers that speak only English should be parallel forked to all office workers that speak fluent English. If the call isn't picked up, then the phones of workers that speak English marginally should be rung. Calls from callers that speak only Spanish should be forked only to workers that speak Spanish.

### 3.9.2. Solution

A user at phone Y1 that speaks English only would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y1@pc.example.com>;languages="en"
```

A user at a phone Y2 that speaks Spanish and a little bit of English would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y2-es@pc2.example.com>;languages="es"
Contact: <sip:Y2-en@pc2.example.com>;languages="en";q=0.2
```

Y2 has registered two contacts. Both of them route to the same device (pc2.example.com), but they differ in their language support and relative q-values. Multiple contacts are needed whenever a UA wishes to express differing preferences for being reached for different feature collections.

A user at phone Y3 that speaks English and Spanish fluently would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y3@pc3.example.com>;languages="es,en"
```

Notice that only a single contact is needed because the same q-value is applied across all feature collections.

For the language-based routing to occur, the caller must indicate its language preferences explicitly:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;languages="en";require
```

The predicate derived from this looks like:

```
(& (languages="en"))
```

This matches the one contact for Y1, the second contact registered for Y2, and the one contact for Y3, all with a score of 1.0. The first contact registered by Y2 does not match, and because of the "require" flag, is discarded. The remaining contacts are sorted by q-value and divided into equivalence classes. There are two

equivalence classes. The first contains Y1 and Y3 with a q-value of 1.0, and the second contains Y2-en with a q-value of 0.2. The contacts in the first class are ordered by Qa. However, since all contacts have the same value of Qa (1.0), there is no change in ordering. Thus, Y1 and Y3 are tried first, followed by Y2-en. This is the desired behavior.

An "explicit" tag is not used because that would cause the exclusion of a contact that does not mention language.

A caller that speaks Spanish only would specify their preference thusly:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;languages="es";require
```

This matches the first contact of Y2 phones, and Y3 phones, all with a score of 1.0. The English contact of Y2, Y2-en, doesn't match and is discarded because of the "require" flag. The remaining contacts are sorted by q-values (Y3, Y2-es) and broken into a single equivalence class containing both contacts. Since the Qa for both contacts is the same (1.0) there is no reordering. The result is that the call is routed to either Y3 or Y2-es.

### 3.10. I Hate Voicemail!

#### 3.10.1. Desired Behavior

AOR Y has two contacts, a phone Y1 and a voicemail service Y2. X wishes to call Y and talk in person. X does not want to be sent to voicemail under any circumstances.

#### 3.10.2. Solution

The phone would register with a Contact that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y1@pc.example.com>
;audio
;mobility="fixed"
```

and the voicemail server would register with a Contact that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y2@pc.example.com>
      ;msgserver
      ;automata
      ;attendant
      ;audio
      ;q=0.2
```

The voicemail server registers with a lower q-value so that it is used only after the phone itself is rung. Note that the voicemail server need not actually register. There can be a configured contact and feature set defined for it instead.

A caller that wishes to avoid voicemail can include an explicit preference to avoid it. A caller would do this with the Reject-Contact header field:

```
INVITE sip:Y@example.com SIP/2.0
Reject-Contact: *;msgserver
```

Since this feature set contains a feature tag that is not contained in the registration for Y1, the feature set is discarded when examining Y1. However, the registration for Y2 contains all feature tags listed in the feature set, and so the rule is considered. There is a match, and therefore, Y2 is discarded. The result is that the user is never routed to voicemail.

### 3.11. I Hate People!

#### 3.11.1. Desired Behavior

The situation is similar to Section 3.10, except the caller wishes only to leave a message, not actually speak to the person.

#### 3.11.2. Solution

The caller would send an INVITE that looks like, in part:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;msgserver;require;explicit
```

This caller preference matches both Y1 and Y2. Y1 matches, but with a score of zero. Y2 matches with a score of 1. Since both the

"require" and "explicit" flags are set, Y1 is discarded. Therefore, the call is routed to Y2, the voicemail server, as desired.

Because of the presence of the "require" and "explicit" tags, if these preferences are used with a user that doesn't have voicemail or that fails to indicate it with a msgserver capability, the call will fail completely with a 480 Temporarily Unavailable error, rather than connect to the user.

### 3.12. Prefer Voicemail

#### 3.12.1. Desired Behavior

The situation is similar to that of Section 3.10. However, the caller prefers to leave a message. If voicemail is not available, they are willing to talk to a person.

#### 3.12.2. Solution

It had been hoped that RFC 3841 could provide a solution for this case, but it does not, because doing so would require a re-ordering of the callee contacts, which is not done. The caller may achieve the intended effect by making two call attempts:

- o First, make an attempt requiring voicemail, as described in Section 3.11.
- o If that fails with a 480 error, send an invitation with no Accept-Contact or Reject-Contact headers.

### 3.13. Routing to an Executive

#### 3.13.1. Desired Behavior

Y is the AOR of an executive. It has three contacts. Y1 is the phone on the executive's desk. Y2 is the phone on the desk of the executive's assistant. Y3 is the address of an auto-attendant system that can answer general questions, route calls to other parties, etc. By default, calls to Y should be directed to Y2, and if that fails, to Y3. If Y3 doesn't answer, then Y1 should ring.

#### 3.13.2. Solution

This is primarily a called party feature and is best accomplished with a CPL (Call Processing Language) script [5]. However, it can be accomplished with caller preferences alone by properly setting the q-values across the three devices. Assuming this coordination is possible, here are the settings that would be made:

Y1 would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y1@pc.example.com>;q=0.1
```

Y2 would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y2@pc2.example.com>;attendant;q=1.0
```

Y3 would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y3@pc3.example.com>;attendant;automata;q=0.5
```

Note that, in reality, the automated attendant would probably not use REGISTER. Since the attendant would be used for every employee in the company, a static contact would probably be added administratively for each user in the enterprise. However, the information in that static contact would be identical to the information in the registration above.

When X makes a call to the executive, Y, and expresses no preference, the proxy computes an implicit preference to support INVITE. All three contacts match such a preference, even though they have not indicated explicit support for INVITE. Thus, no contacts are discarded. Since each contact has a different q-value, the caller preferences do not cause any reordering. The result is that the call is first routed to Y2, then Y3, then Y1, all as a result of the proper setting of the q-values.

### 3.14. Speak to the Executive

#### 3.14.1. Desired Behavior

This case is similar to that of Section 3.13, but this time the caller, X, has a preference. X calls Y, but wants to speak directly to the executive. X doesn't want the call to ring either the assistant or the auto attendant (automaton).

### 3.14.2. Solution

X's INVITE would look like, in part:

```
INVITE sip:Y@example.com SIP/2.0
Reject-Contact: *;attendant
Reject-Contact: *;automata
```

Note that the caller uses two separate Reject-Contact header field values, rather than a single one with two separate feature parameters. The distinction is important. If X had to use a single value with two parameters, a matching UA would need to declare that it was BOTH an attendant and an automaton. If it only declared that it was one of these, based on the matching rules in the caller preferences specification, it would not be rejected.

The above request would result in the elimination of both Y2 and Y3 as contacts. The call would then be routed to Y1, as desired.

This case indicates why a CPL script, or some other programmed version of the feature, is preferable. With caller preferences, a caller can override the desired ring sequence and disturb the executive without any kind of authorization. A proper version of this service would simply not permit caller preferences to force the call to go directly to the executive.

### 3.15. Mobile Phone Only

#### 3.15.1. Desired Behavior

The situation is similar to that in Section 3.13. However, the executive also has a mobile phone that they have registered. Caller X knows that the owner of Y is traveling, and that an assistant is covering the office phone. X wants to call Y and ring only the mobile phone.

#### 3.15.2. Solution

The mobile phone would generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:Y4@mobile.example.com>;mobility="mobile";q=0.1
```



The caller would express their preference by generating an INVITE that looks like, in part:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;mobility="mobile";require;explicit
```

All four contacts match. However, Y1 through Y3 match with a score of zero. Y4 matches with a score of 1. Because of the "require" and "explicit" tags, Y1 through Y3 are discarded, and only Y4 is used, as desired.

Note that this only works if the mobile phone specifies the mobility feature in its registration.

### 3.16. Simultaneous Languages

#### 3.16.1. Desired Behavior

AOR Y is as in Section 3.9. Caller X, fluent in both English and Spanish, has discovered that the company's Spanish language documentation is inconsistent with the English language documentation and wants to discuss the differences between the two. So X wants to speak with one of the workers that is fluent in both English and Spanish.

#### 3.16.2. Solution

The caller would generate an INVITE that looks like, in part:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;language="en";require
Accept-Contact: *;language="es";require
```

This will require a Contact URI to match both constraints. That means it needs to support English and Spanish. This will achieve the desired property.

Note that there are two separate Accept-Contact header fields. If the caller had instead used this INVITE:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;language="en,es";require
```

It would have connected them to a UA that speaks either English or Spanish, which is not what is desired here.

An "explicit" option is not used, because it would bypass contacts that do not include a language tag.

### 3.17. The Number You Have Called...

#### 3.17.1. Desired Behavior

Consider once more the case of the executive, where the caller wishes to reach only their mobile phone (Section 3.15). However, there is a twist. The callee Y has moved to new address YY, and all the configuration described for the callee now applies to YY. The old address Y remains with a pair of statically assigned contacts. One contact is YY. The other is M, referencing an automaton that generates a voice message reporting that the number has been changed. The caller is unaware of the move and calls Y, requesting to reach the mobile phone in exactly the same way they did in Section 3.15. The call should connect to the mobile.

#### 3.17.2. Solution

There would be four registrations against YY:

YY1, the executive, would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:YY@example.com
Contact: <sip:YY1@pc.example.com>;q=0.1
```

YY2, the attendant, would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:YY@example.com
Contact: <sip:YY2@pc2.example.com>;attendant;q=1.0
```

YY3, the answering service, would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:YY@example.com
Contact: <sip:YY3@pc3.example.com>;attendant;automata;q=0.5
```

YY4, the mobile, would generate a REGISTER that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:YY@example.com
Contact: <sip:YY4@mobile.example.com>;mobility="mobile";q=0.5
```

Although it would be configured administratively, there are two registered contacts for Y. The first is for the forwarding:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:YY@example.com>;q=1.0
```

and the second for the automated answering service:

```
REGISTER sip:example.com SIP/2.0
To: sip:Y@example.com
Contact: <sip:machine@example.com>;automata;q=0.5
```

The caller, not knowing that Y has moved, calls Y and asks for their mobile phone:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;mobility="mobile";require;explicit
```

This reaches the example.com proxy, which finds two registrations. Only one of these (the automaton) is associated with feature parameters. The other has no feature parameters and is therefore immune to caller preferences processing. The caller preferences are applied to the automaton's contact. The feature sets match, but have a score of zero. Since the "require" and "explicit" tags are present, the contact for the automaton is dropped. The other contact, YY@example.com, is then added back in as the sole contact. The proxy therefore sends the call to sip:YY@example.com. There, there are four registrations, all of which are associated with feature parameters. The caller preferences are applied. Only YY4 matches explicitly, however. Because of the presence of the "require" and "explicit" flags, all other contacts are dropped. As such, the call is forwarded to YY4, and the mobile phone rings.

### 3.18. The Number You Have Called, Take Two

#### 3.18.1. Desired Behavior

This use case is nearly identical to that of Section 3.17. However, this time, the caller wishes to contact the personal phone of Y. They don't feel strongly about it, and will accept other devices.

#### 3.18.2. Solution

The INVITE generated by the caller in this case will look like:

```
INVITE sip:Y@example.com SIP/2.0
Accept-Contact: *;class="personal"
```

This reaches the example.com proxy. Once more, the first registration (which forwards to the address-of-record for YY) is unaffected by the caller preferences computation. The other contact, for the automaton, is a match, but its score is zero. Its caller preference  $Q_a$  equals zero. The other contact is added back in with a  $Q_a$  of 1.0. The contacts are sorted based on  $q$ -value, resulting in YY ( $q=1.0$ ) followed by machine ( $q=0.5$ ). These are broken into equivalence classes. There are two classes, one for each contact. As a result, the caller's preferences have no impact on the ordering, and the call is routed to YY.

When the request for YY@example.com is processed, all four contacts match. However, the score for all of them is zero (none are the personal phone). As such, the contacts are ordered based on  $q$ -value. Each contact has a different  $q$ -value, so no reordering based on caller preference is possible (not that the caller preference would cause a reordering; all contacts have a  $Q_a$  of 0.0). Thus, the highest  $q$ -value contact is tried, which is the executive assistant.

### 3.19. Forwarding to a Colleague

#### 3.19.1. Desired Behavior

Alice wants to forward her phone to Bob, but doesn't want folks calling her to get Bob's voicemail if he doesn't answer. She wants her callers to get her voicemail.

#### 3.19.2. Solution

Alice would create three registrations. The first, Y1, represents Alice's phone. The second is Bob's AOR. The third is a voicemail server. The three contacts have decreasing  $q$ -values. The registration for Bob's AOR contains an embedded Reject-Contact header field, which rejects message servers.

```
REGISTER sip:example.com
To: <sip:alice@example.com>
Contact: <sip:Y1@192.0.2.150>;q=1.0
```

```
REGISTER sip:example.com
To: <sip:alice@example.com>
Contact: <sip:bob@example.com?Reject-Contact=*;msgserver>;q=0.3
```

```
REGISTER sip:example.com
To: <sip:alice@example.com>
Contact: <sip:alice-drop@msgcenter.example.com>
        ;msgserver;
        ;automata
        ;attendant
        ;q=0.1
```

Meanwhile, Bob is registered as follows:

```
REGISTER sip:example.com
To: <sip:bob@example.com>
Contact: <sip:bob3@192.0.2.212>;q=0.8
```

```
REGISTER sip:example.com
To: <sip:bob@example.com>
Contact: <sip:bob-drop@msgcenter.example.com>
        ;msgserver
        ;automata
        ;attendant
        ;q=0.2
```

Carol calls Alice and doesn't include any caller preference parameters. As such, the example.com proxy constructs an implicit preference for INVITE. This preference matches all three registered contacts, with a score of zero. Because each contact has a different q-value, there is no reordering of contacts. So, the proxy tries the highest q-value Contact, Alice's desk phone (Y1). The proxy cancels after a few seconds (no answer). The proxy then tries the next Contact, which is Bob's AOR. When constructing the request for this Contact, the proxy includes the embedded Reject-Contact header field in the INVITE. This INVITE undergoes caller preferences processing based on Bob's registered Contacts.

Bob has two registered Contacts. The second is a message server, and it matches the Reject-Contact in the INVITE. Thus, this contact is discarded. The other remaining Contact, Bob's phone, is tried. Bob is not around, so his phone rings for a while. Upon timeout, the proxy determines it is unable to reach Bob's AOR. So, the proxy handling Alice tries the final remaining contact, which is Alice's message server.

## 4. Capability Use Cases

The callee capabilities spec [2] allows the Contact header field in OPTIONS responses and dialog initiating messages to contain capabilities of the UA. These capabilities can be very useful for developing new applications. In the subsections below, several usages are outlined.

### 4.1. Web Redirect

A caller sends an INVITE to the called party. However, the called party is not present. The proxy server representing the called party would like to redirect the caller to a web page, where they can find out more information on how to reach the called party. However, the proxy needs to know whether or not the caller supports redirects to web pages. If it doesn't, the proxy would connect the user to an interactive voice response (IVR) device, which would execute an answering machine application.

The proxy could make such a determination if the caller included the "schemes" feature tag in the Contact header field of its INVITE:

```
INVITE sip:callee@example.com SIP/2.0
Contact: <sip:host22.example.com>;schemes="http,sip,sips,tel"
```

This tells the proxy that the UAC can be redirected to an http URI. The INVITE from a normal "black phone" that lacked this capability would look like:

```
INVITE sip:callee@example.com SIP/2.0
Contact: <sip:host22.example.com>;schemes="sip,sips,tel"
```

This indicates that it needs to be connected to the IVR.

### 4.2. Voicemail Icon

On the circuit network, when a user makes a call, and an answering machine picks up, the caller usually requires several seconds to determine that they are speaking to an answering machine. It would be helpful if a phone could display an icon immediately on call completion that indicated that an answering machine was reached.

This indication can be provided by the "msgserver" feature parameter. When the answering machine picks up, its 200 OK looks like, in part:

```
SIP/2.0 200 OK
Contact: <sip:server33.example.com>;msgserver;automata;attendant
```

This tells the caller that it's an answering machine.

## 5. Usage of the Feature Tags

The caller preferences extension briefly enumerates a list of media feature tags that can be registered by a device and included in the Accept-Contact and Reject-Contact header fields in a request. Proper operation of caller preferences depends strongly on consistent interpretation of these feature tags by the caller and the callee. In this section, we provide some guidelines on the usage of these feature tags.

Generally speaking, the more information a device provides when it registers, the more effective the caller preferences extension is. This is why the callee capabilities extension recommends that a device register as much information as it can. This point cannot be overstated.

If devices explicitly registered features that they don't support, such as 'video="false"', the operation of RFC 3841 would be improved. However, given the open-ended nature of capabilities, it will never be possible to ensure the registration of negative values for all capabilities of interest to a caller. Furthermore, attempting to do so would significantly bloat registrations. Instead, it is recommended that all "unusual" capabilities be explicitly registered.

The subsections below show example registrations from typical devices.

### 5.1. Traditional Cell Phone

A VoIP cell phone capable of making voice calls would generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:user@example.com
Contact: <sip:cell-phone@example.com>
;audio
;class="business"
;duplex="full"
;+sip.extensions="100rel,path"
;mobility="mobile"
;methods="INVITE,BYE,OPTIONS,CANCEL,ACK"
;schemes="sip,sips,tel"
;uri-user="<cell-phone>"
;uri-domain="example.com"
```

## 5.2. Traditional Work Phone

A traditional landline IP PBX phone would generate a registration that looks like:

```
REGISTER sip:example.com SIP/2.0
To: sip:user@example.com
Contact: <sip:ippbx-phone@example.com>
;audio
;class="business"
;duplex="full"
;events="dialog"
;+sip.extensions="100rel,privacy"
;mobility="fixed"
;methods="INVITE,BYE,OPTIONS,CANCEL,ACK,SUBSCRIBE"
;schemes="sip,sips,tel"
;uri-user="<ippbx-phone>"
;uri-domain="example.com"
```

This device also supports the dialog event package and several SIP extensions that would be typical in an IP PBX phone.

## 5.3. PC Messaging Application

A PC messenger client, capable of just doing presence and IM (no voice) would generate a registration that looks like:

```
REGISTER sip:example.com SIP/2.0
To: sip:user@example.com
Contact: <sip:pc-msgr@example.com>
;class="personal"
;mobility="fixed"
;methods="OPTIONS,MESSAGE,NOTIFY"
;schemes="sip,sips,im,pres"
;uri-user="<pc-msgr>"
;uri-domain="example.com"
```



#### 5.4. Standalone Videophone

A standalone IP videophone, capable of audio and video, would generate a registration that looks like, in part

```
REGISTER sip:example.com SIP/2.0
To: sip:user@example.com
Contact: <sip:vp@example.com>
;audio
;video
;class="business"
;duplex="full"
;mobility="fixed"
;methods="INVITE,BYE,OPTIONS,CANCEL,ACK"
;schemes="sip,sips,tel"
;uri-user="<vp>"
;uri-domain="example.com"
```

#### 6. Example of Implementation of Preference and Capability Matching

RFC 3841 [3] utilizes the definitions and feature matching algorithm defined in RFC 2533 [6]. This provides a precise normative specification of the algorithm. However, that specification isn't ideal as a guideline for implementation because it is more complex than is required for the restricted use employed by RFC 3841. (The simplification is primarily because a particular feature tag may only appear once in each Contact, Accept-Contact, or Reject-Contact header.)

This section provides a sample approach to implementing the matching of caller preferences to callee capabilities; it does not require the use of the notation and techniques of RFC 2533. It is not normative, but is believed to be consistent with that definition. It may be considered an alternative for that portion of RFC 3841 beginning with Section 7.2.3 and extending to the end of page 13 in the middle of Section 7.2.4.

In this section, there are frequent references to syntactic elements defined by ABNF in RFC 3840, Section 9, and RFC 3841, Section 10. Here, ABNF elements are enclosed to single quotes -- for example, 'feature-param'. Such a reference identifies a sequence of octets within a SIP request that match the corresponding ABNF element when the sip request is parsed according to RFCs 3261, 3840, and 3841.

## 6.1. Extracting a Feature Set from a Header

Contact header fields, Accept-Contact header fields, and Reject-Contact header fields each contain zero or more 'feature-param's, each in turn may contain one or more 'tag-value's, or a 'string-value'. The first step is to extract from each header field a more useful representation as a feature set, herein called an FS. (This FS representation of a feature set representation differs from that in RFC 2533.) This process is the same for each type of header.

An FS consists of a set of one or more feature params denoted by FP. Each FP has a name, denoted FP.NAME, and a set of one or more value ranges denoted by VR. Each VR consists of:

- o A type (VR.TYPE): either token (TOKEN-TYPE), string (STRING-TYPE), or number-range (RANGE-TYPE)
- o A negation flag (VR.NEGATION): either NEGATED, or NON-NEGATED
- o The actual value, differing by type:
  - \* For TOKEN-TYPE and STRING-TYPE, a sequence of octets (VR.OCTETS)
  - \* For RANGE-TYPE, a pair of signed real numbers (VR.LB and VR.UB) representing the lower and upper bounds on the range, inclusive.

A single FS is created to represent the features of one header. (Contact, Accept-Contact, Reject-Contact.) Within the FS, an FP is created for each 'feature-param' in the header. To create an FP, a 'feature-param' is examined as follows:

- o If the 'feature-param' contains an instance of 'other-tags', then FP.NAME is the value matched by 'ftag-name'.
- o Otherwise, the 'feature-param' contains an instance of 'base-tags'. If the value matched by 'base-tags' is "language" or "type", then FP.NAME is just the value matched by 'base-tags'. If not, then FP.NAME is the value matched by 'base-tags' and prefixed with "sip."
- o The value of the 'feature-param', if any, is processed (according to the rules in the next section) to extract a set of one or more VRs that are associated with the FP.

## 6.2. Extracting Values from a Feature Parameter

The value of a 'feature-param' is an encoded representation (as specified in RFC 3840) of one or more value ranges of the corresponding feature. There are several data types that these values may take on: boolean, token, string, number, or numeric range. The type is determined by the encoded form of the value. (These types and their representations are specific to this implementation.)

(Note: numeric values can explicitly represent a range of values. The other types only represent single value: a degenerate range. The term value range is used to encompass all of these.)

The value of the 'feature-param' ('string-value', 'tag-value-list', or none) is converted to VR form as follows:

- o If there is no value, then a single new VR is created with VR.TYPE = TOKEN-TYPE, VR.NEGATION = NON-NEGATED, and VR.OCTETS set to "true".
- o If the 'feature-param' contains a 'string-value', then a single new VR is created with VR.TYPE = STRING-TYPE, VR.NEGATION = NON-NEGATED, and VR.OCTETS is set to the octets matching 'qdtex'.
- o Otherwise the 'feature-param' contains a 'tag-value-list', and a new VR is created for each 'tag-value' in the 'tag-value-list', as follows:
  - o If the 'tag-value' begins with "!", VR.NEGATION = NEGATED; otherwise, VR.NEGATION = NON-NEGATED.
  - o If the 'tag-value' contains a 'boolean' or 'token-nobang', then VR.TYPE = TOKEN-TYPE, and VR.OCTETS is set to the octets matched by 'boolean' or 'token-nobang'.
  - o If the 'tag-value' contains a 'numeric', VR.TYPE = RANGE-TYPE and:
    - \* If 'numeric-relation' is "<=", VR.UB is set to the numeric value matching 'number'. VR.LB is set to MIN-REAL (a negative number with the largest expressible magnitude.)
    - \* If 'numeric-relation' is "=", both VR.LB and VR.UB are set to the numeric value matching 'number'.
    - \* If 'numeric-relation' is ">=", VR.LB is set to the numeric value matching 'number' plus a small epsilon. VR.UB is set to MAX-REAL (a positive number with the largest expressible magnitude).

- \* Else the 'numeric-relation' consists of two 'number's separated by a colon. In this case, VR.LB is set to the numeric value of the smaller of the two numbers, and VR.UB is set to the numeric value of the larger of the two numbers.

### 6.3. Comparing Two Value-Ranges

Two VRs match if their ranges overlap. The comparison is done according to type, and only comparisons between like types are defined. When two VRs of differing types are compared, they are considered not to overlap. Either or both of the VRs may be NEGATED. Comparison proceeds as follows:

- o If the VRs are of different types, the match is false.
- o Otherwise:
  - \* Two VRs with VR.TYPE = RANGE-TYPE match if  $\max(\text{VR1.LB}, \text{VR2.LB}) \leq \min(\text{VR1.UB}, \text{VR2.UB})$ .
  - \* Two VRs with VR.TYPE = TOKEN-TYPE match if their respective VR.OCTETS values compare equal by case-insensitive comparison.
  - \* Two VRs with VR.TYPE = STRING-TYPE match if their respective VR.OCTETS values compare equal by case-sensitive comparison.
- o The result (true/false) is then negated if VR1.NEGATION = NEGATED, and negated again if VR2.NEGATION = NEGATED.

### 6.4. Feature Set to Feature Set Matching

In RFC 2533, the matching of two feature sets is commutative, but as applied to caller preferences matching it is not. In this application, one feature set comes from an Accept-Contact or Reject-Contact header, and the other comes from a Contact header. For purposes of this description, these will be termed the preferred-features (FSp) and the capability-features (FSc), respectively. Non-commutativity arises from explicit tests for the presence among capability-params of feature param names used in preferred-features.

A preferred-features feature set FSp may be matched to one capability-features feature set FSc, and this yields the following metrics:

- o NPF - The number of preferred-features.
- o NCF - The number of preferred-features for which there is a capability-feature of the same name.

- o NVM - The number of value matches between corresponding features of the two feature sets.

For a particular pair of FPP and FPC, these metrics are computed as follows:

- o All the metrics are set to zero.
- o The following steps are applied for each feature param (FPP) of the FSP:
  - \* NPF is incremented.
  - \* A corresponding FP with the same name is sought (using case-insensitive comparison) in the FSC.
  - \* If a corresponding feature param (FPC) is found:
    - + NCF is incremented.
    - + Every VR of FPP is matched to every VR of FPC.
    - + If any of those matches succeed, NVM is incremented.

## 6.5. Selecting and Ordering Contacts Based on Caller Preferences

### 6.5.1. Reject-Contact Processing

The reject processing specified in Section 7.4.2 of RFC 3841 may be performed as follows:

- o For each candidate Contact in the target set, match the feature set of each Reject-Contact to it.
- o If (NVM == NPF) & (NCF == NPF), remove the contact URI from the target set.

### 6.5.2. Accept-Contact Processing

The matching of an Accept-Contact against a Contact and subsequent scoring of the match specified in Section 7.4.2 of RFC 3841 may be performed as follows:

- o Match the feature set of the Accept-Contact to that of the Contact as specified in Section 6.4.

- o If (NVM < NCF), then the match failed. If the Accept-Contact had its "require" flag set, then discard the corresponding contact URI from the target set.
- o Compute the score as NVM/NPF.
- o Apply the "require" and "explicit" flags as specified in the text and Figure 7 of RFC 3841.

## 7. Security Considerations

This document provides explanation and examples of the use and implementation of RFCs 3840 and 3841. The security considerations sections of those documents apply to the material presented here.

## 8. Acknowledgements

The authors would like to thank Rohan Mahy for his input in this specification.

## 9. Informative References

- [1] 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.
- [2] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.
- [3] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, August 2004.
- [4] Mankin, A., Bradner, S., Mahy, R., Willis, D., Ott, J., and B. Rosen, "Change Process for the Session Initiation Protocol (SIP)", BCP 67, RFC 3427, December 2002.
- [5] Lennox, J. and H. Schulzrinne, "Call Processing Language Framework and Requirements", RFC 2824, May 2000.
- [6] Klyne, G., "A Syntax for Describing Media Feature Sets", RFC 2533, March 1999.
- [7] Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, November 2005.

- [8] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.
- [9] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.
- [10] Campbell, B., "The Message Session Relay Protocol", Work in Progress, July 2006.

#### Authors' Addresses

Jonathan Rosenberg  
Cisco Systems  
600 Lanidex Plaza  
Parsippany, NJ 07054  
US

Phone: +1 973 952-5000  
EMail: [jdrosen@cisco.com](mailto:jdrosen@cisco.com)  
URI: <http://www.jdrosen.net>

Paul Kyzivat  
Cisco Systems  
1414 Massachusetts Avenue  
Boxborough, MA 01719  
US

Phone: +1 978 936-1881  
EMail: [pkyzivat@cisco.com](mailto:pkyzivat@cisco.com)

## Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Acknowledgement

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



