BLISS WG	J. Elwell	
Internet-Draft	Siemens Enterprise Communications	
Intended status: BCP	April 30, 2010	
Expires: November 1, 2010		

TOC

An Analysis of Automatic Call Handling (ACH) Implementation Issues in the Session Initiation Protocol (SIP)
draft-ietf-bliss-ach-analysis-06.txt

Abstract

This discusses problems associated with automatic call handling (ACH) when using the Session Initiation Protocol (SIP) and specifies some best practices to help achieve interoperability.
This work is being discussed on the bliss@ietf.org mailing list.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
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.”
This Internet-Draft will expire on November 1, 2010.

Copyright Notice

Table of Contents

1. Introduction

TOC

The Session Initiation Protocol (SIP) [RFC3261] (Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, “SIP: Session Initiation Protocol,” June 2002.) establishes calls or sessions for real-time communication

between users. When a call is targeted at a called user, often the call is subject to some automatic treatment to determine whether to present the call to the user or take some alternative action such as forwarding to voicemail. Similarly, if some condition arises after presenting a call to the called user but before answer, automatic treatment can lead to some alternative action. Automatic treatment is in accordance with policy determined in advance by the user or the user's organization. This automatic treatment of incoming calls is referred to as automatic call handling (ACH) in this document.

In order to encourage innovation, ACH is deliberately not specified in RFC 3261 or in RFCs that specify extensions to SIP. However, the flexibility that this affords has sometimes led to problems, where different implementations have approached the issue in different ways, leading to unexpected and often unwanted behavior when those implementations are deployed together. This document analyses the sources of problems with ACH and specifies some best practices to help achieve interoperability.

A number of other Standards Development Organisations (SDO) (e.g., 3GPP, ETSI) have specified specific features or "services" that involve specific forms of ACH.

A survey was conducted prior to IETF70 to get a feeling for what are common practices in this area. Although the number of responses was disappointingly small (see results), in some cases it did give a clue as to the most common practices. In the remainder of this document, this is referred to as "the survey".

---

2. Terminology

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

---

3. Examples of ACH

TOC

ACH can occur prior to or instead of presenting an incoming call to a called user or after presentation but before the called user answers the call. The particular treatment applied to a call is generally dependent on a number of factors, examples of which are as follows:

* Whether there are other registered contacts that can handle the call (e.g., a registered audio User Agent (UA) for an audio call).

*Whether the user's UA (or UAs) is known to be busy on another call.

*Whether the user has failed to answer the call within a given number of seconds.

*Whether the user is known to be unwilling to receive calls at the present time (a condition often known as Do Not Disturb, DND).

*Whether the user is known not to be available (e.g., on vacation).

*Whether an alternative user (e.g., a colleague, an assistant, another family member) is known to be available.

*Whether the Address Of Record (AOR) at which the call is targeted represents a single user or a team or group of users.

*Time of day, day of the week, date, etc..

*The type of call (e.g., audio, audio plus video, messaging, etc.).

*The source of the call (e.g., whether the caller is anonymous, whether the caller is blacklisted or whitelisted, which organization the caller belongs to, etc.).

The conditions above are detected though local information at the entity performing ACH, by access to presence information or through information received via SIP signalling (e.g., a UA's response to an INVITE request).
Examples of particular treatment to be applied to a call if appropriate conditions are met are as follows:

*Reject the call, with an appropriate indication to the caller. This indication may or may not reveal the actual condition that led to rejection.

*Forward the call to another UA serving that user (e.g., voicemail, a mobile UA, a UA at another location).

*Forward the call to another user, e.g., the next member of a team, an assistant.

*Modify the nature of the call (e.g., downgrade from audio to messaging).

*Any of the above, but impacting presentation of the call at a given UA, without impacting presentation at other UAs serving the user.

A user can specify quite complex sets of rules for ACH. For example, "if presence indicates I am in a meeting, or if my desk phone is busy, or if I do not reply within 15 seconds, forward calls to my assistant between the hours of 09.00 and 17.00, Monday to Friday, but at other times forward to my voicemail, unless the call is from my home or my partner's mobile phone, in which case forward to my mobile phone".

---

4. Known problem areas with ACH

TOC

4.1. Conflict between proxy and UA

TOC

A significant problem area with ACH is interactions between proxies (or Back-To-Back User Agents, B2BUAs) that perform ACH and UAs that perform ACH. The domain proxy for a user is configured to treat incoming calls in a certain way under certain conditions. One of the user's UAs is configured to treat incoming calls in a different way under the same or overlapping conditions. If the condition can be detected by the proxy without presenting the call to the UA, the proxy will win and the user may wonder why the action configured at the UA is not being taken. For example, if the proxy detects a DND condition from a presence server and forwards calls to voicemail, any script at the UA to forward private calls to a mobile phone would never execute. This may or may not be what the user (or his/her organization) desires to happen. Alternatively, if a condition is detected by a UA before it is detected at the proxy, the action determined by the UA will "win", unless the proxy is somehow able to figure out what has happened and apply its own action. For example, if a phone determines it is busy and returns a 302 response code to forward the call elsewhere or performs "call waiting" action, this might prevent the proxy taking whatever action it would have taken on receipt of a 486 response. This may or may not be what the user (or his/her organization) desires to happen.
If a UA attempts ACH, it may be possible for the proxy to override it, e.g., by taking account of the response code returned or, in the case of a 3xx response code, whether the UA has provided further information concerning the reason for redirection in accordance with RFC 4458 [RFC4458] (Jennings, C., Audet, F., and J. Elwell, “Session Initiation Protocol (SIP) URIs for Applications such as Voicemail and Interactive Voice Response (IVR),” April 2006.).
The survey showed that the majority of proxies perform ACH without first presenting the call to the UA, at least for certain types of ACH. The survey also showed that the majority of UAs implement some form of ACH. This does seem to confirm the potential for conflict between proxy and UA. If, as a result of ACH at the UA, the call required

redirection, 302 was the response code used in the majority of situations. Only a minority of such implementations used RFC 4458 to provide more information about the reason for redirection.

4.2. Conflict between UAs

TOC

Where an incoming call is forked to multiple UAs, there is potential for different UAs to be configured to perform different ACH actions under the same or overlapping conditions. With parallel forking (where the INVITE request is sent to each UA at approximately the same time), results can be indeterminate and might depend on which UA responds first. With serial forking, this is likely to be more deterministic, but UAs would need to be configured taking into account the order in which the proxy presents calls to the UAs.
When a proxy forks a call, it can invoke ACH based on the first UA to respond, can wait for all UAs to respond or behave in some other way (e.g., act immediately on certain response codes). The survey did not show a bias towards any one behavior.

4.3. Obtaining information from UA for ACH at proxy

TOC

When ACH is performed at a proxy, it sometimes requires information from the UA, in response to the INVITE request. If this information does not arrive in the form expected by the proxy (e.g, a particular response code), ACH will be adversely impacted. For example, if the proxy is configured to perform forwarding on DND and relies on the DND condition to be indicated in the INVITE response, it depends on the UA indicating the condition in the form expected by the proxy. As there is no standardized means of indicating DND in a response (see [I-D.elwell-bliss-dnd] (Elwell, J. and S. Srinivasan, “An Analysis of Do Not Disturb (DND) Implementations in the Session Initiation Protocol (SIP),” November 2007.)), this can be a problem. Furthermore, there might be more than one flavour of DND (e.g., with/without forwarding to voicemail), requiring different responses.
The survey showed that 486 (Busy Here) is the response code most commonly expected by proxies for indicating DND, but that accounted for less than half of the responses. Other known response codes in use include 406 (Not Acceptable), 480 (Temporarily Unavailable), 600 (Busy Everywhere) and 603 (Decline).
The survey also showed that even for the busy condition, proxies expected different response codes. Although a small majority expected 486 (Busy Here), other expectations included 480 (Temporarily Unavailable) and 600 (Busy Everywhere).

The survey did not yield significant information concerning response codes issued by UAs.

4.4. Informing the calling UA

TOC

A related problem is informing the calling UA, and hence the caller, what has happened. In the case where ACH results in rejection of the call, this might be just a case of sending back an appropriate response code. Considerations are similar to those for a proxy in Section 4.3 (Obtaining information from UA for ACH at proxy), except that privacy might require the proxy to send a different response code rather than the one reflecting the condition encountered. For example, the user might not wish the caller to know about his absence.

The choice of response code might not be an interoperability issue if the calling UA is relatively dumb, but might be an issue if there is an application that takes the response code into account. Where there is forking proxy between the entity performing ACH and the calling UA, information may be lost because of the Heterogeneous Error Response Forking Problem (HERFP).

Where ACH results in forwarding (to a different AOR or a different contact for the same AOR), this can be achieved by retargeting or redirection. In the case of retargeting, the calling UA receives no information, apart from a final response and perhaps identity from the retargeted-to user. On the other hand, if redirection is used, the calling UA will receive a 3xx response, the contact Universal Resource Identifier (URI) in which could indicate the source of the redirection and possibly the reason, in accordance with RFC 4458 [RFC4458] (Jennings, C., Audet, F., and J. Elwell, “Session Initiation Protocol (SIP) URIs for Applications such as Voicemail and Interactive Voice Response (IVR),” April 2006.).

4.5. Scope of conditions

TOC

When an INVITE request is forked to multiple UAs, the user may or may not require a condition at one UA to be considered as applying to other branches. This includes branches already active (through parallel forking) or branches yet to be activated (through serial forking). This can impact when to invoke ACH at the proxy, i.e., whether to perform ACH when one UA reports an appropriate condition (cancelling other active branches if necessary) or to wait for the outcome on other branches.

Although to a large extent this issue can be handled by appropriate scripting at the proxy, an important consideration is how to treat the 6xx class of responses. For example, if a UA issues a 600 Busy

Everywhere response (as opposed to a 486 Busy response), what is the scope of "everywhere"? A simple interpretation is that it literally means "everywhere", and all other branches should be abandoned and the 6xx response passed back to the caller if no other ACH is prescribed for this condition. However, other interpretations might seem reasonable. If a user has several phones, it might be reasonable to interpret a 600 response from one phone as meaning that all other phones are busy, but if the user also has voicemail it is unlikely that that too should be treated as busy. Also, if ACH requires forwarding to a different user (different AOR) on busy, it might be expected that this would take place even on receipt of a 600 response from a UA. Another example is the 603 Decline response code. This is often intended to be applied everywhere.
There is also a question of whether a proxy should trust a UA to decide that all other branches need to be abandoned, particularly in applications like call centres, where the different branches might be different agents, rather than leading to different devices belonging to the same user. It might be wise to consider this a policy matter.
The survey gave only a very small number of answers on the issue of handling 6xx responses, with no conclusions to be drawn other than that forwarding to voice mail is sometimes allowed following a 6xx response.

4.6. Configuring the proxy

TOC

If ACH is performed at the proxy, the user needs a means to configure the proxy with the required rules. There is no SIP means of doing this, but a number of mechanisms can perform the basis for this task, e.g.:

*Via a web page.

*By uploading a CPL [RFC3880] (Lennox, J., Wu, X., and H. Schulzrinne, “Call Processing Language (CPL): A Language for User Control of Internet Telephony Services,” October 2004.) script.

*Via a web services interface based on SOAP.

*Via Computer Supported Telecommunication Applications (CSTA) [CSTA] (, “International Standard ISO/IEC 18051 "Information Technology - Telecommunications and information exchange between systems - Services for Computer Supported Telecommunications Applications (CSTA) Phase III",” .).

*Via XCAP [RFC4825] (Rosenberg, J., “The Extensible Markup Language (XML) Configuration Access Protocol (XCAP),” May 2007.)

*Using standard Hyper-Text Transfer Protocol (HTTP) primitives, a technique commonly known as REpresentational State Transfer (REST).

The survey showed that web pages and SOAP-based web services were the most common mechanisms supported by proxies, but the sample was very small. The majority of UA implementations provided a web user interface.

Related to this is the means by which a UA (and hence the user) can discover how the proxy is configured. Most of the mechanisms listed above are applicable, and also a SIP SUBSCRIBE/NOTIFY mechanism could be used. The survey indicated that only a minority of proxies provided support in this respect.

Some of the above mechanisms (e.g., a web page) are unsuitable for automatic use by the UA (as opposed to direct interaction between the user and the proxy). For example, suppose the UA has a button that can be pressed to activate or deactivate forwarding, and an associated lamp or icon to show that forwarding is active. In order to support ACH at the proxy, the UA would need a means for instructing the proxy to activate or deactivate forwarding, and also a means to obtain from the proxy the current forwarding state for controlling the lamp or icon. A web page would be unsuitable for this purpose, but most of the other mechanisms might be suitable.

Without a single standardized way for a UA to configure a proxy for ACH and obtain a proxy's ACH configuration, there is a danger that the UA and proxy might not support a common method, requiring the user to employ other means (e.g., using a different device, contacting a support centre). Furthermore, it might lead the user to configuring ACH at the UA when in practice ACH at the proxy would serve the user's needs better.

---

5. Discussion

TOC

5.1. Proxy versus UA

TOC

The end-to-end principle of SIP would suggest that ACH at the UA is more appropriate than ACH at the proxy. However, certain considerations make ACH at the proxy more viable or even essential.

ACH in the event that there is no registered contact obviously can only be performed by the proxy.

A proxy is more easily able to take account of the state of other UAs, e.g., by waiting for all branches of a forked call to respond before invoking ACH. Although a UA can use techniques such as the registration

event package [RFC3680] (Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Registrations," March 2004.) in combination with the dialog event package [RFC4235] (Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)," November 2005.) to determine the state of other UAs, this is complex, may not yield the information required, and may suffer from timing-related inconsistencies.
A proxy needs to be configured once and can perform ACH independently of the number of UAs involved. Obtaining consistent behaviour using ACH at the UA may involve configuring multiple UAs and keeping their configurations aligned. The UA configuration framework [I-D.ietf-sipping-config-framework] (Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery," February 2010.) may be a suitable mechanism for this and would require a means for the user to configure the profile delivery server. However, there can be no guarantee that all UAs will download a revised configuration at the same time, so it can lead to a time window when inconsistent behaviour may occur.
With these considerations in mind, a proxy will often turn out to be a more suitable place for performing ACH.
On the other hand, there may be situations in which UA-specific ACH may be required, and it may not be feasible to configure the proxy to provide this level of granularity. For example, it may be required to take one action if the desk UA is busy but a different action if the mobile UA is busy. Convincing use cases for this are hard to find, but it cannot be ruled out. A possible approach here is to use proxy-based ACH as the default handling for all UAs and UA-based ACH for any UA-specific exceptions.

5.2. Avoiding inconsistent configurations

TOC

Given that there is frequently a need to perform ACH at the proxy, problems can be avoided by turning off ACH at all UAs. There may be exceptions to this, e.g., where there is need for a specific UA to perform actions different from default actions carried out by the proxy, or where there is a requirement for behavior not supported by the proxy. Where ACH does need to be configured at one or more UAs, care must be taken to avoid unintentional conflicts. Use of the SIP configuration framework can help to ensure consistent handling at all UAs. One consideration during the work on profiles for use with the SIP configuration framework might be the downloading of policy relating to ACH, such that ACH could be suppressed in order to ensure that proxy-based ACH operates correctly.

TOC

5.3. Enterprise and carrier environments

Considerations for ACH will often differ between enterprise and carrier environments. In enterprise environments, enterprise policy will often govern what a user can and cannot do. This does not necessarily mean that ACH will be done at a proxy, because the enterprise will probably manage UAs too and ensure that they behave in line with policy, although proxy-based ACH will often be easier to accomplish for other reasons discussed in Section 5.1 (Proxy versus UA).
In a carrier environment, everything can be expected to be under the control of the user. Proxy-based ACH is still relevant, however, particularly for mobile devices that are often out of reach or turned off.
Handling such as team calls (where any team member can be selected according to availability) is perhaps more likely in enterprise, although in a residential environment it could be used for finding any family member.
Despite these different considerations, requirements are similar to a large extent and the same solution should be sought for both environments.

---

6. Potential measures that could be taken

TOC

In this section we explore potential measures that can be taken to some of the problems identified above.

6.1. Conflict between proxy and UA

TOC

This appears to be an important problem to solve, in order to have proxies and UAs from mixed vendors.
One approach is to specify particular features that must or must not be implemented in a proxy and particular features that must or must not be implemented in UA. This is likely to fail for a number of reasons:

- There are far too many possible features, and enumerating and standardizing individual features is contrary to the philosophy of SIP and likely to inhibit innovation.

- For a given feature, there will be some deployments where it makes sense to do it at the proxy and other deployments where it makes sense to do it at the UA. It will often be impracticable to choose one.

- Proxy vendors and UA vendors will want to provide as many features as possible on their products and are likely to ignore any recommendation not to implement a particular feature.

Stipulating that ACH as a whole must always be done at the proxy or must always be done at the UA is clearly out of the question, because each has some advantages, depending on circumstances, and also because vendors of one or the other will not be prepared to give up producing features that play an important part in differentiating their products. Therefore it has to be accepted that ACH will be implemented on proxies and UAs, with feature overlap between the two. The challenge then is to ensure that, when deployed, the two can co-exist in a sensible way.
It should be possible to control whether a proxy defers to a UA or vice versa. For a proxy to defer to a UA, it requires the proxy to deliver an INVITE request to a UA before taking any ACH action. Depending on the response of the UA, the proxy may then perform its own ACH action. For a UA to defer to a proxy, it should report any conditions back to the proxy (e.g., by means of a suitable response to the INVITE request) rather than taking unilateral action such as redirecting or placing the call in a waiting state. In other words, it should be possible to turn off ACH at a UA.
There are several ways to achieve this control:

- Configure the UA and proxy independently. The SIP configuration framework [I-D.ietf-sipping-config-framework] (Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery," February 2010.) is one possible means of configuring the UA.

- Configure the UA (e.g., by means of the SIP configuration framework [I-D.ietf-sipping-config-framework] (Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery," February 2010.)) and use SIP to instruct the proxy (e.g., by means of an indicator in REGISTER requests).

- Configure the proxy and use SIP to instruct the UA (e.g., by means of an indicator in inbound INVITE requests or in the 200 response to a REGISTER request).

The configuration approach is the simplest and does not require any enhancements to the SIP protocol. In general, positive action has to be taken to configure any sort of ACH, i.e., ACH is turned off by default. Therefore by default there should not be a problem, because ACH will be turned off in both places. If the user is in control of ACH at the UA and ACH at the proxy, it is the user's responsibility not to configure conflicting behaviours.
The situation is slightly different where there is more than one authority involved, e.g., if the user is able to configure the UA but some other authority is responsible for configuring the proxy. This

might arise in an enterprise environment, where the enterprise administration might configure the proxy. In this case, the UA user could potentially configure the UA in a conflicting way. In such cases it would be useful if the administration could prevent the user configuring ACH at the UA, i.e., place ACH configuration under administration control. Many UAs aimed at the enterprise market have this form of control already. In future there might be a standardised way of doing this based on the SIP configuration framework [I-D.ietf-sipping-config-framework] (Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery," February 2010.) . There does not seem to be a need for BLISS to specify anything further to address this issue at present.

6.2. Conflict between UAs

TOC

This can really only be addressed by configuration. The SIP configuration framework can help here. In fact, that would normally configure all UAs having the same AOR with the same information. Configuration outside this framework (e.g., local actions at the device) might introduce differences (intentional or otherwise). There seems little action that BLISS can take to address this issue.

6.3. Obtaining information from UA for ACH at proxy

TOC

There seems to be a case for more precisely defining or at least recommending information given to the proxy when rejecting an inbound call, in order to assist the proxy in providing the most relevant ACH, if any. Ideally this information needs to be given as a SIP response code, although potentially a response code could be supplemented by a header field. In choosing a particular response code, two factors need to be taken into account:

- *the reason for rejection;
- *the desired scope of rejection.

The following sub-sections discuss these two factors. Best practices in Section 7.2 (Use of response codes for reporting ACH-related conditions) propose the use of existing response codes for the conditions identified, avoiding the need to specify new response codes.

TOC

6.3.1. Reason for rejection

The most relevant reasons for rejection (from an ACH perspective) are as follows:

- *Busy. UA resources are busy as a result of another call and further calls cannot be accepted until resources become free. The condition may also be visible via a presence system.

- *Explicit rejection. The user has indicated an unwillingness to accept the call at the present time. The user may have indicated this in advance, so that any incoming calls (or those fulfilling certain conditions) would be rejected in this way (a feature often known as "do not disturb"). The user may impose such a condition during a meeting or while working on a critical task. The user may indicate in advance a time at which she expects the condition to be lifted. The condition may also be visible via a presence system. Alternatively the user may indicate an unwillingness to accept a particular call in response to being alerted.

- *Silent rejection. The user has responded to this call by rejecting it, but does not wish the reason to be revealed to the caller. A user would use this facility when not currently in a position to answer a particular call or when she feels that it would be better handled elsewhere (e.g., by voicemail, by an assistant).

Note that silent rejection could also be achieved by returning a 180 response to the INVITE request, and then waiting (without alerting the user) until the call is cleared or times out. In the meantime, the proxy would be unaware of what is happening and would be unable to take other action, such as cancelling other branches. On the other hand, indicating silent rejection relies on the proxy to take alternative ACH action (e.g., waiting for a certain time before forwarding to voice mail or reporting to the caller that the call has timed out), rather than revealing silent rejection to the caller. Therefore silent rejection is suitable for use only when it is known that the proxy will take appropriate action.

6.3.2. Desired scope of rejection

TOC

When rejecting a call the user (or the UA on the user's behalf) may desire the rejection to have one of the following scopes:

- *Local. Rejection impacts only the branch concerned or any branches to the user's other devices. Forwarding to voicemail or

to an assistant, for example, is not prevented and may be instigated by the proxy as a result of receiving a local rejection.

*Global. Rejection impacts all branches, including voicemail.

It is a matter for the proxy to determine what ACH to perform for a local rejection or a global rejection (although this may be based on per-user settings, which the user may have some control over, e.g., via a web page, see Section 6.6 (Configuring the proxy)). For local rejection the proxy might, for example, cancel other branches (to the user's other devices) and forward immediately to voicemail or to an assistant. On the other hand it may leave other branches unaffected. For global rejection the proxy might reject the call outright, cancelling all other branches, although policy might require a less severe action to be taken. For example, in a call centre there might be a requirement that all calls be answered.
4xx response codes are appropriate for local rejection and 6xx response codes are appropriate for global rejection.

6.4. Informing the calling UA

TOC

Whilst this might be interesting, it is unlikely to impact interoperability and is not seen as a priority issue for BLISS.

6.5. Scope of conditions

TOC

In view what is specified in [RFC3261] (Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, “SIP: Session Initiation Protocol,” June 2002.) for 6xx response codes and with some, but not all existing practice, it is safest to regard 6xx response codes as impacting all branches of a forked INVITE request. If this is not the desired behaviour when a particular condition arises at the UA (e.g., if forwarding to voicemail is desired), a 4xx response code should be used instead. The proposals in Section 6.3 (Obtaining information from UA for ACH at proxy) take this into account.

TOC

6.6. Configuring the proxy

General methods for configuring proxies (including synchronization of multiple proxies serving a domain) are considered outside the scope of BLISS work.
However, a means is required for a UA to view and modify ACH configuration at a proxy. Although several methods are used in practice, some of these being mentioned in Section 4.6 (Configuring the proxy), the one that is perhaps simplest and in line with industry trends is HTTP using a REST architecture. With this approach, different URIs represent different resources, and standard HTTP methods such as GET, PUT, POST and DELETE are used to manipulate those resources. More information is available in [I-D.zourzouvillys-bliss-ach-http-api] (Zourzouvillys, T., "Basic HTTP API interface for ACH," March 2009.)
OPEN ISSUE. The above reference to be replaced by whatever is adopted as work item for a framework for RESTful configuration.
Under some circumstances a setting can change other than by action of the UA, and therefore if the UA relies on knowing the value of a setting (e.g., to provide a continuous indication to the user), the UA needs to be informed if the setting changes. One example is where there are two or more UAs registered as contacts for an AOR. If one of the UAs changes a setting, the other UAs might wish to know. Another example is where a user or administrator changes a setting from a web page. For such situations it would be useful to have an event package whereby the UA can subscribe to receive notifications of changes to ACH settings. The notification would only need to indicate that a change has occurred, and the UA could then issue GET requests to pull down the new settings. A possible solution to this is specified in [I-D.roach-sip-http-subscribe] (Roach, A., "A SIP Event Package for Subscribing to Changes to an HTTP Resource," February 2010.).

---

7. Best practices for ACH

TOC

7.1. Avoiding conflict between ACH at proxy and ACH at UA

TOC

A UA MUST be able to be configured to operate when ACH is provided by the proxy. Typically this means being able to be configured with all ACH features turned off, which typically would be the default configuration.
A proxy MUST be able to be configured to operate when ACH is provided by the registered UA or UAs for a given address of record. Typically this means being able to be configured with all ACH features turned off

for a given address of record, which typically would be the default configuration.

7.2. Use of response codes for reporting ACH-related conditions

TOC

UAs and MUST use the following response codes for rejecting INVITE requests encountering ACH-related conditions and proxies MUST interpret these response codes accordingly.
The following response codes are for local rejection:

- Response code 486 for busy/local. As indicated in RFC 3261, this can be accompanied by a Retry-After header field to indicate when the user expects to be available again.

- Response code 480 for explicit rejection/local. As indicated in RFC 3261, this can be accompanied by a Retry-After header field to indicate when the user expects to be available again.

- Response code 487 for silent rejection/local. This is the same response code that would be used if a proxy were to issue a CANCEL request.

The following response codes are for global rejection:

- Response code 600 for busy/global. As indicated in RFC 3261, this can be accompanied by a Retry-After header field to indicate when the user expects to be available again.

- Response code 603 for explicit rejection/global. As indicated in RFC 3261, this can be accompanied by a Retry-After header field to indicate when the user expects to be available again.

No code is specified for silent rejection/global. It is not clear what use cases exist for this. One possible use case, however, is for dealing with SPIT, where the user can determine (e.g., from caller ID) that the call is unwanted and does not wish the call to go to voicemail even. In this case, other issues arise, such as indicating the SPIT status to an entity responsible for handling SPIT. This should be pursued as part of anti-SPIT measures and is outside the scope of BLISS.

TOC

7.3. UA configuration of ACH at the proxy

OPEN ISSUE. To specify ACH-specific use of the REST framework, when available, by specifying URL structures for forwarding, DND, etc. and giving examples.

7.4. Notifying a UA of an ACH configuration change at the proxy

TOC

OPEN ISSUE. To specify ACH-specific use of event package, when available.

---

8. IANA considerations

TOC

None.

---

9. Security considerations

TOC

This document just discusses interoperability issues relating to ACH. It does not define any new protocol or practices and therefore does not introduce any security issues, other than the possible user desire not to disclose ACH actions to callers.

Appendix A. Survey results

TOC

In the tables below, the "Question" column shows the questions that were asked. In the case of multi-choice questions, the "Answer" column indicates the choices and the "No." column indicates the number of responses for the given choice. For other questions the "Answer" column contains the free-text answers received.
Survey responses for proxy implementations are shown in Table 1.

Question	Answer	No.
Q1A Configuration mechanism that are supported	Web UI	7

	XCAP	1
	FTP/FTPS/SFTP	0
	SOAP	6
	CPL	1
	CSTA	1
	Others	4
	Not applicable	1
Q1B Most commonly supported mechanism for configuration	Web UI	4
	SOAP	2
	Not applicable	1
Q2A Status Code used for Indicating DND	406	1
	486	5
	480	2
	600	1
	603	1
	Others	3
	Not applicable	1
Q2B Status Code used for Indicating Busy	486	7
	480	3
	600	1
	Others	1
Q3 When does ACH initiates when multiple contacts are registered for the target AoR	ASAP	e
	When all responds	2
	Others	2
	Not applicable	1
Q3A If indicated others for Q3 describe	- If a 302 or 603 response is received, then all other contacts are cancelled and the 302/603 is handled. Such responses are considered user initiated/ configured. On the other hand, if a 480, 486, 606 etc response is received, then the B2BUA waits for all the other contacts to respond. And then takes a decision.	

	- Depending upon the service instantiated, the UA may generate no response but stop alerting, generate a 486 response which would kill that fork, or a 603 which would kill all forks. 600 would kill all forks and go to voicemail.	
Q4 If two different responses were received which can initiate different ACH what then?	A priority list among the response received is followed.	
	- Whichever response arrives first	
	- Priority list is used	
	- Priority on return codes will lead to only one action	
	- The 3xx responses are redirected The 4XX/5XX/6XX responses are canceled	
	- Individual responses are followed until one accepts the request. For example, an INVITE that is 300 redirected by one contact and 180 ringing by another, the redirected contact is handled internally and attempted at the new contact.	
	- SIP response code order of precedence and order of arrival are considered. Upstream proxies may normalize some response codes generated by end clients or gateways into a 480 response code.	
Q5 Support for discovering where to configure ACH?	Yes	2
	No	5
Q5A How is it done?	- Depending on the manufacturer's UA, at least two ways: IP phone display client "toolbar" for the UA using SOAP	
	- Device Configuration	
Q6 Do Proxy/B2BUA execute ACH without routing the call first to the contact?	Yes	8
	No	1
Q6A If you answered Yes to Q6, when is it executed?		

	- Call Forward Always.? Call Forward Busy in some cases	
	- If the provisioned user configuration in the database tells this (the DB content is normally adjustable via a web interface).	
	- Unconditional forward. Call screening. Not available (not registered) Forward	
	- The proxy (app server) is also a B2BUA, if the automatic handling is configured there, it will handle the call.	
	- A contact can be behind another proxy (address configured statically) and hence will route to that Proxy.	
	- For the Call Forward unconditional case.	
	- Screening depending upon other conditions such as time of day, calling party ID, etc.	
Q7A How does it treat 6xx response if there are multiple contacts registered for the target AoR?	- A 6xx response is considered a terminal failure, equivalent to all other terminal failures. The automatic handling continues as configured (e.g. try another ITSP or CO, forward to operator or voicemail, etc).	
	- As Proxy:? Forward the message. As B2BUA: Treated as final response and no further action performed.	
	- The B2BUA follows configured re-routing rules or have a specific treatment?	
	- Will stop processing further routes. May forward to voicemail for 600 response, or not for 603 response.	
Q7B How does it treat 6xx response if other contact outside the domain is registered	- A 6xx response is considered a terminal failure, equivalent to all other terminal failures. The automatic handling continues as configured (e.g. try another ITSP or CO, forward to operator or voicemail, etc).	

	- As Proxy:? Forward the message. As B2BUA: Treated as final response and no further action performed.	
	- The B2BUA follows configured re-routing rules or have a specific treatment?	
	- Certain features allow the proxy to route the call to an alternate endpoint. (e.g. backup SIP trunk on a disaster recovery scenario)	
	- Will stop processing further routes. May forward to voicemail for 600 response, or not for 603 response.	
Q8 Is there a way to communicate the current setting of ACH to the UA?	- No, the B2BUA hides all of the contact information from the UA and performs all automatic handling silently.	
	- Subscribe/Notify mechanism. Synchronizes the call forward state between the endpoint and the B2BUA server. The phone subscribes for the services it is interested in (e.g., DND, CFA, CFB, CFNA). Notifies are sent either way when the service is modified.	
	- Yes, via uaCSTA	
	- Yes, in some cases. Mechanism varies.	
Q9 Does it override the 302 response without considering RFC 4458?	Yes	4
	No	5

Table 1

Survey responses for UA implementations are shown in Table 2.

Question	**Answer**	**No.**
Q1 Does UA implement ACH	Yes	7
	No	2

Q1A If 3xx used for ACH which code?	300	1
	301	1
	302	7
	Not applicable	2
Q1ex: Does it use RFC 4458?	Yes	2
	No	6
Q1B If 4-6xx used which code?	400	1
	403	1
	404	1
	405	1
	408	1
	415	1
	420	1
	480	1
	481	1
	482	1
	486	2
	491	1
	500	1
	600	1
	603	2
	Not applicable	7
Q2 Can the UA configure the proxy remotely via local UI?	Web UI	5
	SOAP	2
	CSTA	1
	Not applicable	2
Q3 How does UA indiate DND?	480	1
	486	2

Table 2

TOC

10. Acknowledgements

The author would like to acknowledge the assistance of Francois Audet, Martin Dolly, Jason Fischl, Jonathan Rosenberg, Shida Schubert, Srivatsa Srinivasan and Theo Zourzouvillys in writing this draft, and also input on specific implementations from various members of the BLISS WG.

---

11. References

TOC

11.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC3261]	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 (TXT).

11.2. Informative References

TOC

[RFC3680]	Rosenberg, J., “A Session Initiation Protocol (SIP) Event Package for Registrations,” RFC 3680, March 2004 (TXT).
[RFC3880]	Lennox, J., Wu, X., and H. Schulzrinne, “Call Processing Language (CPL): A Language for User Control of Internet Telephony Services,” RFC 3880, October 2004 (TXT).
[RFC4235]	Rosenberg, J., Schulzrinne, H., and R. Mahy, “An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP),” RFC 4235, November 2005 (TXT).
[RFC4458]	Jennings, C., Audet, F., and J. Elwell, “Session Initiation Protocol (SIP) URIs for Applications such as Voicemail and Interactive Voice Response (IVR),” RFC 4458, April 2006 (TXT).
[RFC4825]	Rosenberg, J., “The Extensible Markup Language (XML) Configuration Access Protocol (XCAP),” RFC 4825, May 2007 (TXT).

[I-D.elwell-bliss-dnd]	Elwell, J. and S. Srinivasan, “An Analysis of Do Not Disturb (DND) Implementations in the Session Initiation Protocol (SIP),” draft-elwell-bliss-dnd-01 (work in progress), November 2007 (TXT).
[I-D.ietf-sipping-config-framework]	Channabasappa, S., “A Framework for Session Initiation Protocol User Agent Profile Delivery,” draft-ietf-sipping-config-framework-17 (work in progress), February 2010 (TXT).
[I-D.zourzouvillys-bliss-ach-http-api]	Zourzouvillys, T., “Basic HTTP API interface for ACH,” draft-zourzouvillys-bliss-ach-http-api-01 (work in progress), March 2009 (TXT).
[I-D.roach-sip-http-subscribe]	Roach, A., “A SIP Event Package for Subscribing to Changes to an HTTP Resource,” draft-roach-sip-http-subscribe-07 (work in progress), February 2010 (TXT).
[CSTA]	“International Standard ISO/IEC 18051 "Information Technology - Telecommunications and information exchange between systems - Services for Computer Supported Telecommunications Applications (CSTA) Phase III".”

Author's Address

TOC

	John Elwell
	Siemens Enterprise Communications
Phone:	+44 1908 855608
Email:	john.elwell@siemens-enterprise.com