An Analysis of Automatic Call Handling Implementation Issues in the Session Initiation Protocol (SIP)
draft-ietf-bliss-ach-analysis-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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

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

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

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

This Internet-Draft will expire on August 2, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

This discusses problems associated with automatic call handling (ACH) when using the Session Initiation Protocol (SIP).

This work is being discussed on the bliss@ietf.org mailing list.
Table of Contents

1. Introduction .............................................. 3
2. Examples of ACH ........................................... 3
3. Known problem areas with ACH ......................... 4
   3.1. Conflict between proxy and UA ..................... 4
   3.2. Conflict between UAs ............................. 5
   3.3. Obtaining information from UA for ACH at proxy ...... 5
   3.4. Informing the calling UA .......................... 5
   3.5. Scope of conditions ............................... 6
   3.6. Configuring the proxy ............................. 6
4. Discussion ................................................ 7
   4.1. Proxy versus UA ................................... 7
   4.2. Avoiding inconsistent configurations ............. 8
   4.3. Enterprise and carrier environments ............... 8
5. Conclusions ................................................. 8
6. IANA considerations ...................................... 9
7. Security considerations ................................. 9
8. Acknowledgements ....................................... 9
9. Informative References .................................. 9
Author’s Address ........................................... 10
Intellectual Property and Copyright Statements .......... 11
1. Introduction

The Session Initiation Protocol (SIP) (RFC 3261 [1]) 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.

2. Examples of ACH

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 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 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.).
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".

3. Known problem areas with ACH

3.1. Conflict between proxy and UA

A significant problem area with ACH is interactions between proxies (or 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.

3.2. Conflict between UAs

Where an incoming call is forked to multiple UAs, there is potential for different UAs to be configured to perform different 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 are likely to 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.

3.3. Obtaining information from UA for ACH at proxy

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 [6]), this can be a problem.

3.4. Informing the calling UA

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 3.3, 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 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 URI in which
could indicate the source of the redirection and the reason, in
c accordance with RFC 4458 [5].

3.5. Scope of conditions

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, this interpretation is not always 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.

3.6. Configuring the proxy

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.
- Via a web services interface based on SOAP.
- Via Computer Supported Telecommunication Applications (CSTA) [8].

Without a single standardized way of configuring a proxy, 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.

Related to this is the means by which a UA (and hence the user) can discover how the proxy is configured.

4. Discussion

4.1. Proxy versus UA

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 (RFC 3680 [2]) in combination with the dialog event package (RFC 4235 [4]) 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 [7] 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.
4.2. Avoiding inconsistent configurations

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.

4.3. 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.

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.

5. Conclusions

The discussions above raise a number of questions, which need to be addressed by the BLISS WG:

- Do we need to make any recommendations on proxy-based versus UA-based ACH?
o Is provisioning (e.g., using the SIP configuration framework [7]) a suitable solution for ensuring consistent behaviour between UAs and between a UA and its proxy?
o Do we need to make any recommendations on error codes and other information in responses in order to achieve good interoperability between a UAS and its proxy?
o Do we need to make any recommendations concerning the meaning and use of 6xx response codes?
o Do we need to specify a recommended or default method by which a UA can configure its proxy and discover how its proxy is configured?
o To what extent is it necessary to make recommendations concerning keeping the calling UA informed of what is happening?

6. IANA considerations

None.

7. Security considerations

This document just discusses issues relating to DND. 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 a DND condition to callers.

8. Acknowledgements

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

9. Informative References


Initiated Dialog Event Package for the Session Initiation

Protocol (SIP) URIs for Applications such as Voicemail and
Interactive Voice Response (IVR)", RFC 4458, April 2006.

draft-elwell-bliss-dnd-01 (work in progress), November 2007.

Initiation Protocol User Agent Profile Delivery",
draft-ietf-sipping-config-framework-14 (work in progress),
November 2007.

[8] "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

John Elwell
Siemens Enterprise Communications GmbH & Co KG
Hofmannstrasse 51
D-81379 Munich
Germany

Phone: +44 115 943 4989
Email: john.elwell@siemens.com