Basic HTTP API interface for ACH
draft-zourzouvillys-bliss-ach-http-api-01

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), 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 September 9, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This document defines a RESTful HTTP API that enables a SIP device (or agent acting on behalf of) a way to configure, enable, or disable services provided by the network.
# Table of Contents

1. Introduction .................................................. 3  
   1.1. Relationship to XCAP ................................. 3  
2. Terminology ................................................... 3  
3. Scope .......................................................... 3  
4. Protocol Model ............................................... 4  
   4.1. Example ................................................... 5  
5. General HTTP requirements .................................. 5  
   5.1. Server requirements .................................... 5  
   5.2. Client requirements .................................... 6  
   5.3. Cookies .................................................. 6  
6. Configuration Scope ......................................... 6  
7. Call forwarding ............................................... 6  
   7.1. Classes .................................................. 6  
      7.1.1. all .................................................. 7  
      7.1.2. noanswer ........................................... 7  
      7.1.3. unreachable ........................................ 7  
      7.1.4. busy ................................................ 7  
   7.2. Settings ................................................ 7  
      7.2.1. target ............................................... 7  
      7.2.2. status ............................................... 8  
      7.2.3. timeout ............................................. 8  
   7.3. Example ................................................ 8  
      7.3.1. Querying status of unconditional call forwarding .... 8  
      7.3.2. Enable unconditional call forwarding ............... 8  
8. Incoming Call Barring ........................................ 9  
   8.1. Anonymous Calls ........................................ 9  
9. Outgoing Call Barring ........................................ 9  
10. Do Not Disturb (DND) ........................................ 9  
11. Monitoring of changes ...................................... 10  
12. Security Considerations .................................... 10  
13. Acknowledgements .......................................... 10  
14. Normative References ...................................... 10  

Author’s Address .............................................. 10
1. Introduction

There are many situations in which a SIP phone (or user agent acting on behalf of a user) would like to have a way to programatically request the network perform actions on behalf of the user. For example, when a SIP phone is configured to forward all calls, it should be able to configure the network to provide this service rather than rely on the phone sending a 300-class response, as this would not work if the phone is offline.

It is common for SIP service providers to provide this feature for network configuration via a web interface, however a standardised API for changing this configuration setting would be beneficial for end users.

The mechanism defined in this specification allows a client to make simple HTTP requests to a server to query and modify network provided settings.

1.1. Relationship to XCAP

XCAP was not chosen for remote configuration. Due to additional requirements, a simple approach became very complex, perfect suited to modify XML documents and parts of it. But for our purpose, this is not really necessary. For this draft, a simple RESTful approach has been taken. Nevertheless valuable ideas, for example the XCAP URI structure, have been adopted.

2. Terminology

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].

3. Scope

This document is limited to a subset of network provided features and does not support more complex operations such as time-of-day routing. The following features have been identified by the BLISS REST design team to be provided in the first version of this protocol:

(1) Call forwarding unconditional (all calls), on busy, no answer, or not reachable: either to voicemail or to any other number or URI.
(2) Barring on all or certain classes of outgoing calls
(3) Barring on all or anonymous incoming calls.
(4) Enabling/Disabling call forwarding or DND.

OPEN-ISSUE: there are other scopes to consider, for example, each binding within the AOR, as well as the target URI for resulting in this AOR being reached (i.e., the "called number").

4. Protocol Model

The client is configured with a base API HTTP URL for the server. This document does not define any mechanisms for discovering a base API URL, and could be the subject of further study. Examples of mechanisms for discovering the base API URI could be through static configuration, by including a URI in a REGISTER response or through the UA configuration framework.

The configuration model defined in this document is outlined as a tree below. Each node is represented as a path in the HTTP URL that is formed by starting with the base API URI, and then appending each of the nodes with a ‘/’ separating them.
4.1. Example

Example base API URL: <https://api.example.com/alice>

The path for DND status would be:
https://api.example.com/alice/ach/dnd/status

5. General HTTP requirements

5.1. Server requirements

The HTTP server that implements this API MUST support HTTP/1.1.

If the server does not support any of the network services, it should return 404 for any of the nodes relevant to that service.
5.2. Client requirements

Any client accessing this API MUST support HTTP/1.1, and HTTP Digest authentication.

5.3. Cookies

The server MUST NOT rely on cookie support being provided by the HTTP user agent.

The client SHOULD NOT send any cookies, even if previously set by the server.

6. Configuration Scope

All configuration settings defined in this document apply to the AOR scope. In other words, a setting is applied to requests as they are received at an AOR rather than for each binding within the AOR.

This has the implication that any configuration change will apply to all bindings within the AOR. If 2 SIP devices are registered against an AOR, enabling call forward on one will also apply to the other.

Future work may introduce other scopes that settings can be applied at; for example, a specific instance binding rather than than the whole AOR.

7. Call forwarding

The call forwarding setting requests that the SIP proxy forwards all calls received for the given AOR to the instead be forwarded to the target URI configured.

OPEN-ISSUE: Should the SIP response codes what each class gets triggered by be defined by this document, or left as implementation specific?

7.1. Classes

There are 4 classes of call forwarding. This document defines the following classes:
(1) all
(2) noanswer
(3) busy
(4) unreachable

OPEN-ISSUE: These need precedence for the case of multiple matches; Which take priority? (the above seems to make sense to me)

7.1.1. all

This class applies at all times, and is also commonly known as "unconditional call forwarding".

7.1.2. noanswer

This class applies after a given number of seconds passed from the time the INVITE was original processed.

7.1.3. unreachable

This class applies when this AOR is unreachable, which could be either due to no bindings being available, or each of them has returned a failure response. It also applied when bindings are available but an attempt to send an error that indicates the UA is unreachable, for example a 480 or .

OPEN-ISSUE: does this apply to a proxy server that recurses on a 3xx and then gets an unconditional result?

7.1.4. busy

This class applies when bindings are available, but each returned a 486 or 600 response code.

If also applies if any of the devices returns a 603 (in addition to normal CANCEL processing).

7.2. Settings

Within each class, there are 2 configuration settings:

(1) target
(2) status

7.2.1. target

Content-Type: text/plain
This setting, which must be a valid URI, is the target of the new SIP request when the class in which it is set is enabled and the request matches.

OPEN-ISSUE: The content-type really should be something else - for example text/uri, except it doesn’t exist!

7.2.2. status

Content-Type: text/plain

This setting must be a value of either ‘enabled’ or ‘disabled’, and indicates if the class in which this setting is in is either enabled or disabled.

7.2.3. timeout

Content-Type: text/plain

This setting is the number of seconds which must pass since the request is original received before any outstanding branches are CANCELled and the request forked to the URI in the ‘target’ setting.

7.3. Example

7.3.1. Querying status of unconditional call forwarding

F1 Alice -> HTTP Server

GET /alice/ach/forwarding/all/status HTTP/1.1
Host: ach.api.example.com

F2 HTTP Server -> Alice

HTTP/1.1 200 OK
Content-Type: text/plain

enabled

7.3.2. Enable unconditional call forwarding

In this example, unconditional call forward is enabled for an AOR with the API root of ‘https://ach.api.example.com/’.
PUT /ach/alice/forwarding/all/target HTTP/1.1
Host: ach.api.example.com
Content-Type: text/plain
tel:+441908764181

PUT /ach/alice/forwarding/all/status HTTP/1.1
Host: api.example.com
Content-Type: text/plain

enabled

8. Incoming Call Barring

OPEN-ISSUE: perhaps add CPC baring?

8.1. Anonymous Calls

TODO: document

9. Outgoing Call Barring

Setting outgoing call barring is useful for example to restrict users of the phone from making a certain class of call by mistake, i.e premium rate numbers. The network may also be configurable to lock down the ability to change this setting.

TODO: document

10. Do Not Disturb (DND)

DND provides a mechanism to automatically reject all calls to an AOR with a status code and reason phrase configured by the user.

OPEN-ISSUE: While it initially seems like unconditional call forward is the same thing, DND is seen by many as a separate "service" due to setting it overwriting a previous setting target of unconditional forwarding.

OPEN-ISSUE: DND could perhaps be achieved by defining some service URNs that return the status code, for example 480. e.g: urn:sip:status:unavailable, and then setting unconditional call forward to it - thoughts?
11. Monitoring of changes

OPEN-ISSUE: should we reference draft-roach-sip-http-subscribe-01 for monitoring of settings?

12. Security Considerations

TODO: add text on outgoing call barring security considerations.

TODO: add other security considerations

13. Acknowledgements

The author would like to thank the members of the BLISS REST design team: Andy Hutton, Christian Schmidt, Salvatore Loreto, Sanjay Sinha, Shida Schubert and Simo Veikkolainen, as well as Michael Procter and John Elwell.

14. Normative References


Author’s Address

Theo Zourzouvillys
VoIP.co.uk
Commerce House
Telford Road
Bicester, Oxfordshire OX26 4LD
UK

Phone: +44 1908 764 181
Email: theo@voip.co.uk