Network Working Group
Request for Comments: 2324
Category: Informational

L. Masinter

1 April 1998

Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)

Status of this Memo

Copyright Notice

Abstract

This document describes HTCPCP, a protocol for controlling, monitoring, and diagnosing coffee pots.

1. Rationale and Scope

There is coffee all over the world. Increasingly, in a world in which computing is ubiquitous, the computists want to make coffee. Coffee brewing is an art, but the distributed intelligence of the web-connected world transcends art. Thus, there is a strong, dark, rich requirement for a protocol designed espressoly for the brewing of coffee. Coffee is brewed using coffee pots. Networked coffee pots require a control protocol if they are to be controlled.

Increasingly, home and consumer devices are being connected to the Internet. Early networking experiments demonstrated vending devices connected to the Internet for status monitoring [COKE]. One of the first remotely _operated_ machine to be hooked up to the Internet, the Internet Toaster, (controlled via SNMP) was debuted in 1990 [RFC2235].

The demand for ubiquitous appliance connectivity that is causing the consumption of the IPv4 address space. Consumers want remote control of devices such as coffee pots so that they may wake up to freshly brewed coffee, or cause coffee to be prepared at a precise time after the completion of dinner preparations.

This document specifies a Hyper Text Coffee Pot Control Protocol (HTCPCP), which permits the full request and responses necessary to control all devices capable of making the popular caffeinated hot beverages.

HTTP 1.1 ([RFC2068]) permits the transfer of web objects from origin servers to clients. The web is world-wide. HTCPCP is based on HTTP. This is because HTTP is everywhere. It could not be so pervasive without being good. Therefore, HTTP is good. If you want good coffee, HTCPCP needs to be good. To make HTCPCP good, it is good to base HTCPCP on HTTP.

Future versions of this protocol may include extensions for espresso machines and similar devices.

2. HTCPCP Protocol

The HTCPCP protocol is built on top of HTTP, with the addition of a few new methods, header fields and return codes. All HTCPCP servers should be referred to with the "coffee:" URI scheme (Section 4).

2.1 HTCPCP Added Methods

2.1.1 The BREW method, and the use of POST

Commands to control a coffee pot are sent from client to coffee server using either the BREW or POST method, and a message body with Content-Type set to "application/coffee-pot-command".

A coffee pot server MUST accept both the BREW and POST method equivalently. However, the use of POST for causing actions to happen is deprecated.

Coffee pots heat water using electronic mechanisms, so there is no fire. Thus, no firewalls are necessary, and firewall control policy is irrelevant. However, POST may be a trademark for coffee, and so the BREW method has been added. The BREW method may be used with other HTTP-based protocols (e.g., the Hyper Text Brewery Control Protocol).

2.1.2 GET method

In HTTP, the GET method is used to mean "retrieve whatever information (in the form of an entity) identified by the Request-URI." If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

In HTCPCP, the resources associated with a coffee pot are physical, and not information resources. The "data" for most coffee URIs contain no caffeine.

2.1.3 PROPFIND method

If a cup of coffee is data, metadata about the brewed resource is discovered using the PROPFIND method [WEBDAV].

2.1.4 WHEN method

When coffee is poured, and milk is offered, it is necessary for the holder of the recipient of milk to say "when" at the time when sufficient milk has been introduced into the coffee. For this purpose, the "WHEN" method has been added to HTCPCP. Enough? Say WHEN.

2.2 Coffee Pot Header fields

HTCPCP recommends several HTTP header fields and defines some new ones.

2.2.1 Recommended header fields

2.2.1.1 The "safe" response header field.

[SAFE] defines a HTTP response header field, "Safe", which can be used to indicate that repeating a HTTP request is safe. The inclusion of a "Safe: Yes" header field allows a client to repeat a previous request if the result of the request might be repeated.

The actual safety of devices for brewing coffee varies widely, and may depend, in fact, on conditions in the client rather than just in the server. Thus, this protocol includes an extension to the "Safe" response header:

```
Safe                = "Safe" ":" safe-nature
safe-nature         = "yes" | "no" | conditionally-safe
conditionally-safe  = "if-" safe-condition
safe-condition      = "user-awake" | token
```

indication will allow user agents to handle retries of some safe requests, in particular safe POST requests, in a more user-friendly way.

2.2.2 New header fields

2.2.2.1 The Accept-Additions header field

In HTTP, the "Accept" request-header field is used to specify media types which are acceptable for the response. However, in HTCPCP, the response may result in additional actions on the part of the automated pot. For this reason, HTCPCP adds a new header field, "Accept-Additions":

```
    Accept-Additions = "Accept-Additions" ":"
                       #( addition-range [ accept-params ] )

     addition-type   = ( "*"
                       | milk-type
                       | syrup-type
                       | sweetener-type
                       | spice-type
                       | alcohol-type
                       ) *( ";" parameter )
     milk-type       = ( "Cream" | "Half-and-half" | "Whole-milk"
                       | "Part-Skim" | "Skim" | "Non-Dairy" )
     syrup-type      = ( "Vanilla" | "Almond" | "Raspberry"
                       | "Chocolate" )
     alcohol-type    = ( "Whisky" | "Rum" | "Kahlua" | "Aquavit" )
```

2.2.3 Omitted Header Fields

No options were given for decaffeinated coffee. What’s the point?

2.3 HTCPCP return codes

Normal HTTP return codes are used to indicate difficulties of the HTCPCP server. This section identifies special interpretations and new return codes.

2.3.1 406 Not Acceptable

This return code is normally interpreted as "The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request. In HTCPCP, this response code MAY be returned if the operator of the coffee pot cannot comply with the Accept-Addition request. Unless the request was a HEAD request, the response SHOULD include an entity containing a list of available coffee additions.

In practice, most automated coffee pots cannot currently provide additions.

2.3.2 418 I'm a teapot

Any attempt to brew coffee with a teapot should result in the error code "418 I'm a teapot". The resulting entity body MAY be short and stout.

3. The "coffee" URI scheme

Because coffee is international, there are international coffee URI schemes. All coffee URL schemes are written with URL encoding of the UTF-8 encoding of the characters that spell the word for "coffee" in any of 29 languages, following the conventions for internationalization in URIs [URLI18N].

```
coffee-url  =  coffee-scheme ":" [ "//" host ]
                ["/" pot-designator ] ["?" additions-list ]

coffee-scheme = ( "koffie"                       ; Afrikaans, Dutch
                  | "q%C3%A6hv%C3%A6"          ; Azerbaijani
                  | "%D9%82%D9%87%D9%88%D8%A9" ; Arabic
                | "akeita"                    ; Basque
                | "koffee"                    ; Bengali
                | "kahva"                     ; Bosnian
                | "kafe"                      ; Bulgarian, Czech
                | "caf%C3%E8"                 ; Catalan, French, Galician
                  | "%E5%92%96%E5%95%A1"       ; Chinese
                  | "kava"                     ; Croatian
                | "k%C3%A1va"                 ; Czech
                | "kaffe"                     ; Danish, Norwegian, Swedish
                | "coffee"                    ; English
                | "kafo"                      ; Esperanto
                  | "kohv"                     ; Estonian
                | "kahvi"                     ; Finnish
                | "%4Baffee"                  ; German
                | "%CE%BA%CE%B1%CF%86%CE%AD" ; Greek
                | "%E0%A4%95%E0%A5%8C%E0%A4%AB%E0%A5%80" ; Hindi
                | "%E3%82%B3%E3%83%BC%E3%83%92%E3%83%BC" ; Japanese
                | "%EC%BB%A4%ED%94%BC"       ; Korean
                | "%D0%BA%D0%BE%D1%84%D0%B5" ; Russian
                | "%E0%B8%81%E0%B8%B2%E0%B9%81%E0%B8%9F" ; Thai
                )

   pot-designator = "pot-" integer  ; for machines with multiple pots
   additions-list = #( addition )
```

All alternative coffee-scheme forms are equivalent. However, the use of coffee-scheme in various languages MAY be interpreted as an indication of the kind of coffee produced by the coffee pot. Note that while URL scheme names are case-independent, capitalization is important for German and thus the initial "K" must be encoded.

4. The "message/coffeepot" media type

The entity body of a POST or BREW request MUST be of Content-Type "message/coffeepot". Since most of the information for controlling the coffee pot is conveyed by the additional headers, the content of "message/coffeepot" contains only a coffee-message-body:

coffee-message-body = "start" | "stop"

5. Operational constraints

This section lays out some of the operational issues with deployment of HTCPCP ubiquitously.

5.1 Timing Considerations

A robust quality of service is required between the coffee pot user and the coffee pot service. Coffee pots SHOULD use the Network Time Protocol [NTP] to synchronize their clocks to a globally accurate time standard.

Telerobotics has been an expensive technology. However, with the advent of the Cambridge Coffee Pot [CAM], the use of the web (rather than SNMP) for remote system monitoring and management has been proven. Additional coffee pot maintenance tasks might be accomplished by remote robotics.

Web data is normally static. Therefore to save data transmission and time, Web browser programs store each Web page retrieved by a user on the user’s computer. Thus, if the user wants to return to that page, it is now stored locally and does not need to be requested again from the server. An image used for robot control or for monitoring a changing scene is dynamic. A fresh version needs to be retrieved from the server each time it is accessed.

5.2 Crossing firewalls

In most organizations HTTP traffic crosses firewalls fairly easily. Modern coffee pots do not use fire. However, a "firewall" is useful for protection of any source from any manner of heat, and not just fire. Every home computer network SHOULD be protected by a firewall from sources of heat. However, remote control of coffee pots is

important from outside the home. Thus, it is important that HTCPCP cross firewalls easily.

By basing HTCPCP on HTTP and using port 80, it will get all of HTTP's firewall-crossing virtues. Of course, the home firewalls will require reconfiguration or new versions in order to accommodate HTCPCP-specific methods, headers and trailers, but such upgrades will be easily accommodated. Most home network system administrators drink coffee, and are willing to accommodate the needs of tunnelling HTCPCP.

6. System management considerations

Coffee pot monitoring using HTTP protocols has been an early application of the web. In the earliest instance, coffee pot monitoring was an early (and appropriate) use of ATM networks [CAM].

The traditional technique [CAM] was to attach a frame-grabber to a video camera, and feed the images to a web server. This was an appropriate application of ATM networks. In this coffee pot installation, the Trojan Room of Cambridge University laboratories was used to give a web interface to monitor a common coffee pot. of us involved in related research and, being poor, impoverished academics, we only had one coffee filter machine between us, which lived in the corridor just outside the Trojan Room. However, being highly dedicated and hard-working academics, we got through a lot of coffee, and when a fresh pot was brewed, it often didn't last long.

This service was created as the first application to use a new RPC mechanism designed in the Cambridge Computer Laboratory - MSRPC2. It runs over MSNL (Multi-Service Network Layer) - a network layer protocol designed for ATM networks.

Coffee pots on the Internet may be managed using the Coffee Pot MIB [CPMIB].

7. Security Considerations

Anyone who gets in between me and my morning coffee should be insecure.

Unmoderated access to unprotected coffee pots from Internet users might lead to several kinds of "denial of coffee service" attacks. The improper use of filtration devices might admit trojan grounds. Filtration is not a good virus protection method.

Putting coffee grounds into Internet plumbing may result in clogged plumbing, which would entail the services of an Internet Plumber [PLUMB], who would, in turn, require an Internet Plumber's Helper.

Access authentication will be discussed in a separate memo.

8. Acknowledgements

Many thanks to the many contributors to this standard, including Roy Fielding, Mark Day, Keith Moore, Carl Uno-Manros, Michael Slavitch, and Martin Duerst. The inspiration of the Prancing Pony, the CMU Coke Machine, the Cambridge Coffee Pot, the Internet Toaster, and other computer controlled remote devices have led to this valuable creation.

9. References

[RFC2068] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[RFC2186] Wessels, D., and K. Claffy, "Internet Cache Protocol (ICP), version 2," RFC 2186, September 1997

[CPMIB] Slavitch, M., "Definitions of Managed Objects for Drip-Type Heated Beverage Hardware Devices using SMIv2", RFC 2325, 1 April 1998.

[HTSVMP] Q. Stafford-Fraser, "Hyper Text Sandwich Van Monitoring Protocol, Version 3.2". In preparation.

[RFC2295] Holtman, K., and A. Mutz, "Transparent Content Negotiation in HTTP", RFC 2295, March 1998.

[SAFE] K. Holtman. "The Safe Response Header Field", September 1997.

[CAM] "The Trojan Room Coffee Machine", D. Gordon and M. Johnson, University of Cambridge Computer Lab, <http://www.cl.cam.ac.uk/coffee/coffee.html>

[CBIO] "The Trojan Room Coffee Pot, a (non-technical) biography", Q. Stafford-Fraser, University of Cambridge Computer Lab, <http://www.cl.cam.ac.uk/coffee/qsf/coffee.html>.

[RFC2235] Zakon, R., "Hobbes' Internet Timeline", FYI 32, RFC 2230, November 1997. See also <http://www.internode.com.au/images/toaster2.jpg>

[NTP] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992.

[URLI18N] Masinter, L., "Using UTF8 for non-ASCII Characters in Extended URIs" Work in Progress.

[PLUMB] B. Metcalfe, "Internet Plumber of the Year: Jim Gettys", Infoworld, February 2, 1998.

[COKE] D. Nichols, "Coke machine history", C. Everhart, "Interesting uses of networking", <http://www-cse.ucsd.edu/users/bsy/coke.history.txt>.

10. Author's Address

Larry Masinter
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

EMail: masinter@parc.xerox.com

11. Full Copyright Statement