Network Working Group
Request for Comments: 2109
Category: Standards Track

D. Kristol
Bell Laboratories, Lucent Technologies
L. Montulli
Netscape Communications

February 1997

HTTP State Management Mechanism

Status of this Memo

1. ABSTRACT

This document specifies a way to create a stateful session with HTTP requests and responses. It describes two new headers, Cookie and Set-Cookie, which carry state information between participating origin servers and user agents. The method described here differs from Netscape's Cookie proposal, but it can interoperate with HTTP/1.0 user agents that use Netscape's method. (See the HISTORICAL section.)

2. TERMINOLOGY

The terms user agent, client, server, proxy, and origin server have the same meaning as in the HTTP/1.0 specification.

Fully-qualified host name (FQHN) means either the fully-qualified domain name (FQDN) of a host (i.e., a completely specified domain name ending in a top-level domain such as .com or .uk), or the numeric Internet Protocol (IP) address of a host. The fully qualified domain name is preferred; use of numeric IP addresses is strongly discouraged.

The terms request-host and request-URI refer to the values the client would send to the server as, respectively, the host (but not port) and abs_path portions of the absoluteURI (http_URL) of the HTTP request line. Note that request-host must be a FQHN.

Hosts names can be specified either as an IP address or a FQHN string. Sometimes we compare one host name with another. Host A's name domain-matches host B's if

* both host names are IP addresses and their host name strings match exactly; or

* both host names are FQDN strings and their host name strings match exactly; or

* A is a FQDN string and has the form NB, where N is a non-empty name string, B has the form .B', and B' is a FQDN string. (So, x.y.com domain-matches .y.com but not y.com.)

Note that domain-match is not a commutative operation: a.b.c.com domain-matches .c.com, but not the reverse.

Because it was used in Netscape's original implementation of state management, we will use the term cookie to refer to the state information that passes between an origin server and user agent, and that gets stored by the user agent.

3. STATE AND SESSIONS

This document describes a way to create stateful sessions with HTTP requests and responses. Currently, HTTP servers respond to each client request without relating that request to previous or subsequent requests; the technique allows clients and servers that wish to exchange state information to place HTTP requests and responses within a larger context, which we term a "session". This context might be used to create, for example, a "shopping cart", in which user selections can be aggregated before purchase, or a magazine browsing system, in which a user's previous reading affects which offerings are presented.

There are, of course, many different potential contexts and thus many different potential types of session. The designers' paradigm for sessions created by the exchange of cookies has these key attributes:

1. Each session has a beginning and an end.

2. Each session is relatively short-lived.

3. Either the user agent or the origin server may terminate a session.

4. The session is implicit in the exchange of state information.

4. OUTLINE

We outline here a way for an origin server to send state information to the user agent, and for the user agent to return the state information to the origin server. The goal is to have a minimal impact on HTTP and user agents. Only origin servers that need to maintain sessions would suffer any significant impact, and that impact can largely be confined to Common Gateway Interface (CGI) programs, unless the server provides more sophisticated state management support. (See Implementation Considerations, below.)

4.1 Syntax: General

The two state management headers, Set-Cookie and Cookie, have common syntactic properties involving attribute-value pairs. The following grammar uses the notation, and tokens DIGIT (decimal digits) and token (informally, a sequence of non-special, non-white space characters) from the HTTP/1.1 specification [RFC 2068] to describe their syntax.

```
av-pairs        =       av-pair *(";" av-pair)
av-pair         =       attr ["=" value]        ; optional value
attr            =       token
value           =       word
word            =       token | quoted-string
```

Attributes (names) (attr) are case-insensitive. White space is permitted between tokens. Note that while the above syntax description shows value as optional, most attrs require them.

NOTE: The syntax above allows whitespace between the attribute and the = sign.

4.2 Origin Server Role

4.2.1 General

The origin server initiates a session, if it so desires. (Note that "session" here does not refer to a persistent network connection but to a logical session created from HTTP requests and responses. The presence or absence of a persistent connection should have no effect on the use of cookie-derived sessions). To initiate a session, the origin server returns an extra response header to the client, Set-Cookie. (The details follow later.)

A user agent returns a Cookie request header (see below) to the origin server if it chooses to continue a session. The origin server may ignore it or use it to determine the current state of the

session. It may send back to the client a Set-Cookie response header with the same or different information, or it may send no Set-Cookie header at all. The origin server effectively ends a session by sending the client a Set-Cookie header with Max-Age=0.

Servers may return a Set-Cookie response headers with any response. User agents should send Cookie request headers, subject to other rules detailed below, with every request.

An origin server may include multiple Set-Cookie headers in a response. Note that an intervening gateway could fold multiple such headers into a single header.

4.2.2 Set-Cookie Syntax

The syntax for the Set-Cookie response header is

```
set-cookie      =       "Set-Cookie:" cookies
cookies         =       1#cookie
cookie          =       NAME "=" VALUE *(";" cookie-av)
NAME            =       attr
VALUE           =       value
cookie-av       =       "Comment" "=" value
                |       "Domain" "=" value
                |       "Max-Age" "=" value
                |       "Path" "=" value
                |       "Secure"
                |       "Version" "=" 1*DIGIT
```

Informally, the Set-Cookie response header comprises the token Set-Cookie:, followed by a comma-separated list of one or more cookies. Each cookie begins with a NAME=VALUE pair, followed by zero or more semi-colon-separated attribute-value pairs. The syntax for attribute-value pairs was shown earlier. The specific attributes and the semantics of their values follows. The NAME=VALUE attribute-value pair must come first in each cookie. The others, if present, can occur in any order. If an attribute appears more than once in a cookie, the behavior is undefined.

NAME=VALUE
: Required. The name of the state information ("cookie") is NAME, and its value is VALUE. NAMEs that begin with $ are reserved for other uses and must not be used by applications.

The VALUE is opaque to the user agent and may be anything the origin server chooses to send, possibly in a server-selected printable ASCII encoding. "Opaque" implies that the content is of interest and relevance only to the origin server. The content may, in fact, be readable by anyone that examines the Set-Cookie header.

Comment=comment
Optional. Because cookies can contain private information about a user, the Cookie attribute allows an origin server to document its intended use of a cookie. The user can inspect the information to decide whether to initiate or continue a session with this cookie.

Domain=domain
Optional. The Domain attribute specifies the domain for which the cookie is valid. An explicitly specified domain must always start with a dot.

Max-Age=delta-seconds
Optional. The Max-Age attribute defines the lifetime of the cookie, in seconds. The delta-seconds value is a decimal non-negative integer. After delta-seconds seconds elapse, the client should discard the cookie. A value of zero means the cookie should be discarded immediately.

Path=path
Optional. The Path attribute specifies the subset of URLs to which this cookie applies.

Secure
Optional. The Secure attribute (with no value) directs the user agent to use only (unspecified) secure means to contact the origin server whenever it sends back this cookie.

The user agent (possibly under the user's control) may determine what level of security it considers appropriate for "secure" cookies. The Secure attribute should be considered security advice from the server to the user agent, indicating that it is in the session's interest to protect the cookie contents.

Version=version
Required. The Version attribute, a decimal integer, identifies to which version of the state management specification the cookie conforms. For this specification, Version=1 applies.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.