David M. Kristol
AT&T Bell Laboratories

Expires March 22, 1995

Proposed HTTP State-Info Mechanism

Status of this Memo

This is author’s draft 1.15.

1. ABSTRACT

HTTP, the protocol that underpins the World-Wide Web (WWW), is stateless. That is, each request stands on its own; origin servers don’t need to remember what happened with previous requests to service a new one. Statelessness is a mixed blessing, because there are potential WWW applications, like ‘‘shopping baskets’’ and library browsing, for which the history of a user’s actions is useful or essential.

This proposal outlines a way to introduce state into HTTP. A new request/response header, `State-Info`, carries the state back and forth, thus relieving the origin server from needing to keep an extensive per-user or per-connection database. The changes required to user agents, origin servers, and proxy servers to support `State-Info` are very modest.

2. TERMINOLOGY

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

3. STATE AND SESSIONS

This proposal outlines how to introduce state into HTTP, the protocol that underpins the World-Wide Web (WWW). At present, HTTP is stateless: a WWW origin server obtains everything it needs to know about a request from the request itself. After it processes the request, the origin server can ‘‘forget’’ the transaction.

What do I mean by ‘‘state?’’ ‘‘State’’ implies some relation between one request to an origin server and previous ones made by the same user agent to the same origin server. If the sequence of these requests is considered as a whole, they can be thought of as a ‘‘session.’’

Koen Holtman identified these dimensions for the ‘‘solution space’’ of stateful dialogs:

- simplicity of implementation
- simplicity of use

- time of general availability when standardized
- downward compatibility
- reliability
- amount of privacy protection
- maximum complexity of stateful dialogs supported
- amount of cache control possible
- risks when used with non-conforming caches

The paradigm I have in mind obtains the same effect as if a user agent connected to an origin server, carried out many transactions at the user's direction, then disconnected. Two example applications I have in mind are a ''shopping cart,'' where the state information comprises what the user has bought, and a magazine browsing system, where the state information comprises the set of journals and articles the user has looked at already. Note some of the key points in the session paradigm:

1. The session has a beginning and an end.
2. The session is relatively short-lived.
3. Either the user agent or the origin server may terminate a session.
4. State is a property of the connection to the origin server. The user agent itself has no special state information. (However, what the user agent presents to the user may reflect the origin server's state, because the origin server returns that information to the user agent.)

4. PROPOSAL OUTLINE

The proposal I outline here defines 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 of the proposal 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-Info` support. (See **Implementation Considerations**, below.)

4.1 Origin Server Role

The origin server initiates a session, if it so desires. (Note that ''session'' here is a logical connection, not a physical one. Don't confuse these logical sessions with various ''keepalive'' proposals for physical sessions.) To initiate a session, the origin server returns an extra response header to the client:

`State-Info:` *opaque information*

The *opaque information* may be anything the origin server chooses to send, encoded in printable ASCII. ''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 `State-Info` header.

If the origin server gets a `State-Info` request header from the client (see below), it may ignore it or use it to determine the current state of the session. It may send back to the client the same, a different, or no `State-Info` response header. The origin server effectively ends a session by sending back a `State-Info` header that has a null value.

An origin server must be cognizant of the effect of possible caching by other agents of responses that have a `State-Info` header. Generally a document that has associated `State-Info` should not be cached: the cached resource or `State-Info` may be specific to a particular user agent. The origin server must notify upstream agents not to cache them. To inhibit caching, the origin server should use one of the standard mechanisms that inhibit caching, such as `Cache-control: no-cache` or `Expires:` *<yesterday>*.

4.2 User Agent Role

The user agent keeps track of `State-Info` for each origin server (distinguished by name or IP address and port). The extent of its bookkeeping is to note that it does or does not have `State-Info` for the origin server.

The user agent goes from the ‘‘no `State-Info`’’ state to the ‘‘have `State-Info`’’ state when it receives a non-empty `State-Info` response header from the origin server. (The user agent saves the `State-Info` value.) It returns to the ‘‘no `State-Info`’’ state if it receives a `State-Info` response header with a null value. It stays in the ‘‘have `State-Info`’’ state if it receives a non-null `State-Info` response header; the new value overwrites the old one. If the user agent receives no `State-Info` response header, it stays in the same state (‘‘have `State-Info`’’ or ‘‘no `State-Info`’’). The behavior described above applies for all response codes from the origin server.

When it sends a request to an origin server, the user agent sends a `State-Info` request header if it’s in the ‘‘have `State-Info`’’ state; otherwise it sends no `State-Info` request header.

A user agent usually begins execution with no remembered `State-Info` information. The user agent may be configured never to send `State-Info`, in which case it can never sustain state with an origin server. (This would also be true of user agents that are unaware of how to handle `State-Info`.)

A user agent (at the user’s direction) can terminate a session with an origin server by discarding the associated `State-Info` information (moving to the ‘‘no `State-Info`’’ state).

When the user agent terminates execution, it discards all `State-Info` information. Alternatively, the user agent may ask the user whether `State-Info` should be retained; the default should be ‘‘no.’’ Retained `State-Info` would then be restored when the user agent begins execution again.

User agent programs that can display multiple independent windows should behave as if each window were a separate program instance with respect to `State-Info`. Thus `State-Info` obtained in one window would have no effect on links followed in another. (The user agent would have to store `State-Info` tagged by window number, as well as origin server address and port.) When a window terminates, all associated `State-Info` information gets discarded.

4.3 Caching Proxy Role

One reason for separating state information from both a URL and document content is to facilitate the scaling that caching permits. To support `State-Info`, a caching proxy must obey these rules already in the HTTP specification:

- Honor requests from the cache, if possible, based on cache validity rules.
- Pass along a `State-Info` request header in any request that the proxy must make of another server, and return the response (including any `State-Info` response header) to the client.
- Cache the received response subject to the control of the usual headers, such as `Expires` and `Cache-control: no-cache`.

5. IMPLEMENTATION CONSIDERATIONS

Here I speculate on likely or desirable details for an origin server that implements `Server-Info`.

5.1 `State-Info` Content

An origin server’s content should probably be divided into disjoint application areas, some of which require the use of `State-Info`. The application areas can be distinguished by their request URLs. The `State-Info` header can incorporate information about multiple sessions that a user agent might start as follows. Imagine that a single session’s state information takes the form

URL opaque

The *opaque* information might be a *uuencoding* of application-specific information. The *URL* might be the actual URL of a resource, or it might be the prefix for all URLs that comprise a particular application. The `State-Info` header for multiple sessions can be formed by concatenating the session state information of all sessions, separated by commas, as in

```
State-Info: /A YXBwbGljYXRpb246MQ==, /B YXBwbGljYXRpb246Mg==
```

The session information can obviously be clear or encoded text that describes state. However, if it grows too large, it can become unwieldy. Therefore, an implementor might choose for the session information to be a key into a server-side database. Of course, using a database creates some problems that the `State-Info` proposal was meant to avoid, namely:

1. keeping real state on the server side;
2. how and when to garbage-collect the database entry, in case the user agent terminates the session by, for example, exiting.

The origin server software should probably be designed to separate the session information for different applications and only present to a particular application the session information that applies to it.

5.2 Stateless Pages

Caching is a good thing for the scalability of WWW. Therefore it's important to reduce the number of documents that have state embedded in them inherently. For example, if a shopping-basket-style application always displayed a user's current basket contents on each page, those pages could not be cached, because each user's basket's contents would be different. On the other hand, if each page contained just a link that allowed the user to ‘‘Look at My Shopping Basket,’’ the page could be cached.

6. PRIVACY

An origin server can create a `State-Info` header to track the path of a user through the server. Users may object to this behavior as an intrusive accumulation of information, even if their identity is not evident. (Identity might become evident if a user fills out a form that contains identifying information.) The `State-Info` proposal therefore gives a user some control over this possible intrusion by recommending that

- a user agent be able, as a configuration option, never to create stateful sessions.
- a user agent allow a user to discard `State-Info` at any time.
- a user agent notify a user when a stateful session begins, giving the user the opportunity not to continue the session.
- a user agent display a visual indication that a page is part of a stateful session.
- terminating a user agent's execution (or the execution of a window, for multi-window user agents) cause `State-Info` to be discarded.

Note that caches intervening between a user agent and origin server can distort the tracking information.

7. OTHER, SIMILAR, PROPOSALS

I'm aware of two other proposals to accomplish similar goals. Netscape proposes a `Cookie` request header and `Set-Cookie` response header. Netscape cookies have expiration times and other information that require more complicated processing by the user agent than does my proposal. Furthermore, the user has no control over cookies' lifetime.

Brian Behlendorf proposed a `Session-ID` header that would be user-agent-initiated and could be used by an origin server to track ‘‘clickstreams.’’ It would not carry any origin-server-defined state, however.

Koen Holtman has made a proposal that is similar in flavor to, but different in detail from, this one. Many of his ideas have been incorporated in this proposal.

8. SECURITY CONSIDERATIONS

The information in the `State-Info` headers is unprotected. Two consequences are:

1. Any sensitive information that is conveyed in a `State-Info` header is exposed to intruders.
2. A malicious intermediary could alter the `State-Info` header as it travels in either direction, with unpredictable results.

These facts imply that information of a personal and/or financial nature should only be sent over a secure channel. For less sensitive information, or when the content of the header is a database key, an origin server should be vigilant to prevent a bad `Session-Info` value from causing it to fail.

9. ACKNOWLEDGEMENTS

My thanks go to correspondents on the `http-wg` and `www-talk` mailing lists who contributed ideas and criticism that found its way into this proposal. Special thanks to Bob Wyman, Koen Holtman, Shel Kaphan.

10. AUTHOR'S ADDRESS

David M. Kristol
AT&T Bell Laboratories
600 Mountain Ave. Room 2A-227
Murray Hill, NJ 07974

Phone: (908) 582-2250
FAX: (908) 582-5809
Email: `dmk@research.att.com`