



**Framework Implementation Project
Technical Note**

The Use of HTTP Versus Sockets

July 25, 2001



In response to the ongoing discussion about the choice of communications protocol to be used in the Framework, we have developed a list of issues relating to the choice of the IPC 2501 standard without the addition of HTTP-specific requirements. Although HTTP is better known, there are important reasons for not using it in this instance. In sum, HTTP would not add functionality or ease of use; in fact, it is likely to be counter-productive in this application. Other Framework members and developers agree with our reasoning, which is the outcome of our own development experience during the past year. The major points are summarized below.

1) The nature of our communications

We are employing a publish-subscribe paradigm in the Framework (via the Message Broker). HTTP was created specifically for request-response environments. Because of this, using HTTP in the Framework would require that we use HTTP in a very non-standard way, both in theory and in implementation. To implement a multi-node publish-subscribe system like the Framework, a full-duplex connection must be established for any client node. HTTP was not intended for full-duplex communication. Although the HTTP protocol could be used to accomplish full duplex communication, it could require that every entity in the Framework, even clients, act as an HTTP server. This would also require work-arounds and extensions to COTS (commercial off-the-shelf) HTTP software or APIs, resulting in a nonstandard HTTP implementation.

2) The place of the IPC 2501 standard

The 2501 standard defines an envelope for data transmitted in the Framework, as well as control messages, error messages, and intelligent status messaging for all clients connected. This act of defining a 'header' on all data put in a communications system (as well as the definition of errors, etc.) closely parallels what is done by HTTP. HTTP, however, does not define all the data points required to implement a system like the Framework. The 2501 was created because existing protocols are insufficient for controlling a system like the Framework. By use of the 2501, the XML sent becomes entirely self-describing.

3) HTTP as a layer on top of sockets

HTTP itself is a layer over sockets. Sockets were a precursor to HTTP, and sockets are employed in nearly all network systems, inclusive of any that use HTTP. As seen in (2), the additional functionality that HTTP provides over pure sockets is not sufficient for our situation, and thus we have defined a protocol (the IPC 2501) that makes the HTTP layer redundant or simply unused.

4) The past effectiveness of HTTP implementations

In previous attempts to use HTTP in the Framework, HTTP functionalities went almost entirely un-used. Software opened the HTTP connection and then effectively changed that connection into sockets. This applies to both the server (MSB) and client software. Programmers extracted raw communication streams from the socket being used by HTTP, thus negating any functionality

provided by HTTP. Also, since any HTTP streams used were held open for communications, no verification or error messaging could occur on a per-message basis.

5) Future Framework requirements

New capabilities have been suggested for implementation in the Framework, such as guaranteed delivery and security. HTTP does not specifically allow for guaranteed delivery without some modification (status codes and software interfaces must be customized). Also, HTTPS (the secure version of HTTP) is HTTP built on secure socket technology. Again, HTTP is built on sockets, which provide the functionality we are really after. Thus, it is our view that HTTP does not provide any additional functionality either in the current Framework, or in future Framework implementations.

6) Variations in HTTP implementations

HTTP is a protocol that has existed for some time, and has been used for many purposes, some of which have proven to be inappropriate. As a result, different COTS HTTP software implementations will tend to provide different functionality. There have been several attempts to make HTTP a more robust protocol, such as server-push, but they have all ended up as proprietary implementations that were neither successful nor widely accepted.

In conclusion, we believe the use of sockets is a more suitable choice for the Framework since sockets are simpler, faster, more standardized, and more reliable than HTTP. In addition, we believe making use of HTTP may confuse implementers since it would be applied in such a nontraditional manner, requiring modifications that would make the system more complex and less efficient.

Submitted by: Daniel Lee Osiecki, Jeffrey M. Gerth