CONTROLLING EPICS FROM A WEB BROWSER

K. Evans, Jr.†
Argonne National Laboratory, Argonne, IL

Abstract
An alternative to using a large graphical display manager like MEDM [1,2] to interface to a control system, is to use individual control objects, such as text boxes, meters, etc., running in a browser. This paper presents three implementations of this concept, one using ActiveX controls, one with Java applets, and another with Microsoft Agent [3]. The ActiveX controls have performance nearing that of MEDM, but they only work on Windows platforms. The Java applets require a server to get around Web security restrictions and are not as fast, but they have the advantage of working on most platforms and with both of the leading Web browsers. The agent works on Windows platforms with and without a browser and allows voice recognition and speech synthesis, making it somewhat more innovative than MEDM.

1 INTRODUCTION
The concepts described in this paper fall into two groups: (1) the browser objects: ActiveX controls and Java applets, and (2) the agent application. We will speak of the browser objects first and leave the agent application for the end. We will only consider the use of these controls in EPICS [4].

There are advantages to using a Web browser rather than a large program like MEDM to access a control system, particularly if the control system is small or you have special needs. The browser replaces MEDM’s EXECUTE mode, and your favorite HTML editor replaces MEDM’s EDIT mode. Only the objects need to be provided. The rest is done for you by large and presumably competent groups of programmers and designers working for well-known companies.

In place of the somewhat fixed objects that are available in a program like MEDM, the browser objects can be about anything that a person wants. They are relatively small and are largely self-contained. The ease with which they can be modified solves the extensibility limitations with the large graphical control-system interfaces, such as MEDM. Once the boilerplate code that makes these controls work with the control system and the browser is written, the specific functionality of whether they are, say, a meter or a text entry is relatively simple and easily changed.

In addition to being easily changed, the objects can communicate with each other and to other browser objects via their methods and properties in ways that MEDM-type objects do not.

2 HTML FOR BROWSER OBJECTS
It is important to keep in mind that these browser objects are, indeed, objects in the sense of object-oriented design. That is, they have properties and methods, and they respond to events. How they do this is typically encapsulated in the object and is often of no concern to the person who uses them. When such an object is used in a Web page, it is manipulated via these properties, methods, and events.

The types of objects we are describing are incorporated into a browser page in much the same way as the images we see all the time. Figure 1 shows typical HTML for images, applets, and controls. The PARAMs, which are the basic difference from an image, specify the object’s properties.

![Figure 1: HTML for three kinds of browser object.](image-url)
Figure 2 shows a browser object that we have imple-
mented both as an ActiveX control and as a Java applet. It is a text area that includes
the name of the process variable and its value as an alphanumeric string. It has prop-
erties such as its foreground and background colors, the
process variable name and value, whether the name
shows, and whether it is monitoring. It has methods such
as GetValue, PutValue, and AboutBox. It does not, but
could, respond to events. Other objects, such as check
boxes, in the browser page may access our object's meth-
ods and properties. For example, the text entry for the
name and the push button next to it allow you to change
the process variable name (and connect to the new name).
The Monitor check box allows you to set it to monitor or
not. This interaction between objects is something that
cannot be done in MEDM. Further, you can put other
sorts of browser things, like links, in the page, so there is
no need for MEDM's Related Display object.

2 ACTIVEX CONTROLS

ActiveX controls are primarily useful in Windows. They
work with Internet Explorer [5] but do not work with Net-
scape [6]. ActiveX controls are written in a language,
such as C, that makes platform-dependent code and which
is not safe. That is, the control can contain code that, for
instance, deletes files on your computer. Internet Ex-
plorer allows you to specify whether you want to run such
controls at all, have it ask before running them, or always
run them without asking. The default is to not run them,
but it is easily changed. These "unsafe" controls are more
relevant to an Intranet than the Internet. Just like MEDM,
which is also written in C and can delete files on your
computer, you must trust them.

Since our controls must interact with EPICS Channel
Access, which is written in C, they are written in C. Since
C is a strong and well-developed language and is
optimized for a particular platform, this makes them fast
and efficient. Figure 2 shows a Web page handling 100
of our controls, set to have the name not show and
attached to process variables that are updating at 10 Hz.
The browser displaying this page is running on a 200-
MHz PC that is connected over an ISDN line through a
PV Gateway. The performance approaches that of
MEDM. Note that the menu controls labeled "Change"
are ActiveX controls that come with Internet Explorer.
They change the scan rate for various blocks of the page
by communicating with the ActiveX controls that access
the appropriate SCAN process variables (the ones mostly
displaying Passive, which are overridden by the global
control with the red foreground.).

3 JAVA APPLETS

One of the primary benefits of Java is that it is platform
independent. Java applets work in most browsers and on
most platforms, while ActiveX controls work on Win-
dows and require Internet Explorer. It is difficult to use
them on UNIX. We have made a Java applet that looks
like and has the same functionality as the ActiveX one. It
has the same methods and properties and works the same
way. The Web page looks essentially the same.

One problem that arises is that Java security is different
than ActiveX security. Java applets running in a browser
have a "sandbox" in which they must operate. One of the
rules is that they may not access files or sockets on your
machine. This makes them safe. They are allowed to
access files and sockets on the server machine, the one
that served the Web page and the applet. (What the server
lets its applets do is their problem and is not a security
issue for you.)
A second problem is that EPICS Channel Access is written in C, not Java. Java does provide a means, JNI, for using native languages, such as C, with Java. The result is not "Pure Java," and it is not platform independent. Moreover, one of the sandbox rules is that applets cannot run JNI code on your machine. We are stuck with the facts that we must use JNI in order to use Channel Access and that we cannot use it in the applet.

Consequently, we need to serve the applet from an HTTP server and provide a Channel-Access server to talk to the control system. The arrangement is shown in Fig. 4. For ActiveX, the controls can live on the workstation and talk directly to the EPICS input/output controller (IOC). For Java, they must live on the HTTP server and talk through sockets to the Channel-Access server on the same machine as the HTTP server. The Channel-Access server then talks to the IOC. On the positive side, Java is strong in network capabilities, and it is relatively easy to write the Channel-Access server in Java and to have the applet communicate with it.

To the user in his browser, the two controls appear to operate the same up to a point. The performance of Java is slower than that of C, and, to date, we have not found it possible to make a high-performance page, such as the one in Fig. 3, work well for Java. Less ambitious pages work fine. The primary advantage of Java is that the applets can be used in Netscape and consequently on UNIX.

It should be noted that there are means to overcome the security restrictions for Java applets in a browser. Also, you can run ActiveX controls in UNIX and in Netscape. What we have described are the restrictions when doing things the normal way.

4 AGENT APPLICATIONS

Microsoft Agent is a set of software services that supports the presentation of software agents as animated, interactive personalities. It is a glorification of the Office Assistant found in Microsoft Office [7]. It will work either from a browser page or as an application. Like with the browser, most of the programming has been done for you. You just have to implement the little bit you need. Among the capabilities provided for you are voice recognition and speech synthesis. Figure 5 shows the agent, in the form of a genie, getting the value of a process variable from the control system. To get him to do this, you would say something like "Genie, get me a process variable," then enter the name in a dialog box he gives you. He speaks the words in the balloons as well as displaying them. (The dialog box is used because he is not yet up to recognizing the abstruse types of names typically used for process variables, though he does pretty well at pronouncing them.) We are truly at the point where we can demonstrably and feasibly converse with the control system and say things like, "Accelerator, correct the orbit," or "Telescope, move twenty degrees to the East," as fictionally happened some time ago with Hal in 2001 and happens regularly on Star Trek.

REFERENCES

[5] Internet Explorer is a product of Microsoft Corporation, Redmond, WA.
[6] Netscape is a product of Netscape Communications Corporation, Mountain View, CA.
[7] Microsoft Office is a product of Microsoft Corporation, Redmond, WA.