16th Int. Conf. on Accelerator and Large Experimental Control Systems ICALEPCS2017, Barcelona, Spain JACoW Publishing
ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA181

WEB EXTENSIBLE DISPLAY MANAGER*

R. Slominski, JLab, Newport News, VA 23606, USA
T. Larrieu, JLab, Newport News, VA 23606, USA

Abstract

Jefferson Lab's Web Extensible Display Manager (WEDM) allows staff to access EDM control system screens from a web browser in remote offices and from mobile devices. Native browser technologies are leveraged to avoid installing and managing software on remote clients such as browser plugins, tunnel applications, or an EDM environment. Since standard network ports are used firewall exceptions are minimized. To avoid security concerns from remote users modifying a control system, WEDM exposes read-only access and basic web authentication can be used to further restrict access. Updates of monitored EPICS channels are delivered via a Web Socket using a web gateway. The software translates EDM description files (denoted with the *edl* suffix) to HTML with Scalable Vector Graphics (SVG) following the EDM's *edl* file vector drawing rules to create faithful screen renderings. The WEDM server parses *edl* files and creates the HTML equivalent in real-time allowing existing screens to work without modification. Alternatively, the familiar drag and drop EDM screen creation tool can be used to create optimized screens sized specifically for smart phones and then rendered by WEDM.

INTRODUCTION

WEDM employs native web technologies including Web Sockets, HTML 5 and SVG to deliver faithful renderings of control system EDM screens for remote users who can view them using nothing more than a web browser. The ease at which screens can be viewed facilitates on call troubleshooting and off-site monitoring of control systems. Figure 1 illustrates an EDM controls screen rendered in a browser via WEDM.

The initial goals for the deployment of WEDM include (1) enable improved response times of the accelerator support personnel (reduce the Mean Time To Repair, MTTR, improve system availability), (2) reduce the number of staff granted accounts on control system workstations, and (3) reduce the number of special-purpose EPICS kiosks in use.

Improved Response Time

With WEDM available, a support personnel alerted to a problem need no longer have access to a full-blown PC with network access dependent upon 2-factor hardware authentication in order to view screens. As long as cellular service or Wi-Fi is available, identical menus and screens as available to the control room operator are accessible through any web browser. Even without the ability to make changes directly, they can provide diagnosis and guidance or even walk the operator through making changes to specific settings.

Fewer Control System Computer Accounts

Prior to WEDM, it was necessary to grant users full accounts on control system workstations in order to run EDM and view screens. A significant number of these users have no need to update settings and can now access their screens via the web simply using their standard JLab username and password. For example, the facilities management department is switching from expensive proprietary monitoring software to EPICS for monitoring their gas, water, and electricity meters. These users will rely on WEDM to view the outputs of their meters.

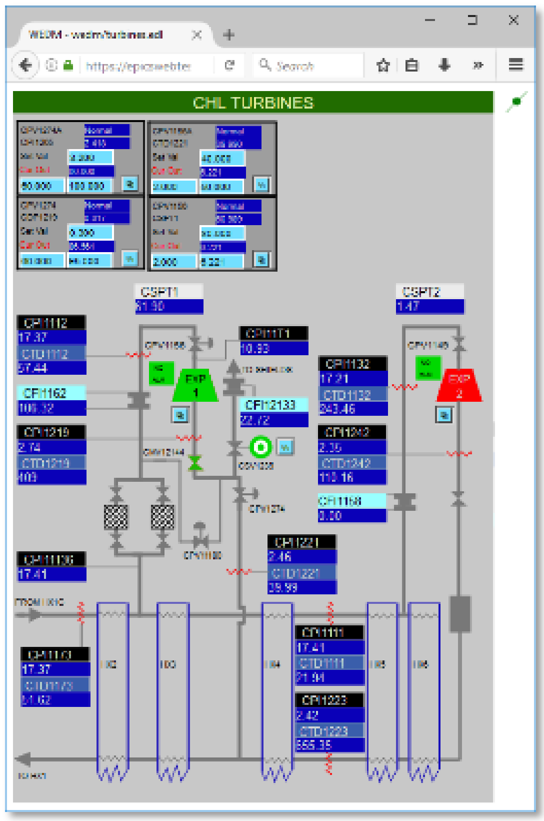


Figure 1: An EDM screen rendered in WEDM.

Reduce Reliance on EPICS Kiosks

A number of EPICS "kiosks" exist at Jefferson Lab. These include a dedicated xterm that allows security guards to monitor special alarm screens at times of year when the control room is not staffed, as well as a number of xterms on roll-around carts in service buildings and tunnels. It is envisioned that over time the number of these

*Authored by Jefferson Science Associates, LLC under US DOE Contract № DE-AC05-06OR23177 kiosks will be reduced. The guards will be able to watch their screens using a standard PC web browser, while techs will make more frequent use of portable tablets when working in service areas.

PROCESS OVERVIEW

Server Request Processing

The WEDM server provides a web page for users to browse a file system for an *edl* file and a web service to generate an HTML view of a selected screen file. The server code is written in Java and the initial file translation is done by creating an object representation for each EDM widget. These intermediate objects are provided the traits found in the *edl* file and generate the HTML / SVG sent to clients.

Client Rendering and CA Updates

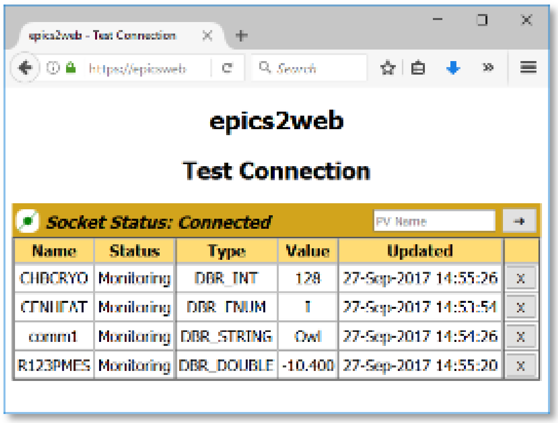
The WEDM server's response to the client web browser contains all necessary HTML to render a screen and setup CA monitors. Data attributes embedded in the HTML for each widget provides the information necessary for CA monitoring. After the web page completes loading in the browser JavaScript is used to scan the document for all widgets and create an object for each widget which requires PV monitoring. These objects are called PvObservers and are responsible for handling PV updates such as from alarm state and value changes. The PvObserver objects are specialized using prototypal inheritance and are tailored for each widget type.

For the WEDM screen drawing implementation, we chose to use HTML and SVG elements thereby preserving the ability to style via Cascading Style Sheet rules and interact with and respond to events on elements via the JavaScript event model. The HTML 5 canvas is also capable of rendering the EDM screen, but requires a custom layout and event framework. In order to present a screen in WEDM, each EDM object is converted to either a generic HTML *div* element or an embedded SVG element and positioned in an absolute layout. An HTML *div* is used for widgets with text to leverage the excellent text handling in HTML. Shapes are modeled using SVG as it provides a natural fit.

Monitoring of EPICS CA updates is done using the epics2web gateway. A Web Socket channel is established between the client web browser and the epics2web service and a monitor is requested for each PV needed by the PvObserver objects. A lookup map is created in the browser, linking a PV with all PvObservers that are interested. Therefore each time a CA update is delivered from the epics2web service for a PV, the map can be consulted to determine which PvObservers to notify of the update.

WEB GATEWAY

The epics2web gateway is a companion web service application created to provide a client JavaScript API for querying EPICS Channel Access over a Web Socket and a server backend to handle the requests. The server itself is implemented in Java and uses the EPICS collaboration's CAJ Java interface to CA [1]. The web service proxies requests to a standard EPICS CA Gateway. Internally the service uses JavaScript Object Notation (JSON) to encode messages between the client and server making the service simple to troubleshoot (Figure 2).



epics2web

Test Connection

Socket Status: Connected

Name	Status	Type	Value	Updated	
CHDCRYO	Monitoring	DBR_INT	128	27-Sep-2017 14:55:26	x
CTNIIFAT	Monitoring	DBR_ENUM	1	27-Sep-2017 14:53:54	x
comm1	Monitoring	DBR_STRING	Owl	27-Sep-2017 14:54:26	x
R123PMES	Monitoring	DBR_DOUBLE	-10.400	27-Sep-2017 14:55:20	x

Figure 2: Test page for epics2web.

EXTENSIBILITY

True to its name WEDM strives to be extensible. The project page can be found on github.com and the source code can be easily forked [2]. To add a new widget, one must create a new Java class which implements the WEDMWidget interface and handles parsing the *edl* and generating HTML. A configuration file named wedm.properties contains a map from Java class name to EDM object name and must be updated to reflect the new widget. On the client, a new instance of the PvObserver object should be defined and placed in a new JavaScript file named after the EDM object. The entire application does not need to be recompiled after a widget is added, but it does need to be repacked to include the new files. Packaging is done via an Ant build file. A new JavaScript file placed in the widget directory will be automatically concatenated and minified into the distributable package by the build script.

PERFORMANCE

Parsing an *edl* file and translating it to HTML every time a screen is requested can be time consuming for large screens. A caching layer has been implemented and screens are stored in memory as they are created and returned on later requests. On each screen request, the last modified date of the *edl* file is consulted to determine if the screen has been modified and if so, the cached version is cleared and the screen is regenerated.

General use at JLab has shown that WEDM has good performance. Page load times are less than a second for most screens. Most *edl* screens can be parsed and converted to HTML screens server side in a few milliseconds and are transferred to clients encoded with gzip to minimize network I/O. One of the largest screens at JLab, the RF Captain screen, contains 4,426 EDM objects, creates 1,546 PV monitors, and has a dimension of 1200px x 1015px, and results in an uncompressed HTML file 1.73 MB in size. It takes about 2 seconds to generate the screen server side on our hardware, but since screen generation is cached this cost only happens the first time the screen is requested. Total screen load time for WEDM on a desktop computer is comparable to native EDM.

IMPLEMENTATION DETAILS

Fonts

Unlike the consistent desktop environment used by operators in the control room, remote viewing of EDM screens in a web browser results in a much more diverse set of client computers including mobile devices, and therefore a much more diverse set of client fonts. Web browsers do a good job of substituting missing fonts, but often the rendered text bounds varies on differing platforms. To overcome this, WEDM dynamically resizes all fonts on the client web browser at runtime to fit within their bounding box.

Mouse Events

The EDM mouse event model differs from the model in web browsers. In EDM, mouse events are propagated to all elements in a stack of elements at a given point. In JavaScript mouse event handling depends on how the element is positioned and only the top most absolutely positioned element receives mouse events. To deal with this in WEDM the top most element event handlers must search for elements underneath and manually propagate mouse events to these other elements. EDM screen designers at JLab like to stack buttons on each other so that what appears like clicking one button is actually clicking multiple. Also, stacking Related Displays with the "invisible" attribute is a common way to allow clicking on a shape to result in a new screen or screen menu.

Invisible Elements

There are many ways to make elements invisible in HTML / SVG. In EDM invisible elements must respond to mouse events. This means we cannot use style rules such as "display: none" or "visibility: hidden". Instead we must set the border, background, and foreground color to fully transparent.

Stacking Order

In EDM and HTML / SVG the order in which an element appears in the document from top to bottom determines the stack order. However, EDM will move an element to the top of the stack each time it redraws an element, ignoring stack order. This includes, drawing the outline around a control widget such as a Related Display or button element on mouse hover or each time a PV update results in a new background, border, or other graphical change. This always draw on top behavior, remains an outstanding difference between WEDM and EDM. In practice, this discrepancy has not been a problem on the screens at JLab for anything other than hover outlines so a simple partial solution is used to handle this one case: on hover the ":before" Cascading Style Sheet style selector is used to insert a pseudo *div* inside the widget that is then positioned via z-index above all elements in the same group, even if they are defined earlier in the document.

CALC Expressions

The ability to execute expressions can be accomplished in JavaScript using the "eval" statement, but the expression syntax differs between EDM and JavaScript. To overcome this, EDM CALC expressions are transformed via character substitution to JavaScript friendly expressions.

Colors / Color Rules

Colors in EDM are specified as an index into a color palette stored in a color file (colors.list). Each EDM color is defined as either a static RGB or as a color rule (expression). Since the RGB colors on the web have a depth of 256 and EDM has a depth of 56K (or optionally 256) a conversion generally must take place. Since colors are dynamically evaluated and assigned, the color file is translated to a set of JavaScript objects so that the script running on the client browser can quickly access the colors.

Color rules are handled similar to CALC expressions in that character substitution is used to prepare the expression for evaluation using JavaScript "eval". However, the color rule operands are implicit and the format is of a conditional statement. In order for the rules to be evaluated explicit operands are inserted and the expression is converted to a JavaScript switch statement.

LOC Variables

During web page initialization when JavaScript objects are created for each WEDM widget LOC (Local) variables are handled specially to ensure they are not sent to the epics2web server requesting EPICS monitoring, but instead are added to a JavaScript collection of known LOC variables. Mapping of LOC variables to widgets is handled just like with EPICS PVs so that when a LOC variable is updated (perhaps via button press) then all widgets interested can be notified. LOC variables are stored with their "LOC\\" prefix to allow mixing them in the same collections and maps containing EPICS PVs, which always have their "EPICS\\" prefix stripped off if originally present.

Macros

HTTP URL parameters are used for EDM macros (a.k.a. symbols / screen parameters). Macros are passed via URL in the form "$(<key>)=<value>", which is the key format that is used in *edl* files to store macros (dollar sign prefix and parenthesis around key) thereby allowing the search and replace to be executed without further string manipulation required.

JLAB INTEGRATION

At JLab, we use a desktop directory service named JMenu to organize and search for screens, applications, and documentation. Browsing for screens by file name and file system path or creating a menu link structure on a master EDM screen are not always the most efficient ways to find screens. To make usage of the WEDM user-friendly, a front-end that mimics the desktop screen launcher has been provided. We created a mobile-friendly web application named WMenu to extend JLab's directory service to WEDM (Figure 3).

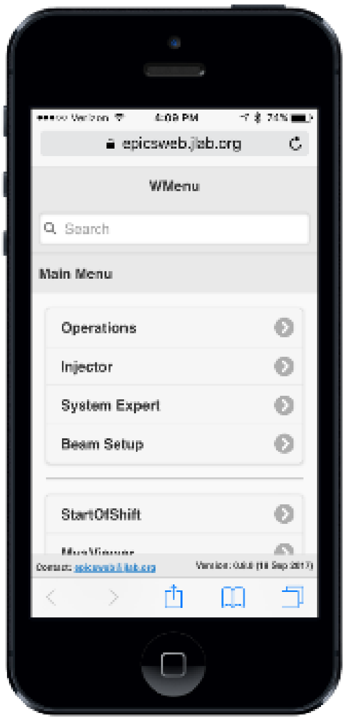


Figure 3: JLab's web-based screen menu.

JLAB SECURITY

To ensure that WEDM does not become a back-door allowing remote modifications to the control system we rely on three layers of defense: (1), All users must authenticate to the web server with a username and password; (2) the application itself is designed to be read-only and does not directly expose any means to update values; (3) network access controls force WEDM to access the control system through a read-only EPICS CA gateway.

RELATED WORK

WebOPI

This project does for Control System Studio (CSS) screens what WEDM does for EDM: OPI files created from CSS BOY can be used as is on the web. A key difference is that WebOPI forwards all click events to the server for processing, generating a lot of network traffic and putting most processing logic on the server. Widget updates are communicated to clients as all new layout instructions. The server also uses long polling instead of Web Sockets for notifications [3].

WebPDA

Created in response to the performance limitations of WebOPI, this toolkit leverages Web Sockets to expose EPICS PVs to the web. A small set of custom widgets are included and a JavaScript API is provided with authentication at the application layer. A limitation of WebPDA is that one must create new screens from scratch using JavaScript and HTML.

Diirt / WebPODS / pyPODS

Originally starting out as a Java CA wrapper library named pvmanager and integrated into CSS, this project is now a more generalized library and includes a protocol for accessing CA over Web Sockets. The WebPODS web service and its Python cousin pyPODS are similar to epics2web, but epics2web only exposes EPICS CA whereas WebPODS provides a much more generic protocol [4].

Others

There are many other EPICS on the web projects. Projects which rely on a Java Applet include CAML / WebCA [5] and JDM [6]. Projects which use Web Sockets include web-epics [7] and NodeCA [8]. Projects which are capable of parsing either an *edl* or *adl* file include WebMEDM [9], EMF [10], and JDM.

FUTURE WORK

There are many interesting opportunities for expanding WEDM. Several widgets are yet to be supported such as the X-Y graph widget and the analog needle meter widget. Further, widget versioning is not implemented and it would allow better interoperability with legacy screens if added. Finally, it would be interesting to allow modifications of the control system via WEDM, but would require careful consideration and use of security mechanisms.

CONCLUSION

The WEDM application provides an easy to deploy out-of-the-box solution for viewing existing EDM screens remotely. It also allows technicians to use the already familiar drag and drop EDM screen creation tool to create new mobile friendly screens.

REFERENCES

[1] M. Šekoranja, R. Šabjan, "Channel Access Java (CAJ)", presented at EPICS Collaboration Meeting, Menlo Park, CA, USA, April 2005.
[2] WEDM, `https://github.com/JeffersonLab/wedm`
[3] X. Chen and K. Kasemir, "Bringing Control System User Interfaces to the web", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, October 2013, paper THCOAAB03.

[4] C. Carcassi, K. Shroff, "WebPODS: Accessing Control Data Through Web Standards (WebSockets, JSON, HTML, CSS)", presented at ICALEPCS'15, Melbourne, Australia, October 2015, paper THHC3O02, unpublished.
[5] T. Pelaia, M. Boyes, "Introducing CAML II", in *Proc. ICALEPCS'09*, Kobe, Japan, October 2009, paper FRA001.
[6] JDM, `https://www.jlab.org/cdev/java/jdm/UserDoc/`
[7] web-epics, `https://github.com/AustralianSynchrotron/web-epics`
[8] A. Uchiyama, K. Furukawa, Y. Higurashi, "EPICS Channel Access Using WebSocket", in *Proc. PCaPAC'12*, Kolkata, India, December 2012, paper WECC02.
[9] A. Bertrand, R. Krempaska, "EPICS on the WEB", in *Proc. ICALEPCS'05*, Geneva, Switzerland, October 2005, paper PO2.087-5.
[10] R. Farnsworth, C. Myers, A. Starritt, "The Flange for Controls System to Internet Applications", in *Proc. ICALEPCS'09*, Kobe, Japan, October 2009, paper THP106, unpublished.