INTRODUCING CAML II

Thomas Pelaia II, ORNL★, Oak Ridge, TN 37831 U.S.A.
Matthew Boyes, SLAC✳, Menlo Park, CA 94025 U.S.A.

Abstract

Channel Access Markup Language (CAML) is a XML based markup language and implementation for displaying EPICS channel access controls within a web browser. The CAML II project expanded upon the work of CAML I adding more features and greater integration with other web technologies. The most dramatic new feature introduced in CAML II is the introduction of a namespace so CAML controls can be embedded within XHTML documents. A repetition template with macro substitution allows for rapid coding of arbitrary XHTML repetitions. Enhancements have been made to several controls including more powerful plotting options. Advanced formatting options were introduced for text controls. Virtual process variables allow for custom calculations. An EDL to CAML translator eases the transition from EDM screens to CAML pages.

INTRODUCTION

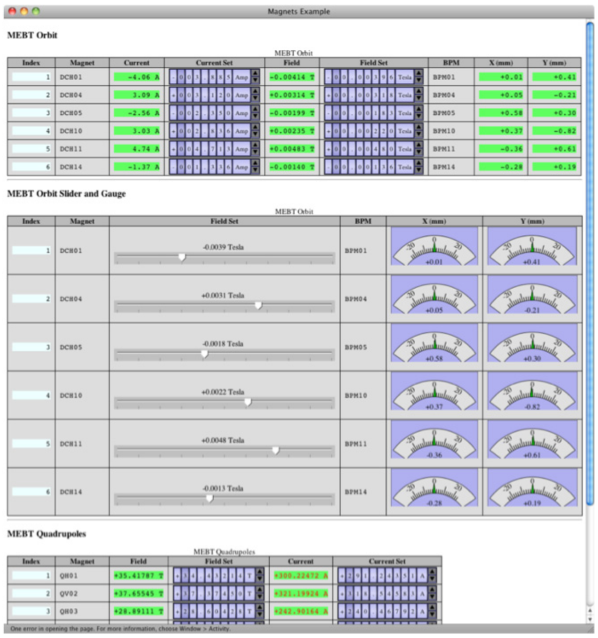
Channel Access Markup Language (CAML) [1] is a declarative XML based language and implementation for displaying EPICS [2] channel access controls within a web browser. CAML uses the WebCA [3] plugin to handle the channel access communication. Initially, CAML was a standalone pure XML specification which included both layout and controls. This approach made it difficult to integrate HTML content and it also limited the layout to the few options that were implemented directly in CAML.

The most significant feature for CAML II is support for XHTML which assigns CAML elements a namespace and allows them to be mixed with other XHTML constructs. Other new features include new and enhanced controls, advanced formatting options, virtual process variable support and an EDL [4] to CAML translator.

CAML ARCHITECTURE

CAML consists of a XML specification and namespace, stylesheet transform, controls library and the WebCA plugin. A CAML document is written in XHTML and allows for any combination of HTML, CAML and other XML code. The CAML namespace allows CAML elements to be uniquely identified through the "caml:" prefix. This allows CAML controls to be embedded within an HTML layout.

When a web page containing CAML is loaded, the browser's XSL transformer parses the document according to the specified CAML XSLT which identifies CAML elements, parses their attributes and substitutes each element in place with corresponding HTML, CSS, SVG and JavaScript to render the control and wire it for channel access events. The document also loads the WebCA plugin which handles the native channel access communication. The page is then presented the user within the web browser as shown, for example, in Figure 1.



MEBT Orbit

MEBT Orbit

Index	Magnet	Current	Current Set	Field	Field Set	BPM	X (mm)	Y (mm)
1	DCH01	-4.06 A	-003.885 Amp	-0.00414 T	-00.00396 Tesla	BPM01	+0.01	+0.41
2	DCH04	3.09 A	+003.120 Amp	+0.00314 T	+00.00318 Tesla	BPM04	+0.05	-0.21
3	DCH05	-2.56 A	-002.350 Amp	-0.00199 T	-00.00183 Tesla	BPM05	+0.58	+0.30
4	DCH10	3.03 A	+002.836 Amp	+0.00235 T	+00.00220 Tesla	BPM10	+0.37	-0.82
5	DCH11	4.74 A	+004.713 Amp	+0.00483 T	+00.00480 Tesla	BPM11	-0.36	+0.61
6	DCH14	-1.37 A	-001.336 Amp	-0.00140 T	-00.00136 Tesla	BPM14	-0.28	+0.19

MEBT Orbit Slider and Gauge

MEBT Orbit

Index	Magnet	Field Set	BPM	X (mm)	Y (mm)
1	DCH01	-0.0039 Tesla	BPM01	+0.01	+0.41
2	DCH04	+0.0031 Tesla	BPM04	+0.05	-0.21
3	DCH05	-0.0018 Tesla	BPM05	+0.58	+0.30
4	DCH10	+0.0022 Tesla	BPM10	+0.37	-0.82
5	DCH11	+0.0048 Tesla	BPM11	-0.36	+0.61
6	DCH14	-0.0013 Tesla	BPM14	-0.28	+0.19

MEBT Quadrupoles

MEBT Quadrupoles

Index	Magnet	Field	Field Set	Current	Current Set
1	QH01	+35.41787 T	+34.43214 T	+300.22472 A	+291.24351 A
2	QV02	+37.65545 T	+37.37450 T	+321.19924 A	+318.54583 A
3	QH03	+28.89111 T	+28.60428 T	+242.90164 A	+240.46792 A

One error in opening the page. For more information, choose Window > Activity.

Figure 1: Sample page with CAML controls.

All required web technologies are covered by open specifications and supported by all major standards compliant web browsers including Firefox, Safari, Opera and Chrome. Specifically, we have tested CAML on Safari and Firefox. One or more of these targeted web browsers is available on each of the major operating systems (Linux, Mac OS X, Windows).

Motivation

While many options exist for building channel access applications using high level languages, it is desirable to offer a modern declarative language for rendering controls that is also platform independent. XML is a natural choice for building a modern declarative language due to its wide adoption and availability of supporting tools. Web browsers make a natural choice as a display manager due to their wide adoption, support of rich media and adherence to cross platform standards. Particularly, the HTML 5 WHATWG [5] specification has empowered web browsers with new capabilities such as the canvas element for 2D graphics, built in client side database and more. Recently, JavaScript engines have gained

★ORNL/SNS is managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725.
*This work was supported in part by the Department of Energy contract DE-AC02-76SF00515.

significant performance improvements [6, 7] and CSS now offers very rich styles and layout control.

Clearly web browser vendors are committed to advancing capabilities and improving performance. Given the powerful platform independent capabilities of modern web browsers, the ability to render XHTML documents and their wide availability, this makes the web browser an ideal platform for rendering controls using a XML specification such as CAML as the declarative language.

CAML ELEMENTS

CAML defines several elements that correspond to channel access controls, and other specialized elements. Each element supports attributes for configuration (e.g. the associated process variable for reading or writing). A default style sheet allows for a common look and feel to be applied to CAML elements across all web pages. For example, you can specify a text style such as background or foreground color to indicate alarm status.

Channel Access Controls

Table 1 shows the currently supported channel access controls grouped by type. Text, incremental and gauge controls allow for custom formatting of the string representation of associated events including value, status, severity and/or time stamp. For controls that support writing, a visual cue indicates whether write access is granted for the specified process variable and the control is accordingly enabled or disabled.

Table 1: CAML Controls

Type	**Controls**
Enumerated	Menu, Radio, Toggle Button, Bit/LED
Incremental	Slider, Wheel Switch
Text	Text Entry, Text Update
Plotting	2D Scatter, Line, Strip, Bar, Waterfall, Intensity
Meter	Gauge

Every control has a contextual menu accessible through a right click on the control. The contextual menu has options to inspect the associated process variable attributes, copy the process variable address, copy the current value and record and display the recent history of the process variable. Unfortunately, the contextual menu can obscure the contextual menu which many browsers already have and the contextual menu doesn't provide the flexibility that is needed for many elements (e.g. those with more than one process variable). A better alternative is to replace the contextual menu with a control inspector accessible through a small "i" button that appears only when the mouse hovers over the control.

Several plot types are supported, but generally plot performance is poor and the plot options are limited as most plots were not designed specifically for CAML or rapid update. Our own work with plotting indicates that we can obtain high performance plots if we code the plots ourselves. Also, we are confident we can make the plots more feature rich without sacrificing performance.

Virtual Process Variables

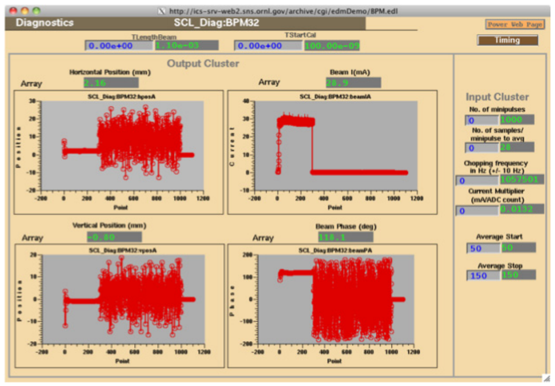
CAML supports the definition of virtual process variables that exist within the context of the current web page. A virtual process variable can execute arbitrary JavaScript that can operate on any number of real process variable values (at the time of execution) without having to make any direct channel access calls. A virtual process variable can be used with any CAML control that takes a process variable as a property.

Repetition

Frequently, one wishes to display a repeated block of code such as a table row or list item in which some data varies adhering to a common pattern. CAML provides a solution for this scenario through repetition with substitution. CAML defines a repetition element within which one includes a template element and a list element. The template element can contain arbitrary XHTML with references to named macros. The list element contains an item element for each record of repetition. The item element can define one or more named macro value pairs. For each item in the repetition list, the template is evaluated with the associated named macros substituted with its corresponding value in the current item.

EDL TO CAML TRANSLATOR

CAML ships with an EDL to CAML Translator to translate EDM pages to CAML. The translator is written in Java; however, we would like to port this to run as a script to avoid the overhead associated with Java. The translator does a good job of translating most controls but has limitations. For example, the translator doesn't translate background colors or images. Figure 2 shows an actual EDM screen used at SNS, and Figure 3 shows this screen translated into a CAML page.



TLengthBeam 0.00e+00 1.10e-03
TStartCal 0.00e+00 100.00e-09
Horizontal Position (mm) Array 2.16
Beam I(mA) Array 28.9
Vertical Position (mm) Array -0.80
Beam Phase (deg) Array 118.1
No. of minipulses 0 1000
No. of samples/ minipulse to avg 0 28
Chopping frequency in Hz (+/- 10 Hz) 0 1057501
Current Multiplier (mA/ADC count) 0 0.0132
Average Start 50 50
Average Stop 150 150

Figure 2: Original EDM screen.

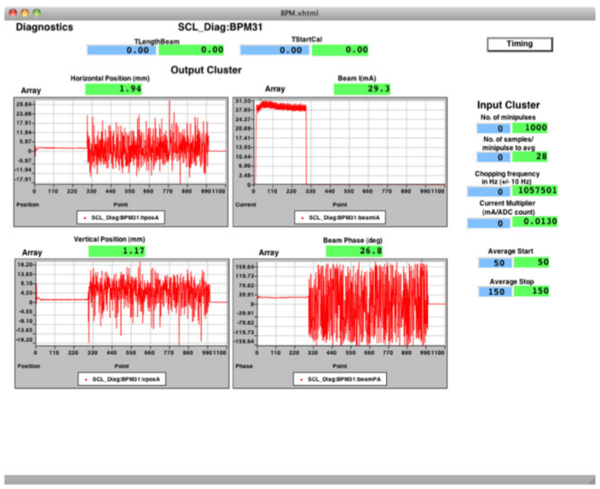


Figure 3: CAML page translated from EDM.

EDM supports absolute layout of elements, so the translator also attempts to generate CAML code with absolute layout. However, since CAML uses HTML for element layout and HTML allows for (and more typically uses) dynamic layout, it is advisable to modify the resulting CAML code for a more natural and consistent user experience on the browser platform.

WEB CA PLUGIN

Web CA is a browser plugin that provides a JavaScript API for several channel access client functions and makes calls through the native channel access client libraries. When writing CAML pages, one does not typically interact directly with the plugin. Rather, calls to the plugin are generated automatically by the stylesheet transformation. Common channel access functions on process variables which are supported are monitor with callback, get with callback, put with callback and fetching information such as access rights.

Obtaining and configuring this plugin is often the biggest obstacle to adopting CAML. A channel access installer for Mac OS X 10.5 and later includes the plugin with support for both 32 and 64 bit web browsers. A Web CA plugin installer for Windows installs the CA Repeater and the plugin for Firefox and Safari and configures the channel access environment.

We would like to improve the plugin to avoid configuration. The author of hosted web pages should know the channel access configuration information, so we would like to modify the plugin to support channel access environment configuration within the web page. Furthermore, we must address potential security issues in which a remote website may host pages which load the plugin and make channel access calls. Also, the plugin is not the best option for all users, and in particular office users would like to access CAML pages without having to install and configure the plugin. To address this requirement, we would like to develop a web service which provides channel access readonly access as a fallback when the plugin has not been installed.

CURRENT SIGNIFICANT EFFORTS

Recently, SNS and LCLS have begun working together to advance the CAML project. As mentioned previously, we would like to improve plotting performance and add more plotting options. We intend to develop a CAML element inspector to replace the current contextual menu. We must address security concerns with the Web CA plugin. Also, we are working to provide a zero configuration alternative to the plugin for office users.

OBTAINING CAML

Please visit our CAML website, http://www.ornl.gov/~t6p/Main/CAML.html, to learn more about CAML, subscribe to the newsfeed and download CAML and the Web CA plugin.

ACKNOWLEDGEMENTS

The authors are grateful for the work of Matej Sekoranja and his Cosylab team who developed and implemented both CAML and the WebCA plugin under contract for the Controls Group at the Spallation Neutron Source in Oak Ridge. Also we would like to thank Vidya Kumar and David Purcell for their contributions to CAML and WebCA.

REFERENCES

[1] Thomas Pelaia II, "CAML and Web CA Status", EPICS Collaboration Meeting, Legnaro, Italy, 2008; http://agenda.infn.it/getFile.py/access?contribId=10&amp;resId=0&amp;materialId=paper&amp;confId=715.
[2] http://www.aps.anl.gov/epics/
[3] Thomas Pelaia II, "EPICS With Cocoa", EPICS Collaboration Meeting, Argonne, IL, 2006; Slide 14 of http://www.aps.anl.gov/epics/meetings/2006-06/Infrastructure/EPICS_with_Cocoa.pdf
[4] http://ics-web.sns.ornl.gov/edm/index.php
[5] http://www.whatwg.org/
[6] http://webkit.org/blog/214/introducing-squirrelfish-extreme/
[7] http://arstechnica.com/open-source/news/2008/08/firefox-to-get-massive-javascript-performance-boost.ars