KICK-OFF A NEW PROJECT LIKE A PRO:

How Schweitzer Engineering Laboratories Capitalized on Ergonomic Engineering with zenon

When a new automation project gets kicked off, there are many important decisions that must be made. Often people jump right into the SCADA development without first understanding the main design and development challenges that are ahead of them in the next several weeks or months. This is a critical time period in which one seemingly harmless decision can be the determining factor on whether a project is delivered on time, meets a budget and is within specification or whether it will be late, sloppy and prone to errors.

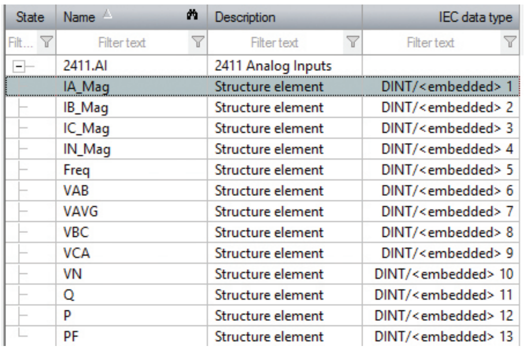
Let's take a recent example in which COPA-DATA collaborated with Schweitzer Engineering Laboratories (SEL) to deliver a large substation automation solution.

THE CHALLENGE

Our main task was to work with the SEL engineers in the design phase of their SCADA projects. After some discussions on communication protocols and drivers, we took a detailed look at the zenon variable or tag creation process. In zenon, there are quite a few ways to create process variables or tags.

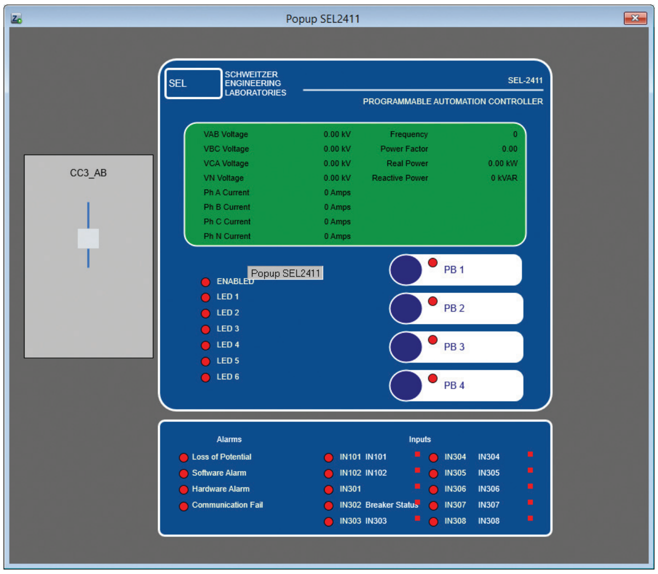
On the one hand, variables can be created manually, one by one. But this method is often prone to human error and can be quite time consuming. Therefore, we ruled this option out quickly. For some specific drivers, zenon offers the ability to import the variables via Online or Offline modes. For smaller projects, this is a gigantic time saver. The driver Online/Offline import could theoretically import all tags for one specific device in less than ten minutes. However, this option was also ruled out because the devices were already in the field, operational, and a few continents away from us. The other disadvantage here is that this import would result in a flat, unstructured list of variables. Another option we considered was a CSV import. zenon also offers a CSV import option to allow for the creation or modification of variables based on a CSV definition file. This is a common approach, and it is a nice way to make bulk changes very quickly. However, due to the size of

IU Published in
INFORMATION UNLIMITED
THE COPA-DATA MAGAZINE
No. 29, April 2016
© Ing. Punzenberger COPA-DATA GmbH
www.copadata.com/iu



State	Name	Description	IEC data type
Filt...	Filter text	Filter text	Filter text
⊟	2411.AI	2411 Analog Inputs	
	IA_Mag	Structure element	DINT/<embedded> 1
	IB_Mag	Structure element	DINT/<embedded> 2
	IC_Mag	Structure element	DINT/<embedded> 3
	IN_Mag	Structure element	DINT/<embedded> 4
	Freq	Structure element	DINT/<embedded> 5
	VAB	Structure element	DINT/<embedded> 6
	VAVG	Structure element	DINT/<embedded> 7
	VBC	Structure element	DINT/<embedded> 8
	VCA	Structure element	DINT/<embedded> 9
	VN	Structure element	DINT/<embedded> 10
	Q	Structure element	DINT/<embedded> 11
	P	Structure element	DINT/<embedded> 12
	PF	Structure element	DINT/<embedded> 13

Figure 1:
Setup of structured datatypes for the SEL-2411 Programmable Automation Controller.



Popup SEL2411

CC3_AB

SEL SCHWEITZER ENGINEERING LABORATORIES SEL-2411
PROGRAMMABLE AUTOMATION CONTROLLER

VAB Voltage	0.00 kV	Frequency	0
VBC Voltage	0.00 kV	Power Factor	0.00
VCA Voltage	0.00 kV	Real Power	0.00 kW
VN Voltage	0.00 kV	Reactive Power	0 kVAR
Ph A Current	0 Amps		
Ph B Current	0 Amps		
Ph C Current	0 Amps		
Ph N Current	0 Amps		

Figure 2:
An SEL-2411 popup screen, showing live data from the device.

this project and the fact that several engineers would be simultaneously working on the zenon project, it was foreseeable that each engineer would use slightly different methods during the CSV export, modification, and import which could potentially yield inconsistent results.

THE SOLUTION

Finally, we came to our solution – zenon datatypes: an object-oriented method of creating variables which ensures consistency and supports inheritance. In the electrical system of this specific substation automation application, there are hundreds of instances from about a dozen different SEL IED types. By using structured variables in zenon, it was possible for us to create one datatype for each device type. The initial time invested in setting up the datatype with appropriate objects and properties is quite small compared to the time savings and organization gained as a result of its use.

For example, we set up structured datatypes for the SEL-2411 Programmable Automation Controller as shown in *Figure 1*. Since every SEL-2411 is configured with identical DNP Maps, we were able to even go so far as to set at the datatype level all of the alarm and event conditions, identification labels, control properties, as well as the DNP addresses.

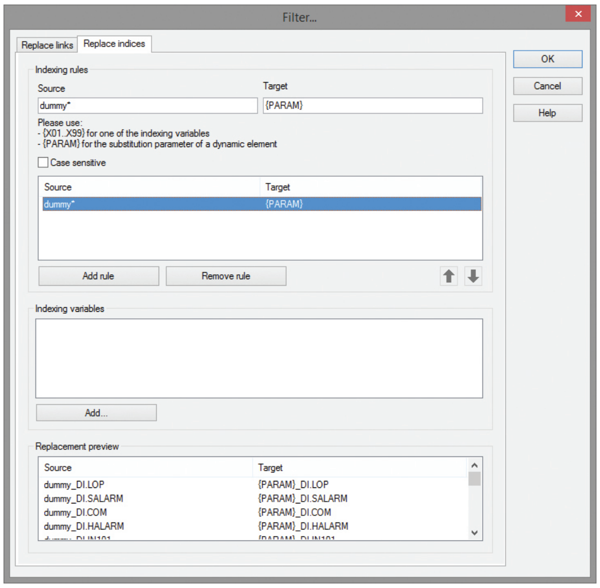
Once the structure is created, it can still be enhanced or modified later, even if variables (instances) were already created based upon it. For example; to add an alarm to the existing structure element 24.11AI.PF at the datatype level, all instances of this type will inherit this modification automatically.

BUT THIS WAS JUST THE BEGINNING ...

While leveraging the benefits of zenon's datatypes, we pushed the object-oriented concept a step further to include the topic of popup screens. In this system, one faceplate or popup screen for each device instance was planned. If a user clicks on the SEL-2411 from the oneline screen, an SEL-2411 popup screen will be displayed (see *Figure 2*), showing live data from that specific device. This is a common task in an HMI/SCADA system, but our goal was to accomplish this in the easiest, fastest and the most reliable way possible.

To start, we created a single screen in zenon for the SEL-2411. We set the frame to have a border so that it could be dragged around in the runtime, and set the frame so that it could be opened multiple times (e.g. to compare two separate SEL-2411 devices side by side). Then, by using symbols and native zenon screen elements, we created a 1:1 replica of the device. As a pre-condition, we had also created a set of "dummy" internal variables in zenon to act as an engineering side placeholder on the screen itself.

To visualize and represent the 100 different instances of the SEL-2411, we knew that we could use a single screen together with the zenon screen substitution, thus essentially replacing the variable linking behind the screen objects. This has been possible with zenon for many years now. However, up until zenon 7.11, in order to represent the 100 different instances of the SEL-2411 by using standard functionality, it would have been necessary to create 100 different screen switch functions, one for each instance.



Filter...

Replace links | Replace indices

Indexing rules

Source: dummy*
Target: {PARAM}

Please use:
- {X01..X99} for one of the indexing variables
- {PARAM} for the substitution parameter of a dynamic element

☐ Case sensitive

Source	Target
dummy*	{PARAM}

Add rule | Remove rule

Indexing variables

Add...

Replacement preview

Source	Target
dummy_DI.LOP	{PARAM}_DI.LOP
dummy_DI.SALARM	{PARAM}_DI.SALARM
dummy_DI.COM	{PARAM}_DI.COM
dummy_DI.HALARM	{PARAM}_DI.HALARM

OK | Cancel | Help

Figure 3:
Based on the replacement rule – Source: dummy* / Target {PARAM} – in the screen switch function and depending on the button the user clicks, the correct variables are displayed in runtime mode.

SOMETIMES, IT CAN BE THAT EASY

As we were using zenon 7.11 in this project, we were also able to take advantage of screen substitution with parameterization. What this eventually meant for us, using native functionalities, was that we could visualize the 100 different instances of the SEL-2411 device type by creating only one screen and only one screen switch function!

So how does this actually work, you may ask? Ok to start, based on our structure and following a naming convention, we created an instance variable called A1_CC1_AB_2411, and to represent our second device, a set called A2_CC1_AB_2411.

In the screen switch function, which opens the SEL-2411 screen, we visited the tab called "Replace indices", where we entered the following replacement rule (see *Figure 3*):

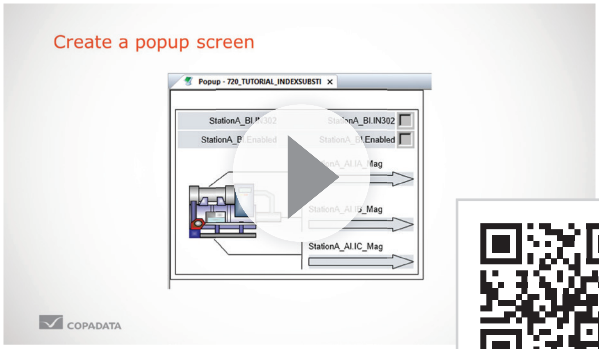
Source: dummy*
Target: {PARAM}.

The contents of {PARAM} are filled during runtime, and this parameter is read from the calling element. In our case, this was the button which resides on the oneline screen. In zenon 7.11 and newer that button is linked to our single SEL-2411 screen switch function, but it also has a text property included for the parameter for substitution. That is where we entered the unique instance name in our example, either A1_CC1_AB_2411 or A2_CC1_AB_2411. In runtime mode, the correct variables will be displayed depending on the button the user clicks.

ERGONOMICS WITH ZENON

This is just a small, but effective, example of how smart decisions made early on in the HMI/SCADA project design and use of supporting software can pay off many times over.

Video:
You too can save time and money by using the ergonomic engineering functions of zenon as described in this case study.
Scan & Play!



http://kaywa.me/mZRO1

LOUIS PAGLIACCETTI,
TECHNICAL CONSULTANT