INTEGRATING WEB-BASED USER INTERFACE WITHIN CERN'S INDUSTRIAL CONTROL SYSTEM INFRASTRUCTURE

A. Voitier, M. Gonzalez-Berges, P. Golonka, CERN, Geneva, Switzerland

Abstract

For decades the user interfaces of industrial control systems have been primarily based on native clients. However, the current IT trend is to have everything on the web. This can indeed bring some advantages such as easy deployment of applications, extending HMIs with turnkey web technologies, and apply to supervision interfaces the interaction model used on the web. However, this also brings its share of challenges: security management, ability to spread the load and scale out to many web clients, etc... In this paper, the architecture of the system that was devised at CERN to decouple the production WINCC-OA based supervision systems from the web frontend and the associated security implications are presented together with the transition strategy from legacy panels to full web pages using a stepwise replacement of widgets (e.g. visualization widgets) by their JavaScript counterpart. This evolution results in the on-going deployment of web-based supervision interfaces proposed to the operators as an alternative for comparison purposes.

INTRODUCTION

Supervisions for industrial control systems are applications that have to remains stable and shielded from any potential factor of disruption. As such, they are usually deployed on dedicated and isolated servers to minimize risks and control its access.

However, users of these systems have a pressing demand of accessing, or at list viewing, the state of their systems without the protective hurdles. Another limitation is the need to install specific software on the computer used for accessing remotely the supervision applications.

These requirements are following the tendency of the present computing era where we got accustomed to seamlessly access any remote services through a simple web-browser or a mobile application. A so-called Web UI (Web User Interface) for the SCADA tool used at CERN is implemented to answer these requests, provided security is strictly enforced.

This UI allows to view existing panels and synoptic used in the current native UI. The graphical engineering editor of the SCADA tool is also accessible with a web browser, suppressing the need to install any specific software. Another important advantage is to profit from standard and modern web technologies to enhance the usability of user interfaces as a wide range of versatile graphical widgets are available to web-enabled technologies [1].

For the end users it brings new paradigms of using supervision applications. It allows accessing them from a field intervention, or from a non-professional place for rapid remote diagnostics. Yet another usage is to broadcast the state of a particular machine to hundreds or thousands of users of this machine, which is a need for physics experiment collaborations.

This paper will present how this project went through several phases of iterative researches. First by prototyping a Web UI with classic web components. Then, facing important limitations, introducing an alternative approach similar to a remote desktop software embedded in a browser.

WEB ACCESS FOR INDUSTRIAL CONTROL SYSTEMS

At CERN, supervisions for industrial control applications amount to a total of 600 systems. All production applications are hosted on a dedicated network isolated from the intranet and internet by firewalls. To enable a web access, the following four main requirements have been postulated.

Security: Making control systems of very costly machines accessible to the entire world means security is paramount. A powerful access control has to be used while keeping the balance with usability. Indeed, with a very secure but hard to use system users have a tendency to circumvent the secure measures, putting the protected systems at risk. Another point is to stick to web security standards extensively reviewed and tested by experts and not forge custom solutions.

Legacy: In the past 15 years at CERN a lot of scripts and panels have been developed for the SCADA tool. This created an important base of legacy work that cannot be deprecated. The Web UI has to be compatible with existing developments. Yet, at the same time it should remain flexible and open to improve the existing panels with user experience paradigms the web can offer.

Scalability: The new Web UI infrastructure should support up to a thousand simultaneous clients in read-only mode. This is for the use case of broadcasting a machine state to the many users interested in these information. One related requirement is that the new UI system should minimize any extra load on the production data servers running SCADA systems. At best, a decoupling and/or buffering should exist between a SCADA system and its numerous clients.

Integration: Industrial control systems are about integrating off-the-shelves components. A Web UI for it should have the same philosophy of reusing readily available web servers and services. In particular, it should integrate with web-hosting services centrally supported within an organisation running industrial control systems. As such, it emphasises the need to use compatible web standards and be platform independent. In a similar way, it

ISBN 978-3-95450-148-9

should also integrate naturally on the client side. Users should not have to install any special plugin. The services should only make use of standard web technologies. Today those are HTML in its 5th version for structuring data, Javascript for scripting, and CSS for styling.

Four use cases have been initially intended:

1. One user running one panel. In that case, not only the user can see the information the panel display, such as process values. It should also let the user click on buttons, interacts with other widgets and open new panels. Such display can be used on mobile devices for field interventions or remote debugging (while an on-call person is away from a control room for instance).
2. Many users can view one panel. This is intended for broadcasting information in read-only mode. Therefore, the actions of one user should not affect the control system and also what the others can see. In that case, interactions are limited to zoom in and out of trends, and to navigate between different broadcasted panels.
3. Editing panels and scripts. This amount to replicate or enable the original graphical engineer editor of the SCADA tool to run in a browser.
4. Shift away from the classical SCADA paradigm of showing panels (merely the content of a framed window) to rather display web pages (where the content is flowing horizontally and/or vertically). In that case, usual web page generation methods apply. The control system only need to provide ways to get, set and subscribe to data, as well as call certain functions remotely.

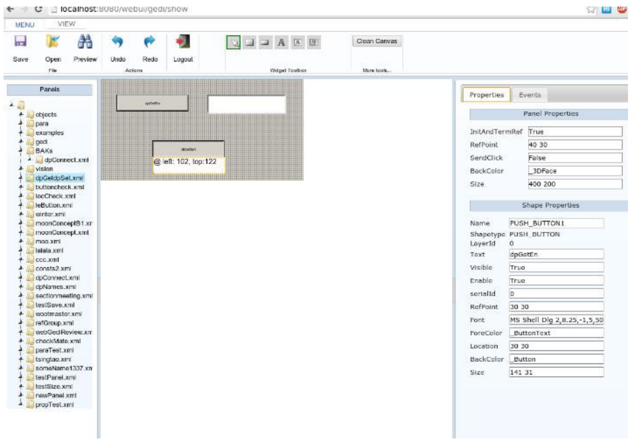
FULLY WEB BASED SOLUTION

The first approach chosen was to replicate all UI functionalities using conventional dynamic web page generation. This meant two important capabilities: Firstly, being able to read the original description format of the SCADA tool panels. Conveniently, these can be exported to XML. Secondly, re-implementing all the editing features of the graphical engineering tool.

Two major limitations arose while implementing prototypes for this solution. The first was that panels contain code. This code is written in a proprietary language for the SCADA tool. This makes it difficult to execute it on client-side as the only commonly accepted mean of scripting in a browser is Javascript. The company of the tool attempted to write a translator. However, it resulted in a solution implying modification to the original codes. A more exotic approach was considered, compiling the code to LLVM [2] bytecode, and executing it in the browser through the use of the software package Emscripten [3]. The initial test has shown that it would require a lot of developments for the compiler. Panels can also contains code that is not purely doing user interface logic. Indeed, sometimes these panels contain business logic code that should be executed on server-side. Solving this problem would require a major refactoring of the existing code base.

The second major limitation was that interpreting the XML files to draw graphic primitives of the panel is a heavy process. There is a large combination of graphical features, themselves dependent on the way it is rendered today by the native Qt-based UI. This made it difficult to keep a compatibility to the pixels, a feature needed by the synoptic drawings. Similarly, the engineering tool would also had to be replicated in the same way. As this tool is quite complex and feature full, it made the task almost impossible to write it entirely in another technology (Fig. 1).

Figure 2 describes the architecture used. This full-web replication approach was dismissed based on all these limitations.



Panel Properties

InitAndTermRef	True
RefPoint	40 30
SendClick	False
BackColor	_3DFace
Size	400 200

Shape Properties

Name	PUSH_BUTTON1
Shapetype	PUSH_BUTTON
LayerId	0
Text	dpGetEn
Visible	True
Enable	True
serialId	0
RefPoint	30 30
Font	MS Shell Dlg 2,8.25,-1,5,50
ForeColor	_ButtonText
Location	30 30
BackColor	_Button
Size	141 31

Figure 1: Screenshot of the engineering tool replicated in a prototype of the full web solution.

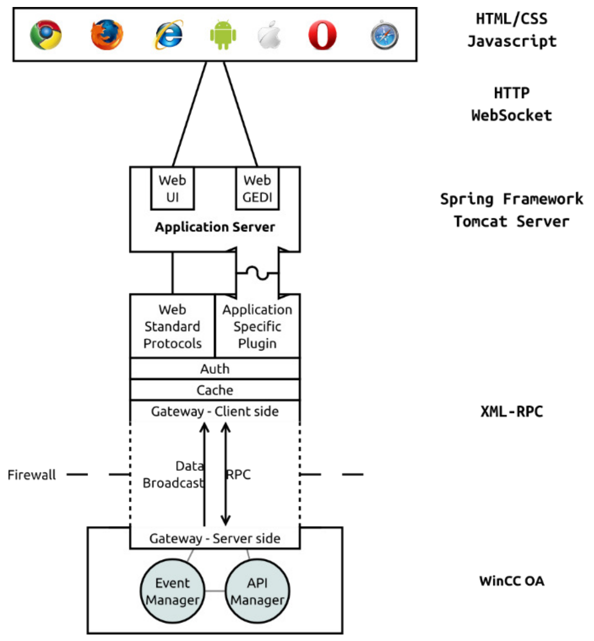


Figure 2: Architecture of the full web solution.

ISBN 978-3-95450-148-9

INTERMEDIATE SOLUTION

As a full web solution is not yet possible, the company providing the SCADA tool proposed a more pragmatic solution. It consists of rendering native panels on the server-side, and send its representation to the browser through a special protocol (Fig. 3). The browser reconstruct the panel representation using HTML5 canvas. Initially, the solution was thought to send the graphical primitives as binary commands driving the Javascript drawing engine. In the end, a protocol similar to VNC, usually used for remote desktop applications, was chosen instead. A Javascript client-side program takes care of implementing the protocol and draws on the canvas.

This solution presents the advantages of keeping full compatibility with the legacy panels and to run the panel code directly on the server. Several performance evaluations were carried out until representative test panels could be displayed over bandwidth-restricted mediums such as copper landlines and mobile phone networks.

However, although this approach is meeting most of the requirement, when many users view the same panel, the interactions of one user are visible to all users. This prevent the inclusion of scrollable trends and navigation buttons. To work around this issue, Javascript widgets can be overlaid on top of the VNC canvas. To implement this functionality, a hook was provided to the panel's developer by reusing a previous CERN project integrating Javascript widgets and code into native panels [1]. In such case, instead of being rendered on the server-side, the Javascript insertion is sent to the client browser. All interactions with these client-side widgets are limited to a single user. Nevertheless, in case of widgets making use of process data (e.g. trends) it is needed to disassociate the data transfer from the VNC protocol within the same websocket.

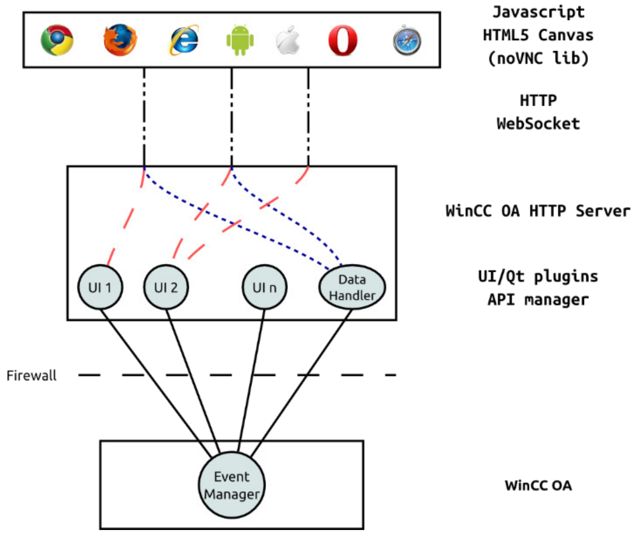


Figure 3: Architecture of the VNC-based solution. Red dashed lines are the communication of rendering primitives. Blue dotted lines are for the process data. Dashed and dotted lines are a mix of both types going through a single websocket.

SECURE IMPLEMENTATION

In the present state-of-the-art, two-factor authentication is often considered a secure and usable measure. It is understood by users that the additional minor hurdles are a necessary evil to benefit from the additional level of protection. Therefore, its usage is warranted to authenticate users accessing the Web UI. CERN IT service provides a single sign-on (SSO) implementation (based on Shibboleth) reusable by all web services across the organisation. This service provides two-factor authentication based on the user password plus several second factors: SmartCard, Yubikey, SMS OTP, and Google Authenticator (Fig. 4). The user's password is actually never shared with the Web UI application front-end. Once a user is identified, a central LDAP service provides the information about which groups the user belongs to. This is used to define the roles and allowed actions within the supervision application [4].

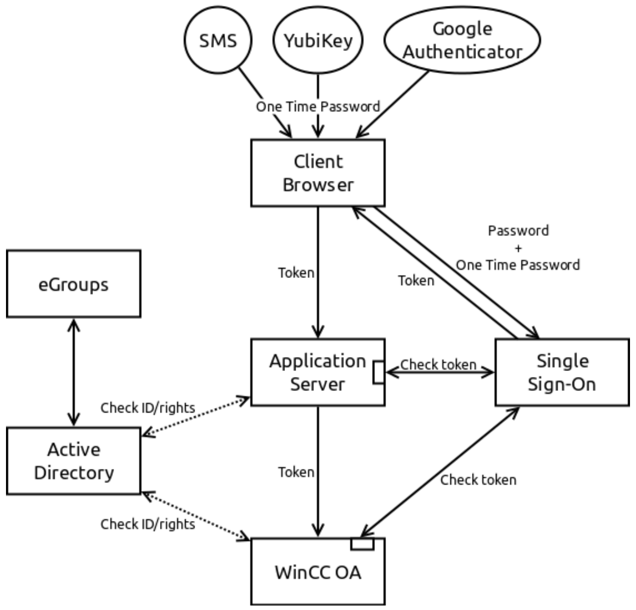


Figure 4: Single sign-on implementation for control system Web UIs.

CURRENT DEPLOYMENT

The VNC solution is expected to enter in production at CERN by the end of 2015. To accommodate the various requirements, the web server provided by the SCADA tool will not be the one facing incoming requests. Instead, the requests will first arrive into an application server managing the initial SSO negotiation, spawning the UI processes doing the rendering and connecting these processes to the original web clients through a websocket proxy. This implementation is thought to be scalable by spawning the UI processes on a farm of virtual machines managed by CERN's Openstack service. These machines will be part of a set of computer used to define firewall rules such that the production SCADA servers are exposed only to these machines. In turn, these machines will never be reached by public traffic as all their communications

ISBN 978-3-95450-148-9

with clients will transit through the application server proxy. Penetration tests organized by CERN's security team will also be performed to validate the overall solution.

CONCLUSION

Enabling web access to industrial control system is an expected feature by many CERN users. Control systems being critical pieces of the installations it is important to ensure a proper implementation. Considerations about security, legacy developments, scalability and integration are fundamental in this project.

Through various researches, prototype implementations, and benchmarking, a suitable solution has been found respecting all requirements. Hence, users of industrial control systems at CERN will soon be able to access their supervision applications through a web browser without using dedicated software. A proper security level will be obtained by reusing proven and standard solutions. The proposed architecture will allow us to scale with a growing amount of users while introducing more security layers. Finally, the reuse of web application servers provided as a central service by the organisation will help to keep the maintenance and administration tasks to a minimum.

This solution is the first step, in the long term, we will look for an implementation closer to traditional web developments that will allow us to use any major and mature web framework.

ACKNOWLEDGMENT

A particular thanks to M. Koller, C. Stoegerer, F. Glege, M. Janulis, S. Lueders, S. Lopienski, and S. Petrovski for their helpful contribution to the presented work.

REFERENCES

[1] P. Golonka et al, "FwWebViewPlus: integration of web technologies into WinCC-OA based Human-Machine interfaces at CERN", CHEP 2013, Amsterdam, Netherland.

[2] The LLVM Compiler Infrastructure website: http://llvm.org/

[3] Emscripten: An LLVM-to-Javascript Compiler project page: https://github.com/kripken/emscripten

[4] P. Golonka et al, "Integrated Access Control for PVSS-based SCADA Systems at CERN", ICALEPCS 2009, Kobe, Japan.