ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOBPP02

DESIGNING A CONTROL SYSTEM FOR LARGE EXPERIMENTAL DEVICES USING WEB TECHNOLOGY

W. Zheng, Y. Wang, M. Zhang, F. Wu, N. Fu, S. Li
International Joint Research Laboratory of Magnetic Confinement Fusion and Plasma Physics, State Key Laboratory of Advanced Electromagnetic Engineering and Technology, School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan, China

Abstract

EPICS is mature in accelerator community. However, there are efforts to improve existing control system software like Tango and EPICS 7 mainly driven by the needs of flexibility of the control system and the development of computer technology. This paper presents a new way of building a large experimental device control system using web technology instead of EPICS toolkit. The goal is to improve the interoperability of the control system allowing different component in the control system to talk to each other effortlessly. An abstraction of the control system is made. The control system components are abstracted into resources. The accessing of the resources is done via standard HTTP RESTful web API. human machine interface is based on HTML and JavaScript in browsers. Web Socket is used for event distribution. The main feature of this design is that all interfaces in the system are based on open web standards, which are interoperable among almost all kinds of devices. The paper also presents a software toolkit to build this kind of control system. A control system for a diagnostic on J-TEXT tokamak built using this toolkit will be presented.

INTRODUCTIOIN

One key characteristic of as large experimental facility control system is its ability to adopt and integrate new systems. As experiment advances new diagnostics or detectors will be added to the machine. On J-TEXT currently there are more than one system been added of modified to the machine. Different types of subsystem are likely to be implemented with different technologies. New technologies are being integrated into those facilities frequently. Efficiently integrated those new technologies into the control system is very important. As the performance of the computer and network keep advancing performance of the supervisory control and data acquisition (SCADA) system is no long the focus of the development. Instead, interoperability became the priority of SCADA system. To achieve interoperability a common language for a control system is needed. There are various control system framework existing in big physics community, each with different characteristics. For fusion community, ITER chose the Experimental physics and industrial control system (EPICS) and Channel Access protocol (CA) as the common language. EPICS has been the common language to the accelerator control system for decades [1]. Now chosen by ITER, it is used by many tokamaks as well [2-5]. It is mature and well supported by the community. But the technologies used in tokamaks are different from those in accelerators. It is not a straight forward job to create EPICS support for equipment used in fusion experiment. EPICS CA was originally designed for performance not interoperability recently there are activities to improve the interoperability of EPICS [6]. Later emerged control system frameworks such as Tango uses object-oriented technique to improve interoperability and flexibility [7-9]. But still, it is hard to have all the equipment in a control system supporting the control system framework that you chose. Is it even possible to develop a control system protocol that everyone supports or it is necessary? There is a technology that is almost supported by all the devices, that's web. Countless web APIs have been published and consumed by all kinds of devices. If a control system is built on web, it could be supported by everyone effortlessly. This work was inspired by web technologies. We proposed an abstract model for control system and a framework that uses web technology to build a control system. It mainly addresses the interoperability issue of very large control systems in large experimental facilities. The proposed framework is based on simple and open web standard which has been used by the web industry for years. Therefore, anyone can implement a system that can be integrated into this control system with tools already available.

This paper first briefly talked about the web technologies and in section 3 we proposed the abstraction of the control system. Based on that abstraction the web technologies are introduced to make control system protocols. Then in section 4 software framework to implement the web based control system is described. Last an application example is presented.

WEB TECHNOLOGIES

Web technologies is an important part of our internet life. We keep using it every day. Web technologies not only power the web site. Today from mobile apps, online games, to smart sensor and IoT application, web plays a big role in them. Many devices have embedded web servers, and many client apps is running in browsers. They communicate using HTTP. So, what is web technologies exactly? There are different interpretations of web technologies. What is common is web is based on HTTP, HTML, and browser.

HTTP is an application protocol on top of TCP. It is HTTP to be specific HTTP/1.1 is a text-based request and response protocol. A client would send a request to a web server and get response. The request and response are fully in text. There are quite a lot of overhead here, but it boosts interoperability as text provide more redundant and easier interpreted information both for man and machines. HTML is a mark language. It describes the structure of a Web page and consists of a series of elements telling the browser how to display the content. The web browser is a software that fetches content using HTTP and renders the HTML content. Modern browsers also run programs like JavaScript script to help user interacting with the content and the server.

You can see that the web technologies and standards are designed for interoperability [10]. These standards are widely used by all kinds of devices and platforms. That is why bringing web technologies to control system may improve the interoperability.

A COMMON MODEL FOR CONTROL SYSTEM

To build an interoperable control system we have to make a common model of the control system. Everyone in this system can be abstracted into this model and understand each other conceptually. This is the basis of interoperability. This The control system discussed here is mainly a SCADA system.

This common model should define the fundamental concepts of a control system with enough abstractions, so it can be adopted by all components in control system without difficulty. For a control system to work, the basic activities are getting information from other systems, knowing their status, and sending commands to make other system to behave as desired. Thus, a control system can be defined as: all the activities in the control system as "accessing resources". There are 5 types of resources: thing, status, configuration, method and event, in which status, configuration, and method are the core resource.

Resource

The control system model can be illustrated in Fig. 1. It is generalized as accessing resources between different control system components.

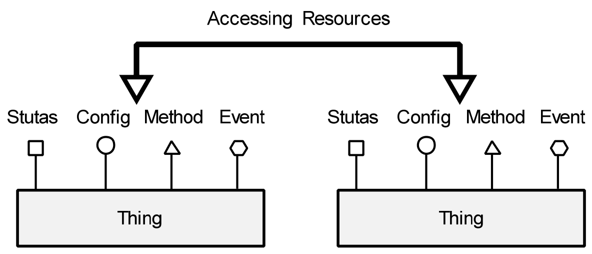


Figure 1: The control system model. Everything in a control system is a resource, and every activity is accessing resources.

Thing: A thing is a control system component. It can have children resources such as status, config, method and event which can be accessed by other things. A thing can be either physical or logical, like the controller for a power supply, or a data archiver software running on some controller.

Status: A status is a property of a thing that others things can observe but cannot change. It can only be changed by the thing itself. Statuses of a thing indicates its current state.

Configuration: A Configuration is a property of a thing that it self cannot change but can only be changed by other things. It represents the desired behavior set by other things. Note that one cannot determine the thing's behavior by its configuration, instead that should be determined by reading the status.

Method: A methods is a property of a thing. A method is like a command, a thing should react to invoking of its methods immediately.

Event: Event is a property that others can subscribe to, and get notified on a certain condition. It always resulted in an invoke of the subscriber's method. It has to be associated with another resource, which is the source of the event.

Access of Resources

The model also abstracts how to access a resource. First there are actions: get, set, invoke, subscribe/unsubscribe. The status only support get action, since it is read-only. The configuration support get and set actions, method only support invoke, subscribe/unsubscribe is for event. Besides actions there are inputs, which is some data passed to the resource when accessing it. This model uses a Uniform Resource Identifier (URI) to locate a resource. In this model, a URI can explicitly locate a control system resource. Lastly in this model, we have sample. Sample is not a resource. It is an immutable object. It is the result of accessing a resource and only valid for this access. Resource can put various meta information in the sample like timestamp, unit, etc. If you are familiar with web technologies, it starts to look like web already. But that is pure coincidence, the control system model is completed independent from any implementations or communication protocol.

HTTP RESTFUL AS COMMUNICATION PROTOCOL

As stated above, accessing resources is the only activities in a control system. Thus, a communication protocol that enables the control system components to access resources on other components is the key. This protocol will be the common language of the control system, it will be supported by every component of the control system. Instead of inventing our own protocol, we use the already massively used standard: the HTTP protocol and the RESTful (Representational State Transfer) practice. HTTP now has become the most common language for the internet. Countless web APIs are published using HTTP with RESTful practice. With a few lines of documentation or even with no documentations but just some poking around, one can start to consume a RESTful API. This makes the interoperability of RESTful API extremely high.

Most web APIs out there do not full comply with RESTful definition. But there are a few characteristics that is followed by almost everyone. That is to locate a resource using URI, presents intended action as HTTP verbs. To be used in the above control system model, the control system resources are located with URL, URL is a legit URI. The actions are mapped to HTTP verbs, the following Table 1 shows the mapping. The sample of an access to the resource is returned as the HTTP. Another thing in RESTful practice is the request and response are presented in platform independent plain text to be specific in JSON format, which can be parsed by any platform. It's even human readable. This makes parsing the data very easy and with some poking there is basically not compatibility issue. Another RESTful practice is Hypermedia. Hypermedia in RESTful APIs responses provide information on how to interact with this resource. This is essential for building autonomous self-organized control system. Hypermedia can be implemented by putting extra information in the sample.

Table 1: Control System Resource Access Action Map to HTTP Verbs Table

Resource Access Action	**HTTP Verb**
Get	GET
Set	Put
Invoke	Post
Subscribe	Post

In a control system, the controllers or a subsystem as long as they have control system resources like status, configuration, method and etc. will implement web servers. So, they can expose the control system as RESTful APIs. When they need to access control system resources on other controllers or systems, they use a HTTP client to consume those RESTful APIs hosted by other systems.

For events it always coexists with resources such as status, configuration and method. Subscribers can start a web socket on the same URI as the host resources. When an event a fired a web socket package are send to the subscriber. For those clients that do not have web socket easily available, they can use a call back method.

WEB PAGES AS HUMAN MACHINE INTERFACES

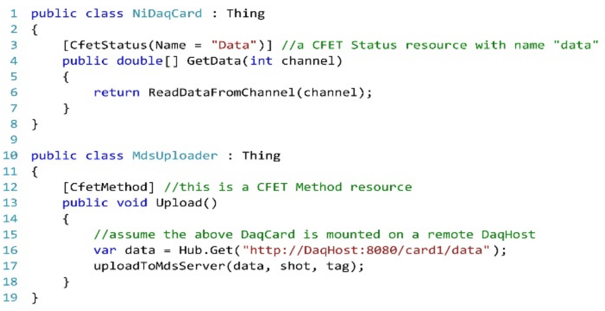
In traditional control systems, human machine interface (HMI) are standalone applications. They are developed separately and deployed on the operator's console. The HMI gets data from the network using protocols like EPICS CA, and renders it on the graphic interface. This means they are developed by tools for specific SCADA system and often runs on limited types of platforms. Another issue is when there are changes to the control system, the HMI need to be modified and need to be re-deployed onto every console. This could lead to inconsistent versioning between IOCs and HMIs.

Using web pages as HMI is a trend nowadays [11]. In web based control system, all HMI is a web site running on web servers. As aforementioned the controllers or other systems implemented web servers and expose control system resources as RESTful APIs. For HMIs those web servers not only host APIs but also web pages. The web pages become the HMIs. The HMI can even be generated automatically according to the resources on the controllers. When a request come from a browser, it uses the URI same as the one used by any RESTful API client, the server would know that and return a web page to visualize the resource instead of a JSON object. This automatically generated HMIs are great for testing and developing a control system, save lots of labor developing an HMI.

CONTROL SYSTEM FRAMEWORK FOR EXPERIMENTAL DEVICES TOOLKIT (CFET)

A toolkit like EPICS is needed to practically build control system applications. We developed the Control system Framework for Experimental Devices Toolkit (CFET). It includes applications and software libraries to make a control system that meet the above standards.

The CFET is implemented as .NET standard libraries which supports major Linux distros, Mac OS and Windows. The design aims to let the control system engineer to focus on the control functions, and ignore the web completely. The developers only need to code an object called thing, which implement the control function like interfacing to IO modules or control scripts. They only need to decorate some properties or method in the object with attribute to tell CFET toolkits they are control system resources as shown in Fig. 2. The CFET will expose them as RESTful APIs. Inherit from Thing is not even mandatory. By inherit from Thing, it will give this object to access CFET services like access other thing's resources subscribe and publish event.



```
public class NiDaqCard : Thing
{
    [CfetStatus(Name = "Data")] //a CFET Status resource with name "data"
    public double[] GetData(int channel)
    {
        return ReadDataFromChannel(channel);
    }
}

public class MdsUploader : Thing
{
    [CfetMethod] //this is a CFET Method resource
    public void Upload()
    {
        //assume the above DaqCard is mounted on a remote DaqHost
        var data = Hub.Get("http://DaqHost:8080/card1/data");
        uploadToMdsServer(data, shot, tag);
    }
}
```

Figure 2: The pseudocode illustrating how to make a thing. In the second method it gets data from a resource located possibly on a remote controller using the hub object.

The core of CFET design uses the mediator design pattern. All the thing access other's resources via the hub object, which hides all implementation of other things and data access details away. It fetches data from one thing and hands to the other no matter if they are on the same host or not. It is also in charges of event distribution. Hub is the mediator between all the things as shown in Fig. 3. RESTful API server and client is implemented as communication module. You can use other protocols like EPICS CA or you own protocol to implement the communication model as long as it follows the control system model proposed above.

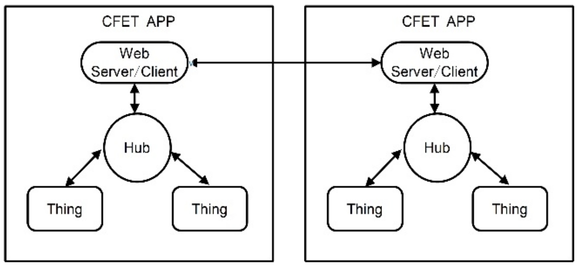


Figure 3: The block diagram of the CFET and the dataflow when accessing resources between components.

For web HMIs, there are 2 types of them. In older versions of CFET, they are server generated pages. But now the HMI is a stand along web applications that severed as static files. The all the logic and HMI generation is done on the client side. This has a benefit of being independent of the server serve as it is all static files. This make the HMI reusable amount different backend. The HMI generating flow chart is shown in Fig. 4. First the CFET web server find out the request is from a browser. It will redirect the browser to the HMI location with a hash fragment indicating the resource the user indent to access. The HMI is loaded into user's browser and the HMI will check if the fragment to see if it is a custom HMI, if so, it will load this custom HMI. If not, it will try to get the resource and use the resource to generate a suitable HMI. The HMI is composed with reusable components called widgets. User can use many widgets to build a custom HMI. It can be saved on local storage or on the server and loaded by the server when a user hits a specific resource.

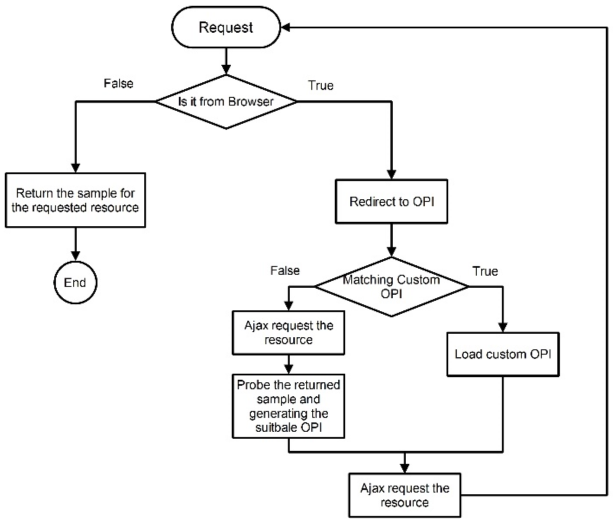

Request
False
Is it from Browser
True
Return the sample for the requested resource
End
Redirect to OPI
False
Matching Custom OPI
True
Ajax request the resource
Probe the returned sample and generating the suitbale OPI
Load custom OPI
Ajax request the resource

Figure 4: The flowchart shows the CFET returns different result based on the request when access the resources and generate suitable HMIs based on the resources.

APPLICATIONS

Currently CFET is at its early stage, and there are not many real-world applications of it. But We did test it on J-TEXT in real DAQ system. The ECEI diagnostic DAQ and data archive system is build using only CFET and nothing else. CFET will a DAQ card thing becomes a DAQ device. CFET with a HDF5 thing and a MongoDB manage thing runs on a server enables user to access the archived data and visualize the data in a browser. No matter if you are trying to get the state of the DAQ card or the waveform acquired by the DAQ card, it is always a RESTful API request but to the different URLs. You view those data using the same application, the browser. The browser run the same web page but it is populated with different widgets. The system block diagram is show in Fig. 5. Beside that, we also have implemented an OPC-UA thing that let others to access PLCs using RESTful APIs and provide a web HMI for PLCs. Later as more and more Things are developed, the CFET will become more useful.

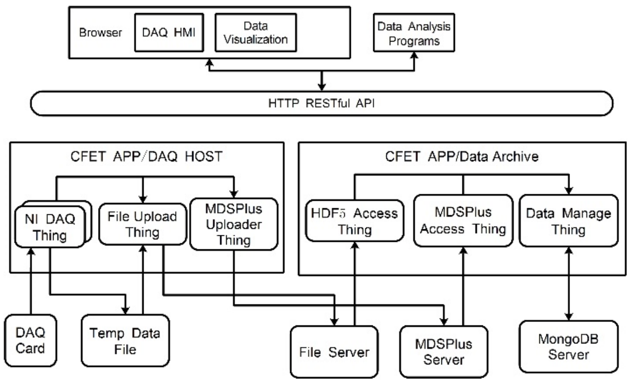


Figure 5: The block diagram of the ECEI DAQ system. The arrows are the data flows. The data flow inside and between CFET APP is handled by the CFET hub.

CONCLUSION

This work is aim for improve the interoperability of control systems. It does not invent ye another control system protocol, but uses web standards for building a control system. First this work proposed an abstract control system model. Control system are abstracted into access to different types of resources, such as status, configuration and method. On top of this abstraction, HTTP RESTful API are used to access all the resources. HTTP RESTful API may not be as efficient as EPICS CA or DDS. The biggest advantage is that is interoperable across many platforms. RESTful API is widely used and mature technology, this allowed a variety of technology and devices can be integrated into a control system seamlessly and effortlessly. A control system toolkit implementing the above design called CFET is introduced. Using it an ECEI diagnostic DAQ system are built.

The CFET is still at its early stage. Build a large experimental device with it is yet risky. However, as the experiment devices get larger and sophisticated, CFET could be an option for future devices.

ACKNOWLEDGMENTS

The authors wish to thank all the members in J-TEXT laboratory. This work is supported by National Natural Science Foundation of China (Nos. 11605068).

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOBPP02

REFERENCES

[1] EPICS, Experimental Physics and Industrial Control System, https://epics-controls.org/

[2] A. Wallander *et al.*, “ITER instrumentation and control—Status and plans”, *Fusion Eng. Des.*, vol. 85, nos. 3-4, pp. 529-534, 2010. doi:10.1016/j.fusengdes.2010.01.011

[3] K. H. Kim *et al.*, “The KSTAR integrated control system based on EPICS”, *Fusion Eng. Des.*, vol. 81, nos. 15-17, pp. 1829-1833, 2006. doi:10.1016/j.fusengdes.2006.04.026

[4] V. Vitale *et al.*, “FTU toroidal magnet power supply slow control using ITER CODAC Core System”, *Fusion Eng. Des.*, vol. 87, no. 12, pp. 2012-2015, 2012. doi:10.1016/j.fusengdes.2012.05.006

[5] W. Zheng, M. Zhang, J. Zhang, G. Zhuang, Y. He, and T. Ding, “The J-TEXT CODAC system design and implementation”, *Fusion Eng. Des.*, vol. 89, no. 5, pp. 600-603, 2014. doi:10.1016/j.fusengdes.2014.03.048

[6] L. R. Dalesio *et al.*, “EPICS 7 provides major enhancements to the EPICS toolkit”, in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 22-26. doi:10.18429/JACoW-ICALEPCS2017-MOBPL01

[7] A. Götz *et al.*, “The TANGO Controls Collaboration in 2015”, in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS' 15)*, Melbourne, Australia, 2015, pp. 585-588. doi:10.18429/JACoW-ICALEPCS2015-WEA3O01

[8] R. Bourtembourg *et al.*, "TANGO kernel development status," in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS' 17)*, Barcelona, Spain, 2017, pp. 27-33. doi:10.18429/JACoW-ICALEPCS2017-MOBPL02

[9] T. Matsumoto, Y. Hamada, and Y. Furukawa, “MADOCA II data collection framework for SPring-8”, in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17),* Barcelona, Spain, Oct. 2017, pp. 39-44. doi:10.18429/JACoW-ICALEPCS2017-MOBPL04

[10] Web standards - Wikipedia, https://en.wikipedia.org/wiki/Web_standards

[11] I. Sadeh, I. Oya, J. Schwarz, E. Pietriga, and D. Dežman, “The graphical user interface of the operator of the Cherenkov Telescope Array”, in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS' 17)*, Barcelona, Spain, 2017, pp. 186-191. doi:10.18429/JACoW-ICALEPCS2017-TUBPL06