GA–A26199

REMOTE CONTROL OF A FUSION FACILITY

by
D.P. SCHISSEL, G. ABLA, D.A. HUMPHREYS, B.G. PENAFLOR,
B.S. SAMMULI, and M.L. WALKER

FEBRUARY 2009

GENERAL ATOMICS

DISCLAIMER

GA–A26199

REMOTE CONTROL OF A FUSION FACILITY

by
D.P. SCHISSEL, G. ABLA, D.A. HUMPHREYS, B.G. PENAFLOR,
B.S. SAMMULI, and M.L. WALKER

This is a preprint of a paper to be presented at the Twenty-Fifth Symposium on Fusion Technology, September 15-19, 2008, in Rostock, Germany, and to be published in *Fusion Engineering and Design*.

Work supported by
the U.S. Department of Energy
under DE-FC02-01ER25455

GENERAL ATOMICS PROJECT 30106
FEBRUARY 2009

GENERAL ATOMICS

ABSTRACT

Magnetic fusion experiments keep growing in size and complexity resulting in a concurrent growth in collaboration between experimental sites and laboratories worldwide. This scientific collaboration activity is strong at existing experimental sites, is a major element of machines just coming on line, and is also a thrust of experiments that will come on line in the next decade. Computer science research into enhancing the ability to scientifically participate in a fusion experiment remotely has been growing in size in an attempt to better address the needs of fusion scientists worldwide. The natural progression of this research is to examine how to move from remote scientific participation to remote hardware control. This paper examines the challenges associated with remote experimental device control and proposes a solution based on a semantic approach that defines a gatekeeper software system that will be the only channel of interaction for incoming requests to the experimental site. The role of the gatekeeper is to validate the identification and access privilege of the requestor and to ensure the validity of the proposed request. The gatekeeper will be a modular system, transparent to end-users, and allow a high volume of activity.

1. INTRODUCTION AND VISION

The next generation of magnetic fusion experiments will be the largest and most expensive scientific instruments ever built for fusion research. Concurrent with their growth in size and complexity is the growth in collaborations between experimental sites and laboratories worldwide. The importance and cost of these devices requires that they operate securely at the highest possible level of scientific productivity. It is believed that for experiments as complex as those carried out in this field, scientific productivity is inextricably linked to the capability and usability of their data and computing systems. Thus, careful consideration must be given to choices for architecture and technologies when designing a system that is so crucial to the overall project's success. Most importantly, the systems must be designed to meet the needs of the hundreds of scientists and engineers who will use them.

It is our vision that the Gatekeeper software system will be the only channel of interaction for incoming requests from experimental sites as well as on-site generated requests. To ensure integrity of the device's systems, the operation of the gateway must be obvious and verifiable. At the same time, it should be transparent to end-users and allow a high volume of activity so as to not provide a work bottleneck.

The vision for the Gatekeeper is that it be a modular system that is simple in design and defined in a way that makes its implementation and operation transparent and obvious. Care must be taken to ensure that it is effective, reliable, reasonably simple, testable, verifiable, and that it robustly supports remote participation.

2. GATEKEEPER DESIGN

The decision making process of the Gatekeeper requires following several specific functionalities: (1) Verify the identity and access control permission of requestor (authentication and authorization). (2) Ensure the safe arrival of requests. (3) Validate the format and appropriateness of requests. Figure 1 represents the pipeline of Gatekeeper's functionalities.

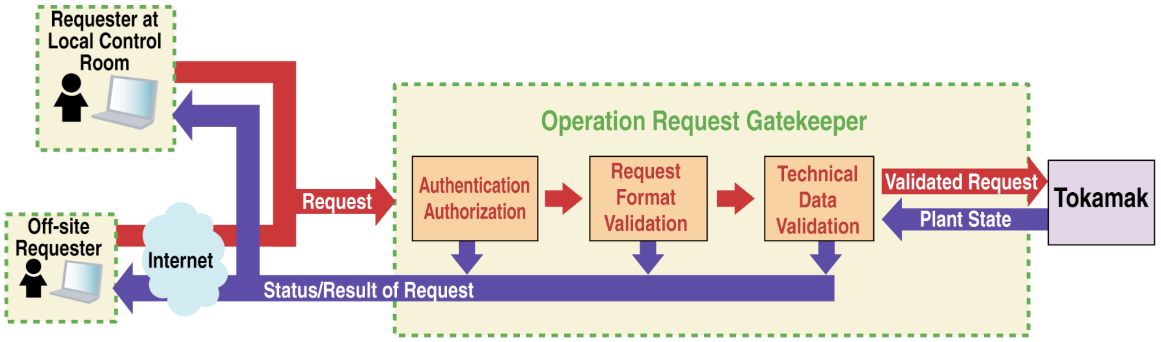
Requester at Local Control Room
Off-site Requester
Internet
Request
Operation Request Gatekeeper
Authentication Authorization
Request Format Validation
Technical Data Validation
Validated Request
Plant State
Tokamak
Status/Result of Request

Fig. 1. The overall functional design of the Gatekeeper system.

The creation of an efficient and reliable Gatekeeper relies on a system design that is standard, flexible and modular. Using standard technologies can guarantee that the Gatekeeper continues to be efficient during the lifetime of the fusion experimental device. Flexibility supports iterative development and helps the system to evolve as technology advances. Modularity allows any of its parts to be updated, if needed, without affecting other components.

Based on these principles, a prototype Gatekeeper has been designed and several technologies chosen to implement the necessary functionalities. Care was taken to make sure that the design was platform independent and that there is a clear interface between request clients, Gatekeeper functionality pipeline components and fusion device control hardware.

This paper describes the design issues and solutions for the components of the Gatekeeper. The front end of the Gatekeeper is designed as a web service. It mainly receives control requests coming from remote sites and provides a response. On the back end, it processes requests by utilizing a validation process pipeline. Based on the validation results, a decision will be made to accept or reject the request. If the request is rejected the client will be notified about the rejection along with appropriate details. Accepted requests will be passed onwards to the hardware control component of the fusion facility and the client will be notified with a status update. In some cases where the request involves numerous steps to complete, a client may receive multiple status updates. The request format uses XML format along with standardized XML schema [1]. The authentication and authorization process for security depends on X.509 technology along with a resource-based authorization scheme. The multiple components of technical content validation communicate and coordinate via a logic module dispatcher based-on a distributed computational framework.

3. REQUEST SECURITY

The Internet is an open system, where the identity of the communicating partners is not easy to ensure. Furthermore, the communication path traverses an indeterminate set of routing hosts and may include any number of eavesdropping and active interference possibilities. Thus, Internet communication is much like anonymous postcards, which are answered by anonymous recipients. The worldwide distribution of next generation fusion experiments creates a Virtual Organization that requires a secure solution to this communication dilemma. In today's distributed world there are actually a number of different ways to solve this problem. These solutions all provide the ability to authenticate the identity of the communicator, the ability to authorize if the communicator is allowed to make the specific request, and the ability to ensure the integrity of the communication so that it cannot be modified while in transit. Authentication is the process in a computerized transaction that gives assurance that a person or computer acting on a person's behalf is not an imposter. Authorization is the process of determining, by evaluating applicable access control information, whether a subject is allowed to have the specified types of access to particular resource. Encryption is a technique of preventing unauthorized access while the request is in transit.

The remote control vision outlined in this paper requires a secure communication solution yet the design is modular so that as security techniques evolve new solutions can be substituted for older ones. For the sake of elucidating the security concepts a specific solution is discussed. The implementation for establishing the identity of a communicator (authentication) and for determining whether an operation is consistent with agreed upon sharing rules (authorization) frames and ultimately defines the virtual organization.

3.1. AUTHENTICATION

Public-key cryptograph [2] is a solution for key distribution that scales to large groups. This system uses two different keys, one public and one private, where it is computationally hard to deduce the private key from the public key. Anyone with the public key can encrypt a message but not decrypt it and only the person with the private key can decrypt the message. Mathematically the process is based on the trap-door one-way function, which are relatively easy to compute but significantly harder to reverse unless the secret is known. That is, given x it is easy to compute $f(x)$, but given $f(x)$ it is hard to compute x. However, there is some secret information y, such that given $f(x)$ and y it is easy to compute x.

Public Key Infrastructure (PKI) is a technology to distribute and use asymmetrical keys. PKI gives trust that the public key being used truly belongs to the person or machine with whom/which they wish to communicate. Trust is established through the usage of certificate authorities (CAs) who issue X.509 certificates where a unique identity name and the public key of an entity are bound together through the digital signature of that CA (certificate = trust + public-key). Typically, a registration authority (RA) is responsible for the identification and authentication of certificate subscribers before the CA issues certificates.

3.2. AUTHORIZATION

Submitted requests need to be analyzed in order to make sure if an already authenticated user has the authorization to send control commands to a specific hardware described in the user's request. For example, a user may be only authorized to make changes to the spectrometer's wavelength settings, but not to the machine control pulse waveforms. In this case, even though a user is authenticated, it is not appropriate to accept their machine waveform change related requests since they are not authorized for this type control.

3.3. REQUEST INTEGRITY

The security system must be able to ensure that a submitted request has not been modified in transit. The ability of an unknown intruder to intercept and modify a control command is clearly not acceptable. Encryption of the request can prevent this from occurring and PKI, mentioned earlier, is a standard method for assuring message integrity.

4. THE CONTROL REQUEST

The control request is the information sent by the requestor to the Gatekeeper in order to issue commands to hardware or make changes to physics parameters. Requests can be simple, such as raising the gain on a diagnostic, or can be complicated such as configuring all aspects of the plasma control system. Therefore, the type and size of the data submitted with the request can vary accordingly.

The Gatekeeper is the front gate that interacts with users. It is the only channel of interaction for incoming requests from experimental sites as well as on-site generated requests. It should be general enough to receive and process a variety of different types of requests regardless of client hardware or software language.

4.1. REQUEST FORMATTING

The need for processing, maintaining, and organizing incoming requests in an efficient and timely manner requires a standard format for the Gatekeeper request description. The standard should be flexible enough to describe all present requests that represent the full variety of control commands and parameters that can be adjusted. It also must be based on an extensible pattern suitable for describing control requests to future hardware equipments or software tools that were not in existence when the Gatekeeper is initially deployed.

The request format standard should be independent from any specific hardware and programming language. Regardless of client software design and hardware features, the Gatekeeper should be able to receive, analyze and process the request. This is only way to provide a Gatekeeper which can evolve as technology progresses during the lifetime of the fusion experiment.

There have been many efforts to standardize the format of fusion experimental data within the community. There are mainly four types of approaches to describe the data: flat files (ASCII or binary), relational databases, application-driven binary databases and XML schema-based file format. Among them, the XML schema is the preferred format for the Gatekeeper. The reasons are: (1) XML can be read by both humans and computers running on a variety of operating systems. Therefore, using XML format can guarantee that requests are transparent on both ends: request submitter and the Gatekeeper itself. (2) There is large support in both the commercial and open source world for XML. Numerous software tools and a rich programming infrastructure exist to edit, validate and process XML data. This trend is accelerating and new computer and network technologies will continue to be built around XML. (3) XML schema supports introducing rules on how data is presented so that a variety of data structures can be easily described and standardized.

The requests that are submitted to the Gatekeeper need to be transparent to both the requester and the Gatekeeper. This means that the requester must always be aware of what format and what kind data are acceptable for the Gatekeeper, and the Gatekeeper must always understand the format, meaning and data type of the requests. This requires a standardization process of format, description tags and grammar for Gatekeeper requests. XML schema supports this need and supports the description and imposing of structure within an XML file. By using XML schema, the request tags, syntaxes, data types, and structure of each request element can be clearly standardized. New request tags can be easily added to the standard in case it is required by future hardware.

While XML is a preferred format to describe the Gatekeeper requests, some requests may contain large amount of data, and transferring it on wide area network is not very efficient. XML syntax is verbose and somewhat redundant, and this disadvantage becomes more obvious when it carries megabytes of data over the Internet. Therefore, the XML schema standard for Gatekeeper request also needs to include description (metadata markups and binary mime types) for binary data sources that are stored in a well-known data formats, such as MDSplus or NetCDF.

5. VALIDATION

Requests that have passed onwards by the security modules are sent onwards only after passing through two steps of validation: the request grammar validation and the technical content validation.

5.1. REQUEST GRAMMAR VALIDATION

The Gatekeeper needs to validate the format of a request before sending it onward for technical validation. Format validation is two-fold. First, the request is verified against regular XML format rules to make sure it is a well-formed XML message. This is straightforward process. Second, the content of the XML document is verified against the Gatekeeper Request Schema rules. All XML content must conform to restraints expressed in Gatekeeper schema, such as element names, attributes, and allowable hierarchies. Metadata markups and mime types are also checked at this step. If the request passes the grammar validation it will be sent to technical content validation. If not, a rejection message along with the appropriate level of detail will be returned.

5.2. REQUEST TECHNICAL CONTENT

Upon successfully verification of the request format, the validity of technical content of the request is then examined. It is possible that the validity checking of all possible requests could be implemented within the Gatekeeper. However, it is felt that this would lead to a very complicated design that would be unacceptable on numerous levels. Instead, the Gatekeeper's prime responsibility is to ensure that the requests are properly formulated and then sent to the appropriate logic module for verification. Valid requests will then be bundled with experiment and state information and sent onward to the logic modules (Fig. 2). The design of the Gatekeeper is such that it can handle any request to the plant systems yet its initial implementation may be for only a subset of all possibilities.

The logic modules are discreet entities that are responsible for determining if a request is valid. These modules are considered part of the Gatekeeper yet they can be authored and submitted (after validation) by any team member. In other words, logic modules are not created by a central autonomous organization but are instead authored within an organization that has the required expertise. The Gatekeeper will define the logic module interface allowing easy adoption of new modules. As a simple example, the request to raise the gain on a diagnostic would be passed to a logic module that would understand the physical limitations on gain and the state of the experiment (if personnel access has been granted to diagnostic areas maybe electrical setting changes are disabled). A more complicated example is a new plasma pulse schedule that will need to be modeled and verified by a rather complex logic module. Approved requests are then transmitted onward for execution either automatically or via a man-in-the-loop for enhanced safety.

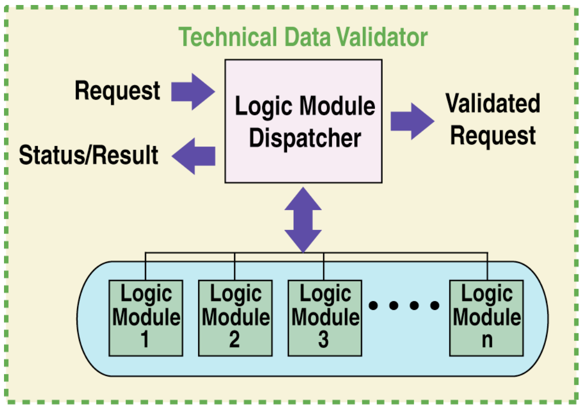

Fig. 2. The technical data validation model structure.

The desire to create request validation via logic modules arises from several reasons. First, the expertise necessary for the required logic for a large fusion experiment is not typically centralized but is spread over all the partners. A modular design allows for much easier division of labor among the various experts. Second, verification and validation of logic modules is easier, as they will be smaller and self-contained. Also, a change in one module has little chance of affecting another module. Third, logic modules can be grouped under different request classifications and therefore have different levels of change control. For example machine safety related transactions could be treated very differently than requests that effect data only (e.g. change in a spectrometer's wavelength setting). Fourth, logic modules can run on a variety of computers and therefore the usage of legacy codes becomes feasible, saving substantial time and quickly integrating the existing large body of knowledge into the program.

It is possible that a number of logic modules will require the usage of an expert system. Some logic modules may not be able to make usage of a more generic expert system and those will need to create their own customized coding. But for the simpler cases, a general expert system will be sufficient and will greatly reduce the amount of redundant coding.

6. IMPLEMENTATION OF GATEKEEPER COMPONENTS

Some of the proposed Gatekeeper components have been tested individually to verify the design concepts mentioned previously while some have been deployed in production environments. Some testing has used the DIII-D Plasma Control System (PCS) [3] which is a software application used to control and monitor various aspects of tokamak plasmas including plasma shape, position, temperature, density, and rotation. It is one of the most widely used plasma control systems and is deployed worldwide at fusion experimental sites in the U.S. (DIII-D, NSTX, MST, MEDUSA), the U.K. (Mast), China (EAST), and South Korea (KSTAR).

The U.S. FusonGrid has successfully deployed X.509 based security for authentication on a worldwide scale. Resource based authorization called ROAM has also been implemented within FusionGrid by designing a database schema that has resource permission information for individual users. This system lacks the concepts of roles and this would have to be added to satisfy Gatekeeper requirements. Details of this work have been published elsewhere [4–6] but the scale of present deployment is sufficient to support present and future tokamak experiments.

Components of Gatekeeper's validation requirements have been investigated with the PCS software since its worldwide deployment requires remote support capability. The PCS software is composed of two main parts, a Graphical User Interface client and the PCS host system. The two components interact with each other remotely utilizing TCP/IP protocol. However, due to stringent network security policies at many sites, the remote PCS client is often blocked from the local PCS host by network firewalls making remote support difficult. Clearly, several aspects of the designed Gatekeeper, such as user authentication and authorization, as well as technical content validation, will increase the security of communications between remote PCS request clients and the on-site PCS system. The Gatekeeper also acts as a proxy between PCS clients and the PCS host and transfers only validated messages to the experimental site. This real benefit along with the modular architecture of DIII-D PCS motivated us to apply the Gatekeeper design to the PCS.

The prototype Gatekeeper software is composed of three main components. The first is a client that composes and sends XML-based PCS requests to the Gatekeeper. The second component is the Gatekeeper front-end script running on an Apache web server that is responsible for accepting the request. The third component is the gatekeeper back-end responsible for parsing and validating XML-based requests, dispatching a technical content validation logic module, and submitting the validated requests to the PCS server. Although the prototype system currently process PCS-related requests, it includes all aspects of the gatekeeper pipeline and therefore extension to other requests is possible.

In the prototype Gatekeeper, the communication between client and Gatekeeper relies on XML-based requests. In order to define the content and structure of the requests, an XML schema has been created. The initial schema design provides a means to describe the PCS action, such as updating the discharge shape algorithm, and a variety of data, such as waveform vertices. The schema not only defines the primitive data types such as, string, integer, float and double, but also has the capability to provide metadata about the filename or location of data in binary format. Figure 3 shows an XML message requesting that the shape algorithm be changed to the IsofluxSingleDivertor algorithm. The request's beginning references the schema-definition file location required for grammar validation. Note that the first two action requests are hierarchical commands specific to the PCS.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<PCS
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation=
 "http://portal.gat.com/xml-/xmlns/gk.xsd">
     <action>
          <name> top </name>
     </action>
     <action>
          <name> select </name>
           <param> Discharge Shape </param>
     </action>
     <action>
          <name> update </name>
          <param> algorithm =IsofluxSingleDivertor </param>
     </action>
</PCS>
```

Fig. 3. A sample XML message for a PCS request.

The prototype request client has been implemented using a simple web-based XML script editor. It provides users with an XML message editor in which users can create Gatekeeper requests manually or by inserting predefined XML tags automatically with button clicks and menu selections. Users can save or submit the XML-based requests through a web browser.

The front end of the Gatekeeper has been implemented as a web service using Django, a python language-based web framework. Django has been chosen due to its support for rapid web application development and "pluggable" architecture [7]. The Django-based server application receives an XML request from clients via HTTP POST method and performs user authentication and authorization.

The XML grammar validation component checks the format and tags of the request against the XML schema and has been written in Python language by using XSV (XML Schema Validator) [8].

The logic module dispatcher is the last component of the Gatekeeper functionality pipeline. It mainly triggers one or more content validation routines based on the request categories. The logic modules may run sequentially or in parallel. Since the content validation codes will most likely be implemented with different software languages and techniques, a simple and flexible communication format between the dispatcher and logic modules is essential. Therefore, a computer-language independent dispatching and communication method is needed. In the prototype system, we utilized Pyro, a python-based Distributed Object Technology system [9] to implement remote code triggering and parameter transmission. The prototype dispatcher was written in Python and the logic module for PCS request content validation was written in C and communication is via XML messages. However, we expect that some logic modules can be very complicated as well as the data passed among them. This may require a more efficient communication framework such as Common Component Architecture [10] and a richer data exchange standard such as Fusion Simulation Markup Language [11]. As a final step, approved requests are sent onward to the PCS host.

The initial prototype Gatekeeper has been completed and has successfully passed initial testing. These tests included sending an XML formatted client request to the DIII-D PCS to change the discharge shape. Note, testing was done in an off-line mode where the PCS was not controlling a live plasma discharge. Preparations are currently underway to test the system for PCS collaborations between DIII-D and EAST including actual machine control.

7. CONCLUDING DISCUSSION

Often when first considered, the concept of sending control commands across the open Internet to directly control sensitive hardware sounds a trifle unrealistic. However, there are clear examples where complex hardware has been controlled entirely by remote control (e.g. NASA's Mars Exploration Rovers). The concepts outlined and demonstrated in this paper illustrate how such remote control of a fusion tokamak can be accomplished in a secure and safe way. The safety aspects employed are automatic validation algorithms that add a layer of safety beyond human input. The secure aspects involve the ability to send communication over the Internet and to guarantee the authenticity of the sender, the integrity of the message, and the authorization level of the sender.

The Gatekeeper as it is conceived is a semantic approach that defines a software system that will be the only channel of interaction for incoming requests to the fusion experimental site. The role of the Gatekeeper is to validate the identification and access privilege of the requestor and to ensure the validity of the proposed request. The initial testing of the Gatekeeper components has been successful and indicates that the vision for the Gatekeeper is realistic. Future work will involve deploying these concepts within the PCS system at DIII-D as well as at select remote sites.

REFERENCES

[1] XML- Extensible Markup Language: http://www.w3.org/XML.

[2] B. Schneier, “Applied Cryptography,” John Wiley & Sons, 1994.

[3] B. G. Penaflor, *et al.*, A structured architecture for advanced plasma control experiments, Proc. of the 19th Symposium on Fusion Technology, Lisbon, Portugal, Vol. 1 (1996) p. 965.

[4] D.P. Schissel, *et al.*, Grid computing and collaboration technology in support of fusion energy sciences, Phys. Plasmas **12**, 058104 (2005).

[5] D. P. Schissel, *et al.*, Building the U.S. National Fusion Grid: results from the National Fusion Collaboratory Project, Fusion Engin. Design **71**, 245–250 (2004).

[6] J. R. Burruss, T. M. Fredian, M. R. Thompson, ROAM: An authorization manager for grids, J. Grid Computing **4**(4), 413-423 (2006).

[7] Django: http://www.djangoproject.com/

[8] XSV: http://www.ltg.ed.ac.uk/~ht/xsv-status.html

[9] PYRO: http://pyro.sourceforge.net/

[10] Common Component Architecture: http://www.cca-forum.org/

[11] S. Shasharina, C. Li, Bull. Am. Phys Soc. **50**(8), 316 (2005).

ACKNOWLEDGMENT

This work was supported in part by the U.S. Department of Energy under DE-FC02-01ER25455.