THE HTTP 'BLACK BOX' PROTOCOL FOR CONTROL AND DATA ACQUISITION AT JET *

C.H.A. Hogben, F.S. Griph, P.J.L. Heesterman, M. Beldishevski, K. Kneupner, R.M.A. Lucock and JET EFDA contributors†

Abstract

The CODAS (Control and Data Acquisition System) and IT department of UKAEA Culham has developed a communication protocol for centralised simultaneous data acquisition, control and monitoring of a large number of processors. It is developed around the Hypertext Transfer Protocol (HTTP)[1] standard.

The protocol has now been in use for about 5 years. It is intended to allow parallel, collaborative developments to take place, by defining communication interfaces between systems, while allowing the internal details of each implementation to remain opaque. For this reason, it is referred to as the 'black box' protocol.

The paper outlines the main factors that led to this protocol choice, and the benefits and experience gained.

The paper briefly describes the functionality of the protocol, and explains how it is being used on a variety of platforms, including Windows, Linux; on PC, VME, and PLC hardware.

PROJECT BACKGROUND

JET[2] is the world's largest Tokamak to date, and has been in operational use since 1983. Sited at Culham, Oxfordshire, UK, it is operated by the United Kingdom Atomic Energy Authority (UKAEA) on behalf of the European Fusion Development Agreement (EFDA)[3].

Today the main role of JET is to support the ITER[4] project with engineering and physics. This requires a steady integration of a large number of equipment upgrades with the existing plant.

CODAS[5] comprises the hardware and software necessary to perform experiment setup, controlled execution of pulsed experiments and retrieval and storage of the resulting experimental data.

SCOPE OF THE PROTOCOL

Since 2000 there has been a need to effectively integrate a rapid increase in new JET instrumentation. The added equipment is often developed by remote collaboration and delivered in kind from EFDA associations. This requires a well defined interface to CODAS. The 'black box' application protocol was developed to provide such a comprehensive interface. The only aspect of the CODAS to plant interface not covered by the protocol is the integration with JET hardware triggers and reference clock, which require direct electrical interfacing.

Collection of Experimental Data

The primary intention of the 'black box' protocol was to provide an interface for systems that acquire measurement data from the JET experiment. The sequence of events for data acquisition is as follows:

- Pre-pulse, at which the system is armed to acquire data. This event normally carries with it some parameters to characterise the data acquisition, for example the number of acquisition triggers to expect from the hardware;
- The JET pulse, during which the instrument acquires data into its own local storage (memory or disk);
- Post-pulse, during which some systems may need to save data from volatile memory to disk;
- Data collection - during which the data acquired during the pulse is transferred to the central data warehouse.

Status and State Monitoring

The JET equipment operational status has a narrow definition: either it is good or there are one or more reasons why the instrument will malfunction, if the next pulse is initialised or triggered at that very moment. The status is used by the Engineer-in-Charge and the Diagnostic Coordinator to decide whether a new pulse can be initialised or, if the initialisation has succeeded, whether the pulse can be triggered to proceed under automatic timing control.

In addition JET equipment state-variables are used by applications in the control room to display or monitor the equipment state. For example a voltage or pressure may be displayed to the end-user on a mimic or be used to raise alarm conditions when equipment need attention.

Setting Parameters

There is a common requirement for control room staff to be able to change various settings of a system. These parameters need to be sent to the system, and read back to verify that the update was successful.

Logging

In order to provide the HTTP server and the JET plant equipment with a way to log information on the CODAS subsystem, a logger interface is provided.

* This work has been performed under the European Fusion Development Agreement.

† See the Appendix of F. Romanelli et al., Fusion Energy Conference 2008 (Proc. 22nd Int. Conf. Geneva, 2008) IAEA, (2008)

TCP/IP Networking

When the design of the protocol began in 2001, it had become clear that Ethernet for the datalink layer and TCP/IP for the network/transport layer would be the dominant networking technologies for the new JET systems under consideration. Thus any software interface would be based upon those foundations.

REVIEW OF OTHER ALTERNATIVES

When deciding on a protocol, our primary motivation was data collection; other aspects followed later. Our goals were several: to allow black box implementers to interface to JET easily; to allow efficient transfer of data; to enable robust handling of error conditions.

Before choosing HTTP, we drew on our past experience with a message protocol, and considered several alternatives for the new protocol.

Past Experience with a Custom Message Protocol

Our previous efforts at interfacing PC-based instrumentation systems to CODAS involved using an existing message protocol used within the JET project. This protocol is layered on TCP/IP and uses small, fixed-size headers and binary data items. A number of drawbacks were identified:

- The use of binary data caused difficulties due to differences in byte ordering between computer architectures.
- A full low level protocol stack needed to be ported or re-implemented.
- The message protocol was primarily designed for small messages, not for large streams of data. The support libraries require a complete message to be assembled in memory by both the sender and receiver. Without significant changes to these libraries, collection of a large amount of data in a single message was therefore impractical. Instead, multiple messages were used, which incurred considerable performance penalties.

File Transfer Protocol (FTP)

The early PC implementations used a file-based mechanism to communicate between the message server and the application: the server wrote a file to request the application to arm for a pulse; the application wrote its acquired data to another file to be read by the server. This led us to consider a simple network equivalent using FTP.

The perceived advantages were: elimination of the message server layer; use of standard FTP server software. The disadvantages: it would not be easily extensible e.g. for error reporting; a file-based backend would be assumed; FTP is an awkward protocol to implement, both client-side and server-side. A separate data port is required, and multiple transactions are needed to establish a session.

Remote Procedure Call (RPC)

We considered several forms of RPC. Advantages: potential re-use of existing software support. Disadvantages: no obvious cross-platform standard; all we examined carried a significant infrastructure burden; as with our existing message protocol, they were not designed for large data streams; we would still have to design an application layer.

Custom Protocol

We could have designed our own protocol as a layer on top of TCP/IP. Advantages: we could make it as simple as we needed - an exact fit to our requirements. Disadvantages: a careful design would be needed to allow for possible future extensions in many directions; with no third-party software available, black box implementers would have to code from the ground up.

BRIEF PROTOCOL DETAILS

The black-box system acts as an HTTP server and software on the CODAS computers as HTTP client. Each interface supported by a server (e.g. status or logging) corresponds to a Uniform Resource Locator (URL) or, in the case of data acquisition, a related set of URLs. A method call on an interface is implemented as a GET or POST request to the URL, with parameters passed using the same mechanism as submitting a web form with multiple fields. As per the HTTP specification[1], a GET must not change server state; when state is changed, a POST must be used.

Full details of the protocol are available elsewhere[6].

Polled Monitoring

For monitoring of state or status, an HTTP GET is used. A state request obtains a simple plain text response, of which each line contains one item name and its value. A status request obtains either an empty response, indicating that the instrument is operational; or a 'Structured Reason' (see later) containing one or more error reports.

Setting Parameters

When a user modifies one or more parameters, they are sent to the instrument using a POST request. This is chosen because there is a consequential state change on the server. Read-back uses a GET, equivalent to state monitoring.

Data Acquisition

Data acquisition is initiated by one or more POST requests, typically one corresponding to a module and one for each channel. The parameters include details such as the pulse number, and how many sample triggers are to be expected. POST requests are also used to signal events such as completion of a pulse or an aborted pulse.

Data collection is a GET request for each channel; the server returns either the collected data or a Structured Reason for any data acquisition error.

Structured Reasons

The 'black box' protocol uses a representation of undesirable states and processing exceptions called Structured Reasons[6]. A Structured Reason report is hierarchical, combining explanatory text and Uniform Resource Identifiers (URI)[7] to pin-point the exact location and nature of a problem. This is used both when monitoring the equipment readiness for a pulsed experiment and when returning reasons why the requested data were not available. It is serialised using the Extensible Markup Language (XML).

BENEFITS AND EXPERIENCE

The choice of HTTP as the basis for the protocol carried a number of advantages, and no significant disadvantages.

Rich and Easily Extensible

The nature of the HTTP protocol meant that the JET 'black box' protocol could be implemented as a very 'thin' layer on top, specialising the semantics for our needs.

HTTP also provides several easy ways to extend the information transferred. Firstly, an HTTP response contains an arbitrary number of header lines; it is easy to add a bespoke header line to convey an item of metadata. Secondly the use of MIME type to describe the content of the message body allows for specialised data formats to be used while remaining within the standard.

The ease of use of the textual part of the response is complemented by the ability for the message body to contain binary data, thus maximising the use of the network bandwidth available for large volumes of data. Furthermore, where bandwidth is the limiting factor, the protocol allows compression to be negotiated between client and server, at the expense of processor power. (Though this has not been needed at JET, it would be straightforward to introduce.)

Widely Deployed Standard

The fact that HTTP is a widely deployed standard (underpinning the World Wide Web) carries other benefits:

- There is a likelihood that developers, both at JET and at collaborating partners, have familiarity with the standard, resulting in a gentler learning curve than would be the case for a less popular or novel protocol.
- There are several robust and well-supported open source and commercial HTTP implementations available, for a variety of programming languages and platforms, both for the server side (black box), and for the client side (CODAS, and black box testing). This affords much flexibility for black box implementers, who may have varied constraints arising from their particular instrument.

Current State

We have nearly 80 deployments of the 'black box' protocol in use. There are more than 60 HTTP servers on Windows PCs and nearly 20 HTTP servers on Linux based systems, that implement the 'black box' protocol. Several more HTTP servers are planned. The Windows PCs normally use an application framework developed at JET using a combination of Microsoft IIS or Java and C++[8]. Linux black box implementations exist using Apache, Tcl and Python. A PLC implementation is planned.

The CODAS HTTP client has been implemented on Solaris using software components. These have been deployed into configured control applications using the Codas Component Framework (CFW)[9]. The General Acquisition Program (GAP)[10] contains another HTTP client, independently implemented using the Curl open source library[11].

DISCUSSION

The approach we adopted has proven successful in interfacing a number of externally developed systems to JET, and could be useful for other collaborative ventures. The key features are these:

- Distill the sets of interactions down to a handful of simple interfaces or aspects
- Define the interfaces but leave implementation details opaque
- Base on a well-known non-proprietary transport layer for maximum familiarity by developers and availability of tools and software
- Where possible, provide reference implementation(s) and test tools.

REFERENCES

[1] R. Fielding et al., "Hypertext Transfer Protocol - HTTP/1.1", IETF RFC 2616, September 2004, http://www.ietf.org/rfc/rfc2616.txt

[2] http://www.jet.efda.org

[3] http://www.efda.org

[4] http://www.iter.org

[5] H. van der Beken, et al., "CODAS: The JET Control and Data Acquisition Systems", Fusion Technology 11(1), pp. 120-137

[6] C.H.A. Hogben, F.S.Griph, "Interfacing to JET Plant Equipment Using the HTTP Protocol", Aug. 2008, EFD-R(08)001, http://www.iop.org/Jet/article?EFDR08001.pdf

[7] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt

[8] P.J.L. Heesterman, et al., "The JetFsm Data Acquisition Framework, and Proposed Usage for ITER", PCaPAC, October 2008

[9] F.S. Griph, C.H.A. Hogben, M.A. Buckley, "A generic component framework for real-time control", IEEE Trans. Nuc. Sci. 51(3), June 2004, pp 558-564

[10] H. van der Beken, et al., "Data acquisitions at JET – experience and progress", IEEE Trans. Nuc. Sci. 36(5), October 1989, pp 1639-1646

[11] http://curl.haxx.se/libcurl/