ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP17

EXTENDING THE REMOTE CONTROL CAPABILITIES IN THE CMS DETECTOR CONTROL SYSTEM WITH REMOTE PROCEDURE CALL SERVICES

R. Jiménez Estupiñán, ETH Zurich, Zurich, Switzerland
Attila Racz, Christian Deldicque, Christian Wernet, Christoph Schwick, Cristina Vazquez Velez, Dainius Simelevicius [1], Diego Da Silva Gomes, Dinyar Rabady, Dominique Gigi, Emilio Meschi, Frank Glege, Frans Meijers, Hannes Sakulin, Jeroen Hegeman, Jonathan Richard Fulcher, Luciano Orsini, Maciej Gladki, Marc Dobson, Michael Lettrich, Philipp Brummer [1], Thomas Reis, CERN, Geneva, Switzerland
Ulf Behrens, DESY, Hamburg, Germany
Audrius Mecionis [2], Jean-Marc Andre, Mantas Stankevicius [1], Nicolas Doualot, Petr Zejdl[3], Remigius K. Mommsen, Srecko Morovic, Valdas Rapsevicius [1], Vivian O'Dell, FNAL, Chicago, Illinois, USA
Christoph Paus, Georgiana-Lavinia Darlea, Guillelmo Gomez-Ceballos, Zeynep Demiragli, MIT, Cambridge, Massachusetts, USA
Andrea Petrucci, Rice University, Houston, Texas, USA
Ioannis Papakrivopoulos, Technical University of Athens, Athens, Greece
Samim Erhan, UCLA, Los Angeles, California, USA
Andre Holzner, James Branson, Marco Pieri, Sergio Cittolin, UCSD, San Diego, California, USA
[1]also at Karlsruhe Institute of Technology, Karlsruhe, Germany, [2]also at Vilnius University, Vilnius, Lithuania, [3]also at CERN, Geneva, Switzerland

Abstract

The CMS Detector Control System (DCS) is implemented as a large distributed and redundant system, with applications interacting and sharing data in multiple ways. The CMS XML-RPC is a software toolkit implementing the standard Remote Procedure Call (RPC) protocol, using the Extensible Mark-up Language (XML) and a custom lightweight variant using the JavaScript Object Notation (JSON) to model, encode and expose resources through the Hypertext Transfer Protocol (HTTP). The CMS XML-RPC toolkit complies with the standard specification of the XML-RPC protocol that allows system developers to build collaborative software architectures with self-contained and reusable logic, and with encapsulation of well-defined processes. The implementation of this protocol introduces not only a powerful communication method to operate and exchange data with web-based applications, but also a new programming paradigm to design service-oriented software architectures within the CMS DCS domain. This paper presents details of the CMS XML-RPC implementation in WinCC Open Architecture (OA) Control Language using an object-oriented approach.

INTRODUCTION

CMS DCS applications are implemented using the SIMATIC WinCC OA [1] platform, mostly written in its native programming language called control (CTRL) language. The CTRL language is a C-like language with a very poor type definition syntax, not suitable for programming complex software architectures. The CMSfwClass [2] is a programming framework to build object oriented (OO) applications using CTRL language. The CMS XML-RPC toolkit relies on the CMSfwClass framework, which permits a better implementation of the software architecture after an extensive planning and design phase. The toolkit provides a set of software classes to build client-server architectures (See Fig. 1), enabling heterogeneous software entities to act as service providers (servers) or service requesters (clients).

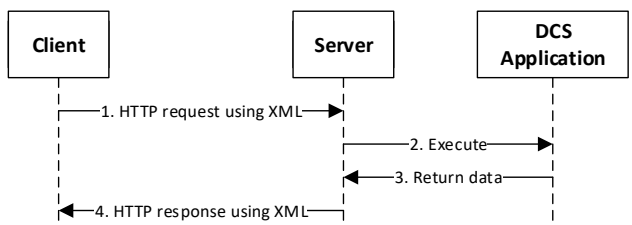

Figure 1: Client-server activity diagram on XML-RPC.

USE CASES

During the design phase, the software models were prepared to support at least the following scenarios: Remote procedure calls from the DCS user interface (UI), service-oriented collaboration between DCS applications, and DCS web services.

Remote Procedure Calls from the UIs

The CMS DCS is a large distributed environment, organized in a hierarchy of nodes and accessed from remote lo-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP17

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

cations. In this context, the code associated to UIs is executed in the client machines while the events and data processing happens in the control nodes. Certain operations need access to resources that are only available in a remote control node (server). The CMS XML-RPC toolkit allows programmers to encapsulate server-only operations in the form of service objects that can be invoked from the UI. This feature allows the UIs to access resources, which are typically not accessible on the client side (E.g. local server files, system calls, etc.). In addition, the OO implementation hides the server details from the UIs; delimiting the concerns of the software and resulting in a cleaner implementation, compared to the classic procedural approach.

Service-oriented Collaboration

Many of the DCS applications are implemented as standalone applications, which are not ready to share functionality without further development. The CMS XML-RPC toolkit introduces a mechanism to model DCS functionality as a service. Different applications can be prepared to act as producers (servers) and consumers (clients) of services, permitting a well-structured collaboration interface between remote peers. In addition, this mechanism provides an abstraction layer to connect applications on different platforms.

DCS Web Services

The CMS XML-RPC toolkit complies with the standard XML-RPC specification using HTTP over TCP. This means that any application implementing these protocols can make use of the features exposed by DCS web services, including standard web applications. The CMS online portal (https://cmsonline.cern.ch) is a web portal hosting DCS related applications. Some of its applications connect to DCS web services; offering a new set of control features to the CMS users worldwide.

ARCHITECTURE

The CMS XML-RPC toolkit does not provide ad-hoc functionality. Instead, it provides a software model that can be used or extended to build new tailored client-server applications. The model includes two abstract classes describing the basic functionality of a service dispatcher and a service client. These classes are unaware of the final implementation details, as well as the message-encoding format, permitting the further extension of the architecture. In addition, the model includes concrete classes to build different client-server architectures.

Service Classes

The service classes are the core structures for building service based applications. The toolkit includes three subclasses derived from the abstract service class. Each of them extends the initial service implementation.

- Service class: This class implements the basic service functionality. Objects of this class offer a list of procedures to be executed remotely in the DCS context.
- Service router class: This class groups multiple service objects; offering a single entry point to clients. Objects of this class are particularly useful to concentrate and minimize the number of connections between the DCS and other platforms.
- Proxy server class: This class implement a message forwarding mechanism. Objects of this class can intercept, filter and divert requests to other service providers.

Service System Interface

The service system interface describes a list of popular RPC functions. The system interface is not part of the XML-RPC protocol itself, but it is a common feature supported by most of the available RPC servers. Using the system functions, clients can request the full list of available methods, their signatures or even the encapsulation of multiple calls to a remote server into a single request.

Client Classes

The CMS XML-RPC toolkit includes a set of service client classes, derived from an abstract service client class. Each of them implement different ways of invoking services.

- Client class: This class implements a basic method invocation by forwarding requests to specific service objects. Objects of this class use the internal WinCC OA run-time database to pass the messages to the service objects.
- Web client class: This class forwards the requests to a web service. Objects of this class listen to specific host ports using the transmission control protocol (TCP).
- Http client class: This class extends the web-service client functionality by using the HTTP protocol over TCP.

Proxy Server Configuration

By combining the server and client functionality into a single class, objects can configure a proxy server capable of serving, requesting, and therefore forwarding requests to other peers (see Fig. 2). A proxy server object can be configured to administer a group of resources, acting as a firewall between the CMS DCS web server and external clients.

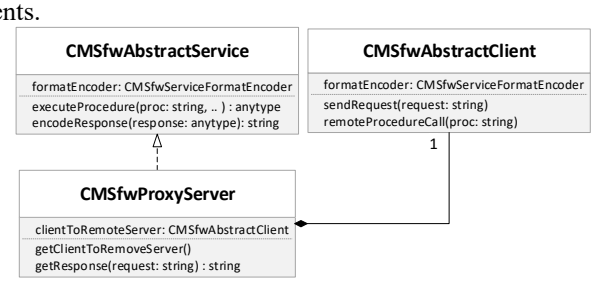

Figure 2: Service-client composition.

Message Formatting Classes

The information exchange between clients and servers require the encoding and decoding of messages. Clients should encode their request and transmit them using one of the available formats. Servers will decode the messages and process the request. The results are sent back to the

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP17

client, repeating the encoding-decoding operations on both sides. The CMS XML-RPC toolkit delegates the format encoding operations to objects implementing the message format. The different formatting classes implement the same interface; which defines the common methods for encoding and decoding messages. Thanks to this model, new formats can be easily included or extended without altering the final architecture.

IMPLEMENTATION DETAILS

The CMS XML-RPC toolkit is composed of 13 classes, 8 class interfaces and a Web service application handler. Once the framework is installed, objects can be created using a factory script or any of the available serialization mechanisms. The configuration of the objects and the relations between them is what determines the final software architecture. In total, the framework comprises around 3000 lines of code, in addition to the code required for building the final application.

Web Service Application Handler

WinCC OA provides an application programming interface (API) for building a standard HTTP server using CTRL language. The CMS XML-RPC toolkit uses the HTTP server API to implements a web-service application handler. Instances of any service class can be passed as parameters to the application handler to run a fully functional web service.

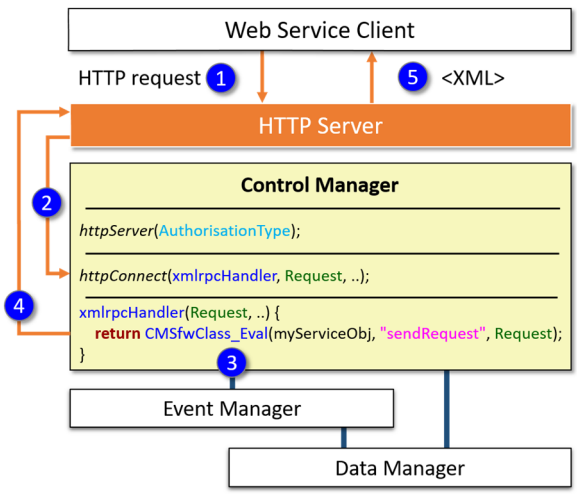

Figure 3: CMS XML-RPC application handler.

As shown in Fig. 3, web clients initiate the procedure with a HTTP request to the server. The application handler configures the HTTP server to listen to a particular port number in the control node. When the request arrives to the server, a call-back function processes it and delegates the execution to a service object. The service object decodes the request, executes the requested operation and encodes the results in the appropriate format. Results are passed back to the HTTP server, which replies to the corresponding client.

Message Format

The CMS XML-RPC toolkit provides two format-encoding classes. The first formatting class complies with the standard XML-RPC protocol specification. The second class implements a variant where responses are formulated in JSON. The usage of the JSON format is only available to web clients, which are able to decode this type of messages. This format is particularly useful in the context of JavaScript applications, since the encoding is part of the language notation. Messages encoded in JSON are lighter than the ones in XML (see table 1) requiring less space and processing time. Using one of the XML-RPC services, we have measured the length of the messages in the different formats (See Table 1).

Table 1: Size of the Messages

Type of message	Format	Bytes
Client request	XML	111 bytes
Server response	XML	1,042 bytes
Server response	JSON	597 bytes

A single query to the system returned a list of 25 items with 534 bytes of content data. The format of the XML message took more than 50% of the total message space. By contrast, the JSON format required only 1% of the message space. For this reason, JSON is the preferred format to exchange data with online applications.

INTEGRATION WITH CMS ONLINE

CMS Online is a web portal using the Oracle WebCenter Portal [3] technology to access technical information of the CMS experiment. The CMS online portal extends the DCS capabilities by offering a set of tools to monitor and administer parts of the DCS. Initially, the data exchange between the CMS Online and the DCS was limited by the usage of databases. Now, the CMS XML-RPC toolkit adds a new connection method to exchange data and perform remote operations from the CMS Online platform. At the moment, two web applications using the CMS XML-RPC toolkit are available:

- Online parametrization browser: This web application allows the users to inspect and change certain parameters in the control systems.
- Online DCS log files browser: This web application exposes the different logs available in the remote nodes, and implements some formatting and filtering features to facilitate the readability.

CONCLUSION

The implementation of the CMS XML-RPC toolkit started as a prototype to prove the feasibility of the XML-RPC protocol within the DCS domain. With the time, after several iterations to extend and refine its design, the toolkit became a consolidated part of the CMS DCS software. As result, the online applications based on the XML-RPC have also become an important tool for the CMS DCS community, allowing experts to access DCS technical information with the only help of a web browser.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP17

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

ACKNOWLEDGEMENTS

The authors would like to thank the Swiss National Science Foundation for the financial support.

We would like also to thank our colleague L. Masetti, who provided insight and expertise in the topics discussed in this paper.

REFERENCES

[1] SIMATIC WinCC OA [online], http://www.etm.at/index_e.asp

[2] R. Jimenez Estupiñan et al. "Enhancing the Detector Control System of the CMS Experiment with Object Oriented Modelling", ICALEPCS'15, Melbourne, Australia, October 2015, paper MOPGF025.

[3] Oracle WebCenter Portal [online], https://www.oracle.com/technetwork/middleware/webcenter/portal/overview/index.html