ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV005

# OPC-UA DATA ACQUISITION FOR THE C2MON FRAMEWORK

E. Stockinger*, B. Copy †, M. Bräger, B. Farnham, M. Ludwig, B. Schofield,
CERN Beams Department, 1211 Geneva, Switzerland

*Abstract*

The CERN Control and Monitoring Framework (C2MON) [1] is a monitoring platform developed at CERN and since 2016 made available under an LGPL3 open source license. It stands at the heart of the CERN Technical Infrastructure Monitoring (TIM) that supervises the correct functioning of CERN's technical and safety infrastructure. This diverse technological infrastructure requires a variety of industrial communication protocols. OPC UA [2], an open and platform-independent architecture, can be leveraged as an integration protocol for a large number of existing data sources, and represents a welcome alternative to proprietary protocols. With the increasing relevance of the open communication standard OPC UA in the world of industrial control, adding OPC UA data acquisition capabilities to C2MON provides an opportunity to accommodate modern and industry-standard compatible use cases.

This paper describes the design and development process of the C2MON OPC UA data acquisition module, the requirements it fulfills, as well as the opportunities for innovation it yields in the context of industrial controls at CERN.

## INTRODUCTION

C2MON must consolidate information from a wide range of sources and so support a diverse set of devices and communication technologies through independent data acquisition modules. OPC UA is considered a key technology to managing heterogeneous machine interoperability and therefore fundamental to developments in advanced industrial environments [3]. As an open protocol, it addresses the common problem of insufficient interoperability by allowing equipment from different vendors to communicate with a host through the same interface [4]. Along with its high data modeling capabilities, OPC UA is a compelling candidate for adapting C2MON to modern use cases.

More generally, trends in industry urge Supervisory Control and Data Acquisition (SCADA) systems towards greater scalability and compatibility with Cloud Computing. While the C2MON server layer has already been adapted to natively support configurability injection and process monitoring [5], the OPC UA data acquisition (DAQ) module presented here pioneers these concepts on the data acquisition layer.

## MONITORING OF THE DAQ PROCESS

The continuous monitoring of software plays a key role in ensuring smooth operation. Effective monitoring solutions enable a timely detection and treatment of issues related to quality-of-service and provide insight into system resource management [6]. The states or qualities to be monitored can be inherently complex and multidimensional, and so elude a direct measurement through single metrics. Kaner et al. [7] suggest the use of multidimensional metrics as a more meaningful representation of a complex quality.

C2MON provides extensive functionality for monitoring the elements within the supervised facilities. The connection between these elements and the processes on the C2MON data acquisition layer is also supervised through C2MON as a fundamental function. However, the DAQs are independent Java processes. Their internal states are not monitored through C2MON, but contain information relevant to ensure the availability, security and usability of the data acquisition infrastructure.

The metrics identified to be of relevance are diverse in nature. For example, the health of the machine and the operating system running the DAQ process can among others be tied to CPU, memory or disk usage, and the network load. Other metrics deal with the JVM and cover garbage collection threads and log messages. These are generic metrics which are provided by several external tools and client libraries. Other metrics are tied to the DAQ and the area of data acquisition. For example, these metrics could expose meta-information regarding OPC UA-specific concepts such as OPC UA subscriptions and their respective filter types.

Within the OPC UA DAQ process, metrics representing such internal states are exposed as observable endpoints through the Spring Actuator [8] project and Micrometer [9]. The metrics can be scraped out-of-the-box through the monitoring toolkit Prometheus [10] and be represented by a multi-dimensional data model. They can then be gathered and aggregated into a form enabling a global analysis by administrators and operators.

The choices of technological solutions were largely taken due to the prevalence of Spring within the C2MON infrastructure, and to be in line with the guiding principles of C2MON which prefer the use of proven technologies and of open-source resources where possible to facilitate reusability in other projects [1].

## CONFIGURABILITY

There are two distinct aspects to configurability within the OPC UA DAQ. On the one hand, there is the need to configure the data to be monitored through C2MON. This use case is integral to the C2MON platform and impacts all tiers of the software architecture. This configuration data is stateful and managed on the application-level across the different DAQ modules and client applications. On the other hand, C2MON acts in heterogeneous environments with diverse hardware requirements. Operators and administrators

* elisabeth.stockinger@live.at
† brice.copy@cern.ch

benefit from a system catering to the high configurability of the OPC UA protocol.

Therefore the OPC UA DAQ module introduces a set of additional configuration options through the Java implementation of the Spring framework. Externalized Spring configurations can be specified in a variety of ways [11], profile specific application property files being the de-facto standard in the C2MON environment. This allows for the specification of different configuration profiles for example in between production and test environments, as well as an easy integration into container orchestration systems such as Docker [12] or Kubernetes [13].

The composition and configuration of required DAQ processes and the servers to be monitored is only known at runtime within the C2MON configuration strategy, and is susceptible to change. Therefore, handling configuration through Spring requires the dynamic creation and management of dedicated configuration scopes. The default Spring configuration scopes [14] do not support configuration options on the level of single connections to OPC UA servers out of the box. Therefore a custom Spring scope was defined to govern the life cycle of all software elements involved in the supervision of individual servers.

The OPC UA DAQ offers configuration options dealing among others with redundancy and connection settings, security, and modification strategies for information defined on the OPC UA servers.

In addition to monitoring, integration of Spring Actuators allows the remote management of applications through JMX and HTTP. Operators can for example change specific configuration settings on running processes, write data directly to OPC UA nodes or execute OPC UA nodes of the method type.

## SECURITY

C2MON as a data communication network faces the risk of cyber-attacks and malware threatening confidentiality, integrity or availability of processes. One of OPC UA's distinguishing features among other low-level networking protocols is its extensive security model [15] which has been verified among others in [16] and [17].

OPC UA can be employed at different levels of the automation pyramid where the security requirements may differ. The uses of OPC UA as a communication interface can span from the isolated shop floor network to access via the internet as shown in Fig. 1. The different use cases are supported by OPC UA's layered approach to security, allowing a trade-off with other quality indicators such as performance or flexibility, depending on the needs of the environment.

OPC UA relies extensively on X.509 certificates [18] for security across these layers, addressing among others application authentication, user authentication and authorization, confidentiality, and integrity.

These certificates support a variety of trust models. Even within a single organization, several trust models can be in place, for example direct trust models where an administrator manages certificates on a case-by-case basis, hierarchical models, or fully-meshed Webs-of-Trust. Given that C2MON is a generic framework which must be adaptable to a variety of environments and use cases, the OPC UA DAQ module should support generic trust models and be configurable in its security modes. There are two major aspects to this solution: the client certificate must be configurable to comply with server requirements, and the server certificate must be processed and validated accordingly.

To satisfy generic use cases, the OPC UA DAQ module can be configured in one of three ways:

- the client certificate may be loaded from local storage,
- a self-signed certificate can be created on the fly, and
- connection can be established without a certificate.

Certificate validation is achieved through a PKI directory, a common approach in security management. This allows administrators to define their own solutions for certificate verification, be it through on-line or off-line methods.

While OPC UA offers rich potential in the area of user authentication and authorization, which would be valuable to exploit for any monitoring application context, this functionality is not currently supported within C2MON.

## REDUNDANCY

Redundancy refers to the integration of several exchangeable instances in an environment. If one redundant component experiences faults, it can be replaced by another. Redundant setups can significantly increase component reliability and robustness and therefore are vital to critical systems.

OPC UA Part 4 [20] defines an extensive and versatile redundancy model encompassing redundant servers, clients, and networks, all of which can co-exist with one another. Server redundancy can be either *transparent* in which case fail-over cases between servers are handled exclusively on the server-side, or *non-transparent*, requiring the client to take action during a fail-over process. The server additionally has a *fail-over mode* which specifies the capabilities and behavior of the redundant setup, each mode imposing different requirements on the client. However, all redundancy modes defined within OPC UA can fall back to a lesser mode, where "Cold Redundancy" is the smallest common denominator.

Complex industrial environments often include servers which communicate through OPC UA but employ different and potentially proprietary models of redundancy. There are several such vendor-specific redundancy models within the context of TIM, demonstrating the need for standardization within industry.

While redundancy is increasingly relevant to ensure high availability, user demand for OPC UA redundancy is currently limited within the TIM environment. Most relevant to the artifact presented here is the use case of server redundancy, where the C2MON DAQ, acting as the OPC UA client, is presented with multiple sources of the same data.

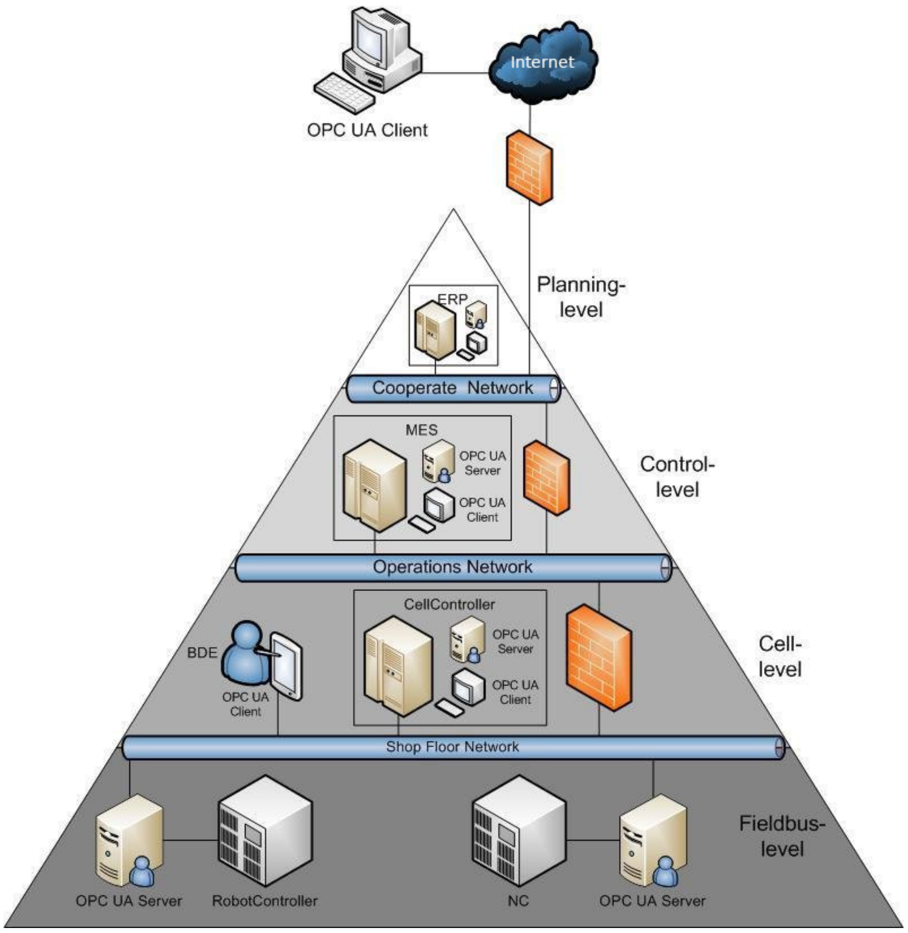

Figure 1: OPC UA as a communication interface in the automation pyramid, as depicted in [19].

For reasons of portability and testability, the design choice was made to only support redundancy in cold failover mode until there is user demand for other modes.

## POTENTIAL AND PROSPECTS

The integration of OPC UA as a powerful open standard into C2MON has implications beyond a greater range of supported devices. There are several potential areas of future research and application within the context of CERN.

### *Configuration Management*

C2MON configuration is passed inbetween the application layers at runtime via asynchronous messaging. This allows the dynamic addition and removal of entities to supervise. However, the configuration itself is static: configuration elements are mapped to specific DAQ processes, and there is only one configuration for each process. The static nature of the configuration of acquisition items is a potential obstacle in deploying C2MON within IoT projects: the conversion is Java-based, resource-intensive, and relies on proprietary services.

As an alternative to being passed between the architectural layers at runtime, it is possible to externalize the configuration. This approach enables a number of design patterns including service discovery, distributed configuration management, or load balancing. Similarly, configuration could be adapted to fit the capabilities and requirements of a DAQ when started within a dynamic environment. In the case of the OPC UA DAQ, instances could be configured to dynamically restrict subscriptions to data points producing very frequent updates on OPC UA servers which do not natively support deadbands. Similarly, data points with a low priority could be omitted if a server were to run low on memory during operation.

Spring supports the externalization of configuration natively. The Spring Cloud Config project [21] provides a dedicated platform for externalized configuration in a distributed system, which would integrate with a dynamic configuration handling within C2MON.

In order to benefit from Spring's support for externalized configuration within software components, these components must be managed by the Spring's inversion of control (IoC) [22], a mechanism abstracting the creation and management of component dependencies. The use cases described here are not currently supported by the C2MON framework core nor required within the TIM environment, but are only offered by the OPC UA DAQ module. The C2MON DAQ core initialization strategy historically relies on Java class reflection, and predates the wide adoption of IoC design patterns in the C2MON technology stack. By externalizing the management of software components to Spring, we allow our OPC UA DAQ module to benefit from the rich options for configuration handling offered by the Spring ecosystem. This new approach to process configuration will hopefully be generalized to the C2MON DAQ core and benefit all other DAQ modules in the near future.

### *OPC UA as an Integration Platform within CERN*

Equipment within a plant shop floor commonly offers several forms of communication, using proprietary communication protocols side-by-side with OPC UA or other open protocols. Redundant server setups may be supported natively only in configurations using proprietary communication protocols. Similarly, vendors may offer redundant server setups for OPC UA servers which rely on custom or proprietary architectural abstractions rather than the extensive redundancy model defined within the OPC UA standard. This non-standardized treatment negatively impacts system interoperability and integration.

Yielding to the prevalence of non-standard redundancy models, the OPC UA DAQ was designed to be extensible towards external models while supporting the OPC UA model of redundancy. The scope of extensions to vendor-proprietary models was restricted to initial connection, failover, and monitoring of connection status to ease the effort of switching from vendor-proprietary protocols to OPC UA without loss of functionality.

Likewise, only minor changes are necessary within the CERN environment to support OPC UA compliant certificates for increased security. CERN offers an established and well-accepted infrastructure regarding certificate authorities and certificate management, which is however not equipped to handle the creation of OPC UA-compliant certificates. Developers and administrators must therefore either forgo an integration of the CERN root certificate, or they must develop and maintain a custom add-on application polling the official revocation list and certificate authority, and handling certificate renewal.

The only factor preventing compatibility in the CERN certificate management services is the integration of certain extensions required by OPC UA. This issue would be easy to solve, thereby positively impacting the standardization of certificate creation and management across the organization.

OPC UA also offers functionality for user authentication and authorization. The exploitation of this functionality would open up new possibilities for auditing and access control. However, this would require C2MON to manage user authentication and authorization through a standardized interface across all DAQs and client applications. Such an interface is not available at present, among other reasons due to the stark differences in capabilities of supported protocols. The implementation of user authentication, authorization and of auditing are relatively open in the OPC UA specification, and out-of-the-box solutions offered by vendors are not cohesive. While this impedes the provision of standard auditing and access control functionality in heterogeneous environments, the integration of such an infrastructure is a worthwhile venture for future investigation.

Finally, OPC UA can reduce the complexity of an existing infrastructure whilst adding functionality. This is the case in some areas of alarm handling in the CERN infrastructure. Some systems supervised through the CERN C2MON instance declare alarms internally and publish them through JMS rather than publishing the data points themselves. Several intermediate steps are required to supervise these alarms in C2MON:

- The alarms must be transformed from into a JMS message that is compatible to C2MON.
- These JMS message are passed through a message broker to a process-specific DAQ.
- The filtered alarms are transformed into the C2MON “DataTag” format and passed to C2MON through the JMS queue.
- The alarms arrive at the C2MON cluster either in the state “on” or “off” and are further processed accordingly.

The devices exposing alarms in this fashion within the TIM environment offer communication through an OPC UA server. These data points can therefore be published directly to the new OPC UA DAQ module rather than as alarms to a separate and process-specific DAQ process.

This would allow the omission of the process-specific DAQ as well as the message format transformation, and so streamline the supervision pipeline in C2MON.

Alarms could in this fashion be defined and managed dynamically based on these data points using the rich configuration and validation functionality offered by C2MON client services [23, 24], avoiding consistency issues and increasing efficiency.

The current and proposed processes for handling alarms are contrasted in Fig. 2.

### *OPC UA as a Simulation Interface*

The high complexity of services and processes acting within a SCADA system requires a holistic evaluation in a close-to-production setting to assure security and safety [25–28]. Ludwig *et al.* have found the sufficient validation of control systems for power supply at CERN not to be feasible using laboratory approaches [29]. They propose a simulation-based validation strategy for introducing new hardware into the environment using a custom simulation engine for power supply crates.

[29] describes a setup of redirecting traffic within OPC UA servers to target their so-called “VENUS” simulation engine instead of the physical crates, whilst keeping all other aspects of the SCADA systems in a production state. This allows for the validation of different hardware compositions within the production environment.

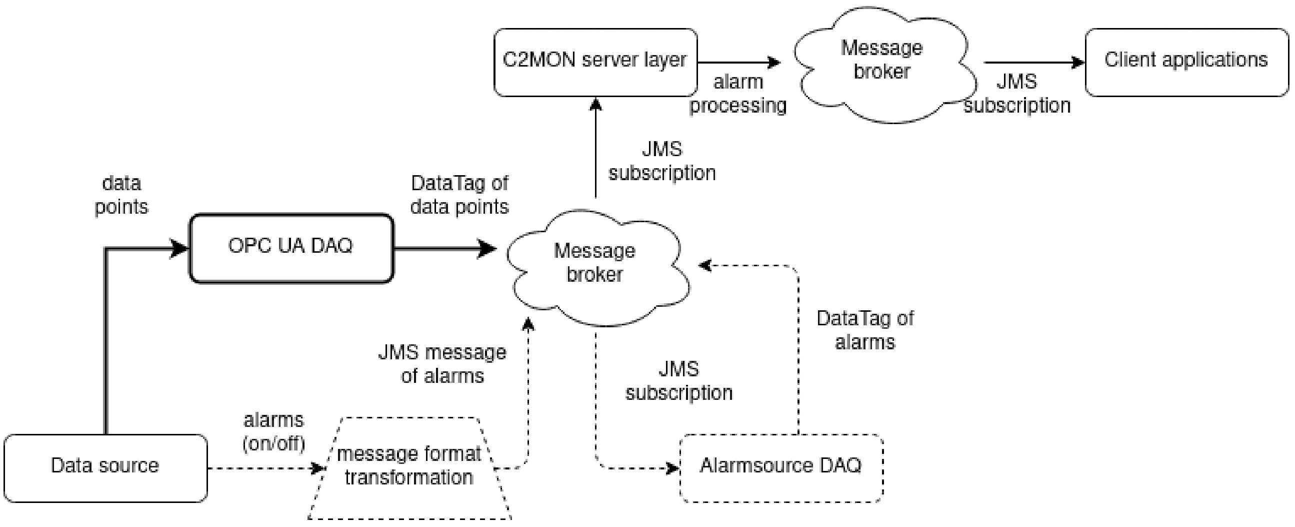

Figure 2: The pipeline of supervising alarms from systems currently publishing alarms as data points. Dotted elements indicate architectural blocks which could be omitted by exposing data points directly through OPC UA. Bold elements depict the alternative path through the OPC UA DAQ.

Rather than change the simulated hardware composition, it is also possible to trigger load and conditions exceeding those nominally observed in production. In this fashion, the SCADA layer can be examined for its tolerance to corner cases and exceptional circumstances. The OPC UA DAQ can so be used as an interface for validating the behavior of the SCADA system in adverse circumstances.

## CONCLUSIONS

Along with extensive OPC UA integration, C2MON combines the powerful open-source Java management platform with OPC UA's rich data model and ubiquity in industry. The C2MON OPC UA DAQ module leverages OPC UA's potential as an integration platform on the plant level. Its support for Java process monitoring and its extensibility towards vendor-custom redundancy models makes it ideally suited to the ever-evolving demands of large-scale infrastructure monitoring. Finally, OPC UA can benefit a range of use cases within modern industry such as testing new hardware compositions within a simulated environment as well as testing the SCADA layer under simulated peak loads.

## REFERENCES

[1] M. Bräger, M. Brightwell, E. Koufakis, R. Martini, and A. Suwalska, "High-Availability Monitoring and Big Data: Using Java Clustering and Caching Technologies to Meet Complex Monitoring Scenarios", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper MOPPC140, pp. 439–442.

[2] OPC Foundation: OPC Unified Architecture: `https://opcfoundation.org/about/opc-technologies/opc-ua`

[3] I. González, A. Calderón. J. Figueiredo and J. Sousa, "A Literature Survey on Open Platform Communications (OPC) Applied to Advanced Industrial Environments", *Electronics*, vol. 8, no. 5, p. 510, Mar. 2019.

[4] J. Fitch and H.Y. Li, "Challenges of SCADA protocol replacement and use of open communication standards", in *Proc. of the 10th IET Int. Conf. on Developments in Power System Protection (DPSP'10)*, Manchester, UK, Mar. 2010, p. 35.

[5] B. Copy, M. Bräger, A. Papageorgiou Koufidis, E. Piselli, and I. Prieto Barreiro, "Integrating IoT Devices Into the CERN Control and Monitoring Platform", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA125.

[6] , A. Van Hoorn, J. Waller and W. Hasselbring, "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis", in *Proc. of the 3rd ACM/SPEC Int. Conf. on Performance Engineering (ICPE'12)*, Boston, MA, USA, Apr. 2012, pp. 247–248.

[7] c. Kaner and W.P. Bond, "Software Engineering Metrics: What Do They Measure and How Do We Know?", in *Proc. of the 10th IEEEInt. Software Metrics Symposium (METRICS'04)*, Chicago, IL, USA, Sept. 2004.

[8] Spring Boot Documentation on Actuators: `docs.spring.io/spring-boot/docs/current/reference/html/production-ready-features`

[9] Micrometer Application Monitoring: `micrometer.io`

[10] Prometheus: `prometheus.io`

[11] Spring Boot Documentation on Externalized Configuration: `docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-external-config`

[12] Docker: `www.docker.com`

[13] Kubernetes: `kubernetes.io`

[14] Spring Boot Documentation on Bean Scopes: `docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-factory-scopes`

[15] D.S. Pidikiti, R. Kalluri, R.K.S. Kumar and B.S. Bindhumadhava, "SCADA communication protocols: vulnerabilities, attacks and possible mitigations", *CSI Transactions on ICT*, vol. 1, no. 2, pp.135–141, Apr. 2013.

[16] OPC Foundation Security Bulletins: https://opcfoundation.org/security

[17] Kaspersky ICS CERT Security Analysis of OPC UA: https://ics-cert.kaspersky.com/reports/2018/05/10/opc-ua-security-analysis

[18] R. Housley, T. Polk, W.S. Ford and D. Solo, "Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.", *RFC Editor*, vol. 5280, pp.1–151, Apr. 2002.

[19] W. Mahnke, S.H. Leitner and M.Dann, "OPC Unified Architecture", Heidelberg, DE: Springer, 2009.

[20] OPC UA Specification: Part 4 - Services, Release 1.04: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-4-services

[21] Spring Cloud Config: spring.io/projects/spring-cloud-config

[22] Spring Documentation on the IoC container: https://docs.spring.io/spring-framework/docs/current/reference/html/core

[23] M. Bräger *et al.*, "Improving Alarm Handling for the TI Operators by Integrating Different Sources in One Alarm Management and Information System", presented at the ICALEPCS'19, New York, NY, USA, Oct. 2019, paper MOPHA118, this conference.

[24] Z. Zaharieva and M. Buttner, "CERN Alarms Data Management: State & Improvements", in *Proc. ICALEPCS'13*, Grenoble, France, Oct. 2011, paper MOPKN011, pp. 110–113.

[25] R.L.Krutz, "Securing SCADA systems", Indianapolis, In, USA: Wiley Publishing Inc., 2005.

[26] M. Brändle and M. Naedele, "Security for process control systems: An overview", *IEEE Security & Privacy*, vol. 6, no. 6, pp.24–29, 2008.

[27] J. Slay and M. Miller, "Lessons learned from the maroochy water breach", in *Proc. of the 1st. Int. Conf. on Critical Infrastructure Protection (ICCIP'07)*, Hanover, NH, DE, Mar. 2007, pp. 73–82.

[28] M. Masera, I.N. Fovino and R. Leszczyna, "Security assessment of a turbo-gas power plant", in *Proc. of the 2nd. Int. Conf. on Critical Infrastructure Protection (ICCIP'08)*, Arlington, VA, USA, Mar. 2007, pp. 31–40.

[29] M. Ludwig, J.A.R. Arroyo Garcia, M. Bengulescu, B. Farnham, P.G.J. Gonzalez Jimenez, and F. Varela, "Developing and Validating OPC-UA Based Industrial Controls for Power Supplies at CERN", in *Proc. 12th International Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'18)*, Hsinchu City, Taiwan, Oct. 2018, pp. 35–37. doi:10.18429/JACoW-PCaPAC2018-WEP04