ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUMPA05

OPC UA TO DOOCS BRIDGE: A TOOL FOR AUTOMATED INTEGRATION OF INDUSTRIAL DEVICES INTO THE ACCELERATOR CONTROL SYSTEMS AT FLASH AND EUROPEAN XFEL

F. Peters[†], I. Hartl, C. Mohr, L. Winkelmann, DESY, Hamburg, Germany

Abstract

Integrating off-the-shelf industrial devices into an accelerator control system often requires resource-consuming and error-prone software development to implement device-specific communication protocols. With recent progress in standards for industrial controls, more and more devices leverage the OPC UA machine-to-machine communication protocol to publish their functionality via an embedded information model.

Here we present a generic DOOCS server, which uses a device's published OPC UA information model for automatic integration into the accelerator control systems of the FLASH and European XFEL free-electron laser facilities, thus reducing software development time and errors.

We demonstrate that the server's and protocol's latency allows DOOCS-based burst-to-burst feedback in the 10Hz operation modes of FLASH and European XFEL and is capable of handling more than 10^4 data update events per second, without degrading performance. We also report on the successful integration of a commercial laser amplifier, as well as our own PLC-based laser protection system into DOOCS.

MOTIVATION

The DOOCS control system [1] of the FLASH and European XFEL free-electron laser facilities extends to thousands of individual sensor and actuator devices, connected to hundreds of distributed computer nodes. Many of these devices are off-the-shelf industrial products and the majority of them feature a device-specific communication protocol. Integrating them into the control system requires implementing the custom protocols and mapping the devices' functionality to the DOOCS data model. This software development process is both resource-consuming and error-prone.

The development of a general OPC UA to DOOCS bridge software enables the immediate integration of OPC UA devices into the control system, based on the device's published OPC UA information model. Thus integration effort and costs are minimized.

INTRODUCTION TO OPC UA

A result of recent progress in industrial controls is the standardization of the OPC UA (Open Platform Communications, Unified Architecture) protocol [2]. It differs significantly from it's predecessors by it's ability to not only transport machine data (such as control variables, measurements and parameters) but also describe them semantically in a machine-readable information model.

In the simplest case, the device-specific part of the OPC UA information model consists of named Variables, Objects and Methods, called *nodes*. The device-specific root node is a defined object, fittingly named *Objects.*

- ***Objects*** contain any number of variables, methods and/or other objects.
- ***Variables*** are either scalars or fixed-size arrays of a fixed type (e.g. 32-bit integer, float, string)
- ***Methods*** have zero or more input arguments as well as zero or more output arguments. The arguments are named and typed, similar to variables.

The OPC UA protocol follows a server-client communication model over TCP/IP. Clients can perform synchronous requests to read and write node values, i.e. get or set the value of a variable and call a method. Additionally, clients can subscribe to nodes to be asynchronously notified when a variable's value changes.

INTRODUCTION TO DOOCS

The DOOCS (Distributed Object-Oriented Control System) data model constists of named and typed *properties*. The type of a property can range from simple scalar values such as bits or floating point numbers, over arrays, up to arbitrarily complex data types, such as camera images with metadata.

Properties are organized into l*ocations*, where one location typically corresponds to one physical device. Each location and it's properties can be identified by their DOOCS addresses. Several locations can run alongside each other in a server process on a specific computer node.

The DOOCS protocol also follows the server-client model over TCP/IP. Clients can read or write the values of properties synchronously via “get” and “put” requests. Alternatively, clients can subscribe to properties, to receive asynchronous updates of the properties value via a “server push” messaging mechanism.

MAPPING OPC UA TO DOOCS

Although there are considerable similarities between the data and communication models of OPC UA and DOOCS, some aspects are incompatible and impose limitations to the bridging between the two protocols.

First, one DOOCS location connects to a single OPC UA server and exports the objects, variables and methods in a subset of it's data model as DOOCS properties. At the same time it is possible that several DOOCS locations connect to the same OPC UA server, mapping separate,

[†] Email: falko.peters@desy.de

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUMPA05

overlapping or even the identical subsets of it's data model.

Second, the components of the OPC UA data model are “translated” to DOOCS properties as follows:

Variables

Each OPC UA variable is mapped to one DOOCS property with an appropriate type (Table 1). The DOOCS server subscribes to OPC UA data-change events for the mapped variable to achieve fast update times. Property value changes are published to the DOOCS system via the publish-subscribe mechanism.

Table 1: Mapping of Types Between OPC UA and DOOCS

OPC UA	DOOCS Scalar	DOOCS Array
Boolean, Sbyte, Byte	D_int	D_bytearray
Int16, Uin16	D_int	D_shortarray
Int32, Uin32	D_int	D_intarray
Int64, Uin64	D_int	D_longarray
Float	D_float	D_floatarray
Double	D_double	D_doublearray
String, LocalizedText	D_string	-

The DOOCS property's name is generated from it's path in the OPC UA data model, prefixed with “OPCUA” to distinguish the mapped properties from other properties related to configuration or DOOCS server diagnostic.

Methods

DOOCS does not directly support the concept of a method call with arbitrary arguments and return values. Thus OPC UA methods are each mapped to several DOOCS properties:

- One DOOCS property for each of the method's input arguments.
- One DOOCS property for each of the method's output arguments. The type mappings for both input- and output arguments are the same as for variables (Table 1).
- One “CALL” property of type *D_int*.

To call the OPC UA method, a client first sets the DOOCS properties for the input arguments. To actually execute the method, the value 1 is written to the “CALL” property. The DOOCS properties for the output arguments are then set to the values returned by the OPC UA method call.

As a consequence, the mapped methods are not “atomic”. If several DOOCS clients access a method's argument properties, race conditions may occur.

Objects

Each child of an OPC UA object (i.e. another object, variable or method) is mapped to DOOCS properties. DOOCS does not support the concept of nesting and all properties in a location exist in a flat structure. Thus the object itself does not appear in DOOCS. Only it's name is retained as a component of the address of properties belonging to the object.

EVENT-BASED COMMUNICATION BETWEEN OPC UA AND DOOCS SERVERS

The communication between the OPC UA and DOOCS Systems involve four distinct components (Fig. 1). The components are typically physically distributed over the facility's TCP/IP control network.

OPC UA Devices are the industrial devices to be integrated into the control system. Often these are based on programmable logic controllers (PLCs).

The **OPC UA DOOCS Server** is the software component discussed here. It is written in C++, using the

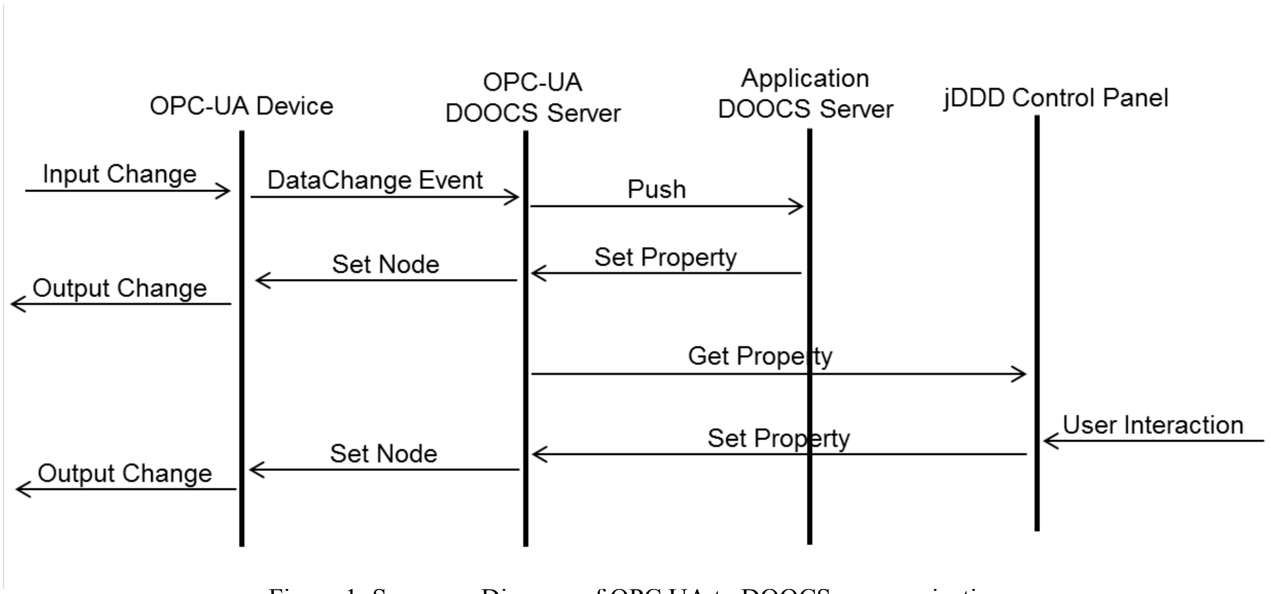

Figure 1: Sequence Diagram of OPC UA to DOOCS communication.

standard DOOCS libraries as well as the open-source FreeOpcUa library for OPC UA communication [3]. It's only task is mapping the communication between the OPC UA and DOOCS protocols. No device-specific logic is programmed into this server. Note that this software component is a DOOCS *server*, but an OPC UA *client*.

Application DOOCS Servers, also called middle-layer servers, can optionally be used to perform device-specific operations on the OPC UA device in an automatic or semi-automatic fashion without direct operator input, e.g. feedback controls or running state machines.

JDDD Control Panels (Java DOOCS Data Display)[4] enable operators to perform control tasks.

Feedback Performance

Using the asynchronous update capabilities of both OPC UA and DOOCS (DataChange event and Server Push, Fig. 1) enables a low-latency operation of controls and feedbacks in device-specific application servers, limited mostly by the performance of the OPC UA device.

To measure feedback performance, we use a Beckhoff [5] CX5140 PLC running an OPC UA server to read an analog input signal. Through the asynchronous update data path as described above, the signal is propagated to a "Feedback" application server, doing nothing but immediately writing the received value back to the OPC UA DOOCS server and to an analog output signal in the PLC. The rising-edge delay between the PLC's input- and output signals is our performance metric (Fig. 2). We observe that feedback latency depends linearly on PLC cycle time. Cycle times up to about 20ms allow burst-to-burst feedback in the 10Hz-operated FLASH and European XFEL facilites.

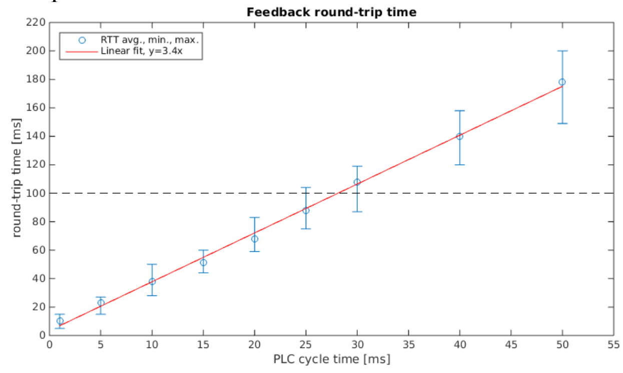

Figure 2: Feedback performance of the OPC UA DOOCS server. The dashed line at 100ms shows the repetition period of the burst-mode operated FLASH and European XFEL facilities.

Behaviour Under Load

To analyze the system under load, we measure the behaviour of the OPC UA DOOCS server with a feedback-setup similar to the one described previously. A number of variables (16-bit integers) are set to random values on a PLC running with 10ms cycle time. Again we measure round-trip time of an analog input signal as our performance metric. Additionaly we monitor server CPU load and network load (Table 2).

We observe constant performance up to more than 10^4 DataChange events per second (100 variables, updated every 10ms). With a larger number of update events, CPU load becomes a limiting factor.

Table 2: OPC UA DOOCS Server Performance Under Load

Number of OPC UA Variables	Round-trip-time avg.	Server CPU load	Network load
1	38ms	3.5%	-
10	35ms	10%	530kB/s
100	36ms	17%	2MB/s
1000	63ms	67%	13MB/s

Note that the use of individual 16-bit integer variables is close to a worst-case in terms of protocol overhead. The same number of updating variables could be transferred more efficiently by using arrays.

INTEGRATION OF COMMERCIAL DEVICES INTO DOOCS

The OPC UA DOOCS server is currently used for integration of several PLC-based systems into the FLASH and European XFEL control systems.

- A commercial laser amplifier is installed in the XFEL backup photocathode-laser currently being commissioned. The laser control system runs on a Beckhoff PLC including an OPC UA server.
- Several Beckhoff PLCs are installed in the laser systems at FLASH and European XFEL used for controlling actuators in the laser- and beamline-systems as well as for monitoring purposes such as lab-temperatures and cooling water flow.
- For the upcoming FLASH2 pump-probe laser system, a PLC-based laser protection system is developed at DESY.

Integration of these devices into DOOCS was achieved without programming work, requiring only configuration of the servers' network connections and, where necessary, the design of jDDD control panels.

CONCLUSION

The emergence of standards such as OPC UA in industrial controls enable rapid integration of conforming devices into existing control infrastructure. The development of a single piece of software, bridging the OPC UA protocol and DOOCS, makes the integration of OPC UA-enabled devices into the control systems of the FLASH and European XFEL free-electron laser facilities easy, quick and inexpensive. Using a standardized communication protocol increases system reliability as well as availability of support in the form of litarature and commercial services.

By choosing appropriate mappings between the data models of the protocols, we achieve a well-performing and reliable, yet simple interplay between the heterogenous components of the control systems.

REFERENCES

[1] G. Grygiel, O. Hensler, K. Rehlich, “DOOCS: A Distributed Object-Oriented Control System on PC’s and Workstations”, PCaPAC conference, 1996.

[2] The OPC Foundation, “OPC Unified Architecture”, http://opcfoundation.org/opc-ua/

[3] FreeOpcUa library, http://freeopcua.github.io

[4] E. Sombrowski. A. Petrosyan, K. Rehlich, W. Schütte, “jddd, a state-of-the-art solution for control panel development”, in Proc. ICALEPCS’11, Grenoble, France, October 2011, paper THBHAUST04.

[5] Beckhoff Automation GmbH, http://beckhoff.com