MIGRATION FROM OPC-DA TO OPC-UA

B. Farnham, R. Barillère, CERN, Geneva, Switzerland

Abstract

The OPC-DA specification of OPC has been a highly successful interoperability standard for process automation since 1996, allowing communications between any compliant components regardless of vendor. CERN has a reliance on OPC-DA Server implementations from various 3rd party vendors which provide a standard interface to their hardware. The OPC foundation finalized the OPC-UA specification and OPC-UA implementations are now starting to gather momentum. This paper gives a brief overview of the headline features of OPC-UA and a comparison with OPC-DA and outlines the necessity of migrating from OPC-DA and the motivation for migrating to OPC-UA. Feedback from research into the availability of tools and testing utilities will be presented and a practical overview of what will be required from a computing perspective in order to run OPC-UA clients and servers in the CERN network.

INTRODUCTION: OPC CURRENTLY AT CERN

At CERN, OPC is heavily used to control and monitor a large variety of devices. One commonly used technology stack allowing a human controller to control physical devices is a HMI and SCADA layer provided by Siemens [3] WinCC OA, WinCC OA communicates via its built-in OPC driver (an OPC Client) to an OPC server and the OPC Server communicates with the physical device in question.

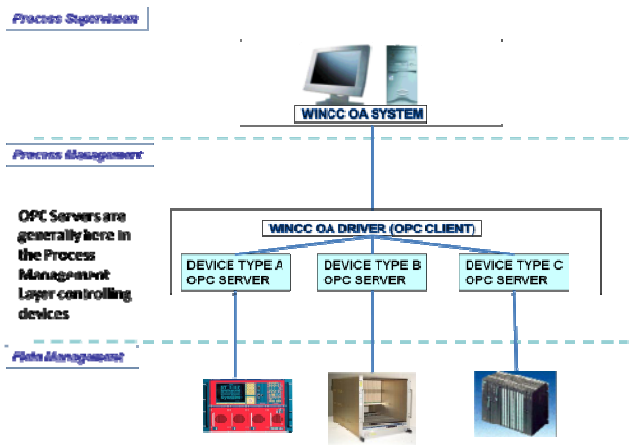

Figure 1: Common current usage of OPC at CERN as device orientated middleware.

The device vendor generally provides hardware plus an OPC Server capable of driving it. Current devices driven by OPC at CERN include:

- PLCs, providing low level control and monitoring of systems.
- High and low voltage power supplies, powering detector tubes and front end electronics.
- VME crates, providing specialist control of peripherals.
- Embedded Local Monitor Boards (ELMBs), a CERN proprietary I/O module for monitor/control of front end equipment.

Prior to OPC-UA, the OPC Foundation produced a series of sibling OPC specifications, amongst which was OPC-DA. These sibling specifications are hereafter referred to as OPC Classic. OPC-UA [1] is intended to be the modern successor to OPC Classic. CERN currently has only OPC Classic applications in production systems, however, there are some OPC-UA evaluation projects currently in progress.

OPC-UA HIGHLIGHTS

Platform Independence

OPC Classic is tied to Microsoft platforms by its dependence on COM/DCOM for security and inter-process communication. The OPC-UA specification has a built-in means of security, transmission and message encoding based on modern industry standards. Dropping the COM/DCOM dependency frees OPC-UA client and server implementations from the Microsoft platform.

Embedded Platforms

An important corollary of the point above is that OPC-UA servers can be written for embedded platforms – a device could have its own OPC-UA server built in. For scalability, the full functionality of the OPC-UA specification is chunked into discrete profiles describing subsets of the full feature set: An embedded OPC-UA server need only implement profiles relevant to its operating constraints.

Improved Security

Classic OPC has no intrinsic security; this is delegated to the COM/DCOM layer. OPC-UA has a comprehensive security model built in based on Public Key Infrastructure providing application identity and secure channel client/server communication. The specification also describes a means for exchanging user identity for application specific user authorization and authentication.

Improved Modelling

The OPC-UA standard provides an extensive vocabulary for modelling devices and processes, including being able to type components (allowing clients to establish semantic information) and to express relationships between components (allowing clients easier browsing between related components).

Message Transmission and Encoding Options

An OPC-UA slogan is 'from the device to the enterprise'. OPC-UA has been designed to allow applications to

encode and transmit messages in a manner most suited to their operating environment and constraints. There are two means of transmission: An efficient low level means, based on TCP/IP, called tcp.opc and ubiquitous HTTP. There are two means of encoding messages: UABinary, which is small on the wire and requires less processing overhead than the alternative XML/SOAP encoding which allows for the possibility of OPC-UA messages to be consumed by a wide variety of applications (i.e. not just OPC-UA). OPC-UA applications providing software interfaces to devices are expected to prioritise efficiency, whereas OPC-UA applications providing high level information publication are expected to prioritise support for the widest client base.

MOTIVATION IN MOVING FROM OPC CLASSIC TO OPC-UA

The primary motivation for migrating from OPC Classic is that its message transmission is based Microsoft's proprietary COM/DCOM. COM/DCOM has been de-emphasized in favour of .NET's Windows Communication Framework. OPC-UA on the other hand describes two non proprietary means of message transmission, tcp.opc or HTTP. OPC-UA applications can choose to support one or both transmission types.

Linux compatibility. Some administrators of Detector Control Systems (DCS) have selected Linux as their preferred operating system but are forced to maintain windows machines in order to host OPC Classic applications. Linux compatible OPC-UA replacements would allow administrators to remove these windows nodes thereby reducing maintenance complexity by homogenizing the host operating system, Linux, across the DCS. Devices supplied with embedded OPC-UA servers would further simplify DCS administration: The device hosts its own OPC-UA Server.

OPC-UA security is standards based and easier to configure. Restricting access to OPC Classic applications is possible but experience has shown that setting up DCOM security configuration is error prone and the results fragile to operating system updates and patches. OPC-UA application authenticity is provided by standard x509 certificates which every OPC-UA application is required to uniquely own. OPC-UA applications only accept session requests from other OPC-UA applications proffering trusted certificates. These same certificates are used to provide secure communications channels for client server sessions in which all messages can be signed (preventing message tampering) or signed and encrypted (additionally preventing eavesdropping).

OPC-UA allows for more intuitive server address spaces. As mentioned in OPC-UA highlights above, OPC-UA has some quite considerable changes and enhancements over OPC Classic. OPC Classic provides a simple address space in a tree structure of branch nodes containing branch nodes and leaf nodes. A common idiom in OPC Classic is to have write-only leaf nodes, which, when written to, invoke an action on the OPC server/device. To turn on/off a channel in an industrial power supply, for example, an OPC Classic user often writes true/false to a write only OPC item. The idiom works but to the uninitiated user it is not obvious that this is the standard means of turning a channel on. OPC-UA servers describe their address spaces in objects (as in Object Orientation) whereby objects contain fields, and methods which have an effect on those fields. An OPC-UA server for an industrial power supply could model a channel as an object with fields representing current, voltage etc. and a method, parameterised with a Boolean, to turn the channel on and off. This interface is more self-explanatory. Views are another useful OPC-UA address space enhancement. Similar to database views, OPC-UA views exist to provide subsets of the full address space tuned to a specific perspective. For example, a cooling and ventilating application is more likely to be interested in temperature sensor values of an industrial power supplies than, say, voltage and current read outs. An industrial power supply OPC-UA server could provide views tuned for such perspectives.

OPC-UA IMPLEMENTATION AVAILABILITY

Stacks

The OPC Foundation provides their corporate members with reference OPC-UA implementations, stacks, of the standard. Stacks are available in three languages: ANSI C, .Net and Java. The primary goal of the stacks is providing communications interoperability. At time of writing only the .Net stack supports both transmission means (opc.tcp and HTTP) and both message encodings (UABinary and XML/SOAP). The C stack and Java stack provide only opc.tcp transmission and support for binary encoded messages. In terms of the C stack this is not unreasonable, it is the most likely to be used for embedded or high performance environments where HTTP/SOAP would not be the natural choice. The .Net stack is the most fully featured but is limited to the .Net runtime, essentially limiting it to Microsoft platforms.

Toolkits

These are built on top of the stacks and aim to provide a much more complete SDK to aid and simplify application development (providing programmatic session management for example). The toolkits generally inherit any omissions or limitations in functionality from the stacks upon which they are built, the Java SDKs surveyed, for example, do not currently support HTTP transmission or XML/SOAP encoding.

There are various stack vendors available but the main players are:

- Unified Automation [5] - Have C, C++ and Java toolkits. Applications built with the Java toolkit are naturally cross platform, application hosts require JRE6. The C and C++ toolkits can, on request, be cross compiled by the vendor to various

Windows/Linux platforms including embedded variants and realtime operating systems.

- Softing [4] - Have a .Net (requiring the .Net 3.5 runtime) and C++ toolkit available. On request the vendor can compile the C++ toolkit for Windows or Linux.
- Prosys [6] - Have a Java toolkit, application hosts require JRE6.

Diagnostic Tools

An important tool for OPC deployment is a simple, visual OPC client, allowing verification and problem diagnosis of server installations. Unified Automation, provide UaExpert: A free, stable and sufficiently fully featured OPC-UA client available on Linux and Windows. For low level diagnosis, ascolab [8] provide a Wireshark [7] plugin allowing an engineer to view client/server messages (for unencrypted client/server sessions).

FUNCTIONALITY TESTS

The following tests were carried out using available implementations from various vendors. In all cases only UABinary message encoding was used and opc.tcp transmission.

Cross Stack and Cross Platform Interoperability

In every attempted permutation (see table below) of client/server toolkit and platform the connecting client was able to connect to, and browse, the server without problems:

Table 1: Permutations tested for cross platform and toolkit client/server interoperability. All succeeded.

Client Description	Server Description	Server Description	Server Description
Java on Windows	Java on Windows	C++ on Windows	C++ on Linux
C++ on Windows	C++ on Windows	C++ on Linux	Java on Windows
C++ on Linux	C++ on Linux	C++ on Windows	Java on Windows
Java on Linux	C++ on Linux	C++ on Windows	Java on Windows

OPC-UA as Device Middleware

OPC Classic is heavily used at CERN as device middleware (see figure 1), OPC-UA's applicability for device middleware was tested. The test stack consisted of a WinCC OA HMI and SCADA layer, using its built-in OPC-UA driver to communicate with a custom made OPC-UA server providing control and monitoring of simulated hardware.

The WinCC OA system (version 3.10) and its OPC-UA driver ran on Windows 7. The OPC-UA server was built using Softing's C++ toolkit, the OPC-UA server and hardware simulation processes ran on Scientific Linux 5.

Table 2: Test results of OPC-UA functionality for device middleware

Functionality	Results/Notes
OPC Classic Functionality in OPC-UA	
Create a simple, browseable address space for device.	The OPC-UA server builds an address space to represent the device, browseable from WinCC OA.
WinCC OA datapoints could be mapped to server variables.	WinCC OA datapoints have a 'peripheral address' configuration, allowing mapping between them and OPC-UA server address space elements.
WinCC OA Datapoints updated as hardware values change.	WinCC OA defines subscriptions on the OPC-UA server, with publishing intervals. The OPC-UA server sends cyclic notifications to WinCC OA, updating the datapoints.
Can write values to server variables	WinCC OA's dpSet() command uses peripheral address mappings to send write commands to the OPC-UA Server, on receipt, the server writes hardware values.
OPC-UA Enhancements	
Server object methods can be browsed and invoked	The OPC-UA Server can provide methods on objects. These can be invoked using the UaExpert client. Siemens confirmed this is not available in WinCC OA version 3.10.
Server views can be browsed and datapoints mapped to view fields	As above, the OPC-UA server can provide views, browseable from UaExpert. Siemens confirmed this is not available in WinCC OA version 3.10.
OPC-UA Security	
OPC-UA applications only communicate with trusted applications	The OPC-UA Server only communicates with trusted clients however it was possible to initiate communications between WinCC OA and untrusted servers. The vendor is investigating this.
OPC-UA client/server traffic can be signed or signed and encrypted.	The OPC-UA Server supports this, tested in session with UaExpert, however attempts to create secure sessions between the server and WinCC OA failed. The vendor is investigating this.

Overall, OPC-UA analogs of OPC Classic functionality, essential for device control and monitoring, work in the surveyed OPC-UA implementations. Nice-to-have OPC-UA device modelling enhancements were not present in the WinCC OA version used. WinCC OA security issues are being investigated by the vendor.

INFRASTRUCTURE REQUIREMENTS OF OPC-UA APPLICATIONS AT CERN

The main difference for system administrators is adapting to OPC-UA's security requirements.

Firewall Settings

OPC-UA has been designed to be firewall friendly. For tcp.opc traffic the administrator need only open the port required by each endpoint. Endpoints using HTTP transport require port 80 open.

Application Certificates

CERN already has infrastructure in place for generating and managing the type of signed certificates OPC-UA applications require: The CERN CA (Certificate Authority). The CERN CA is visible from both CERN's technical and general purpose networks. Currently, obtaining a CERN signed certificate requires non-trivial manual interaction with the CERN CA website. This effort would be required on each application installation and additional effort to renew certificates before they expire (CERN certificates generally expire after one year). Planned CERN CA enhancements include provision for programmatic certificate generation and renewal. Once in place, OPC-UA applications could be delivered with a module which communicates with the CERN CA interface to automate (as far as security policy allows) the process of obtaining signed certificates as part of the installation procedure and certificate renewal.

OPC AS HIGH LEVEL MIDDLEWARE

OPC-UA's features make it an interesting candidate for high-level middleware providing a means of inter-system communication and publication of enterprise level summary data.

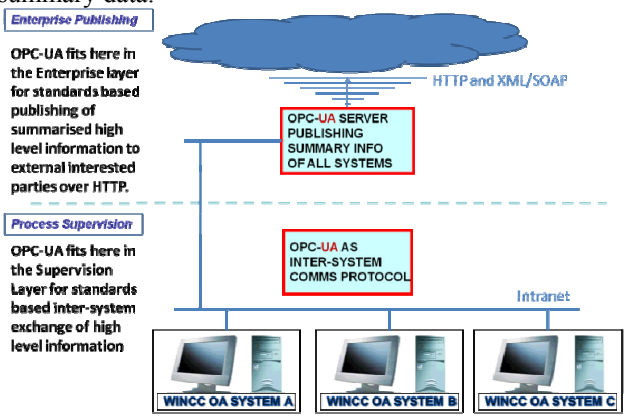

Figure 2: OPC-UA in high level middleware roles.

- OPC-UA is a common standard, compatible with a range of 'off the shelf' components.
- OPC-UA provides secure client/server sessions, with message integrity and confidentiality.
- opc.tcp transmission and UABinary encoding are designed for bandwidth and processing efficiency.
- HTTP transmission and XML/SOAP encoding opens the possibility of delivering messages to a broad base of non OCP-UA specific clients.
- Toolkit interoperability allows for creating a communications system between applications dispersed over heterogeneous platforms.
- Service discovery via OPC-UA discovery servers.

Whilst the features above are very promising facets of OPC-UA for high level middleware, the technology is, at time of writing, insufficiently mature for the purpose for three main reasons:

- During the evaluation, secure sessions failed to work between WinCC OA and an OPC-UA server. The vendor is investigating.
- HTTP transmission and XML/SOAP encoded messages are only currently available in the windows centric .Net tools.
- The final 'Discovery Server' section of the specification is currently in draft; however it is expected to be released this year.

CONCLUSION

OPC-UA is clearly a most compelling candidate for control and monitoring middleware between SCADA systems and device layers. OPC Classic, the incumbent CERN middleware, is based on deprecated COM/DCOM technology and OPC-UA is its successor, based on modern industry standards. The full functionality described in the specification was not found to be available using the current technology and tools surveyed. Despite this, however, current OPC-UA tooling would be sufficient (once security issues are resolved) to provide applications to match and exceed functionality currently provided by OPC Classic applications, most notably in that existing OPC-UA tools allow for interoperable cross platform implementations. Features described in the OPC-UA specification make it an interesting candidate technology for standards based, inter-system communications. However, implementations of some of these features (e.g. server discovery) were absent in the tools surveyed. The evolving OPC UA toolset will be monitored for implementations of these features and verified to ensure they fully cover these expectations.

REFERENCES

[1] The OPC-UA Specification. www.opcfoundation.org
[2] W. Mahnke, S. Leiner, M. Damm, 'OPC Unified Architecture', Springer 2009.
[3] Siemens Automation. www.automation.siemens.com
[4] Softing. www.softing.com
[5] Unified Automation. www.unified-automation.com
[6] Prosys. www.prosysopc.com
[7] Wireshark. www.wireshark.org
[8] ascolab. www.ascolab.com
[9] EN/ICE OPC Support website: www.cern.ch/wikis/display/EN/OPC+Support