The *OPC Security Specification* was not widely used in the past. Therefore, a detailed description of the specification contents in this book has been abandoned in favor of other topics. The importance of security has drastically increased in the past few years. The protection from unauthorized access to OPC servers and their data used to be limited to DCOM Security and was consequently increasingly regarded as a weak point of Classic OPC technology. For this reason, security has played a major role in OPC UA from the very beginning. The fact that the *OPC UA Specification* has dedicated a separate part – *Part 2* – to the topic of security shows just how important this subject is.

2.3 OPC Unified Architecture

2.3.1 Introduction

2.3.1.1 Never touch a running system – why a new OPC?

OPC is unquestionably one of the most successful de-facto standards since the invention of the computer. Today, a user can choose from over 20,000 OPC products offered by more than 3,500 vendors. Millions of installed OPC based products are used in production, processes, building automation, and many other industries around the world. This renders the OPC technology as the undisputed standard for interoperable data exchange between software applications from different vendors. OPC allows automating data transfer between widely distributed installations. OPC interfaces provide easy to use, high-performance connectivity between automation components, control hardware and field devices, thus bridging the divide between heterogeneous automation worlds. The OPC technology is used today for practically all types of data acquisition, for vertical and horizontal data integration, and data management. OPC is the essential link for HMI/SCADA systems for process visualization, for DCS systems or PLCs for process control, and MES and ERP systems to access the underlying automation components. In the early days of the OPC technology only process data or individual parameters were transferred over the OPC interface. Today, OPC is used to transport entire ERP documents, parameter sets, control sequences, or to drive control applications.

OPC is proven, well-established, and globally successful technology. What motivates the OPC Foundation to introduce the *OPC Unified Architecture*? Is OPC UA a "new OPC"? Will OPC UA supersede Classic OPC? What advantages does the new Unified Architecture offer compared to Classic OPC? These and other questions will be discussed in following sections.

2.3.1.2 Ten Reasons for OPC UA

The following section outlines ten key reasons that lead to the development of an entirely new technology generation, the *OPC Unified Architecture*. They are based, on the one hand, on the experiences with OPC as well as the technological changes and trends over the past 13+ years since the beginning of OPC technology. On the other hand, they also take into account many wishes and suggestions from OPC vendors and users.

1. Discontinuation of COM/DCOM

The automated exchange of data between Classic OPC applications is based on Microsoft's COM technology. Distributed COM (DCOM) allows OPC clients and OPC servers to communicate across computer boundaries. As the Windows operating system rapidly became widely used all over the world and promoted the use of Windows computers in automation, it provided ideal conditions for driving the widespread adoption of OPC technology. In early 2002, Microsoft launched its new .NET framework and announced the discontinuation of DCOM. This did not mean that future versions of Windows operating system versions would not support DCOM – the requirements for using existing OPC components and other DCOM applications would continue to be met. But, as a result of the discontinuation, the base technology of Classic OPC would not be further developed and sooner or later become obsolete. With .NET, Microsoft has introduced a framework that uses XML and Web Services as the base technology. The OPC Foundation followed suit with *OPC XML-DA* in 2003, releasing the first OPC specification that used XML and Web Services instead of COM/DCOM. Experiences with the implementation of XML-DA components not only under Windows, but also under Linux and other non-Windows operating systems were promising, even though OPC communication was limited to process data (*Data Access*) at first and the data throughput of the SOAP XML protocol did not reach the required performance level demanded by many automation tasks. But it was a start. The first step into an OPC technology without COM/DCOM had been taken. This approach also showed, however, that a high-performance solution had to be found for exchanging process data, events and historical data.

2. DCOM limitations

With COM/DCOM, Microsoft introduced a set of features in the 90ies that were highly appreciated by both end users on home computers in the non-industrial segment and professional users who used Windows computers as automation components in industrial applications. These features include copy and paste, drag and drop, linking and embedding – technologies that offer an easy way to exchange data between different Windows applications, embed graphics or spreadsheets in text documents, and much more. DCOM also offers the complete communication infrastructure with all the necessary security, such as authentication,

authorization and encryption. DCOM Security controls the access rights to data and programs on remote computers. But DCOM Security at the same time also presents a major challenge for setup engineers, system integrators and developers managing projects that involve OPC communication across PCs. Setting up DCOM Security correctly is a very complicated task and takes a lot of expertise. For example, the access rights granted to a user during Windows login have to be precisely adapted to the DCOM Security settings. As a result, setup engineers and system integrators routinely choose to speed up the process by granting very broad access rights on all networked OPC computers and thus largely disabling the protection from unauthorized remote access. This shortcut collides with IT security requirements and, in the long run, risks damage caused by negligence or sabotage. DCOM Security settings are often a showstopper for the otherwise very easy to configure OPC communication relationships, and they lead the support statistics of OPC product vendors. Another limitation of DCOM are the long – and not configurable – timeouts for detecting broken communication. If the connection breaks between an OPC client and an OPC server on a remote computer, which e.g. acquires data from a PLC, it can take many seconds until the OPC client is informed. These response times are unacceptable in most industrial applications. The DCOM limitations are key drawbacks in the otherwise so successful OPC technology. A popular and widely used strategy to prevent DCOM problems are tunneling mechanisms that avoid DCOM completely and provide their own security and connection monitoring features (see section 4.4.3).

3. OPC communication across firewalls

The possibilities of OPC communication across computer boundaries were recognized very early in the automation industry. The advantages are obvious: Access to remote OPC servers is completely transparent; from the user's point of view there is no difference between accessing local or remote OPC data. Using suitable DCOM Security settings, many communication stations within an intranet can be easily networked via OPC without problems. But what about OPC communication outside an intranet, i.e. over the Internet? Exchanging data over the Internet is only desirable in automation projects if adequate security is ensured. This makes the use of a firewall indispensable in protecting the data of an automation plant against unauthorized access. And this is where DCOM again limits Classic OPC communication. DCOM needs a large number of ports for executing the communication services. Ports are part of a frame address, which assigns the data segments to a network protocol. Fixed ports are, for example, port 80 for HTTP, port 443 for HTTPS and port 21 for FTP. DCOM requires multiple ports for establishing a connection, for authentication, for transmitting data, and for a number of other services. The ports to be used are randomly assigned by DCOM. If these ports are not available, DCOM automatically looks for others. Consequently, many ports have to be opened in a firewall to allow DCOM communication across it. Every open port in a firewall is a security gap and provides a potential target for hacker attacks. If Network Address Translation (NAT) based firewalls are used, no OPC

communication link can be established at all because DCOM cannot handle the address translation. All in all, it is not possible to establish secure DCOM OPC connections across firewalls. OPC Tunneling is a widely accepted strategy to solve DCOM's limitation of the use of Classic OPC products (see section 4.3.3).

4. Use of OPC on non-Windows platforms

DCOM is included in all Windows operating systems. On the one hand, the 'near-omnipresence' of the Microsoft platforms in industrial applications has been a major factor in promoting the rapid acceptance of OPC. On the other hand, DCOM restricts the use of OPC technology to Windows. In many industries, this does not limit the acceptance of OPC. However, there are areas, such as IT, where Windows operating systems are the exception and UNIX or Linux systems the rule. Automation, too, has application areas that categorically refuse to implement Windows operating systems. They are mainly found in conservative industries, such as the chemical or pharmaceutical segments, where the short-lived Windows operating system versions, the sometimes considerable differences between Windows NT, 2000, XP and Vista, and security issues preclude the use of Microsoft operating systems. The embedded area is another area in which Windows hardly features (except for Windows CE or embedded XP). This sector has for the last few years displayed a strong trend towards small and compact automation devices. At the same time, these devices are equipped with increasingly intelligent, high-performance processors. The applications they use are getting more and more complex and are embedded directly in field devices, PLCs, operator panels and other devices running VxWorks, QNX, embedded Linux, RTOS or other embedded operating systems without DCOM. Integration concepts with OPC are doomed to fail in those areas because OPC needs DCOM as the technological basis, and this basis is missing in embedded systems.

5. High-performance OPC communication via Web Services

With the release of the *OPC XML-DA* specification in 2003, the OPC Foundation for the first time showed a way out of the dependency on Windows platforms and the limitations caused by DCOM. Today, many *OPC XML-DA* products demonstrate the possibilities of Web Services based OPC technology. In building automation, energy technology, process technology and other industries, XML-DA products are used in Unix computers or Linux servers, or embedded in field devices under Linux or other embedded operating systems. A wrapper that maps the XML-DA data and services to DCOM DA, and vice versa, allows smooth interoperability of XML-DA products with DCOM OPC products. This way, XML-DA offers the possibility of platform-neutral integration concepts with OPC. To transport the data, XML-DA uses the SOAP XML protocol. OPC communication is based on exchanging HTTP messages with OPC data encoded in text format. As a result, reading and writing OPC data is time consuming because the HTTP messages first need to be created and the OPC data converted to text format, and

then the HTTP messages need to be unpacked and converted back to the original OPC data format. The data throughput of XML-DA communication is slower by a factor between five and seven compared to that of DCOM DA communication. This performance is significantly too slow for many automation tasks. The possibilities offered by Web Services based OPC communication are promising, but a much higher data transfer performance has to be achieved.

6. Unified data model

One of the market's demands on the OPC Foundation came from the process and building automation industries, in particular. It concerned a better integration of alarms and historical data in the address space of a *Data Access Server*. Until now, it takes three different OPC servers – *Data Access*, *Alarms & Events* and *Historical Data Access* – to acquire, for example, the current value of a temperature sensor, the event of the temperature exceeding a preset limit and the historical average of the temperature. DA, AE and HDA also have very different object models. Though it is possible to implement all three object models in one OPC application, there are great differences in the way the values of a DA, AE or HDA object are each accessed. This makes it very time consuming for users to access process data, event and historical data in such different ways. Unifying the three object models would make things a lot simpler not only for the OPC product vendors, but also for system integrators and users.

7. Support of complex data structures

Right from the early days of OPC technology, the OPC Foundation has continually been asked to support structured data types in addition to simple data types. One of the main applications of OPC is the operation and monitoring of devices that are networked through serial communication protocols or fieldbuses. Simple data types, such as byte, integer or real, or arrays thereof, are absolutely sufficient for reading process data and state information and for writing individual operating parameters. To configure devices, however, data types are needed that allow an OPC client to write complex data structures, including the meanings of the data structure elements, to a device via an OPC server. The structure of this configuration data depends on the device type and vendor. Many fieldbus organizations, including PROFIBUS International, Fieldbus FOUNDATION, CAN in Automation, etc., have defined proprietary device description formats to provide a standardized way to configure devices. With the *Complex Data Specification* (see section 2.2.8), the OPC Foundation has created a possibility to describe complex data structures, to distinguish the individual components of these structures by their (simple) data types and to represent the interrelations between the individual components. For example, the specification allows exposing the device descriptions of fieldbus organizations. However, the vast majority of OPC products on the market today has not implemented the *Complex Data Specification*, apart from very few exceptions. The reason is probably that this specification was defined very late, i.e. several

years after the introduction of the *Data Access Specification*, and that thousands of OPC products had already been installed by the time the specification was released. Consequently, the need for a real support of complex data structures and a possibility to add further descriptions to a data point has remained unfulfilled.

8. Process data communication without data loss

Data Access was originally defined to cyclically inform client applications of the current state of process data. Users configure the frequency of the cyclic update process by setting the *UpdateRate*. If the process data changes more frequently than (can be) transferred from the server to the OPC client according to the configured *UpdateRate*, information will be lost. If disturbances occur in the physical communication link between an OPC client and a remote OPC server, the communication will be broken according to the *Data Access Specification*. Communication between client and server can only be resumed by completely re-establishing the OPC connection after the physical link has been re-connected. Data changes that have occurred while communication was broken could not be transferred to the OPC client and were lost. This data loss is not critical with most *Data Access* projects, such as trend recording, process monitoring or process visualization. But OPC has increasingly penetrated application areas where requirements are more critical. For example, OPC technology has become established in areas such as the chemical or pharmaceutical industries, where data must be seamlessly recorded. What made this possible is that vendors have implemented specific extensions. They are based on connection monitoring systems that ensure a fast detection of broken communication, automatic reconnection if communication breaks, data buffering in Data Access servers, redundancy, and store & forward concepts. Useful as these extensions are, they have not been defined in the Classic OPC specifications and vary from vendor to vendor. There is a great demand for a standardized, interoperable OPC definition.

9. Increased protection against unauthorized data access

As a result of the growing trend towards Ethernet based communication in automation, the automation and office networks are intertwining. On the one hand, this opens up new possibilities of vertical integration. The data at the process level can be provided by an OPC server and represented in MS Excel using an OPC client plug-in or archived in a database by an OPC client application, without a need for additional cabling. On the other hand, this type of integration concept involves new security risks. If no special precautions are taken, an installation might not be safe from unauthorized access or data manipulation. OPC is also increasingly used in remote maintenance and remote control concepts. Here, again, more stringent requirements must be met regarding the security of the installation from unauthorized access from the outside. With rising cybercrime, spying and sabotage, IT security is growing more and more important – and so are the requirements

on security when using OPC. Without the proprietary precautions developed by vendors, Classic OPC cannot meet these security requirements.

10. Support of method calls

In many applications, it is not only the reading and writing of values that is important, but also the execution of commands, such as starting or stopping a drive or downloading a file to a device. The *OPC Commands Specification* (see section 2.2.7) defines possibilities to execute commands and call methods, e.g. starting or stopping a drive, executing a program, etc. The *OPC Commands Specification* is available as a draft version and was not completed before work on the *OPC Unified Architecture Specification* started. Therefore, there continues to be a need for method calls over the OPC interface and this has been added as a requirement for the new *OPC UA Specification*.

2.3.1.3 Origin, development and objectives of OPC UA

Considerations to build a new OPC architecture were embraced as early as 2003, as the *Alarms&Events* working group of the OPC Foundation was developing the next generation of the AE specification and its migration to Web Services. These considerations led to the formation of a completely new working group in late 2003. The primary goal of this working group initially was to convert the access to process data (*Data Access*), alarms and events (*Alarms&Events*) and historical data (*Historical Data Access*) to Web Services, and standardize the way they are accessed. The *Unified Architecture* – in short, OPC UA – was born. For over five years employees from 30 companies – several of them market leaders in their respective industries – worked countless hours to develop the new OPC architecture under the auspices of the OPC Foundation. Besides converting Classic OPC to Web Services and unifying DA, AE and HDA, many additional new requirements were also placed on the new OPC UA. They were added by the OPC Foundation after conducting market surveys and consulting a vast number of OPC users, system integrators and vendors. Based on the findings, the OPC Foundation defined the following guidelines and key objectives:

- **Keep it simple**
  To make UA technology easy to use through UA components, despite the multitude of functional requirements and complexity.
- **"Evolution" instead of "revolution"**
  To maintain the terminology, object models and essential communication principles of Classic OPC; to protect investments in the development of Classic OPC products by ensuring their continued usability.
- **Platform independence and scalability**
  To supersede DCOM as the technology base with a service oriented architecture (SOA) to allow using OPC technology at the IT level and in embedded systems