OPC UA – Service-oriented Architecture for Industrial Applications

Stefan-Helmut Leitner, Wolfgang Mahnke

ABB Corporate Research Center
Wallstadter Str. 59, 68526 Ladenburg, Germany
{stefan.leitner|wolfgang.mahnke}@de.abb.com

Abstract: OPC Unified Architecture (OPC UA) is the new standard of the OPC Foundation providing interoperability in process automation and beyond. By defining abstract services, OPC UA provides a service-oriented architecture (SOA) for industrial applications – from factory floor devices to enterprise applications. OPC UA integrates the different flavors of the former OPC specifications into a unified address space accessible with a single set of services. This paper gives an overview over the architecture of OPC UA, its address space model and its services. It discusses the necessary security mechanisms needed to allow secure access over the internet. Finally, migration strategies to OPC UA applications are introduced.

1. Introduction

The OPC Foundation has released a set of standards widely accepted in industry to provide interoperability in industrial automation. OPC DA [DA] allows accessing current data, OPC HDA [HDA] accessing historical data and OPC A&E [AE] accessing alarms and events.

Several reasons motivated the OPC Foundation to develop its new OPC UA [UA1] specification:

Unified data access

While the previous specifications of the OPC Foundation served their purposes they where not connected, i.e. there was no connection between an actual value read with DA to the history read with HDA or events raised based on the value. OPC UA provides all data in its unified address space. Thus current data, historical data and events are related to each other.

Additional requirements

OPC UA supports a set of new features, like accessing historical events, multiple hierarchies and providing methods and programs (also often called commands). A big new achievement of OPC UA is a higher-level data model beyond simple data type information. Whereas the old specification only provided a single hierarchy with items containing data, OPC UA provides an extensible meta model where those items are typed. In addition to providing an item with the data type Float and some meta information like engineering unit, OPC UA allows typing the item, so it can for example be identified as a heat sensor. Section 3 gives more details about the meta model of OPC UA. Finally, equipped with a powerful and extensible type model OPC UA allows adding and deleting items and references between them.

Technology migration

The old OPC Foundation standards base on Microsoft's COM/DCOM technology [BK98]. Microsoft already deemphasized COM/DCOM in favor of cross-platform capable Web Services and SOA. In addition, vendors demand a platform-independent specification that allows running OPC applications on non-Microsoft systems. OPC UA supports this by specifying an abstract set of services and maps them to different technologies like Web Services. Section 4 gives more details about the architecture of OPC UA.

Additional Areas of Applications

Whereas the typical application of the old OPC Foundation specifications is either to bring data into a DCS or to access data of a DCS OPC UA provides a single, interoperable way accessing data from the factory floor to the enterprise. With its capability to map the services to different technologies, OPC UA fulfils the requirements accessing devices in a performing way as well as providing pure Web Services for MES and ERP systems that only allow a generic Web Service integration. Section 2 explains the different technology mappings of OPC UA in more detail.

The OPC UA specification is broken into several parts. [UA1] gives an overview and [UA2] explains the security model. [UA4] defines the abstract services, [UA3] the address space model and [UA5] the information model of OPC UA. [UA6] defines the mapping of the abstract services to a concrete technology. These parts represent the basic of the OPC UA specification. Whereas [UA7] specifies different profiles for OPC UA clients and servers, Part 8 to 11 covers specializations for data access [UA8], historical access [UA11], alarms and conditions [UA9] and programs [UA10]. Currently only Part 1 to 5 and 8 are released, the other parts are announced to be released in 2006.

In the following section, the architecture of OPC UA is introduced, followed by a description of the meta model and an overview of the services. Section 5 describes the security concepts of OPC UA – an important feature when applications start accessing device data over the internet. Afterwards section 6 discusses migration strategies to OPC UA applications and section 7 concludes this paper.

2. Architecture

OPC UA specifies an abstract set of services in [UA4] and the mapping to a concrete technology in [UA6]. OPC UA does not specify an API but only the message formats for data exchanged on the wire. A *communication stack* is used on client- and server-side to encode and decode message requests and responses. Different communication stacks can work together as long as they use the same technology mapping. The following subsection gives an overview over the different components of OPC UA clients and servers followed by a subsection of technology mappings. Subsection 2.3 explains possible interactions of OPA UA servers.

2.1. Client and Server components

An OPC UA client consists of a Client Implementation using an OPC UA communication stack. The Client Implementation accesses the communication stack using the OPC UA API. Note that the API is not standardized. It may vary for different programming languages and potentially for different communication stacks. Several communication stacks may exist for different operating systems, programming languages and mappings. For example, there may be a communication stack for Java and a communication stack for Microsoft's new Windows Communication Foundation (WCF) [MS06]. The client-side communication stack allows the client to create request messages based on the service definitions. The client-side communication stack communicates with a server-side communication stack. The OPC Foundation standardized only this communication. Thus, everybody can develop his or her own communication stack with its own API as well.

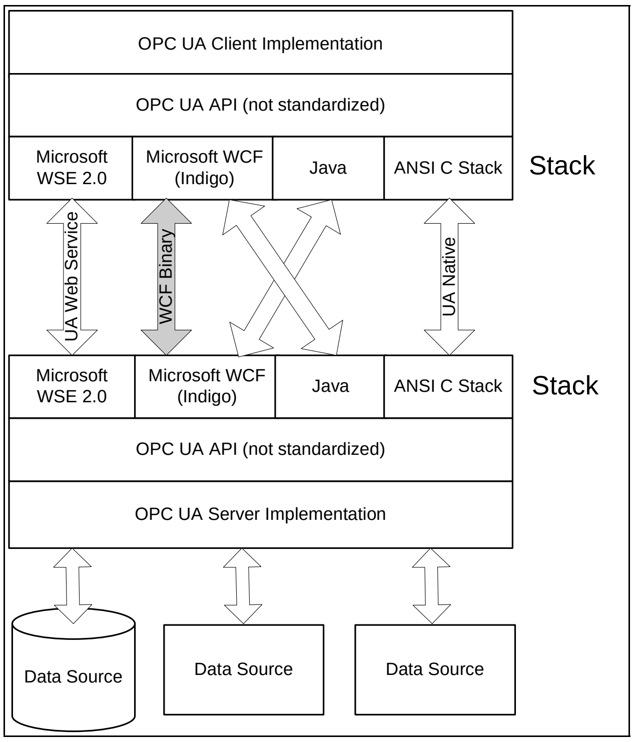

Figure 1: OPC UA Architecture

The server-side communication stack delivers the request messages to the Server Implementation via the OPC UA API. Since the OPC UA API realizes the abstract service specifications, it may be the same as on the client-side. The Server Implementation implements the logic needed to return the appropriate response message. The OPC UA Sever Implementation gets its data from some underlying system. For example, this can be a configuration database, a set of devices or some OPC server.

Figure 1 illustrates the architecture of OPC UA.

2.2. Technology Mapping

[UA6] currently defines two mappings: *UA Web Services* and *UA Native*. The first mapping uses SOAP and the various WS-* specifications (see [UA6] for details). The second mapping uses only a simple binary network protocol and integrates TLS-like security mechanisms.

The encoding of the data can be done in XML or *UA Binary*. UA Binary specifies the serialization of data into a byte string. The UA Binary encoding is faster than the XML encoding since the message size is smaller than for the XML encoding. On the other hand, the XML encoding allows generic SOAP-clients to interpret the data in the SOAP message, while they would only get a binary string using the UA Binary encoding. In theory, the encoding of the data is independent of the mapping. However, the XML encoding will typically only be used in the UA Web Service mapping in combination with the various WS*-specifications.

The protocol of the UA Web Service mapping is SOAP/HTTP(S) while the UA Native mapping typically runs directly on TCP/IP. For bypassing firewalls, the UA Native mapping also allows putting the binary encoded messages into SOAP messages using HTTP(S).

Of course, UA messages can also be encoded and transported with other protocols, for example using WCF Binary as shown in Figure 1. However, by using this technology you are losing interoperability since you only can talk to WCF clients as long as you do not provide another mapping. Since the described architecture separates the Client and Server Implementation from the communication stack, this is easy to realize.

2.3. Aggregating Servers

Build-into the OPC UA specification is the concept of aggregating servers. An aggregating server aggregates one or more OPC UA server and provides the information of those servers – or an excerpt of the information – in its address space. Thus, a client does not have to access several servers but only one server. This mechanism allows a flexible architecture by chaining several OPC UA servers for different clients with different requirements. For example, several OPC UA servers running on small devices will be aggregated by one OPC UA server. Several clients of the DCS system may access this server. Another OPC UA server aggregates this server and provides part of the information to the MES system. The MES system could work as an aggregating server, too. OPC UA supports aggregating servers by allowing to mark the origin of data.

3. Meta Model

The address space model defined in [UA3] is the meta model of OPC UA. The base concept of the meta model is a *node*. Several *node classes* are defined specializing the base node class (see Figure 2). Each node has a fixed set of *attributes* depending on the node class. Some attributes are mandatory and some are optional. For example, each node class has a node id uniquely identifying the node while the description attribute is optional.

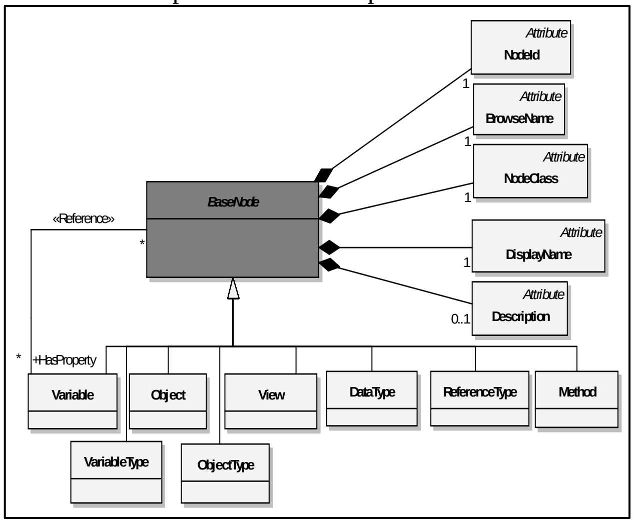

Figure 2: OPC UA Meta Model

Relationships between nodes are realized by *references*. References are no nodes and do not contain any attributes, thus they are a very simple construct. However, each reference is associated to a *reference type*. Although the meta model already defines a reference type hierarchy and uses those references as inherent part of the meta model (e.g. for defining a type hierarchy), the reference hierarchy is extensible.

The specializations of the base node class represent different concepts of the meta model. An *object* is a simple node that is typed by an *object type*. The attributes of the node only contain data describing the object. However, objects are used to represent real-world objects, software objects, etc. These data are stored in several *variables* referenced by the object. A variable has a special attribute called value containing the data. Like objects, variables have types, called *variable types*. Unlike all other attributes, the value attribute has no data type assigned to it. The data type may differ for different variables and therefore each variable points to a *data type* node representing a data type. Data types are extensible, i.e. each server can define additional data types.

Method nodes represent methods in the address space. They contain information how to call the method (input parameters) and what will be returned (output parameters).

View nodes represent an excerpt of the address space. A view typically restricts the data to the needs of a special user group or task and hides unnecessary data. Clients can browse through the address space in the context of a view.

The OPC UA meta model allows to define an information model by defining object, variable and data types as well as reference types. The specification already defines the base information model in [UA5] already containing several base types. Vendors can extend this model to create there own information model. It is expected that other standards like EDDL [EDDL] and FDT [FDT] will define companions specifications to define their domain specific information model accessible via OPC UA.

4. Services

OPC UA groups their services into *service sets*. An OPC UA service is defined by its request and response messages, thus, it is on the same level called operation in [WSDL]. OPC UA defines 34 services – we will only give an overview of the service sets.

SecureChannel Service Set

This service set defines security-related services that are handled by the communication stack that OPC applications use. These services are required to guarantee a secure communication between client and server. Therefore, services to establish a Secure Channel between two communication partners are defined. Establishing a Secure Channel requires knowing which security mechanisms to use for communication. This information can be retrieved by a further service, which is also defined in this service set.

Session Service Set

The Session service set specifies security-related services handled by the OPC UA application directly, like establishing a session on behalf of a specific system user between client and server. An additional service allows also changing the identity of a user of an active session.

NodeManagement Service Set

The NodeManagement service set allows adding and deleting nodes and references in the address space.

View Service Set

The View service set contains services for browsing the address space. The browsing is done in the context of a view; the default view contains the whole address space.

Query Service Set

The Query service set is build to query the address space. Like browsing querying is done in the context of a view. Queries always access a snapshot of the server, i.e. each value is only provided once (no history), but the query allows to specify the time of the snapshot. Thus, the query can access one point in time of the history of the OPC UA server.

Attribute Service Set

The Attribute service set allows reading and writing attributes, including the value attribute. It also allows accessing and updating the history of attributes and events.

Method Service Set

The purpose of the Method service set is calling methods.

Subscription Service Set

This service set must be used to subscribe to data. Services exist to manage subscriptions and to receive data from the subscription.

MonitoredItem Service Set

The MonitoredItem service set allows specifying which data should be returned for a subscription. The services allow specifying a deadband and update rates for attributes as well as filters for events.

5. Security

5.1. Background

Security has been a widespread and often used term in the field of *Information Technoloy* (IT) for many years and is still a very important field within this context. Because today's IT systems mostly work together with different kind of systems and technologies, security becomes more important in other areas, too.

Especially in the *Automation Technologies* (AT) security has grown to a high-priority issue since corporate and automation networks as well as applications are merging together. An excellent example for that are *Manufacturing Execution Systems* (MES) and *Enterprise Resource Planning* (ERP) software products used to control the manufacturing process in a plant and to adjust a company's resources. A security flaw in such kind of critical systems could bring up disastrous financial and environmental impacts.

Different goals, threats and measures have to be taken into account when providing security in and for applications. The following sections briefly describe how to approach.

5.2. Security Assessment

As a first step, the general approach is to arrange a *Security Assessment*: Thereby security experts, system experts, domain experts and user of the target system come together to define security goals, point out threats and risks and introduce counter-measures after analysing the target system and the environment where it is runnig in. The result of a *Security Assessment* is in most cases a document where all the above mentioned issues are written down. In addition to that, different decisions and actions for the next steps are defined, too.

[UA2] specifies such a result of a *Security Assessment* in the context of automation systems, where *OPC UA* is used. Thereby six common security goals (authentication, authorization, confidentiality, integrity, auditability and availability) are shortly introduced and mapped to the needs of automation systems as a kind of industrial application. Additionally possible threats that could occur in OPC UA environments are described and thereby shown, which security goal would be compromised. It has to be taken into consideration that there is no one-to-one relation between goals and threats. A threat could also compromise more than just one goals and a security goal could also be compromised by more than one threat. The list of threats cannot be complete since such environments consist of many different applications and subsystems that have also interdependencies between them. Each of them could have different security flaws and some might be even unknown. This means, that a *Security Assessment* is not an approach that is processed one time and never again. This should happen regularly to be always up-to-date and to be able to maintain and improve a system's security.

The second step would be to apply the counter-measures and process the defined actions. Because industrial applications are complex systems managed by various organizational processes and controlled by different persons, various counter-measures and actions have to be applied at different locations and parts of a system. The following two sections describe some threats and how the security-related actions and counter-measures defined by OPC UA are applied in system's infrastructures and in the application's architecture.

5.3. Secure Infrastructures

It is common that the infrastructure and the environment of applications are very important security-related issues in a system. However, it gets even more important, if a *Service Oriented Architecture* (SOA) – based on Web Services – has been chosen as application-architecture. Several security-related issues that come with the service orientation:

Web-service-based SOA implementations using SOAP offer different mechanisms for discovering servers. Because SOA implementations also use standard HTTP ports for communication and in some scenarios the plant floor networks are merged together with the corporate networks, an attacker from outside could pass many firewalls easily.

Another scenario is that network traffic be recorded and identified messages could be re-sent (message replay) without modification to server or clients by a *Man-In-The-Middle*. Thereby an attacker could resent a *CloseSession* service call and misinform an OPC UA server, which obviously closes the session. Formally, this attack compromises *Authorization* as security goal.

A further possible attack is to capture network traffic, then alter sent messages after identifying them. Hereby an attacker has to know the message structure, which is not very difficult since OPC UA is standardized and published. This could allow the attacker to get illegitimate access to a server. Formally, this attack compromises *Integrity* and *Authorization* as security goals.

The above mentioned attacks can be mitigated in different ways. The first measure to do is to secure the network environment. [UA2] describes the relationship to site security defined in some other common standards (see [IEC]) for network security and references the Defense-In-Depth strategy as a basic security mechanism. Thereby the network of a company is divided into separate zones: corporate network, operations network and plant floor network. Each of this zones have own security policies and restricted user access and allows only the needed protocols and ports.

To mitigate the first scenario (message replay) OPC UA specifies message fields for *SessionID*, *TimeStamp* and *SequenceNumber*. These fields have to be verified for every message, so that no received message can be sent twice.

The second scenario can be mitigated by signing messages before sending them. This signature is verified for every message that is received. If a message was altered then the signature verification would fail. It is common to use a *Public Key Cryptography* (PKC) for signing together with a *Public Key Infrastructure* (PKI). A private key is used to sign a message and the appropriate public key is used to verify the signature. Therefore, the public key has to be provided to all communication participants. This can be done by providing them with certificates offered by a certificate server. A PKI consists of several entities that offer functionality to create (certificate and registration authority), validate (validation service) and provide (certificate provider) digital certificates.

Figure 3 shows a simplified view of how a possible network environment with a PKI could look like.

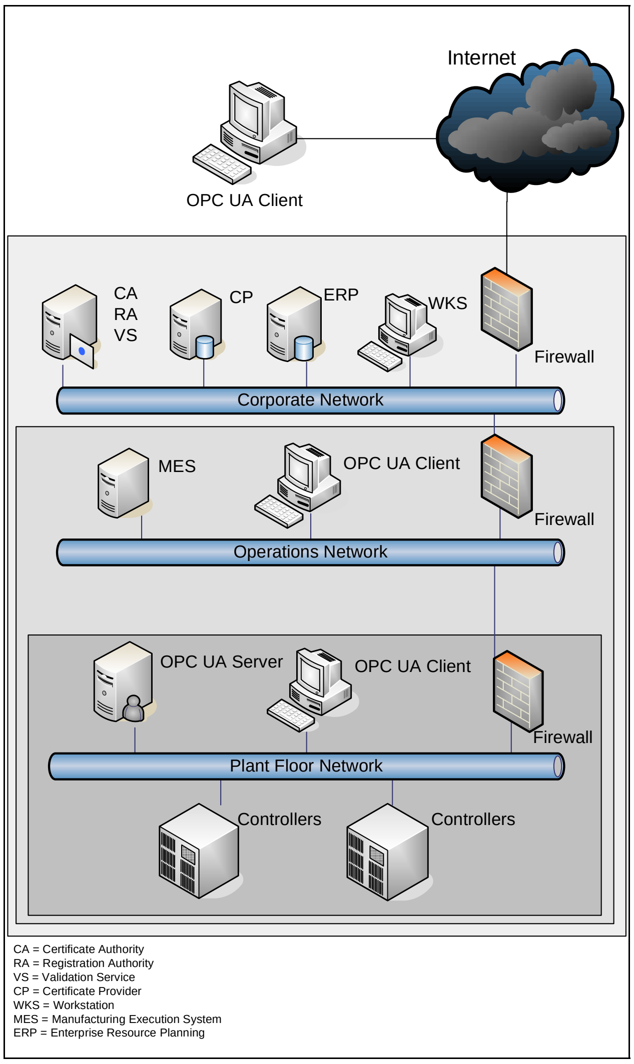

Figure 3: OPC UA Network Environment

5.4. Secure Architectures

Securing an application’s infrastructure is not enough. The application itself has to be secured, too.

Currently many applications have problems with *(remote) buffer overflow* attacks whereby an attacker could execute malicious code. Poor memory handling and the development of big monolithic application are often reasons for that. Using Web Services even increases the probability that such attacks occur, since many of them can be consumed directly or indirectly from outside (internet).

Another problem is that used libraries could have security flaws and even cryptographic algorithms could get unsecure since computing power increases continuously.

An important step to remain secure is to consider security during design and implementation phase when developing applications.

A clear and well-structured application architecture is a good starting point. Thereby different separate layers with strictly defined responsibilities have to be designed. Memory- and security-related measures can be assigned for each separate layer. By applying this approach problems with buffer overflows can be reduced. OPC UA specifies a layered architecture and assigns specific security-related functionality in different layers:

The Application Layer is responsible for User Authorization and Authentication, the Communication Layer verifies and applies Application Authentication and finally the Transport Layer maintains Confidentiality and Integrity as security goals.

This layered architecture is depicted in Figure 4.

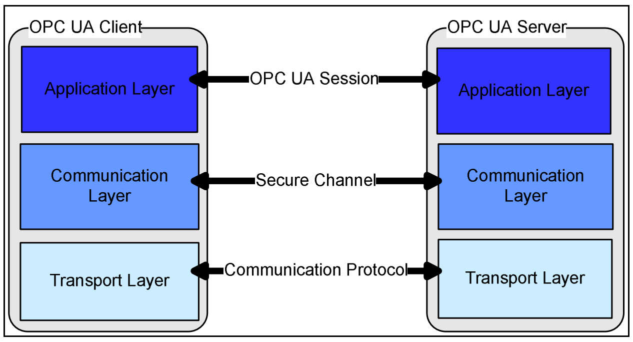

Figure 4: Security Architecture

Security has to be maintained in the future and therefore it is important to check always whether libraries or cryptographic algorithms are still secure. It is even more important to be able to update or change applications with appropriate effort. This implies a modular design of the different layers, so that modules can be replaced without redesigning the whole application.

6. Migration – Wrapping vs. Direct Implementation

An important topic is how to migrate old COM-based-OPC server to the new Web-Service-based Unified Architecture technology.

Current COM-based OPC servers typically use a proprietary interface to access the data of the devices in the control network. There are basically two approaches that can be taken into account: wrapping existing servers or directly accessing device data.

Wrapping existing servers may shorten development times for implementing OPC UA servers, since existing OPC servers can be used. It is only necessary to build a Web-Service-based wrapper on top of the different OPC COM servers. This wrapper exposes the different functionalities of the underlying COM servers in form of Web Services. This approach is shown in Figure 5.

However, there are several drawbacks coming with that solution:

Choosing the wrapper approach implies a heterogeneous server application environment, which is harder to maintain and manage. Especially if specific system components or frameworks are updated then the interdependencies of the Web-Service- and COM-Layer have to be checked again.

From the security point of view, wrapping is not always the best solution. Web Services as well as COM components have different security-related issues to consider. Both technologies used together in one application offer an attacker more possibilities to exploit security flaws.

The other approach is to develop an UA server that directly uses a proprietary interface to access the devices of the control network. Web Services expose thereby the complete functionality. Figure 5 shows this approach.

The main drawback of this solution is that the development time increases since all needed functionality has to be reimplemented.

On the other hand a direct implementation leads to a simple and homogeneous application environment, that is easier to maintain and manage. There are fewer interdependencies than in the wrapping approach and therefore updates of components and frameworks do not entail checking whether interoperability of different technological layers can still be assured.

From the security perspective, the amount of possible security considerations decreases compared to the wrapper approach, but there are still security issues to consider.

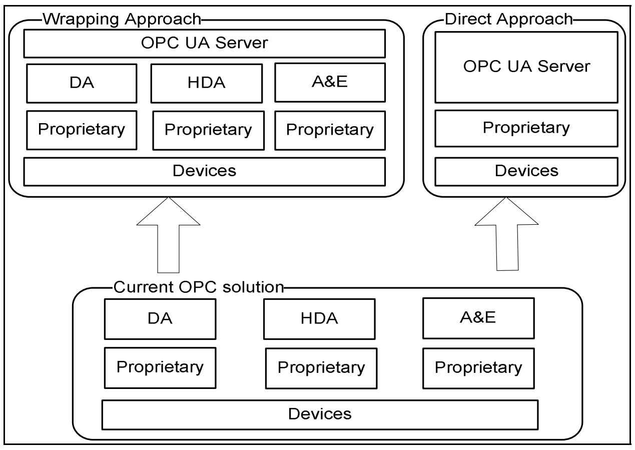

Figure 5: Migration Strategies

7. Conclusion

OPC UA is an important step to integrate new technologies and concepts into industrial applications. The strong meta model, the open and standardized service-oriented architecture and the ability of unifying different kind of OPC servers opens the door for new areas of application. Especially in the field of process automation, it will help to simplify the integration of other applications like MES, HMI or SCADA systems.

Other standards, like FDT [FDT] and EDDL [EDDL], will take advantage of OPC UA and define companion specifications for their domain-specific information model accessible via OPC UA. Since many companies participated in developing the OPC UA specification it is expected that soon there will be many products available supporting OPC UA.

8. Literature:

[AE] OPC Foundation: *Alarms and Events Custom Interface Standard.* Version 1.10, Oct. 02, 2002

[BK98] Brown, N., Kindel, C.: Distributed Component Object Model Protocol - DCOM/1.0. Microsoft Corp., 1998

[DA] OPC Foundation: *Data Access Custom Interface Standard.* Version 3.00, March 04, 2003

[EDDL] IEC 61804-3 Ed. 1.0 – Function blocks (FB) for process control - Part 3: Electronic Device Description Language (EDDL)

[FDT] FDT Joint Interest Group: *FDT Interface Specification,* Version 1.2, May 2001

[HDA] OPC Foundation: *OPC Historical Data Access Specification.* Version 1.20, Dec. 10, 2003

[MS06] Microsoft Corporation: *Windows Communication Foundation*, September 2006, http://msdn.microsoft.com/winfx/technologies/communication/default.aspx

[UA1] OPC Foundation: *OPC UA Specification: Part 1 – Concepts.* Version 1.00, July 28, 2006

[UA2] OPC Foundation: *OPC UA Specification: Part 2 – Security Model*. Version 1.00, July 28, 2006

[UA3] OPC Foundation: *OPC UA Specification: Part 3 – Address Space Model*. Version 1.00, July 28, 2006

[UA4] OPC Foundation: *OPC UA Specification: Part 4 – Services*. Version 1.00, July 28, 2006

[UA5] OPC Foundation: *OPC UA Specification: Part 5 – Information Model*. Version 1.00, July 28, 2006

[UA6] OPC Foundation: *OPC UA Specification: Part 6 – Mapping*. Release Candidate 0.93, June 01, 2006

[UA7] OPC Foundation: *OPC UA Specification: Part 7 – Profiles*. Draft 0.93, July 28, 2006

[UA8] OPC Foundation: *OPC UA Specification: Part 8 – Data Access*. Version 1.00, Sep. 25, 2006

[UA9] OPC Foundation: *OPC UA Specification: Part 9 – Alarms and Conditions*. Draft 0.75, June 28, 2006

[UA10] OPC Foundation: *OPC UA Specification: Part 10 – Programs*. Draft 0.4, Sep. 01, 2006

[UA11] OPC Foundation: *OPC UA Specification: Part 11 – Historical Access.* Draft 0.992, Aug. 22, 2006

[WSDL] W3C: *Web Services Description Language (WSDL) 1.1*. March 15, 2001

[IEC] International Electrical Commision: *IEC 62351: Data and Communication Security*, Jan. 05, 2006