Introduction

Over the last decade the OPC Foundation’s OPC standard for process control and automation has achieved wide adoption throughout industry. The standard specifies an API to decouple process control clients and servers, i.e. an OPC compliant client can interact with any OPC compliant server via a standard API without needing to know the specifics of the server itself (i.e. without needing to know details by which the server read/writes data - accessing underlying hardware for example). The OPC standard is actually a family of standards and is based on Microsoft COM/DCOM.

OPC-UA (UA for Unified Architecture) is the next generation OPC specification. OPC-UA is not an incremental change to the existing family of OPC specifications but a brand new standard representing a profound change. OPC-UA is not intended to be directly backwards compatible with classic OPC (though some inter-operability options exist). OPC-UA provides a complete, modern, secure and platform independent standard for industrial process control.

Headline Benefits of OPC-UA

Platform Independence: COM and DCOM are being relegated to legacy status, Microsoft has de-emphasized COM/DCOM as a means for inter-process communication and instead promotes a service orientated approach and cross platform web services. 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: An embedded OPC-UA server need only implement profiles the client environment requires of it.

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 a Public Key Infrastructure to provide secure channel client/server communication and a means for user authorization and authentication.

Improved Modeling: The OPC-UA standard provides an extensive vocabulary for modeling the devices and processes under control, 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).

Enterprise Level Data Publishing: OPC-UA Servers can publish data via standard SOAP web services with a high level of security. Using this method, disparate OPC-UA servers can securely interchange status information with one another through firewalls. Using a widely adopted standard like SOAP web services over HTTP also allows non OPC-UA specific clients to consume output published by an OPC-UA server.

Service Discovery and Session Initiation

For Classic OPC systems the process of a client discovering a server was a windows registry based operation. OPC-UA, however, has been designed to be platform independent and therefore requires a platform independent approach to server discovery. OPC-UA servers require to be known by a discovery server. Discovery servers reside at a fixed and known address on the network and, when interrogated by OPC-UA clients, yield the connection details of the OPC-UA servers registered with them. Note: Discovery servers themselves need not be full featured OPC-UA servers – they only need comply with a very small subset of the OPC-UA specification related to providing OPC-UA server information. Therefore a standards compliant discovery server can be simply implemented.

OPC-UA servers communicate with clients only via exposed endpoints. Endpoints fall into 2 categories: discovery endpoints and session endpoints. OPC-UA servers expose exactly one discovery endpoint – querying this endpoint yields connection details of the server’s session endpoints. Clients connect to session endpoints to interact with the device or process managed by the server.

The connection information yielded by a discovery server is the URLs of the discovery endpoints exposed by the OPC-UA servers it knows about. Clients choose a server and interrogate its discovery endpoint to ascertain the URLs of the session endpoints and the details of how to communicate with each session endpoint. Note a single server instance can expose many different session endpoints, each with its own combination of security configuration and message format and transport protocol.

The diagram below shows the steps required for an OPC-UA client to discover the OPC-UA server location, discover the server’s endpoints and their security/transport options and start a secure session with a selected session endpoint.
Note that the client can try a direct connection using a cached endpoint description from a previous browse of the OPC-UA server discovery. So long as the endpoint has not changed its configuration this should succeed. If the connection fails the client should start the process from the top.

Security

The OPC-UA is a platform independent standard and relies on cross platform security measures, this is a departure from classic OPC which delegates security to COM/DCOM. OPC-UA security is based on a Public Key Infrastructure (PKI) using industry standard x.509 digital certificates and addresses authentication, authorization, encryption and data integrity.

Below is a diagram from the specification on security giving a high level view of how security is managed in OPC-UA. Note that user authentication and authorization are left to the application layer. Secure channel communication, however, is part of the specification.
**Authentication:** OPC-UA application instances are uniquely identified by their x509 certificates, a session can only be created between two OPC-UA applications if each trusts the other's certificate. A client, for example, cannot initiate a session with a server providing a certificate that the client does not trust. Trust can either be established

1. Directly- Each application has the other's (public) certificate copied directly into some trust folder which the application inspects.
2. Via a ‘tree of trust ’ whereby an application trusts the authority which granted the other application's certificate, thus by extension the application's certificate.

**Authorization:** An endpoint's description defines the user identity token a client application must supply. The OPC-UA specification describes four types of user id an endpoint can demand:

1. Anonymous.
2. User name and password.
3. x509 certificate (note this is distinct from the application instance's x509 certificate - this is a certificate that has been issued to the user).
4. WS-Security compliant tokens (Kerberos tickets, widely used for single sign on, have a WSS token profile).

Note that the specification only describes the permitted types of user tokens and the means by which they are exchanged. For example: Restricting access to the address space based on a user's identity token is application specific — this must be implemented by the OPC-UA server vendor.

**Data Encryption and Data Integrity:** Data encryption is intended to prevent a 3rd party reading messages passed between client and server (snooping on a network with a packet sniffer for example). OPC-UA uses public/private key encryption. Data integrity measures prevent a 3rd party tampering with messages passed between client and server (injecting malicious content to the message for example).

OPC-UA defines a set of common internet standard measures that clients and servers must implement for scrambling the data (encryption) and signing messages (data integrity) in order to guarantee a secure channel. Each session endpoint informs clients of the measures it will use through its description published by the discovery endpoint. The client and server then apply these measures on their respective sides to every message passed within the confines of a secure session.

The options for encryption and signing are:

- None - no encryption, security is turned off. Messages can be read by a 3rd party and tampered with.
- Signed - messages are signed to ensure data integrity but the message body is unencrypted. Messages can be read by a 3rd party.
- Sign and encrypt - as above but with the message body encrypted. Secure, messages are private and their integrity is assured.

The options for encryption method are

- None – no encryption (no security)
- Basic128Rsa15 – 128 bit encryption (fairly secure)
- Basic256 – 256 bit encryption (very secure)

The diagram below shows the client/server process of setting up a secure session between an OPC-UA client and server.
Client Server Communications

A significant improvement OPC-UA offers over classic OPC is platform independence. The specification provides for various platform independent and standards based combinations for inter process client/server communication. The OPC-UA supports two message formats and two transport protocols for inter process communications.

Message Formats

OPC-UA supports two message formats: UA Binary and XML. The format defines how the message data is encoded. The sender of a message must encode the data into the relevant format for transmission and the receiver must be able to decode the transmission contents to reconstruct the original data.
**UA Binary:** This message format encodes the data serialized into a byte array. UA Binary offers reduced computational cost in terms of encoding and decoding but can only be interpreted by OPC-UA compliant clients. UA Binary is more likely to be used in device level communications where processing power is limited and performance is a high priority.

**XML:** XML documents are the ubiquitous method of high level data exchange. XML encoded messages can be interpreted by OPC-UA clients and also by generic clients using the XML schema contract (generic clients being clients with no intrinsic knowledge of OPC-UA). Serializing and deserializing data into XML format is computationally more expensive than the UA Binary format and so XML encoding is more likely to be used towards the enterprise end of the communication spectrum.

### Transport Protocols

OPC-UA supports two transport protocols: OPC TCP and SOAP/HTTP(S). The transmission protocol defines the means by which messages are passed between client and server.

**OPC TCP:** This is a TCP (sockets) based protocol providing a full duplex channel between client and server. Messages are packaged into a structure specified by the OPC TCP binary protocol and the structure is transmitted using a socket or secure socket (depending on the endpoint’s security requirements). As OPC TCP is specific to the OPC-UA specification only OPC-UA clients are capable of receiving data transmitted with OPC TCP.

**SOAP/HTTP(S):** Using this transmission protocol, messages are packaged into the body of a SOAP message and transmitted over HTTP(S). A SOAP message is an XML document, sometimes called an envelope, which conforms to the SOAP XML schema. The SOAP envelope has a ‘holder’ into which the message data is inserted. The entire envelope is then transmitted over HTTP or HTTPS (depending on the endpoint’s security requirements). SOAP/HTTP is an established industry standard and is widely used for enterprise level information exchange, a generic web service client could receive SOAP envelopes transmitted over HTTP/S.

### Sending Messages

In order to pass messages between OPC-UA clients and servers three pieces of information are required:

1. The message format.
2. The transport protocol.
3. The channel security measures.

In theory any possible combination is possible: Unencrypted XML messages could be sent over OPC TCP, encrypted and signed binary messages could be sent over HTTP. In practice however, the most likely format and transport pairings are as follows:

- UA Binary + OPC TCP – this is the leanest method of formatting and transmitting data and is therefore the most likely combination used at the device level. Only OPC-UA clients and OPC-UA servers can exchange information in this form.
- XML + SOAP/HTTP(S) – this combination is the most firewall friendly and most easily consumed from the perspective of generic clients and is therefore the most likely combination used at the enterprise level.

### Reading Device Data

The OPC-UA specification describes two services by which OPC-UA clients access device data via OPC-UA servers.

**Read Service (Poll):** This service provides a simple 'one-off' style read - effectively a poll. The parameters for this service include:

- A list of node IDs, indicating which nodes and which attributes of the nodes are to be read (for example the value attribute).
- The maximum age of the data the client expects (one value for all node IDs in the request). If the data corresponding to the node IDs in the server’s cache is too old to satisfy the stipulated maximum age, then the server should try to retrieve the data direct from the underlying device (Note: a maximum age of ‘0’ instructs the server to query the device). If the server is unable to get the device data directly then the server replies with its best effort.

This service is described in more detail in the OPC-UA specification Part 4, section 5.10.2

**Subscription Service (Publish Subscribe):** This service allows an OPC-UA client to subscribe to OPC-UA server data updates and request that it receive update notifications with a specified periodicity. This service must be used in conjunction with the MonitoredItem service set.

- **MonitoredItem:** Clients instruct servers to create monitored items, this tells the server to initiate a process (non specific) whereby data is gathered from the underlying device. Monitored items have a sampling interval, this is the rate at which the server should try (best effort) to gather device data, (0 means as fast as possible). On each sample cycle a monitored item will gather data from the device and create a notification,
Device Modeling with OPC-UA

Classic OPC has a fairly simplistic approach to modeling underlying devices and processes – data in a tree structure. OPC-UA provides a richer means for modeling the details and complexities of the actual devices and processes using the Object Oriented (OO) paradigm. An OPC-UA client has access to the server’s OPC-UA Address Space. This address space is populated with objects, objects are the core modeling component – an object specifies its variables, methods and events.

Variables: These hold the system data, variables can be sub-divided into 2 main types: Data and properties. Data variables are similar to classic OPC items, they contain the dynamic system data. Properties are used to hold meta information describing the parent object, for example a property might contain expected minimum and maximum values for the parent object’s system data or a description of what the object represents in the process or device. A property would usually be masked to be read-only. A client interacts directly with an object’s variables using read/write commands and can create subscriptions for notifications of variable changes.

Methods: A method on an object encapsulates a piece of functionality specific to the object’s role. Clients can interact with objects by invoking their methods to execute some task on the server side. An OPC-UA client can browse the methods on an object but the details of what a method does is implementation specific.

Events: An object can define the events it emits. An object created to represent some aspect of a device or process would emit events specific to that aspect, clients with an interest in some aspect of the underlying device/process would subscribe to the events emitted by objects that model it.

Objects form the basis of the OPC-UA address space but there are more items in the OPC-UA address space meta model, not all of which come under the scope of this document. Two more address space items merit mention however: Views and references. Both are essentially browsing aids for an OPC-UA client in that they add another layer of organization to the address space over and above the object model.

Views: For those who have worked with SQL views the concept is familiar, an OPC-UA view provides a way of organizing data (data is held within objects) to provide a browse space specific to the views intended purpose. For example, objects could be created modeling the physical components within a high voltage power supply system: the main frame unit, the powered crate controllers, the crates and the boards in the crates delivering a voltage to an end channel. The physical equipment modeled by the objects could be physically located in disparate areas. In this case a view could be created grouping the address space objects by physical location.

References: References provide a way to link logically connected objects together. An object can specify a reference to another object in the address space (or even an object in another server’s address space). A client can browse an object’s references and follow the reference to navigate an interconnected web of objects.

Pragmatic Considerations (Availability)

OPC-UA is a standard, not an implementation. Implementations are built in accordance with market demand. At time of writing (October 2010) there are 3 main OPC-UA implementation stacks available from the OPC Foundation: ANSI C/C++, Java and Microsoft .Net. Not all features of the OPC-UA standard have been implemented on every stack – the grid below shows the availability of key features on a per stack basis:

<table>
<thead>
<tr>
<th>Feature</th>
<th>ANSI C/C++</th>
<th>Java</th>
<th>Microsoft .Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA Binary formatting</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>XML formatting</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>OPC TCP transport</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SOAP/HTTP(S) transport</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

3rd party OPC-UA toolkits tend to be based on the OPC Foundation stacks. Toolkits provide a leaner programming interface than the raw base stacks offered by the OPC Foundation but they are unlikely to exhibit a feature set better than the base stack. As a result at time of writing ANSI C/C++ and Java based implementations do not have the capacity for XML encoding messages or transport over SOAP/HTTP(S). This is only available using the .Net stack, which is bound to Windows.

Summary
The OPC-UA standard provides a comprehensive and secure way to build and operate multi-vendor clients and servers. In addition the OPC Foundation, the body in charge of the OPC-UA specification, has accumulated years of pragmatic automation and process control experience from classic OPC. OPC-UAs platform independence is a major advantage over classic OPC but comes at the cost of increased complexity in the clients and servers as they must support the specified message formats, transports and security measures.

If you have any comments or questions please contact Ben Farnham (BE/ICS group).