

# **Considerations for Data Collection from Serial & Ethernet Devices**

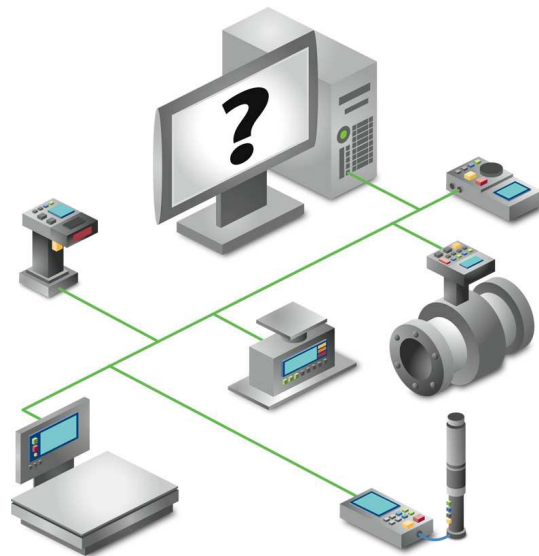
Tony Hartford  
Kepware Technologies  
Portland, Maine USA

Kepware Whitepaper  
March 1, 2007

# Considerations for Data Collection from Serial & Ethernet Devices

## Introduction

Since the early 1980s engineers and scientists have been able to read data from industrial and laboratory devices, into a computer or data acquisition system. In the early days this may have been driven more by an engineer's curiosity than a requirement of their organization or industry. Over the past decade, reliable access to device data has rapidly changed from being a nice afterthought to a necessity of most applications. There have been many factors driving this change including: the need to increase productivity due to global competition, standardization to streamline global corporate operations, rising costs of energy and raw materials, quality improvement initiatives, new focus on reduced downtime, predictive maintenance, the need to reduce the costs to maintain control systems, deregulation, and finally government legislation like 21CFRPart11 and Sarbanes-Oxley. The good news for project engineers these days is they have many technologies to choose from. The bad news may be that there are too many options to consider. If you ask control engineers around the globe, most will tell you that device communications (device drivers and protocols) can be a common challenge area when specifying and commissioning projects.



**The goal of this paper** is to help people who are designing data acquisition/collection systems learn more about choosing the device connectivity option that best meets the needs of their application.

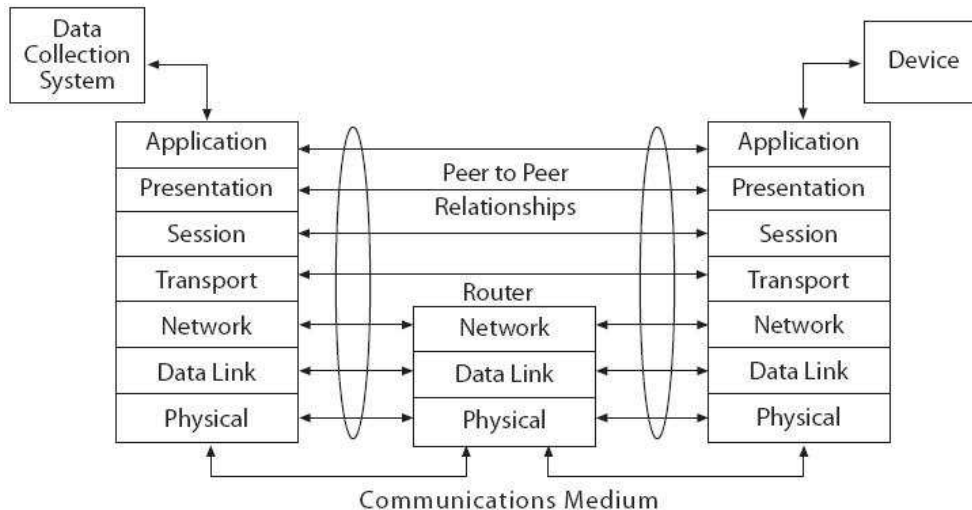
The number of devices and variation of device brands used in most factories and corporations can be mind boggling! The diagram above provides a very small sample of common devices used in industrial applications. Other devices used in industry as well as laboratory applications include:

Common Devices requiring connectivity for Data Acquisition			
Analyzers	Displays	Mixers	Pumps
Balances	Drives	Monitors	Readers
Barcode Scanners	Gauges	Motors	Recorders
Check Weighers	GPS Data	Panels	RFID Readers
Comm Boards	Instruments	Power Relays	Scanners
Controllers	Loggers	Power Systems	Sensors
Counters	Meters	Printers	Weigh Scales

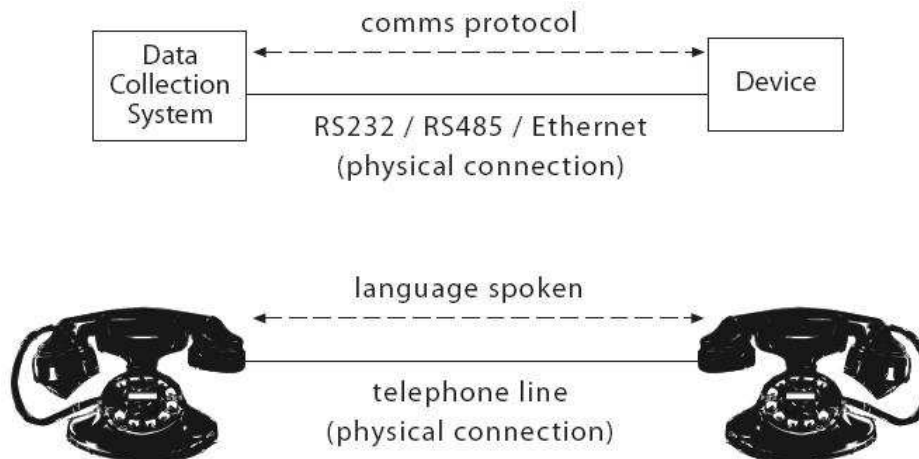
# Considerations for Data Collection from Serial & Ethernet Devices

## Protocols & the OSI Model

Protocols are data structures which are typically used to define and coordinate communications between two or more devices or systems. A common example is a computer being able to “talk” to a printer to output data. To be able to communicate with the printer the computer must have a print “driver” installed. Typically when a printer is purchased it also contains an installation disk/CD which the buyer uses to install print driver(s) on the computer. To improve integration between products from different vendors, the Open System Interconnection (OSI) model was developed by the International Standards Organization. The OSI model is a seven layer model shown in the diagram below.



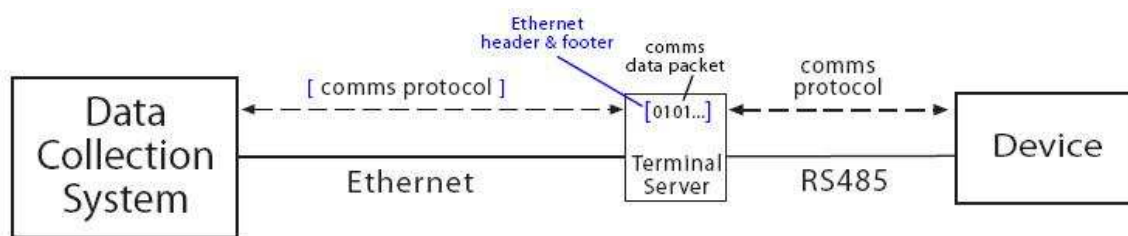
A protocol defines the message packet (sequence of bytes) that travels over the physical layer. Typically a packet begins at the Application layer and travels down through the remaining layers until it reaches the Physical layer. A good analogy for the protocol over a physical layer concept is a telephone connection with a person on either end. In the analogy the protocol is the language spoken by the people who are holding the phones. If the people both speak the same language then they will be able to communicate.



## Considerations for Data Collection from Serial & Ethernet Devices

The OSI model has gained wide adoption and since most devices come from the manufacturer with some type of serial or Ethernet connectivity option, the *physical* connection to devices is no longer a hurdle. For devices in relatively close proximity to the data collection system, wiring an RS485 or 10BaseT Ethernet connection is straightforward. For remote connectivity to devices, the hardware options abound with dialup modems, radio modems, and wireless Ethernet devices.

Connecting serial devices (RS232, RS422, RS485) into a wired Ethernet network is a matter of choosing between several well known manufacturers of device servers (also known as terminal servers or serial-to-Ethernet device bridges). These DIN rail devices can help owners get more life out of legacy hardware that may not be cost effective to replace. The diagram below shows the terminal device server connection.



### Terminal Drivers installed on the Data Collection Computer

In order to Ethernet enable serial devices, device servers “wrap” or Encapsulate an Ethernet header and footer around the serial data stream coming from the legacy device. At the data collection system terminal drivers strip off the Ethernet header and footer then pass the serial stream to the collection application. Use of these terminal servers typically *requires the installation of drivers on the data collection PC*. This can cause conflicts with other software programs, and in the long term the aspect of having a somewhat hidden software program to maintain and update can be a concern. In a scenario where multiple brands of terminal servers are used, multiple brands of terminal drivers would need to be installed on the data acquisition PC. Additionally, there is no standardization of these drivers between device manufacturers, and most of these drivers may not handle connectivity to large numbers of devices or large amounts of data very well (dead locks or blue screens of death are not unheard of). Terminal servers are best at wrapping and routing simple device protocols where timing and error recovery are not concerns. Reliable connectivity to many industrial control networks often requires features related to the timing and error recovery needs of common industrial device protocols. Adjustable connection timeouts, request timeouts, inter-character delays, write optimization, device demotion, and built-in data capture from device diagnostics are common requirements in these applications. A final point to remember is the peer-to-peer or Master / Slave scenario of serial device networks. Typically devices with “native” Ethernet connections allow multiple Masters to poll them for data. Most serial devices on the other hand can only have a single Master polling them for data. This needs to be considered when designing systems where device data must be acquired into multiple data collection nodes.

## Considerations for Data Collection from Serial & Ethernet Devices

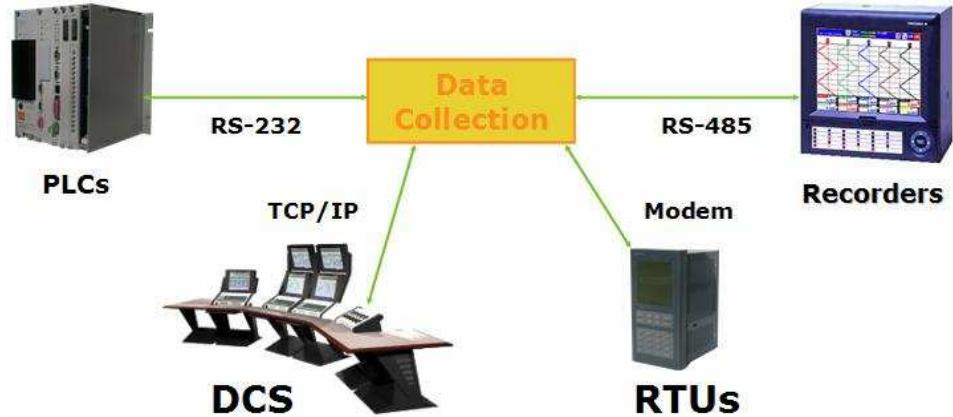
### Protocol Proliferation

The quality, usability, and affordability of the hardware used to connect to industrial devices has continued to improve and the remaining sticking point for reading device data from the lab or factory floor device is often due to protocol mismatch or driver quality and reliability.

Since the creation of the Modicon Bus (Modbus) protocol in 1979 the number of device protocols has grown tremendously. In the industrial market the most common protocols tend to be proprietary protocols promoted by dominant manufacturers of programmable logic controllers

(PLCs) or control systems, as well as open protocols supported by standards organizations. Many

of the protocols are also popular by industry as well as region, for example the most popular protocols in factory automation in North America are not identical to those most popular in factory automation in Europe. The table below shows a small sample of proprietary and open protocols used in industrial applications.



of the protocols are also popular by industry as well as region, for example the most popular protocols in factory automation in North America are not identical to those most popular in factory automation in Europe. The table below shows a small sample of proprietary and open protocols used in industrial applications.

Proprietary Protocols:	Open Protocols:
Allen Bradley DH+, Allen Bradley Ethernet	AS-I
Foxboro I/A	BACnet
GE SNMP, GE SRTP	CANbus
Koyo KSequence	DeviceNET & Ethernet I/P
Mitsubishi FX, Mitsubishi Format 1 Ethernet	DNP
Omron Host Link, Omron FINS	EtherCAT
Siemens AS511, Siemens MPI, Siemens H1	HART
SquareD Serial, SquareD SY/MAX	Modbus RTU & TCP/IP
Telemecanique Uni-telway	Profibus

### SCADA, HMI, Historians & other Client Applications

Over the past decade there has also been significant growth in the number of supervisory control and data acquisition (SCADA) and human machine interface (HMI) systems available in the market, not to mention a variety of mainstream programming environments for building custom client applications. On the one hand this growth led to more options for consumers but initially options were often dependent on the device drivers available with the SCADA, HMI, or client application. In the past, a customer with one vendor's PLCs would only be able to purchase an HMI product which had drivers for that vendor's PLCs supported in the HMI. The other problem with HMI

## Considerations for Data Collection from Serial & Ethernet Devices

drivers was ownership of the data. As soon as one brand of HMI and the corresponding driver were collecting data from the PLC or control system, that closed the system to other control or data collection applications. It often required expensive custom

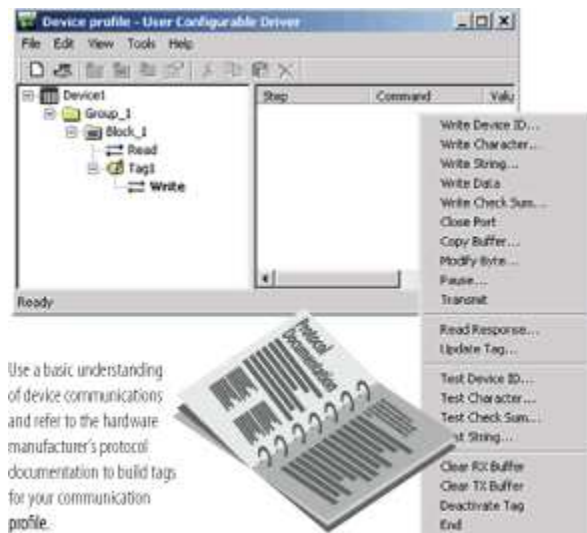


integration to share that data with other systems. Although open protocols like Modbus have made dramatic improvements in interoperability between systems and devices, there were still issues with conformance to protocol specifications as well as quality and reliability of drivers. For certain device types (flow computers for example) the Modbus protocol may not completely cover all connectivity needs of the device so vendors have produced communications based on a “modified” Modbus approach. For a truly competitive list of drivers, each company who produced a client application still had the burden of also producing proprietary drivers, open protocol drivers, and variations of modified open protocol device drivers for the most common devices in the market. To summarize, **client applications with proprietary drivers had many problems including:**

- Resource burden applied to all client vendors
- Distraction from core focus (client application)
- Often had issues with quality and reliability
- Closed system, not easy to connect other systems to the data
- Market not conducive to interoperability standards
- Severely limited consumer choice

### DDE and the birth of Middleware Options

With the increasing demand for data acquisition and the continued proliferation of communication enabled devices, the need for independent middleware became apparent. The birth and subsequent growth of middleware solutions was driven more by the need to acquire data from devices other than PLCs than the desire to interface via common PLC communication protocols. Users of sensors, temperature controllers, barcode scanners, instruments, meters, and weigh scales all needed an easier way to get proprietary serial or Ethernet data from their device. A few off-the-shelf but configurable software applications were created by independent software companies to acquire data from devices and then make the data available to other Windows applications. These applications did not require the experience of a programmer but did require a basic understanding of device communications as well as access to protocol documentation from the device manufacturer. Typically these applications have a **menu driven user interface** which allows the configuration of read and write transactions (data packets) to establish runtime communications with the device. Limitations of most off-the-shelf middleware products are that they are targeted more for simple laboratory applications where small amounts of data are being read into a



## Considerations for Data Collection from Serial & Ethernet Devices

---

simple data acquisition application. Most are not based in common automation standards and may not meet the demands of industrial applications. Typically they are not tested and supported with industrial client applications like SCADA, HMI, and Historian systems, to say nothing about enterprise resource planning (ERP) and manufacturing execution systems (MES). Middleware software products tend to contain configurable capabilities for simple protocols but do not have pre-configured drivers for complex PLC and control system protocols. With the amount of existing hardware requiring proprietary or open protocols listed in the table earlier, a single middleware solution supporting a combination of configurable connectivity combined with additional modules or plug-ins for common industrial protocols would be a more reliable, robust, and better performing solution. Multiple configurable and pre-built communication capabilities through a single application would be much easier to configure, maintain, and scale with future application requirements.

Up until the mid to late 1990s most of these applications made the data available via DDE. Dynamic data exchange (DDE) was created by Microsoft and was a fairly simple way of sharing data between Windows applications. A typical application would entail configuring the software to read data from a laboratory instrument, then pass the data into a cell inside an Excel application. Due to its simplistic design and the dominance of Windows based applications, DDE became a very commonly used methodology in data collection applications. DDE is still in use today and seems to be more common in laboratory applications and can be the easiest way to read data into a spreadsheet for simple applications. There are a few major downsides to DDE, especially for applications where a large amount of data is being collected and distributed throughout the system and in critical 24/7 production

environments so common today. With many DDE server or client applications it may only take one hundred or more data items to seriously degrade the performance and reliability of the application.

Most developers in the automation market have embraced Microsoft's newer component object model (COM) as it provides a more robust and powerful methodology for sharing data

and has proven to be more stable and better suited for

industrial applications. Standards around COM have been developed by the automation industry (more about automation standards later) for the critical demands of industrial applications. Microsoft has not enhanced or updated DDE much in the past 15 years and while they will continue to incorporate legacy DDE code for backwards compatibility with Microsoft products, for several years they have been replacing DDE with their component object model (COM) when they can.



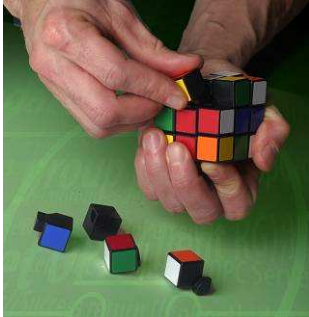
---

## Considerations for Data Collection from Serial & Ethernet Devices

---

### Custom Code

Prior to off-the-shelf middleware for getting data out of a device and into a control system, custom software was often the only choice. In the past, this was written in assembly languages or Visual Basic or one of the "C"-based languages. Custom code



made the most sense when the entire data collection application (HMI for example) was also a custom coded application.

Benefits for users of custom applications include more low-level control of the application, the ability to customize for your specific process needs, royalty-free runtime applications, and much less reliance on products of mainstream automation suppliers. One major drawback to custom applications is that they were often "brittle." Simple changes in the application or a connected device could cause lockups or crashes. For certain industries and applications (e.g. real-time control) it is

necessary to do this kind of work, and integrators and OEMs who continue to write custom code generally now adhere to canons of good software engineering practice, but in the early days, often the code was idiosyncratic, not well commented, and difficult to modify and re-use.

For many organizations looking to standardize whenever possible, custom applications and their resource requirements may not make sense. Custom applications require advanced programming skills and in today's competitive resource environment it can be difficult enough to find qualified engineers to use and maintain common HMI and PLC products, to say nothing about complex custom applications. Custom applications tend to be very sensitive to "feature creep" and can end up costing more money over the life of the application. The classic scenario is the junior engineer who does not understand the entire application requirements (that were never specified by management) but after learning how simple it can be to read a few device data points into a custom VB application, pushes to develop the solution in-house. Later, more complex needs are realized (such as alarming, logging, batching, inter-locks, secure operator logins) and it becomes apparent that the engineer (and the company) may have been better off if the engineer was able to focus on the process needs and overall application, instead of being consumed by the *development* of the core application. Finally, it is important to verify, whether choosing custom code for the application is more about an engineer's desire for something new, than a move that makes lasting financial sense for the company. Reliable access to data, time to market, and focusing on core capabilities are the key ideas to keep in mind. Reinventing, as well as maintaining the wheel can be a time and resource vacuum.

### A Great Leap Forward with OPC

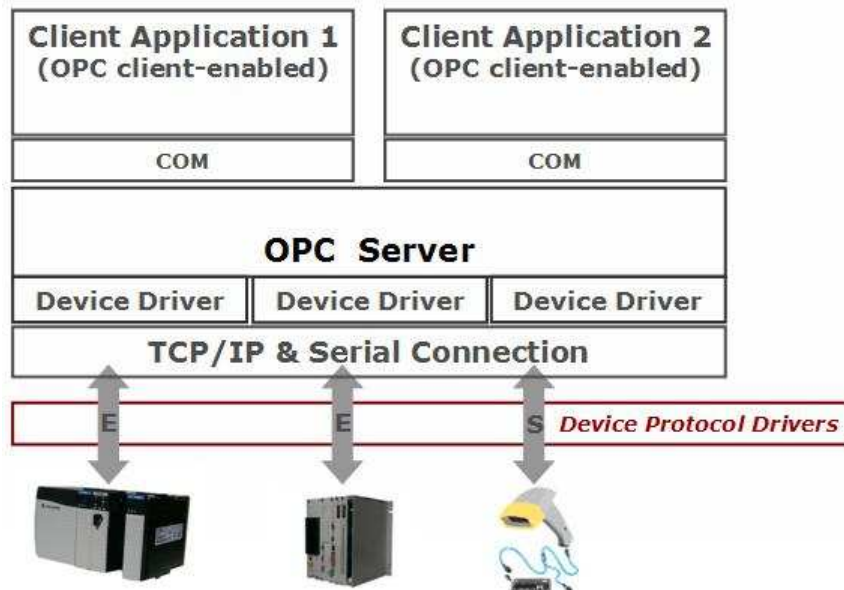
OLE (object linking and embedding) was introduced by Microsoft in 1990 as the successor and evolution of DDE. The idea was to easily automate the ability to move data from one program running under Windows to another. A few years later a handful of engineers from the industrial market came together to create an even better way to share data for automation applications. Their primary goals were to develop a single client/server architecture to allow



## Considerations for Data Collection from Serial & Ethernet Devices

vendor systems to share data in a fast, robust fashion and to drastically reduce or eliminate the burden for vendors to develop, maintain, and support their own communications drivers.

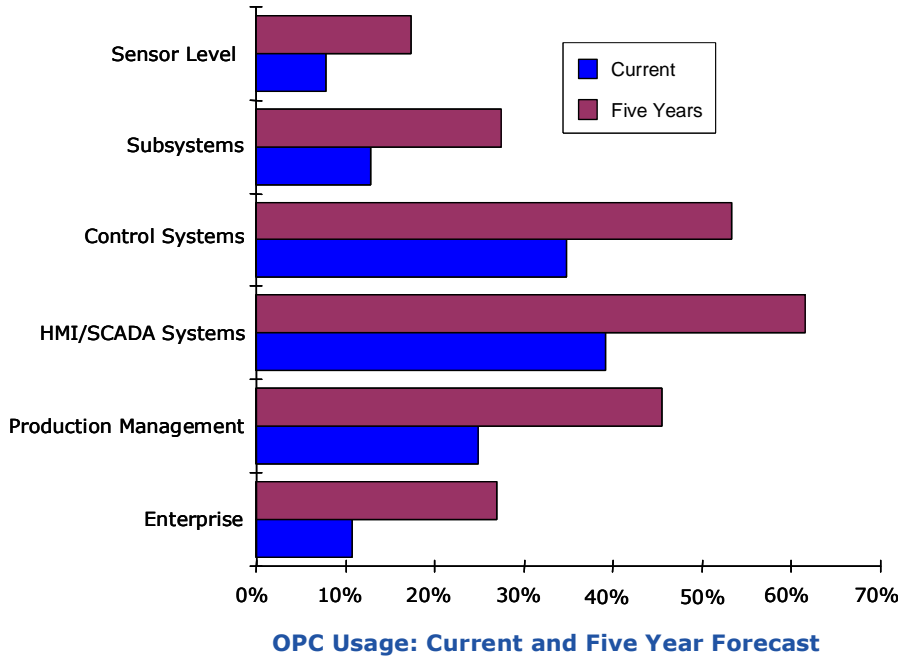
They designed a client/server architecture based on Microsoft's OLE COM (component object model) technology, and by 1996 the OPC (OLE for Process Control) Foundation had been formed and the first OPC Data Access (DA) specification had been released. With a commitment to interoperability in automation, the OPC Foundation sponsored yearly interoperability testing events where member companies could test server and client interoperability in an effort to improve connectivity in the field by customers. The foundation also implemented and has continued to improve compliancy testing tools for member companies to maintain OPC compliancy of their products. The creation and proven success of the OPC client/server architecture has made it possible for client vendors (HMIs, SCADAs, Historians, etc.) to add OPC client capabilities to their products, thus allowing them to focus on their core competencies instead of driver development.



OPC broke the lock on closed systems. Now when an OPC server is used to collect data from a device or control network, that data can be made available to other applications as well. Users of automation products now have the freedom to mix and match products and choose the PLC, HMI, Historian products that best meet their requirements, without the need for custom integration. The success of OPC has also led to improved working relationships between large and often competitive manufacturers. It is now commonplace to integrate various brands of products in an application, and many vendors are becoming skilled at working together to deliver a working solution. When issues arise, hopefully more vendors will be in the mindset of sorting out the problem instead of pointing fingers at others.

## Considerations for Data Collection from Serial & Ethernet Devices

Adoption of OPC has grown steadily as shown by a study commissioned by ARC Advisory Group in Dedham, Massachusetts. Respondents to the survey were asked: “What is the approximate percentage of OPC used for your Plant Connectivity today and in five years”?



From the survey results we can see that OPC is used in all system levels in the production process. The leading area of usage for OPC continues to be at the HMI/SCADA and Control Systems levels. Production Management should be one of the fastest growth areas for future

OPC adoption in manufacturing. Survey respondents forecasted positive growth plans for OPC over the next five years.

### Are All OPC-based Products Equal?

Although the world-wide adoption of OPC has dramatically improved interoperability in automation, it is important to remember that it has not solved all connectivity issues. Many hardware and software vendors found out that implementing high quality OPC client or server capabilities in their products still requires effort. Unfortunately customers have had to experience a bit of pain through this learning curve as well. There are examples of PLC vendors providing very low cost OPC server connectivity for their PLCs, only to have quality and performance issues force their customers to look elsewhere for a third-party alternative. OPC server consistency can be a common problem because most OPC server vendors produce a separate OPC server for each device protocol they support. This means there are often widely varying OPC server interfaces which results in customers encountering noticeable issues with reliability as well as added maintenance and training costs. There is also a wide gap in the product quality and company support commitment between third-party providers of OPC client and server products.

A few things you'll want to consider for the OPC server-based solution you choose for your next device connectivity application:

- Can you download and evaluate the product? Does the quality and reliability of the demo software meet your application needs?

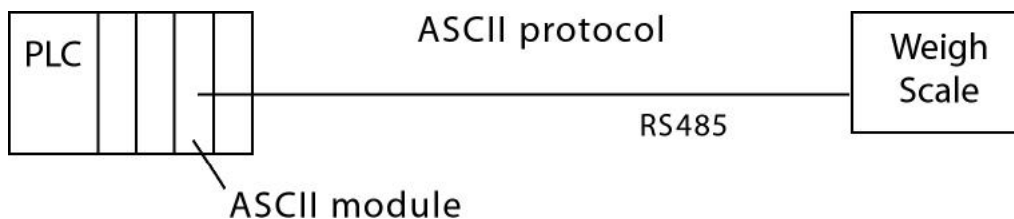
## Considerations for Data Collection from Serial & Ethernet Devices

- Does the manufacturer have formal quality control procedures and infrastructure in place?
- How many types of devices does the OPC server support through one server instance?
- Is the manufacturer in control of the technology or do they rely on toolkits or development resources from outside their company?
- Does the vendor participate in OPC interoperability events and does the product meet OPC compliancy requirements?
- Is the manufacturer's support responsive? What documentation or online help is provided with the product?
- Is the manufacturer more of a sales/marketing company or a product development/support company?



### Non PC Solutions: Interface Modules and Protocol Gateways

With the overview of OPC provided above, there was an assumption made that the device connectivity solution would be installed on a Windows PC. There are certain situations where users prefer to have a connectivity solution which is a stand alone hardware solution. Drawbacks to hardware solutions as compared to software-based solutions are likely to be increased cost, lack of standards in the user interface, impaired scalability, and no easy method for evaluating prior to purchase. The two most common examples of hardware based solutions which allow the collection of device data from other systems are interface modules in PLCs and stand alone device gateways/protocol converters. Interface modules may not be available for all brands of PLCs but are common for many modular & rack-style PLCs by manufacturers like GE Fanuc, Mitsubishi, Omron, Rockwell Automation, Siemens, and Telemecanique just to name a few. For systems where it is desired to have all device data collected into the PLC, interface modules can be the best way to go. This tends to be more common in applications where cost is not a concern, where slots are available in the PLC rack, and where a single brand of PLC hardware is the company standard.

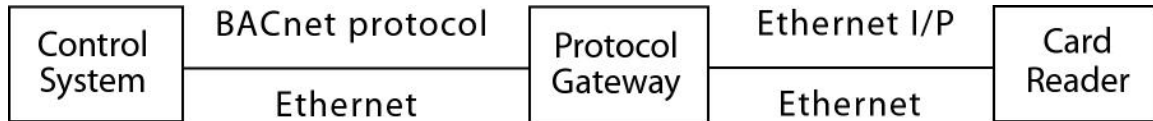


An example is bringing data from a weigh scale into a PLC. The weigh scale terminal may not support native protocols spoken by the PLC and may only have simple but proprietary ASCII communications over an RS485 connection. The interface module would need to support configurable ASCII communications and would need an RS485 port. One drawback of interface modules can be future scalability. To connect to multiple types of non PLC hardware may require multiple slots in the PLC rack, multiple types of

## Considerations for Data Collection from Serial & Ethernet Devices

---

interface modules, likely requires powering off the PLC to install the module, and can entail modifications to the PLC ladder logic to incorporate the desired connectivity. With the proliferation of industrial protocols over the past two decades, stand alone protocol gateways have become another common hardware-based connectivity solution.



An example is a building automation system that expects to collect data from BACnet enabled devices but also needs to interpret data from an Ethernet I/P enabled card reader that reads employee ID cards for secure access to authorized areas. Since the card reader does not support BACnet over its physical Ethernet port, a conversion is needed. Protocol gateways are often DIN rail mountable devices and if they are designed for industrial environments they may be mounted in close proximity to the device you need to acquire data from. When they are specified correctly, protocol converters can be an effective way to connect or bridge between disparate protocols and similar to interface modules, can achieve this without the requirement of a central computer. One advantage over interface modules is that they do not require slot space in a PLC rack and don't require modification of the PLC's runtime program or state. Gateways tend to be more expensive than PLC interface modules, can be perceived as a short-term solution, and are typically not easy to scale as the application grows.

### So Many Devices, So Little Time

In the opening page of this paper a table listing more than 25 different device types commonly found in industrial and laboratory environments was provided. These devices may support common communication protocols like Modbus or Ethernet IP, but often times they only support a proprietary native protocol developed by the manufacturer. This may be fine if the device manufacturer provides a reliable, standards-based, and open way to connect via the native protocol and make the data available to any data



acquisition system. This continues to be fine if you purchase all hardware from a single manufacturer. Since this is rarely the case, it is immediately apparent that most facilities will have multiple brands of hardware, as well as multiple brands of OPC or DDE enabled middleware to connect to the hardware. The resulting data collection system (if on a PC)

would then have multiple brands of middleware and the project development, commissioning, training, and maintenance costs can begin to add up. Customers may still have the option of interface modules in a PLC or stand alone protocol gateways, but initial hardware costs, impaired scalability, and standardization can make hardware-based data collection a deal breaker as well.

## Considerations for Data Collection from Serial & Ethernet Devices

### The Best Choice?

It seems logical that for most applications, the best choice would be a configurable, off-the-shelf, software product which is cost effective, standards based, supports connectivity for DDE as well as COM / OPC, easily scaled, designed and tested for industrial or laboratory data collection needs, with robust timing and error recovery features, full Ethernet Encapsulation support, with pre-built drivers for common protocols, serial and Ethernet supported in one interface, and is supported by a manufacturer dedicated to data communications.

<b>Common User Interface</b>		<b>Project Management</b>	
<b>Application Connectivity</b> OPC 1.0, 2.0, 3.0 OPC Tag Browsing, DCOM DDE CText Advanced DDE, NETDDE Fastdde / Suitelink PDB Interface	<b>System Core</b> Compliance Tested Interoperability Tested Runs as a Service Device & OPC Diagnostics Free Demo and Simulation Factory + Laboratory Proven	<b>Performance / Scalability</b> Multi-threading Error Recovery Adjustable Timeouts Write Optimizations Serial & Ethernet connectivity Built-in Ethernet Encapsulation	
<b>Common Driver Development Layer</b>			
User-Configurable Plug-in Driver A	User-Configurable Plug-in Driver B	Pre-built Protocol Plug-in Driver C	Pre-built Protocol Plug-in Driver D

### Data Collection Design Check-List

Here's an eleven step check-list to assist you in developing your data collection design specifications. If a software-based data acquisition system makes the most sense for your application, then you can begin to evaluate vendors of configurable device drivers and OPC servers, such as those supplied by Kepware Technologies.

### Data Collection [ Design Check-List ]

---



- Step 1:** What Device Types do you want to acquire data from and what are the manufacturer and model types of these devices?
- Step 2:** What physical connections as well as protocol communication options exist for the devices?
- Step 3:** Do you prefer a device connectivity solution based in software or hardware?
- Step 4:** If hardware, will a PLC be a data collection point and do you have present and future rack space available for an interface module? If no PLC, do protocol gateways exist which support the protocols you need to convert?
- Step 5:** If the solution is based in software, does an off-the-shelf product or custom code make more sense for your current and long-term application needs? Is there an off-the-shelf data collection application (client application like an HMI, SCADA, Historian) that the data will be served to? What are the connectivity capabilities of the client application(s)?
- Step 6:** Depending on the protocols determined in Step 2 above, does the solution need to be configurable? Does it offer pre-built driver protocols? Are both configurable and pre-built connectivity available through a single interface?
- Step 7:** What is the initial cost? What are the long-term upgrade and maintenance costs of the solution?
- Step 8:** Is the solution easy and cost-effective to scale later?
- Step 9:** Does the solution need to have robust timing, error recovery, diagnostic, and service mode capabilities for the demands of industrial applications?
- Step 10:** Can you assess the commitment and responsiveness of the manufacturer(s) involved in the solution you choose? Is support offered before and after the sale?
- Step 11:** Can you easily download demo software from the manufacturer's website and evaluate the solution before you purchase? Is pricing information and system requirements and product feature information readily available?