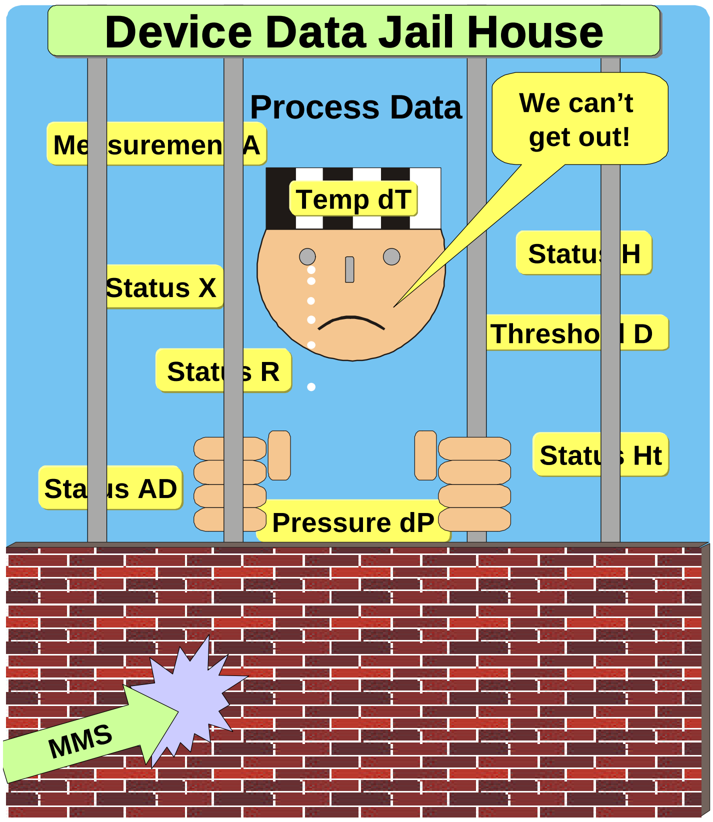
White Paper

DRAFT

Device Data Integration into Systems



provided by

ISO TC 184/SC 5/WG 2 communication and interconnection

Introduction

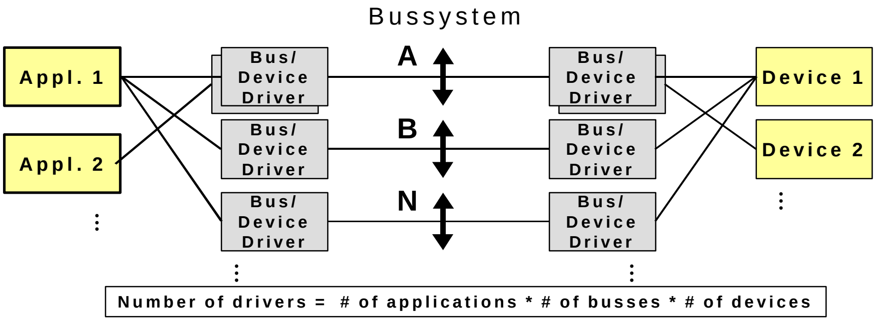
This white paper explains the vision of the standard ISO/IEC 9506 (Manufacturing Message Specification) to dramatically improve **device data integration** in the **enterprise information technology**, reducing costs and increasing the agility of the whole life cycle of a technical process to react to changing business needs. It also describes the architectural approach used by the standard and the compelling business value of working together to achieve heterogeneous interoperability of business software components and any kind of devices installed for controlling and monitoring technical processes.

The ISO TC 184/SC 5/WG 2, which published MMS in the early 90's, was comprised of many of the most prominent stakeholders in the device software interoperability arena in the world. It was formed in response to the rapidly expanding problem of tying together disparate software applications and heterogeneous devices comprising a systems. The requirement to bolt business software and device data together quickly and cheaply is accelerating and customers are spending more time and money on this problem than ever before.

The distribution of systems – or the building systems out of distributed functions residing in scattered devices – requires a standard to **inter-operate under real-time conditions and at reasonable costs**.

The Problem

Imagine if you didn't have common electric outlets and plugs in your house, and every time you bought a new appliance, you had to wire up the appliance to the wires in your wall. And everybody's wires in everybody's walls were different. And everybody's appliance wiring was different. That's really the way it works today with trying to integrate device data into applications. Examples for device data are status, diagnostic information, measurements, configuration description, and control information.



Since the number of drivers to integrate devices into applications that need to be written equals the **number of bus systems** times the **number of application software packages** times the **number of devices**, the situation forces developers of application software and devices to write a tremendous number of drivers.

Costs

Industry experts say: US $82 billion was spent on application integration in 1998 (Forrester, 1999) (= 40% of corporate IT budgets; still growing)

Any 1 % of the Dollars spent for the integration of devices into applications costs some US $800.000.000 per year. How many per cent this integration costs is not known. The costs of integration are not well documented. Many IT organizations do not keep specific records of the cost of integration. Much more information on the true costs of integration will be appearing shortly.

The cost is not limited to installing new applications, customers explain. The larger expense in time and money comes from the overwhelming task of maintaining the API's to existing in-house applications. Many customers tell they believe that savings in maintenance alone is the biggest opportunity for saving time and money for an enterprise.

It is clear that this problem is costing organizations great sums of money as well as lost opportunities because of the inability to support the business needs quickly enough.

Today's solutions are the problems of tomorrow.

To reduce the risk of getting even worse in the future, the integration of – more and more intelligent – devices into the enterprise applications (SCADA, real-time asset, machine diagnostics, ...) is a real challenge for programmers and engineers.

The driving force behind the standardization is to effectively and efficiently perform device data integration and sharing.

DRAFT

Mission

The mission of the ISO IEC TC 184/SC 5/WG 2 is to promote the easy and cost-effective integration of device data into the business application software.

The Group does this by advancing industry awareness of issues and solutions regarding interoperability – under real-time constraints – and working to evolve those solutions into a best.

In this complex and dynamic environment, it is easy to see that single sourceing one's own software will not fix this problem. The notion of a **homogeneous environment** is no longer practical. Some of the most progressive IT organizations are changing their strategy to build a **integration backbone capability** that proactively supports these business conditions, thus providing the ability to react to the business.

Unfortunately, without a common approach, and without the cooperation of the technology vendors, this approach falls short of its promise.

We believe that the publication of the standard ISO/IEC 9506 will decrease costs and complexity for organizations that are wrestling with the problem of tying together their business applications and scattered device data.

The standard ISO/IEC 9506 has built a common model for **interoperability** between applications and any devices. MMS is an exciting, evolutionary, **inter-operation language**, published as an ISO/IEC standard. MMS is actually a language that describes **device data** and **rules on how to access the data**. Since MMS is **communication-system-neutral**, **operating-system-neutral**, and **device-neutral**, it is an effective tool for providing **heterogeneous interoperability**. This is also in complete alignment with the OSI/IEC strategy, which is to be technology aware, but not technology specific.

Vision

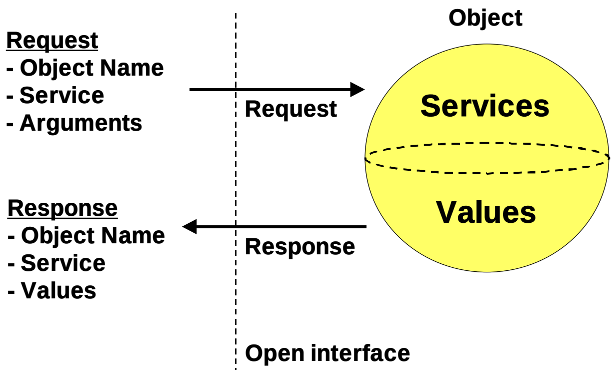
WG 2's vision is to drive for a solution that enables an organization to assemble their application portfolio, as they need to support the business and associated process, with a minimum of time and cost. When customers talk about tying together real-time application software and devices, most of the time they do not care about the technical wonders associated with integrating software. They require the ability to support their organizational needs to be fast, agile, and economical – and meeting the real-time requirements.

How does MMS compare to other standards?

There is one existing standard that relates to MMS, OPC. OPC does not fit the functionality of MMS well. OPC was designed to support factory floor and process control equipment, such as PLCs (programmable logic controllers), DCSs (distributed control systems), and the like. It's design is oriented towards single-register I/O to and from those devices at relatively slow data rates. The MMS objects support all this functionality nicely. However, it would be difficult and even unnatural to try to force this functionality into the OPC models.

MMS is specifying virtual interfaces for the **real-time data exchange** so that customers and suppliers can build real-time applications to these interfaces and thus reap the benefits of mixing and matching applications and devices from different sources, including purchased products and customer built applications.

The MMS virtual object model



Consider the generally accepted approach to accessing an object. One invokes a service on an object by sending it a message with an object name, a service signature, and a set of arguments. The object processes the request and responds to the originator of the message.

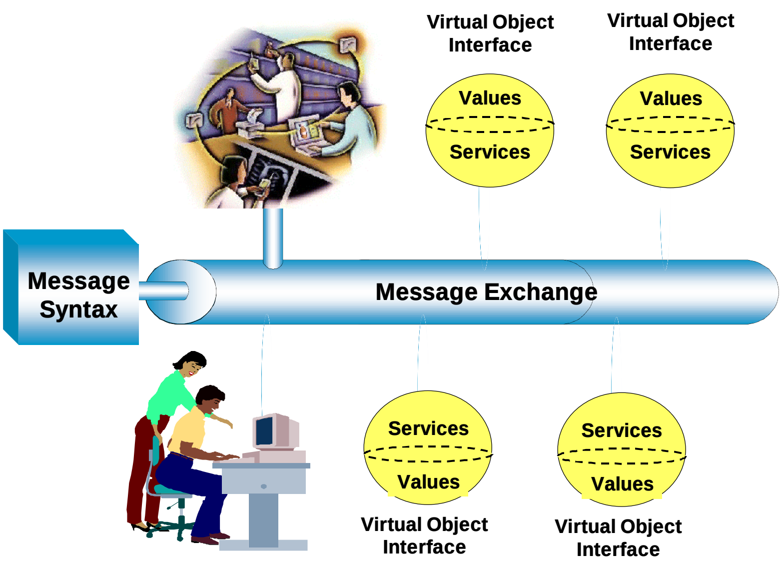
This model encapsulates the private implementation details of the object and enables communication through a public, open interface. This is the conceptual approach to interoperability MMS is taking.

MMS is building a content based virtual device object model that enables a device application to build a **virtual object** wrapper around itself through the use of MMS virtual objects. This interoperability is achieved with object oriented advantages without the requirement to implement a software application in a specific object oriented technology.

The virtual manufacturing device (VMD) represents the real device device data – in the direction of a client. The following definition helps to clarify the modeling in the form of a virtual device:

If it's there and you can see it	It's REAL
If it's there and you can't see it	It's TRANSPARENT
If it's not there and you can see it	**It's VIRTUAL**
If it's not there and you can't see it	It's GONE
Roy Wills	

The VMD is not the real world device – but it is a **standardized representation of the real world data**.



To communicate with a object, in this model, one passes a request in the form of an message to a virtual object interface. In this regard MMS implements a MOM approach (Message-oriented model).

The message itself is not an MMS object. It is a template (or an "document") that is used to convey the request and the necessary data to fulfill the request.

Because it is part of a larger, implied, virtual object model the messages (syntax) is fully described in the Manufacturing Message Specification (MMS).

The messages are understood by the receiving application and can be processed accordingly. The receiving mechanism and the underlying specifics of how the request is processed are irrelevant to the requesting application.

MMS object message interfaces

Objects interacting through messages form the essence of object technology. The concept of **classes** brings **order to the object approach** (e.g. animals) and makes it effective in **representing complex systems**. **Inheritance** mechanism allows methods developed for one class to be **reused** in other, more specialized classes.

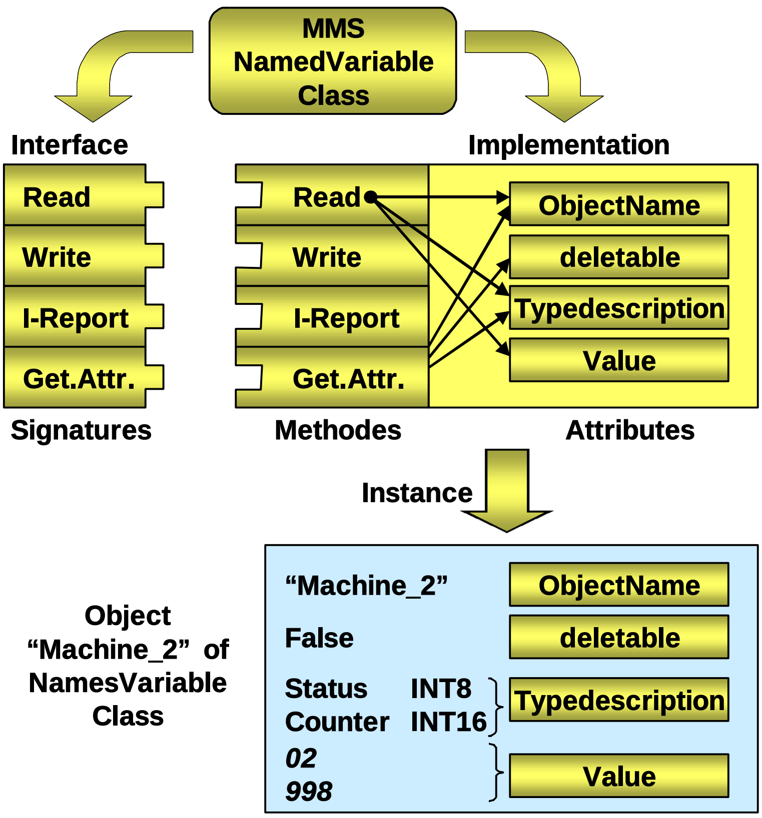
The purpose of a **class** is to specify the **behaviour** of its instances. This specification has – as depicted below – two aspects: a **message interface** and an **implementation** of that interface. The interface specifies **what the class can do**, and it consists of a **list of messages** the class can respond to. The implementation specifies **how** those responses are carried out.

Message interfaces become the driving force behind object designs. A message interface is a named **set of message signatures**.

When WG 2 designed the mechanism to define the messages necessary to build the **MMS model**, they had the foresight to determine that a fixed length mechanism was not flexible enough to accommodate the various needs of communicating between application software and a variety of devices.

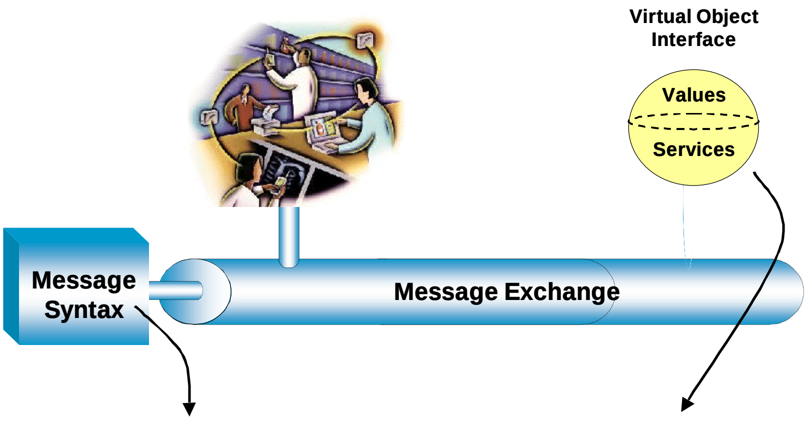
As a result of this thought process, WG 2 built a **self-describing** mechanism thus building a **meta data model**. Meta data is actually data that describes device data and services interfaces (signatures) and enables a flexible mechanism that will describe itself to another application and will ensure that only the information necessary for accomplishing the task is sent.

This architecture provides a model that is faster to develop, easier to support, and ensures higher performance for the end user. The **virtual objects** and the **messages** define an application architecture used to execute the **virtual object interface**.



DRAFT

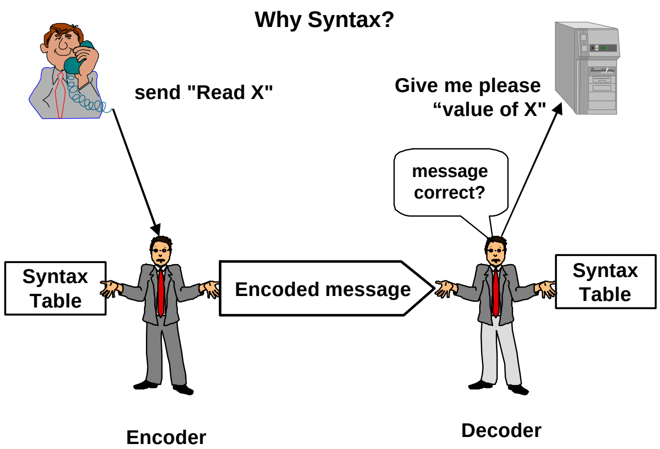
This **common semantic model** of communication is similar to the process of providing translators for all the participants in a United Nations debate. There, everyone speaks in their own language, but everyone also hears in their own language because of the translation that is built into the process.



Service Signature (Messages)	Classes (Objects)
Initiate, Conclude, Abort	Environment and General Management
Unsolicited Status, Get Name List, Identify, Rename	VMD Support
Download, Upload, Delete Domains, ..	Domain Management
Start, Stop, Resume, Kill, Delete, ...	Program Invocation Management
Read, Write, Information Report, Define Variable, ...	Variable Access
Define, Alter Event Condition Monitoring, Get Alarm Summary, Event Notification, Acknowledge Event Notification, ...	Event Management
Define Semaphore, Take/Relinquish Control, ...	Semaphore Management
Write Journal, Read Journal, ...	Journal Management

The messages and their contents, the service request and data, do essentially the same thing, except they go one step further. They also **normalize a set of very common semantics**, or the meaning of the dialog, not just the words, so there is no misunderstanding in the communication process. The various virtual objects and the services defined to communicate with these objects are depicted in the figure above. MMS provides 14 object classes and some 80 services (signatures). MMS is communication-system-neutral, operating system-neutral, and device-neutral. It is an effective tool for defining **heterogeneous interoperability for real-time applications**.

Using MMS as the method to express the messages enables to clearly and concisely define the **device model** in a **machine readable format** to enable software developers to quickly adopt and implement this model. The MMS message format uses the ISO 8824 Abstract Syntax Notation One; other notations may be used, e.g., XML (see below).



The information to be exchanged in MMS messages must be agreed upon between transmitter and receiver. These agreements are for example:

- kind of the request (read or write, ...).
- number and meaning of the parameters which are part of a request.
- distinction of the direction (request or answer).

- distinction of several similar requests; if for example two read requests (Read "A" and Read "X") are sent to the same server within a short time period, then it must be possible to assign the two answers to the requests again (to "A" and to "X").
- encoding of the data (the data types for the transmission must be defined, or at least the length; and the data must be transparent).
- definition, whether the highest-order bit of an octet is sent first or last...

Essential objectives of these definitions are:

- to **facilitate the communication** between transmitter and receiver (communication presupposes syntactic and semantic definitions).
- to ensure **completeness and consistency** of all information to be interchanged.
- to facilitate the receiver to **check the received messages** on correctness and consistency (incorrect messages are directly rejected by the communication).

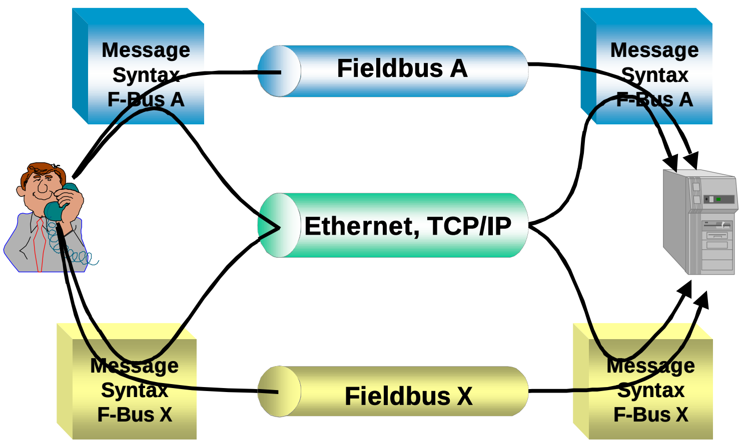
Isn't Ethernet and TCP/IP sufficient?

Ethernet provides a means to **transparently exchange groups of bits** in one data packet (frame). Ethernet doesn't care about the structure and content of the bits carried.

IP (Internet Protocol) provides very well known qualities of the stream of bits between these entities. The IP protocol that defines the IP datagram as the unit of information passed across an Internet and provides the basis for connectionless packet delivery service.

TCP (Transmission Control Protocol) The standard transport level protocol that provides the full duplex, stream service on which many application protocols depend. TCP allows a process or one machine to send a stream of data to a process on another. Software implementing TCP usually resides in the operating system and uses the IP to transmit information across the network.

So we can state that Ethernet, TCP/IP and MMS provide totally **independent** and different aspects of the inter-operation of the connected entities. MMS may make use of TCP/IP and Ethernet to exchange the MMS messages. MMS mainly defines a common semantic (common object classes and services) for use in any kind of automation systems with real-time requirements.



These days it's very popular to run well known fieldbus systems on Ethernet and TCP/IP. This is mostly achieved by **tunneling** the fieldbus protocols transparently through the TCP frames. The various fieldbus common object classes and services will be the same if tunneled or not (see left figure). The applications have to speak/understand the different "fieldbus languages" – even when using the same "Ethernet phone".

The same phone does not help a German to understand Japanese.

DRAFT

ASN.1 or Extensible Markup Language (XML)

Extensible Markup Language (XML) is an exciting, evolutionary, meta data language, approved as a standard by the World Wide Web Consortium in February 1998.

XML is actually a language for creating markup languages that describe data and rules about the data. It requires applications (e.g., the MMS objects and messages) to be defined to it before it can become truly useful. The process of defining applications is done through the use of the **Document Type Definition** (DTD), which defines the **tags and rules** within XML for a well-formed XML document.

The DTD tags and rules just replace the MMS message encoding using ASN.1/BER. Using XML as the method to express the MMS messages enables also the clear and concise definition the MMS model in a machine readable format.

XML is an exciting capability that the IT industry is quickly adopting and now tools vendors are building generally available tools that enable this approach to be based on standard tools and technologies and prevents the need for building custom or proprietary tools to adopt this approach.

The basic idea using XML instead of ASN.1/BER is to standardize rules that would allow for information described in ASN.1 (Abstract Syntax Notation) to be carried in XML (eXtensible Markup Language):

- XML provides a facility for representing character string data structures. XML defines "markup" rules in which a data structure is encoded as a **tag**, **value**, and **end-tag**.
- ASN.1 expresses message structure types and data values. When BER (Basic Encoding Rules) is applied to the ASN.1 expressions, data structures are encoded as octets in the form of **tag**, **length**, and **value** (or an indefinite length value is delimited by an end-tag).

To change an ASN.1/BER data structure to XML, an XML tag and end-tag are derived and the value is wrapped within. To change an XML structure into ASN.1/BER, the appropriate ASN.1/BER tag is derived and the value length is calculated.

For developer communities making significant investments in XML, the use of XML instead of ASN.1/BER may leverage investments already made in ASN.1-based applications and thereby preserve the **hard-won interoperability** already achieved among IT systems and **device applications modeled** according to MMS.

It is the **common MMS device model objects**, the **application specific device models**, and the **common services** that comprise the – most important! – semantic. The messages carrying the object **names, service signatures, and parameters** are absolut the same whether they are encoded in ASN.1/BER or XML. XML is just an other way to write messages – **human** <u>**and**</u> **machine** readable.

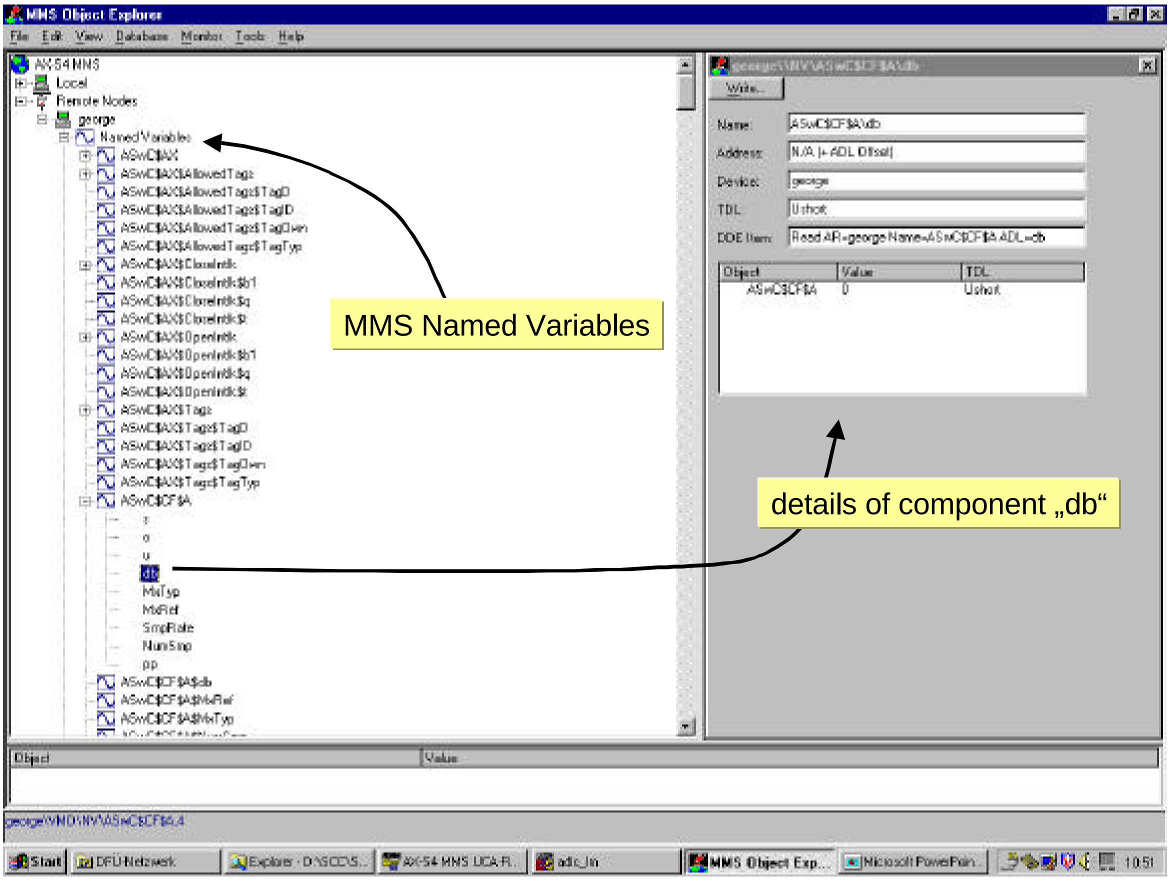
WG 2 is in the process to map the MMS messages completely to XML.

Building applications

MMS comprises **common** object classes like “Named Variable” and a “read named Variable” that allow to communicate any complex or simple structured information. For specific applications, e.g., “High Voltage Transformers”, **specific** Named Variable object classes may be defined. The IEEE UCA™ Technical Report 1550 "Utility Communications Architecture Version 2" specifies tenth of devices found in the electric utilities substations with several hundred hierarchical named object attributes for data acquisition and control. A transformer is, e.g., comprising some 200 attributes. Groups of these attributes are modeled as MMS Named Variables.

Since the representation and the services are defined independently of this specific application, the common MMS object classes and messages can be applied for **accessing and making visible** (at, e.g., a browser interface) the information from a specific instance of the transformer object class. The figure below shows the Named Variables of an instance of the **Automated Switch Control object class** (ASwC) defined in the IEEE Technical Report 1550 (UCA2).

Although MMS was originally developed for ISO/OSI networks most implementations are available for other networks such as the TCP/IP network or simple RS 232 point-to-point links, or radio links. From the point of view of MMS this is insignificant as long as the necessary quality of the connection is guaranteed.



The TCP/IP suite as well as three and seven layer communication stacks are supported by, e.g., UCA2.

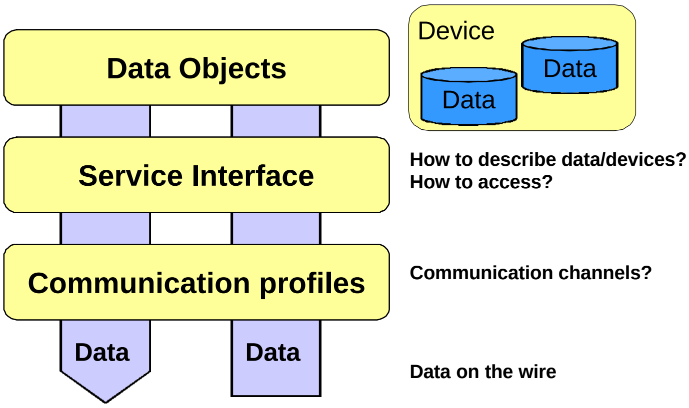
Since the Fast Ethernet is used by more and more vendors even in the field, it is very likely that in the near future each device communicates with each device in the field and with any application of the enterprise directly. Only a common language like MMS allows the applications and devices seamlessly interoperate. What would the Ethernet and TCP/IP provide,

DRAFT

if there is no precise semantic and no rule associated with the information? Not a lot! The ability to exchange row data (just bits or octet-strings) would make it visible that today's devices and enterprise applications speak totally different languages (semantics). A transformer, and the access to the transformer should be independent of the communication stack selected. Today you find as least as many message languages as we have bus systems.

Summary

The benefactors of the results of **open device data integration** span the entire industry and include all of the stakeholders in this industry. The customers are in a position to save large sums of money and time. The vendors who provide solutions that meet or exceed expectations will become very successful. This is an exciting time in the industry with an inexorable move toward practical software components.



The approach used by MMS is depicted in the figure above. The most important issue are the **models of the real device data** and the **rules** (service interface) how to access these data. It is obvious that an appropriate **transport mechanism** (communication profiles), e.g., the TCP/IP or a point-to-point link, must be used to exchange the messages between devices. Finally, the encoding of the messages (ASN.1 or XML) is a secondary issue – both do their job well.

> **One** inter-operation language MMS –
> **many** communication systems (Ethernet, ...).
> or
> **One** communication system (Ethernet) –
> **many** inter-operation languages ?

Since the common language is more important as the common "Phone" it is obvious what is crucial in the distributed automation systems in the future.

Summarizing we can state that the **device data of heterogeneous devices are freed** from specific device data jail houses using a standard service interface (MMS messages):

With MMS – The device data go public.

For More Information

If you would like more information about the application of MMS or the use of MMS in the Utility Communications Architecture (UCA), please visit the following web sites at

http://www.tamarack.com
http://www.sisconet.com/EPRI/UCA
http://www.scc-online.de

or contact Karlheinz Schwarz, Convenor of ISO TC 184 SC 5/WG2
Telephone +49 721 684844; Fax: +49 721 679387; or
e-mail: schwarz@scc-online.de.