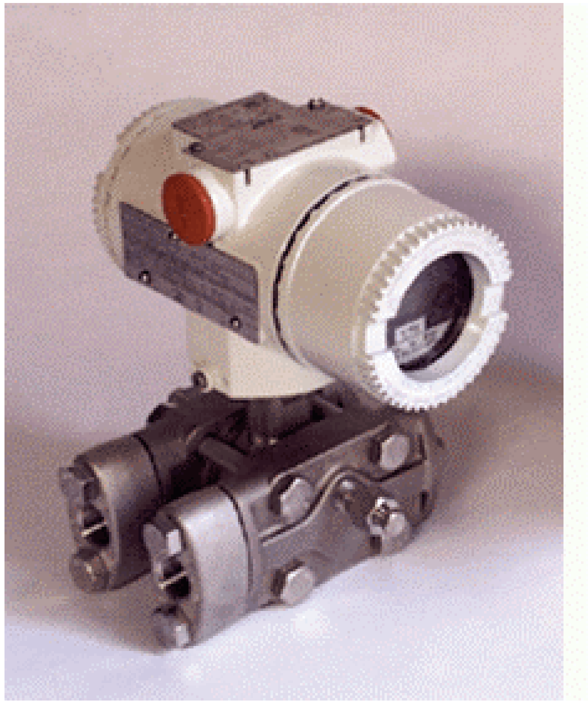
Industrial Automation
Automation Industrielle
Industrielle Automation



4. Device Management Protocols

Protocoles de gestion des appareils
Gerätezugangsprotokolle

4.1.1 The HART Protocol

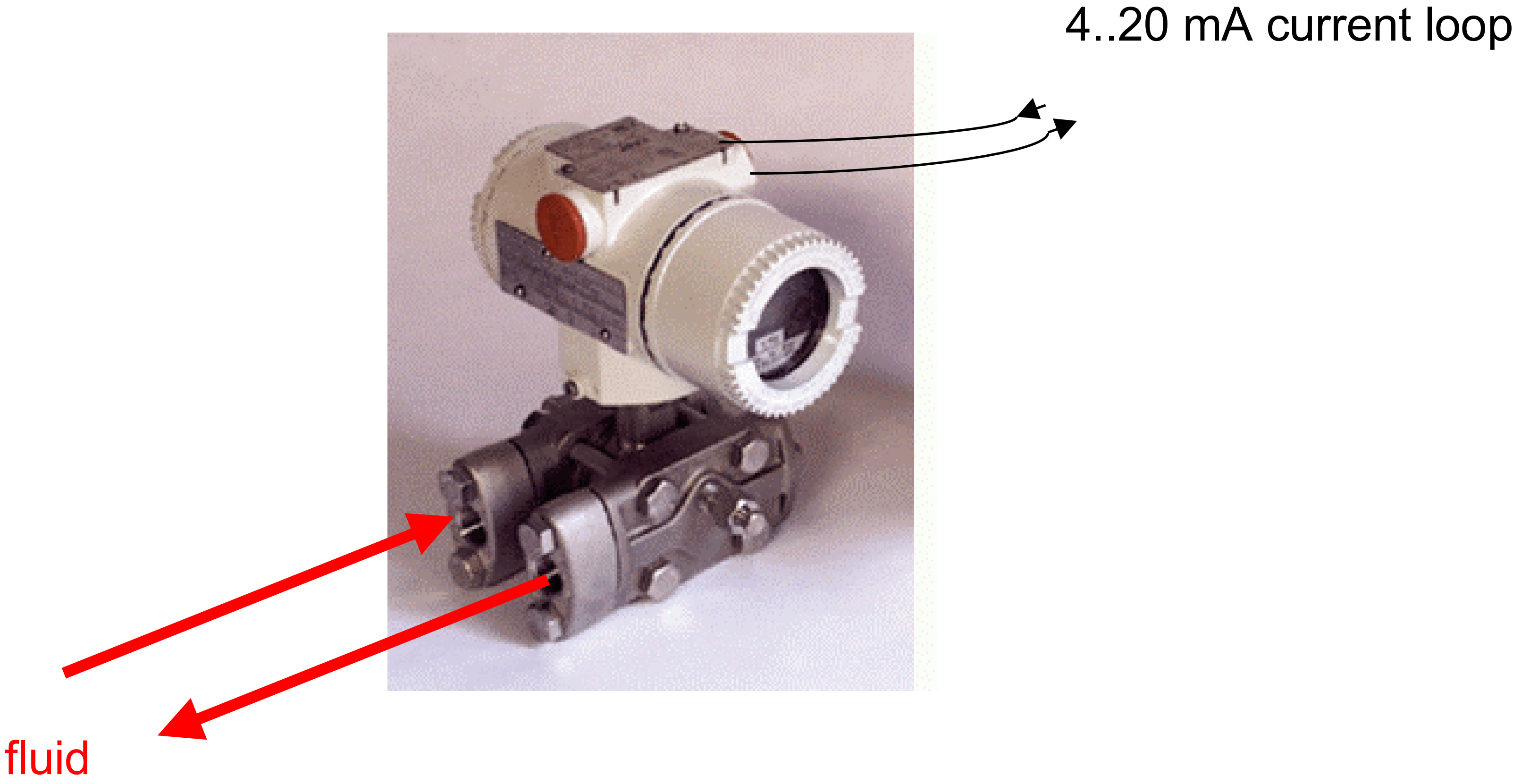
Prof. Dr. H. Kirrmann
ABB Research Center, Baden, Switzerland

2005 April HK

4.1.1 Current Loop

The classical solution for analog values

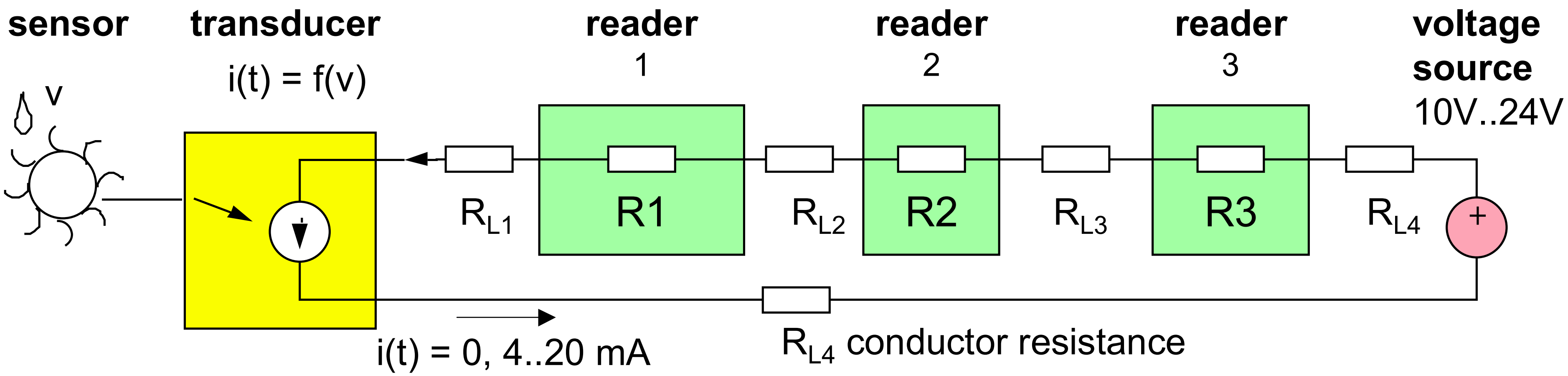
Field device: example differential pressure transducer



The device transmits its value by means of a current loop

4-20 mA loop - the conventional, analog standard (recall)

The 4-20 mA is the most common analog transmission standard in industry



The transducer limits the current to a value between 4 mA and 20 mA, proportional to the measured value, while 0 mA signals an error (wire break)

The voltage drop along the cable and the number of readers induces no error.

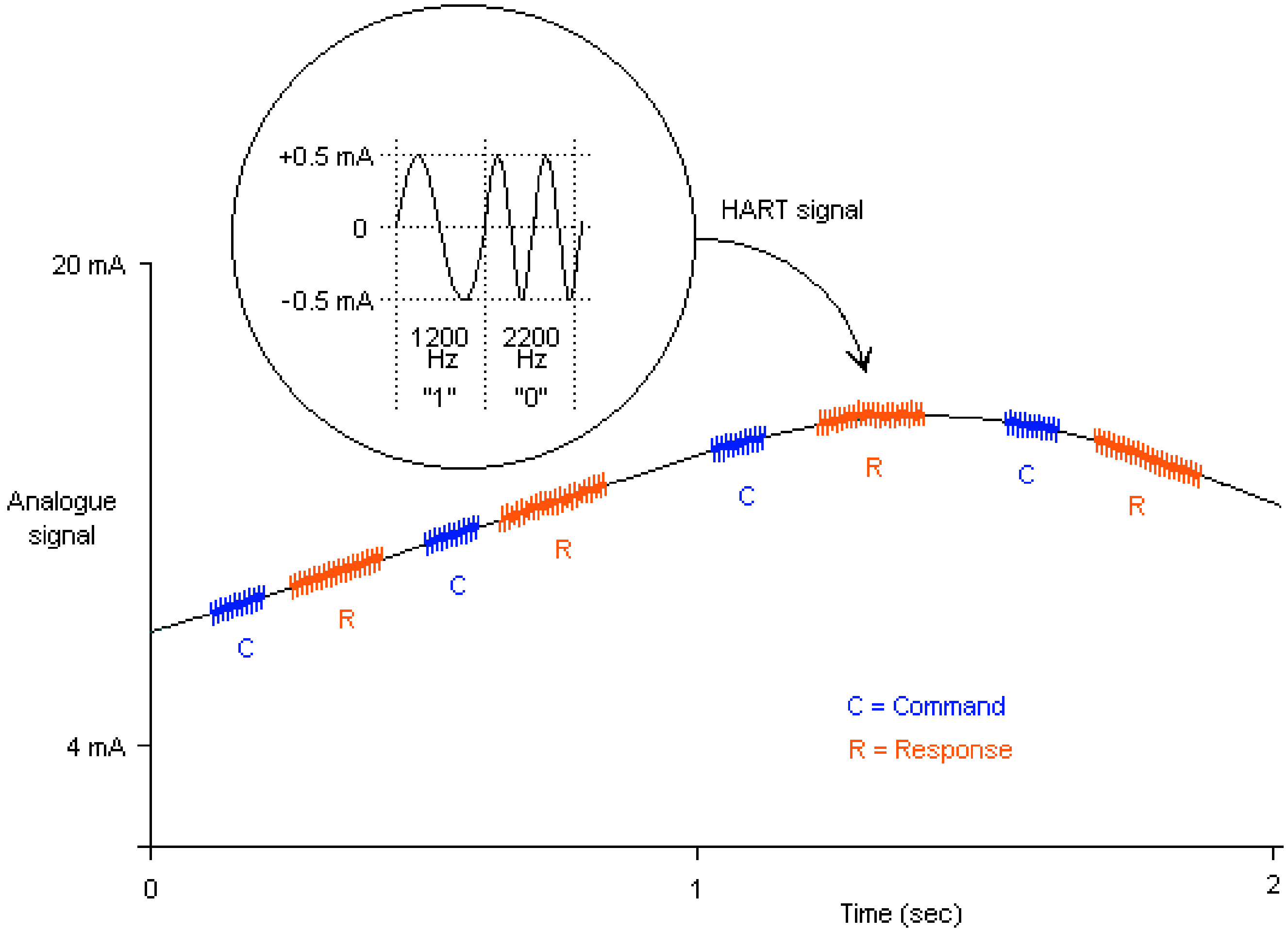
Simple devices are powered directly by the residual current (4mA) allowing to transmit signal and power through a single pair of wires.

4.1.2 HART

Data over 4..20 mA loops

HART - Principle

HART (Highway Addressable Remote Transducer) was developed by Fisher-Rosemount to retrofit 4-to-20mA current loop transducers with digital data communication.

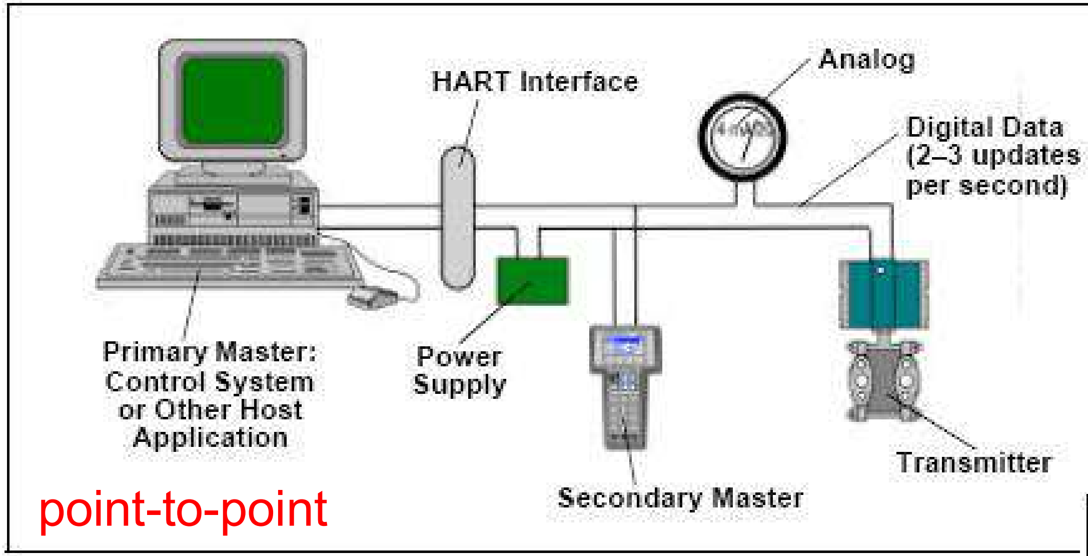


HART modulates the 4-20mA current with a low-level frequency-shift-keyed (FSK) sine-wave signal, without affecting the average analogue signal.

HART uses low frequencies (1200Hz and 2200 Hz) to deal with poor cabling, its rate is 1200 Bd - but sufficient.

HART uses Bell 202 modem technology, ADSL technology was not available in 1989, at the time HART was designed

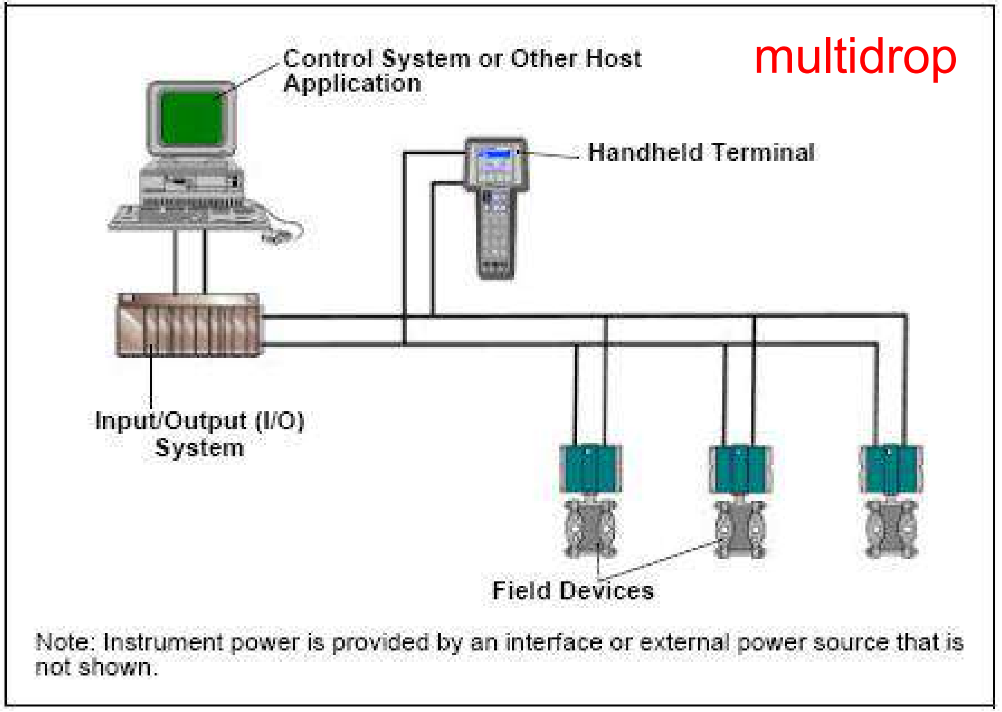
Installation


HART Interface
Analog
Digital Data (2–3 updates per second)
Primary Master: Control System or Other Host Application
Power Supply
Transmitter
Secondary Master

point-to-point

universal hand-help terminal

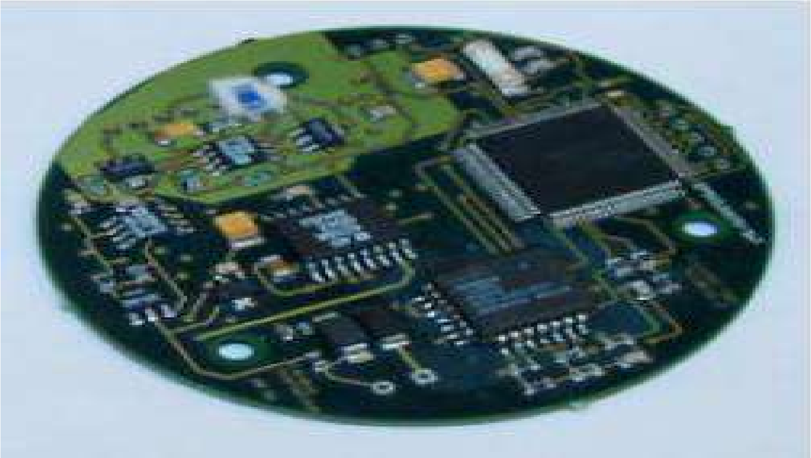
taken from: www.hartcomm.org


Control System or Other Host Application
Handheld Terminal
Input/Output (I/O) System
Field Devices

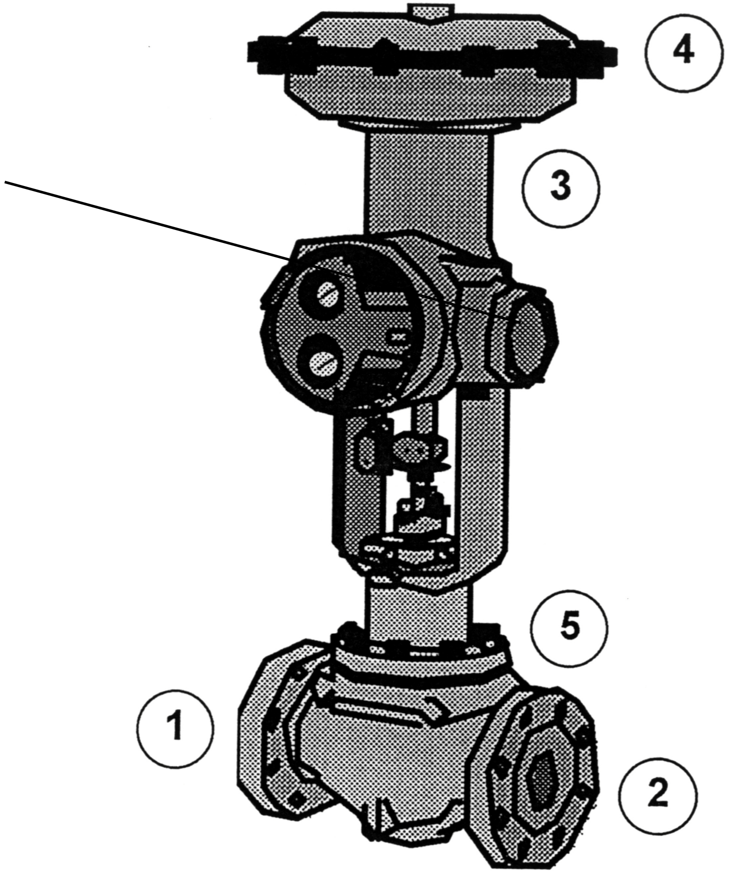
multidrop

Note: Instrument power is provided by an interface or external power source that is not shown.

The Round card



http://www.fint.no/ha-i4012.pdf

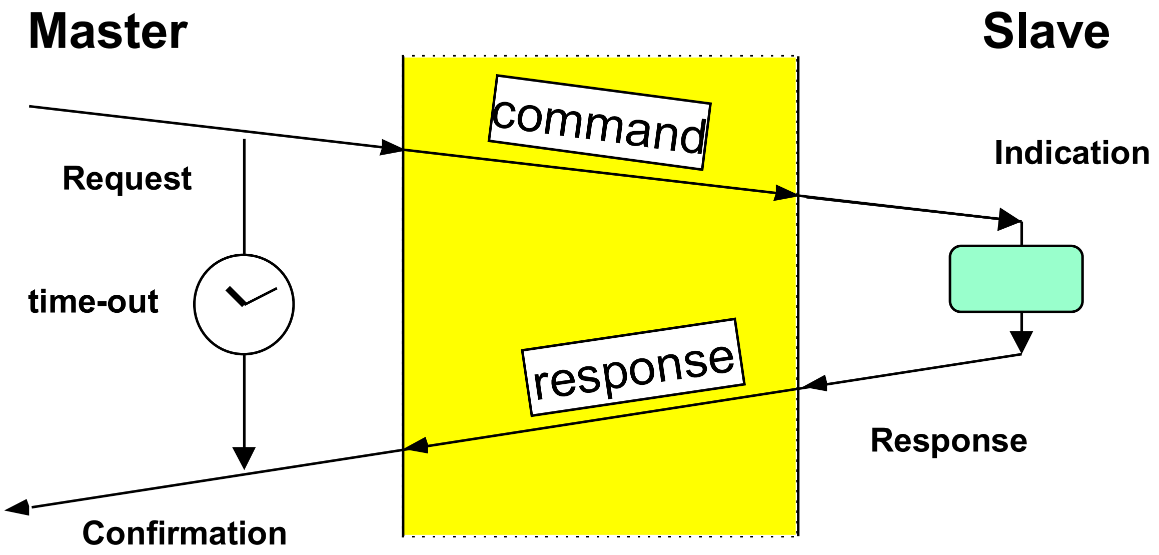


The round card is a standardized printed circuit board that can be mounted in an instrument, containing the modem, a processor, RAM, EPROM and all the logic and software necessary to execute the HART protocol.

It is round because most hydraulic instruments have a round case.

HART - Protocol

Hart communicates point-to-point, under the control of a master, e.g. a hand-held device



Hart frame format (character-oriented):

preamble	start	address	command	bytecount	[status]	data	data	checksum
5..20 (xFF)	1	1..5	1	1	[2] (slave response)	0..25 (recommended)		1

HART - Commands

Universal commands (mandatory):

identification (each manufactured device is identified by a 38-bit unique identifier),
primary measured variable and unit (floating point format)
loop current value (%) = same info as current loop
read current and up to four predefined process variables
write short polling address
sensor serial number
instrument manufacturer, model, tag, serial number, descriptor,
range limits, …

Common practice (optional)

time constants, range,
EEPROM control, diagnostics,…

total 44 standard commands

Transducer-specific (user-defined)

calibration data,
trimming,…

HART commands summary

Universal Commands	Common Practice Commands	Device-Specific Commands (example)
• Read manufacturer and device type • Read primary variable (PV) and units • Read current output and percent of range • Read up to four predefined dynamic variables • Read or write eight-character tag, 16-character descriptor, date • Read or write 32-character message • Read device range values, units, and damping time constant • Read or write final assembly number • Write polling address	• Read selection of up to four dynamic variables • Write damping time constant • Write device range values • Calibrate (set zero, set span) • Set fixed output current • Perform self-test • Perform master reset • Trim PV zero • Write PV unit • Trim DAC zero and gain • Write transfer function (square root/linear) • Write sensor serial number • Read or write dynamic variable assignments	• Read or write low-flow cut-off • Start, stop, or clear totalizer • Read or write density calibration factor • Choose PV (mass, flow, or density) • Read or write materials or construction information • Trim sensor calibration • PID enable • Write PID setpoint • Valve characterization • Valve setpoint • Travel limits • User units • Local display information

HART - Importance

Practically all 4..20mA devices come equipped with HART today

About 15 Mio devices are installed worldwide.

more info: http://www.hartcomm.org/

http://www.thehartbook.com/default.htm

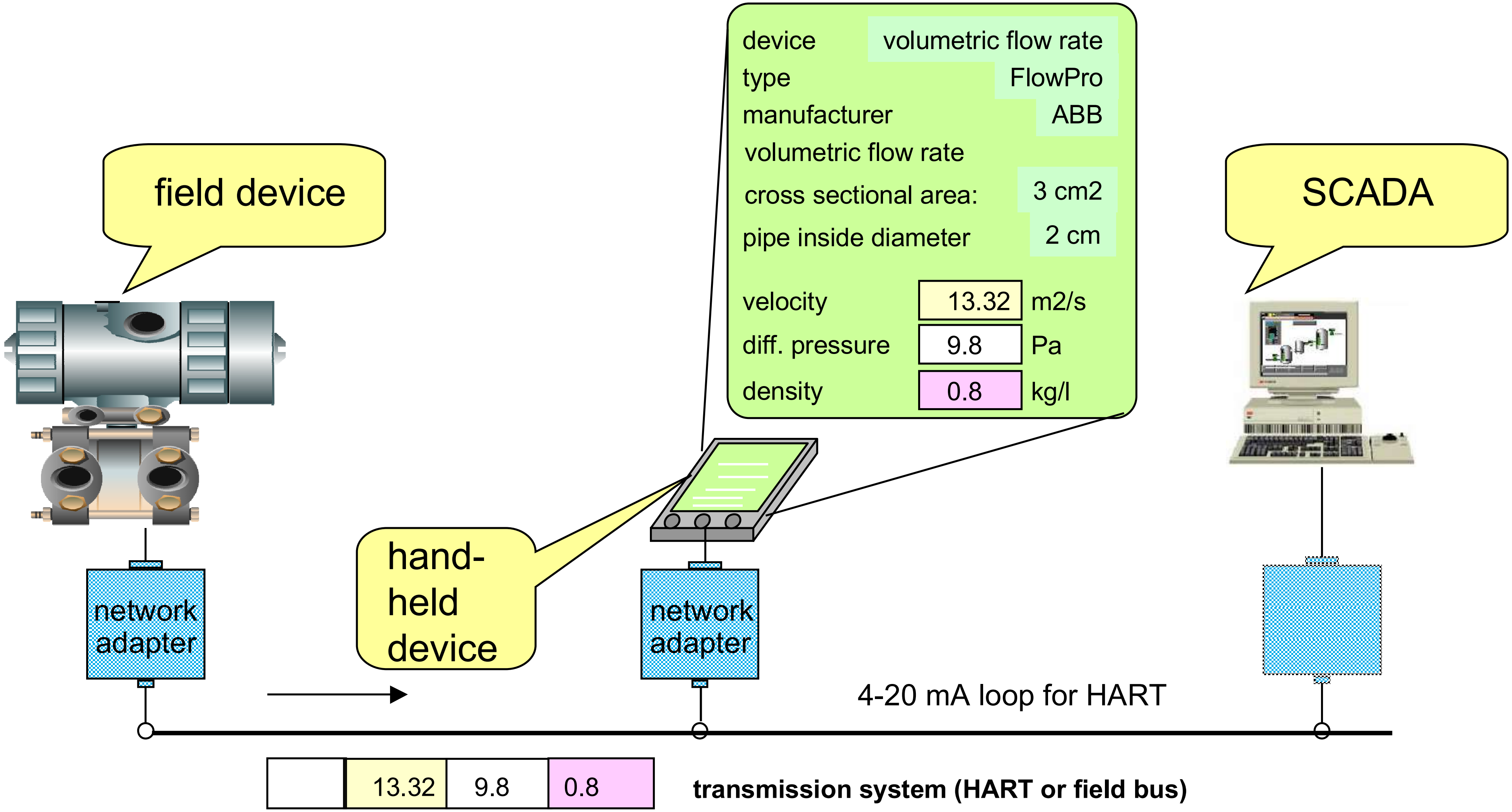
Device Description

Also known as Device Description Language (DDL) or

eDDL (electronic Device Description Language),

“electronic frontplate” (Elektronisches Typenschild, *plaque électronique*))

Device access



field device

hand-held device

SCADA

network adapter

network adapter

4-20 mA loop for HART

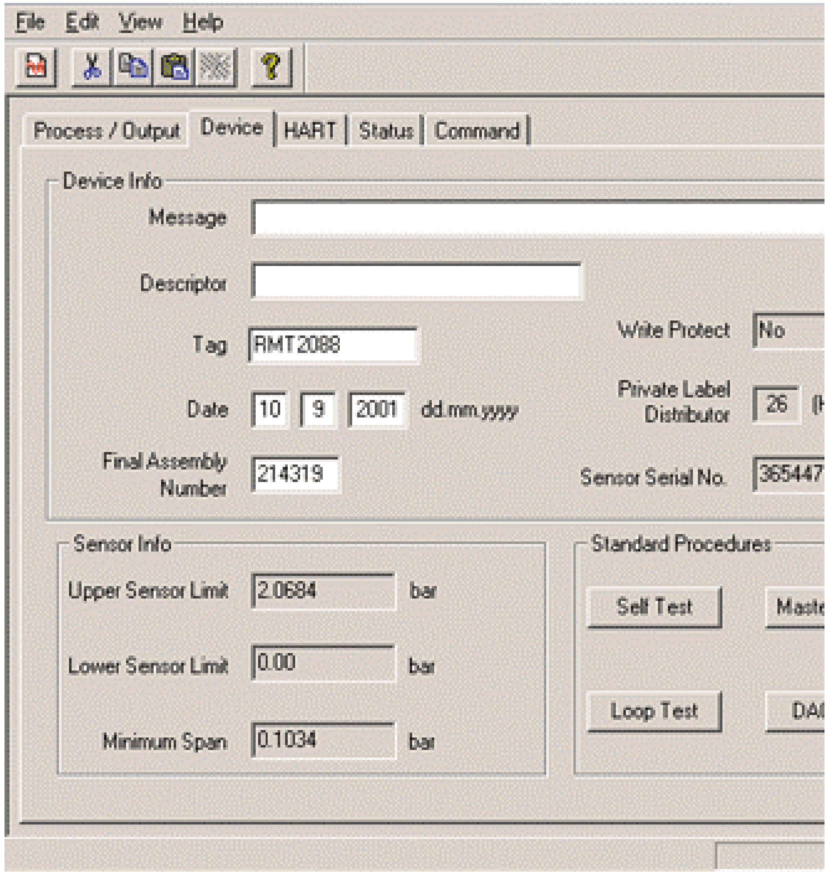
device	volumetric flow rate	
type	FlowPro	
manufacturer	ABB	
volumetric flow rate		
cross sectional area:	3 cm2	
pipe inside diameter	2 cm	
velocity	13.32	m2/s
diff. pressure	9.8	Pa
density	0.8	kg/l

	13.32	9.8	0.8

transmission system (HART or field bus)

Device Description Language

Device Description Language DDL allows a field device (slave) product developer to create a description of his instrument and all relevant characteristics, such that it can be represented in any host (master) device. The objective is common “look-and-feel”, independent of the hand-help or SCADA, similar to HTML for a web server.



Message

Descriptor

Tag RMT2088

Date 10 9 2001 dd.mm.yyyy

Final Assembly Number 214319

Write Protect No

Private Label Distributor 26 (H

Sensor Serial No. 365447

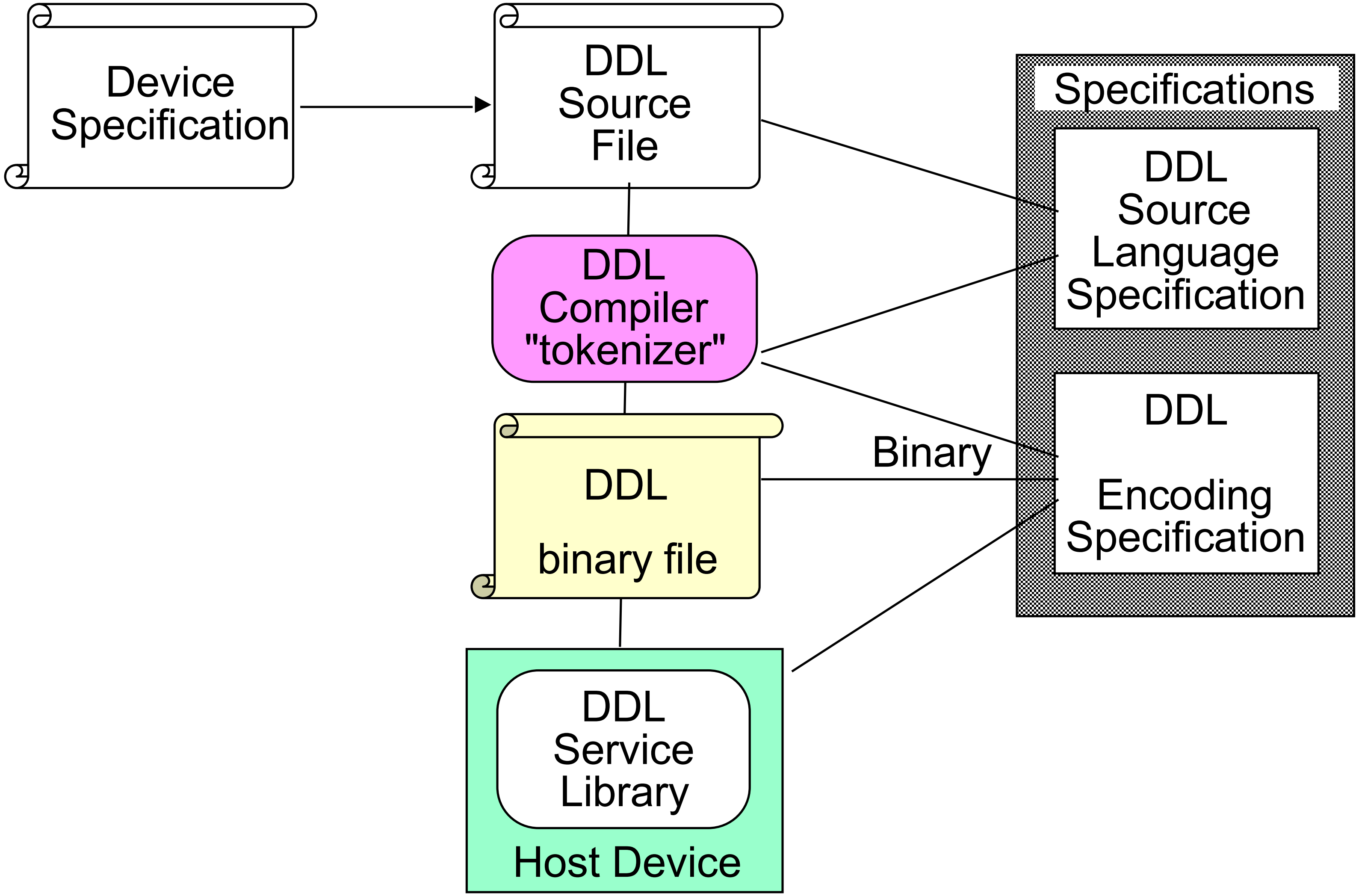
Upper Sensor Limit 2.0694 bar

Lower Sensor Limit 0.00 bar

Minimum Span 0.1034 bar

Why not use HTML ?
special instructions needed !
(C-language is used)

Device Description usage



A binary form of the source is stored in the hand-help device (not in the field device)

Assessment

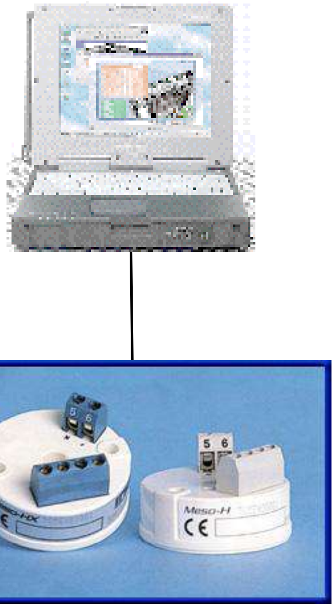
What is the purpose of the HART protocol ?

Which communication is used between a hand-help and a field device ?

Which categories of commands do exist ?

What is the purpose of the Device Description Language ?

Industrial Automation
Automation Industrielle
Industrielle Automation

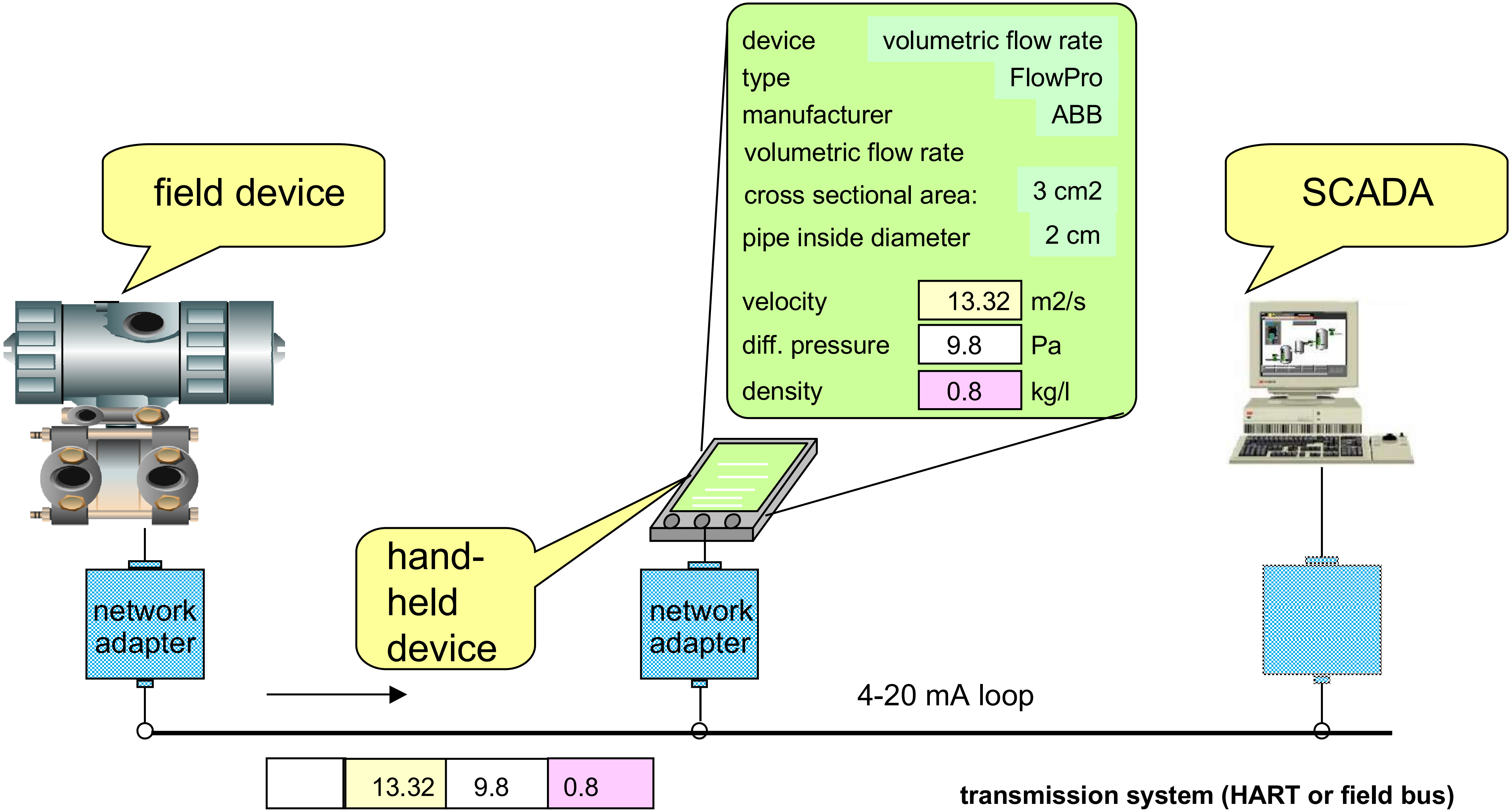


4. Device Management Protocols
Protocoles de gestion des appareils
Gerätezugangsprotokolle

4.4.1 Electronic Device Description

Prof. Dr. H. Kirrmann
ABB Research Center, Baden, Switzerland

Device Model Principle



device	volumetric flow rate	
type	FlowPro	
manufacturer	ABB	
volumetric flow rate		
cross sectional area:	3 cm2	
pipe inside diameter	2 cm	
velocity	13.32	m2/s
diff. pressure	9.8	Pa
density	0.8	kg/l

	13.32	9.8	0.8

transmission system (HART or field bus)

Device Description in HART

DDL Origins

Developed by Fisher-Rosemount for transducers connected over HART

HART = data communication superimposed over 4-10 mA loops

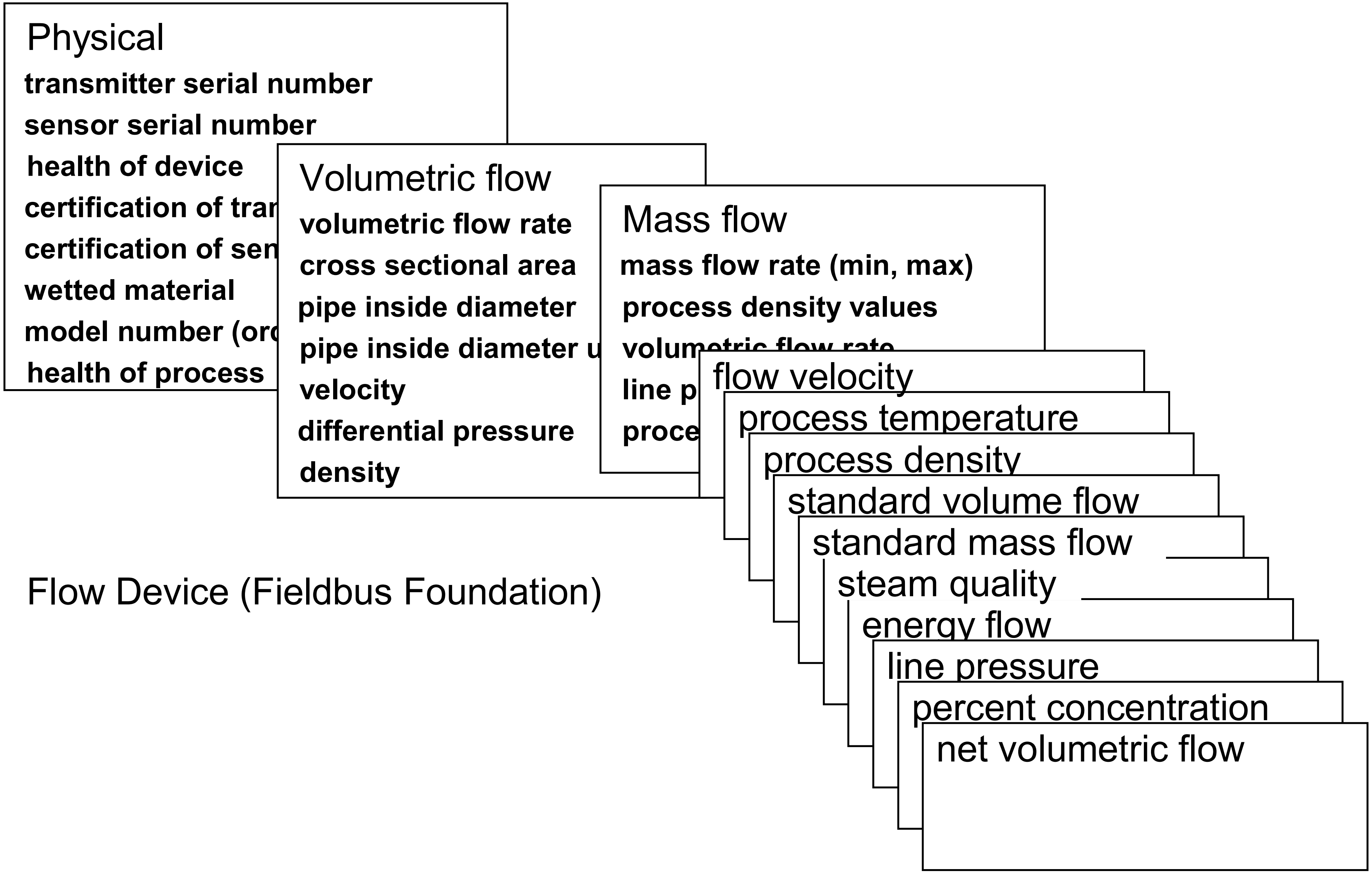
Extended by Fieldbus Foundation (FF-900-1.0 1996)

Objective:

define how a device presents itself to a hand-help terminal or an engineering station

became international standard in 2004 as EDDL (IEC 61804-2)

Example of Function Profile



Physical

transmitter serial number

sensor serial number

health of device

certification of trar

certification of sen

wetted material

model number (orc

health of process

Volumetric flow

volumetric flow rate

cross sectional area

pipe inside diameter

pipe inside diameter u

velocity

differential pressure

density

Mass flow

mass flow rate (min, max)

process density values

volumetric flow rate

line p

proce

flow velocity

process temperature

process density

standard volume flow

standard mass flow

steam quality

energy flow

line pressure

percent concentration

net volumetric flow

Flow Device (Fieldbus Foundation)

Device Description Language objects

Variables:

Variables, Records, Arrays

Relations: relationship between variables, records and arrays

Variables Lists: logical grouping of variables

Menus : presentation of the data to a host

Edit Displays : editing the data by a host

Item Arrays : logical grouping of data

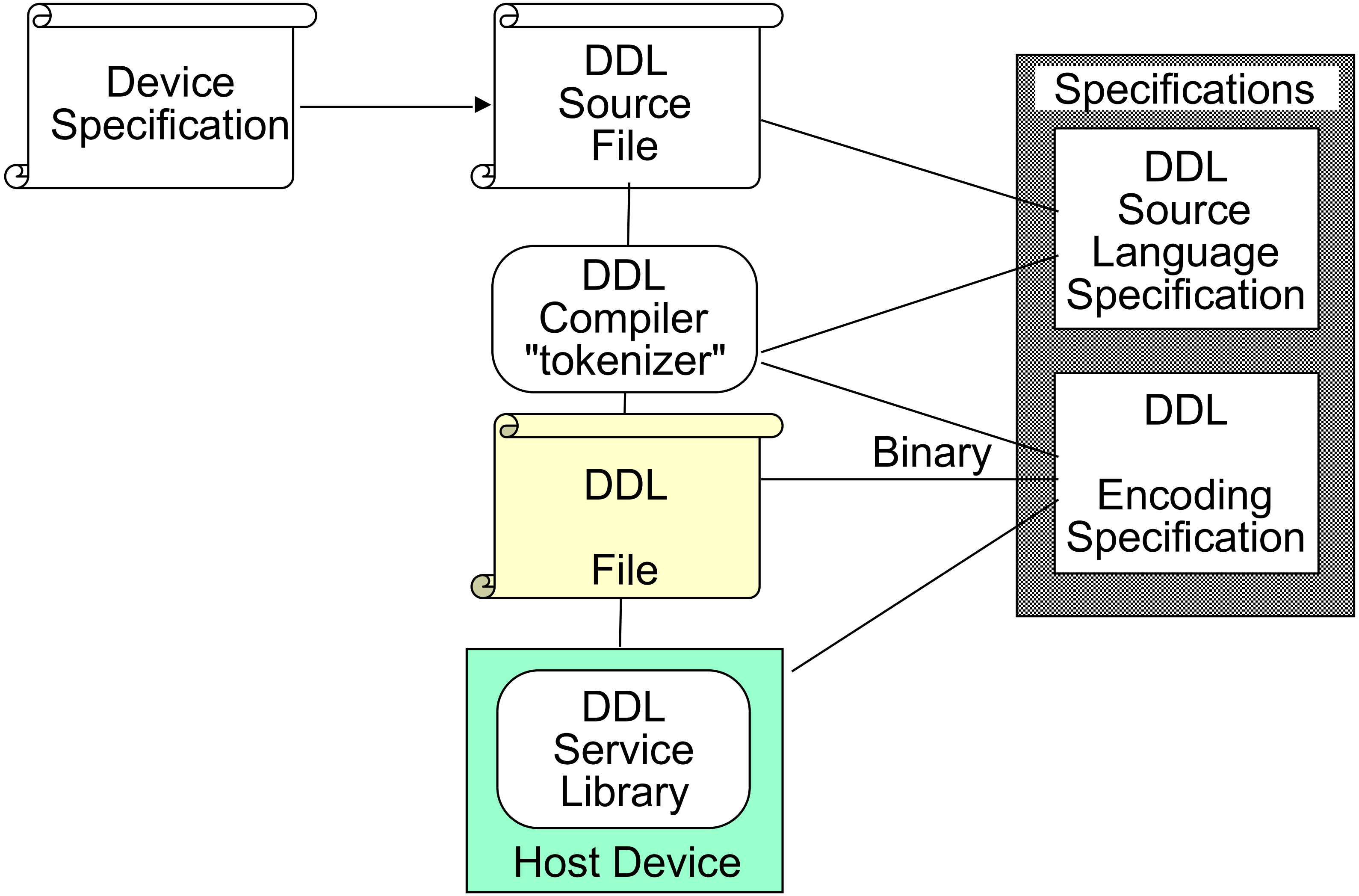
Programs : tasks to start and stop

Blocks: describes Function Blocks

Domains : download/upload of memory

Response codes: status of the request for an object

DDL Usage



A binary form of the source is stored in the hand-help device (not in the field device)

DDL Device Description Information

Information about the device itself

```
MANUFACTURER integer;          // a 24-bit integer identifying the manufacturer
DEVICE_TYPE integer;           // a 16-bit integer which identifies the device type
DEVICE_REVISION integer;       // an 8-bit integer which identifies the revision
DD_REVISION integer;           // an 8-bit integer which identifies the DDL version
```

DDL Variables

```
VARIABLE name                // name of the variable as ASCII string
{
 CLASS = { INPUT, OUTPUT, CONTAINED, // must belong to one of these three
           DYNAMIC, DIAGNOSTIC, SERVICE, OPERATE, ALARM, TUNE, LOCAL // options
         }
 TYPE = { arithmetic,enumerated,string,index, date/time }

 LABEL string;                // text to display along the variable value
 CONSTANT_UNIT string;        // string to be displayed for the units

 HANDLING = {READ, WRITE} //

 HELP string;                 // on-line help string

 PRE_EDIT_ACTIONS {methods}

 POST_EDIT_ACTIONS

 READ_TIME_OUT expression;

 WRITE_TIME_OUT expression;

 VALIDITY boolean;

 RESPONSE_CODES response_code_name;
 }
```

DDL Variables (Types)

```
// arithmetic types
INTEGER, UNSIGNED, FLOAT, DOUBLE,
        // e.g. TYPE INTEGER (size) {option option ...}
        // options:
        DISPLAY_FORMAT string;        // e.g. %4i as in printf
        EDIT_FORMAT string;           // e.g. %d  as in scanf
        MIN_VALUE expression;         // e.g. MIN_VALUE = -10; MIN_VALUE1 = -10;
        MAX_VALUE expression;         // e.g. MAX_VALUE = +10; MAX_VALUE1 = -5;
        SCALING_FACTOR expression;    //
```

```
// enumerated type
ENUMERATED (size)
  {{value,        //
    description,  // text to be displayed when value is taken
    help,         // short text describing the value
  }}
```

```
BIT_ENUM (size)
  {{value         // in reality, bit position in word, not octet
    description   // text to be displayed when bit is set
    help,         // short text describing the bit
    function,     // functional class (see CLASS)
    status_class, // cause, duration, correctability, scope, output, miscellaneous
    methods       // method to be performed when bit is set.
  }}
```

DDL Variables (Strings)

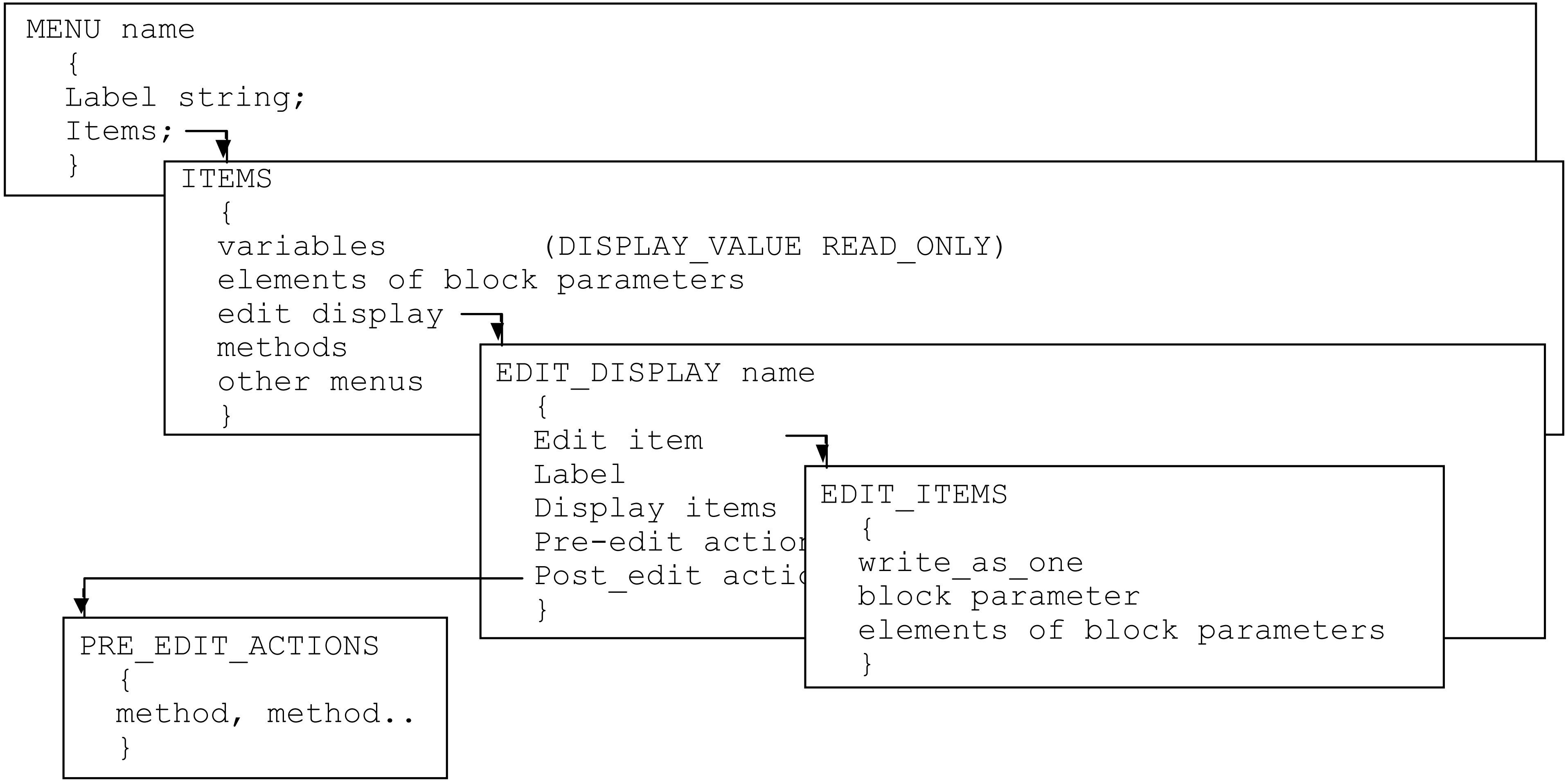
```
// string types
EUC (size);
ASCII (size);
PASSWORD (size);
BITSTRING (length);  // number of bits
VISIBLE (size);
OCTET(size);
```

```
// index type
INDEX (size) item_array;
  // size in octets >1, default 1.
  // item_array see item array
```

```
// data/time types
DATE_AND_TIME;
TIME;
DURATION;
TIME_VALUE;
```

DDL Menu Items

Menu items define screen windows - implementation is free but order is prescribed.



```
MENU name
  {
  Label string;
  Items;
  }
```

```
ITEMS
  {
  variables         (DISPLAY_VALUE READ_ONLY)
  elements of block parameters
  edit display
  methods
  other menus
  }
```

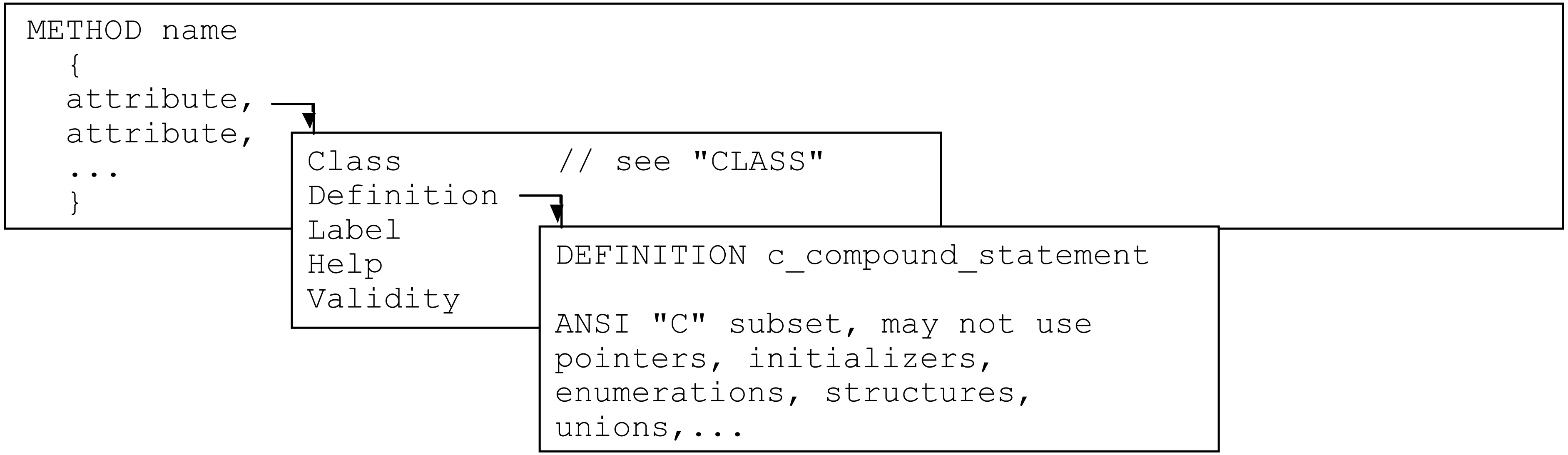
```
EDIT_DISPLAY name
  {
  Edit item
  Label
  Display items
  Pre-edit actio
  Post_edit actic
  }
```

```
EDIT_ITEMS
  {
  write_as_one
  block parameter
  elements of block parameters
  }
```

```
PRE_EDIT_ACTIONS
  {
  method, method..
  }
```

DDL Methods

Methods are piece of code to be executed by the host in response to change of device variables or user commands



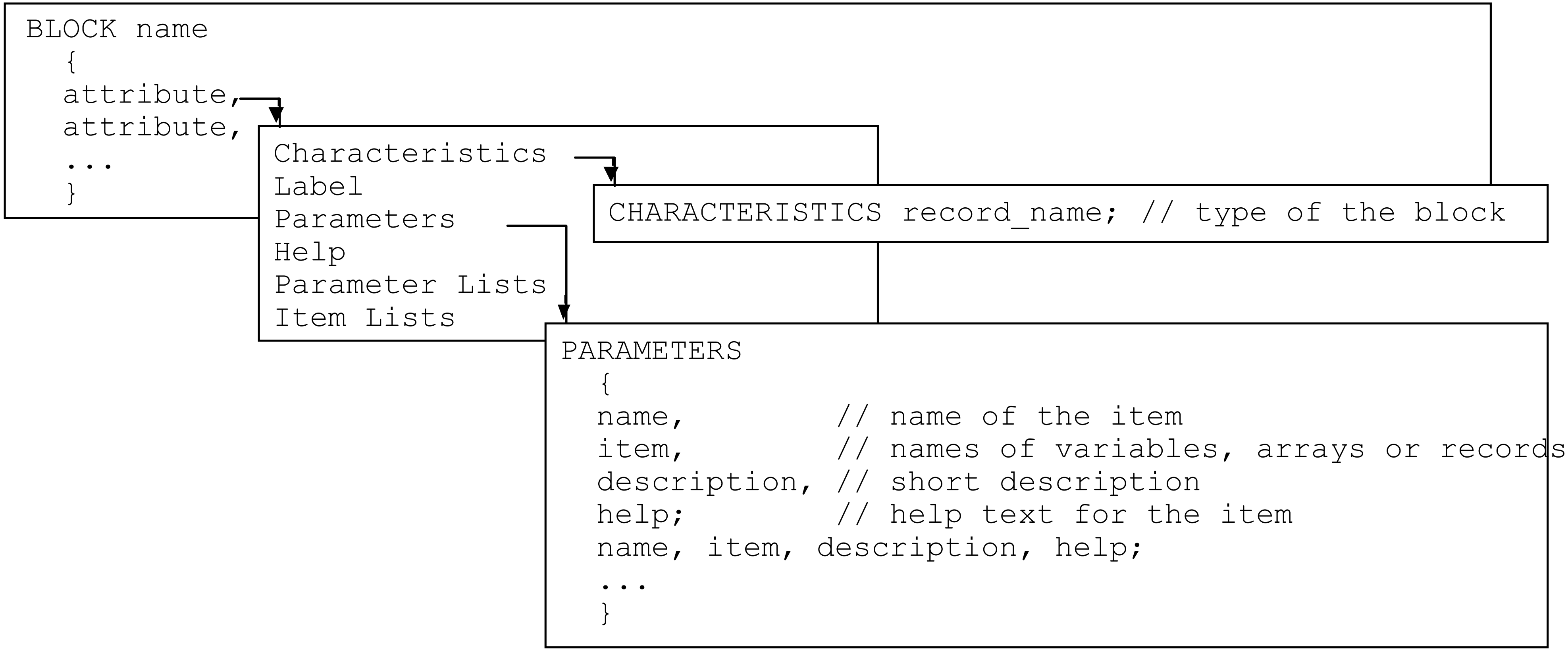
```
METHOD name
  {
  attribute,
  attribute,
  ...
  }
```

```
Class          // see "CLASS"
Definition
Label
Help
Validity
```

```
DEFINITION c_compound_statement

ANSI "C" subset, may not use
pointers, initializers,
enumerations, structures,
unions,...
```

DDL Blocks

Blocks are segments of Function Block Language defined in FMS



```
BLOCK name
  {
  attribute,
  attribute,
  ...
  }
```

```
Characteristics
Label
Parameters
Help
Parameter Lists
Item Lists
```

```
CHARACTERISTICS record_name; // type of the block
```

```
PARAMETERS
  {
  name,         // name of the item
  item,         // names of variables, arrays or records
  description,  // short description
  help;         // help text for the item
  name, item, description, help;
  ...
  }
```

Industrial Automation
Automation Industrielle
Industrielle Automation

4. Device Management Protocols
Protocoles de gestion des appareils
Gerätezugangsprotokolle

4.2 FDT - DTM

Prof. Dr. H. Kirrmann

ABB Research Center, Baden, Switzerland

FDT - DMT purpose

integrate field devices (sensors and actors) of different manufacturers in any control or engineering system

device description file (HART, GSD) -> device type manager (DTM)

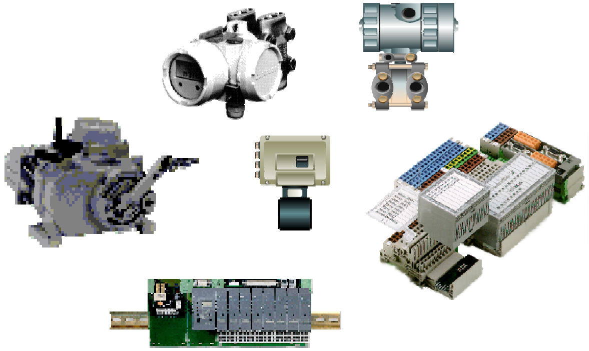
DTM is a software module (a kind of driver) that comes with each device.
DTM encapsulates the device's configuration, functions, parameters and describes the user interface

Field Device Configuration

> 10 Device Suppliers

> 100 Device Types

> 10.000 I/Os

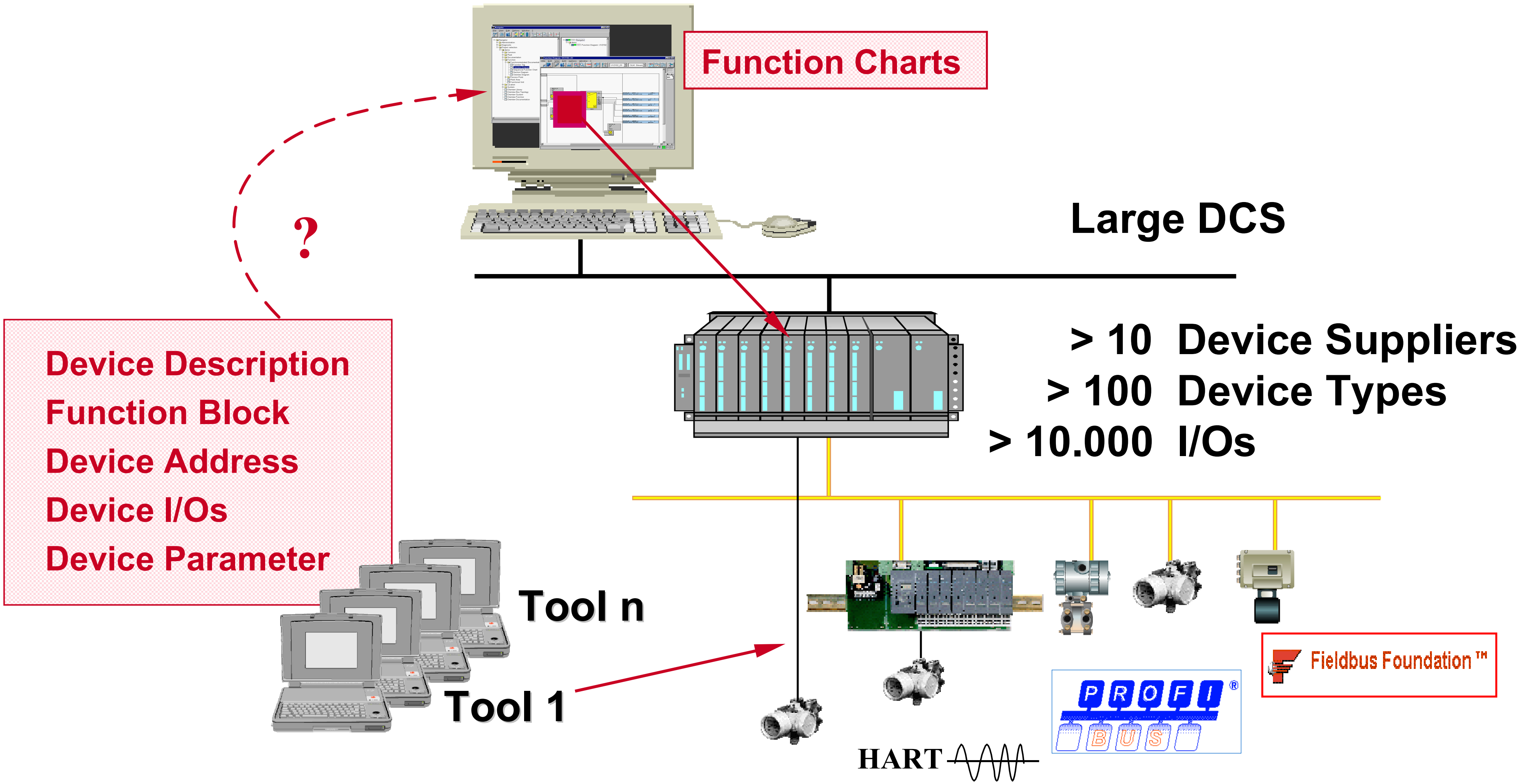


HART

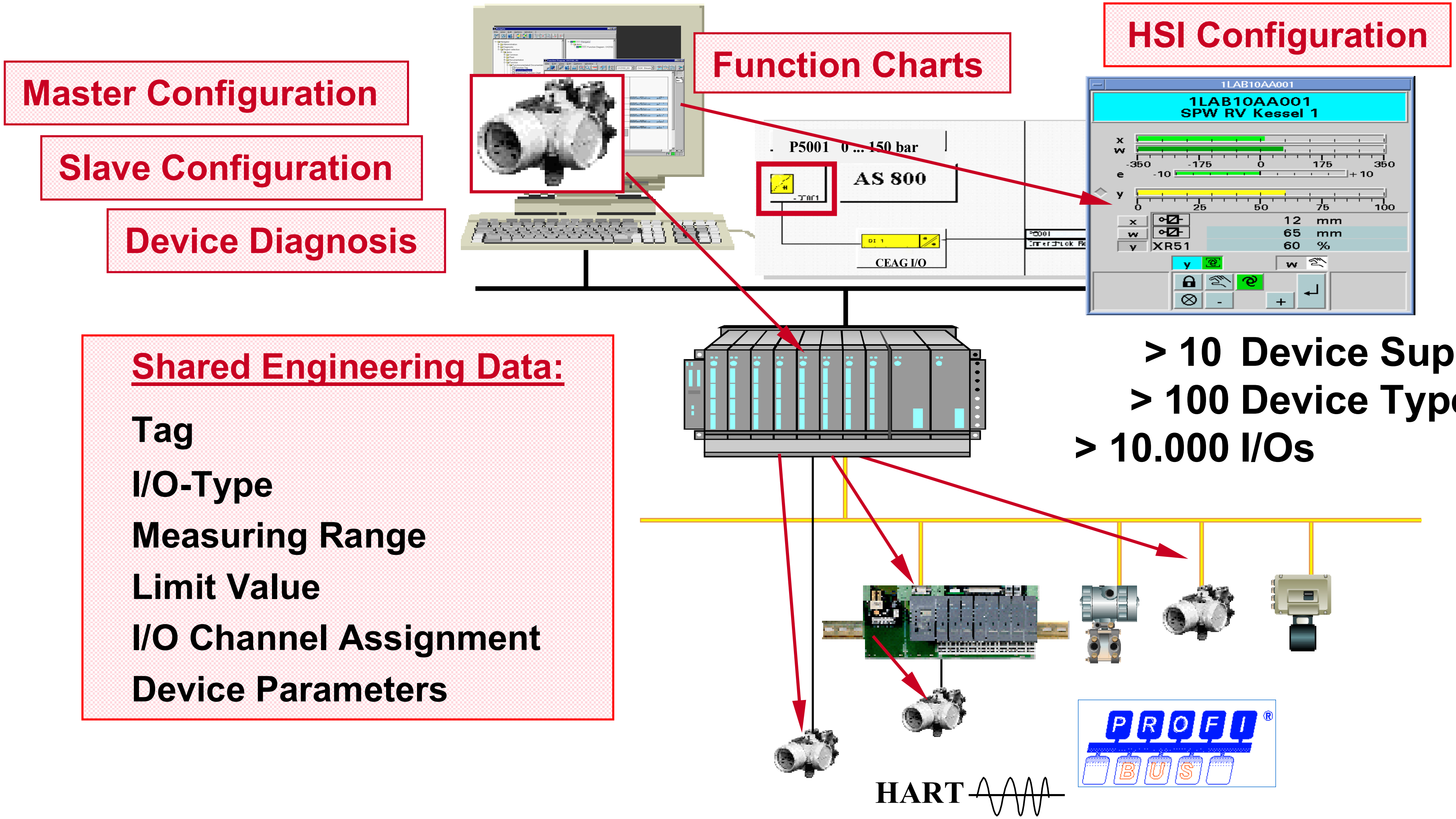
PROFIBUS

Fieldbus Foundation™

The Situation of Today



The Goal



Master Configuration

Slave Configuration

Device Diagnosis

Function Charts

HSI Configuration

Shared Engineering Data:

Tag

I/O-Type

Measuring Range

Limit Value

I/O Channel Assignment

Device Parameters

> 10 Device Suppliers

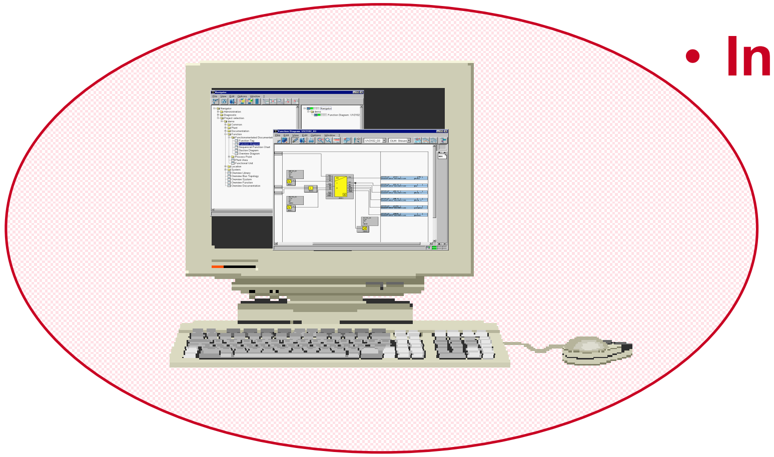
> 100 Device Types

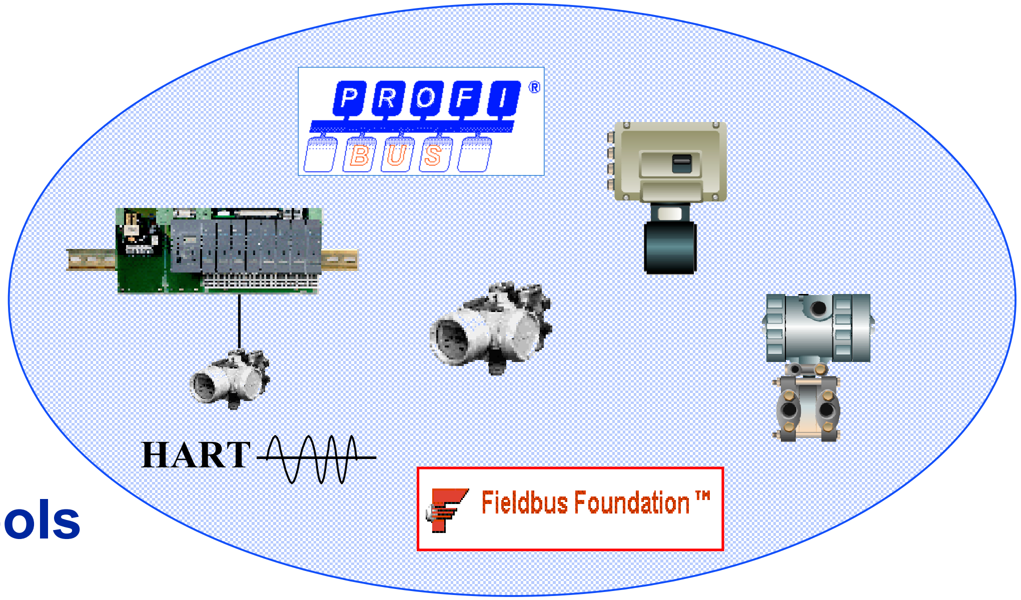
> 10.000 I/Os

Overall Requirements

- Consistent, plant-wide configuration of DCS, Fieldbus and devices
- Integrated device configuration and documentation
- Device integration with smallest effort

DCS



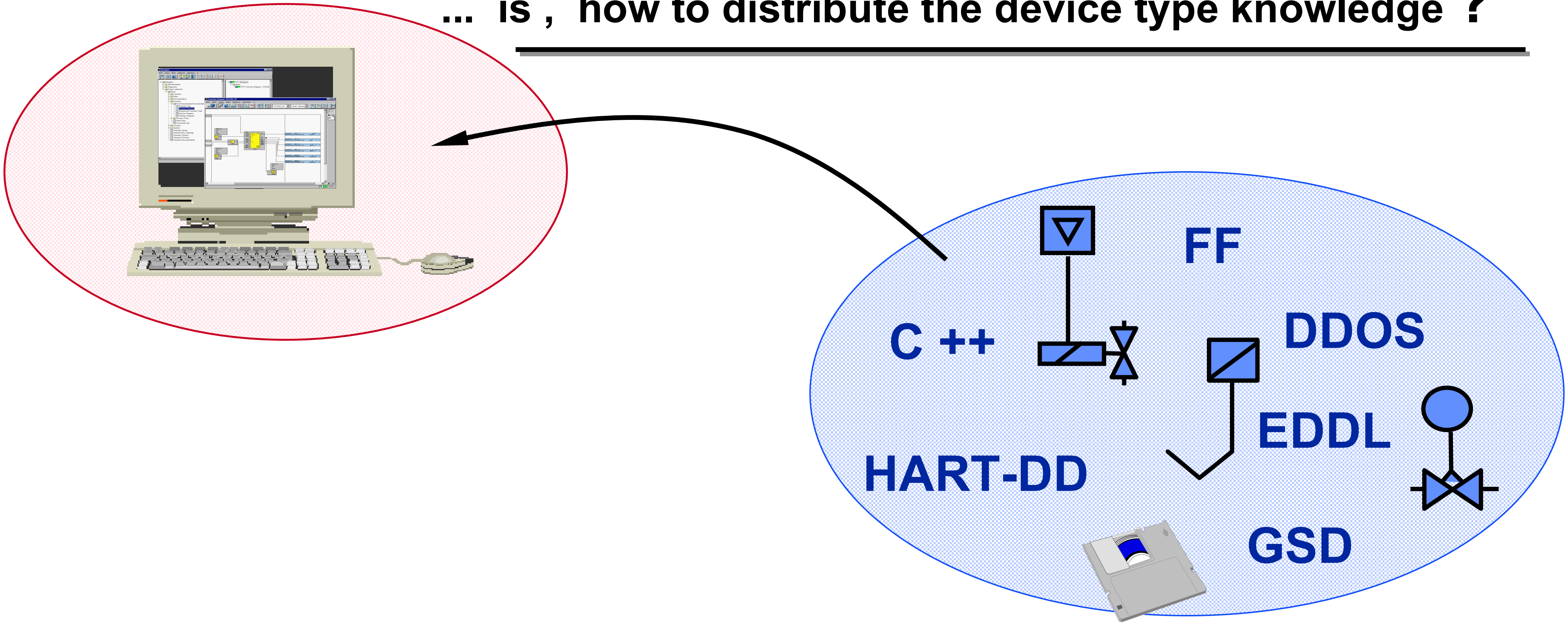


Devices

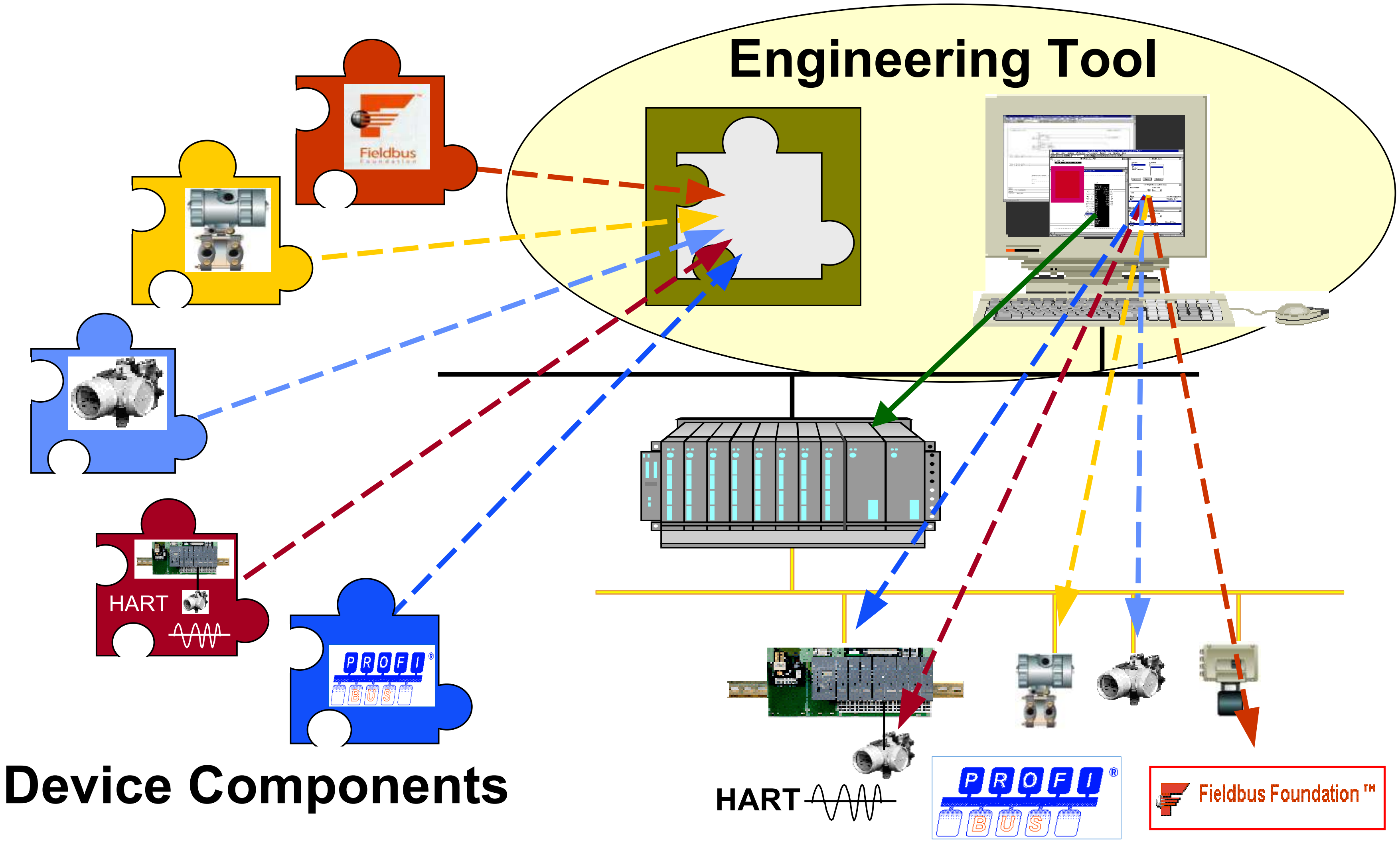
- Integration of devices in all available tools
- Tool support of individual device features

The Still Open Question ...

... is , how to distribute the device type knowledge ?

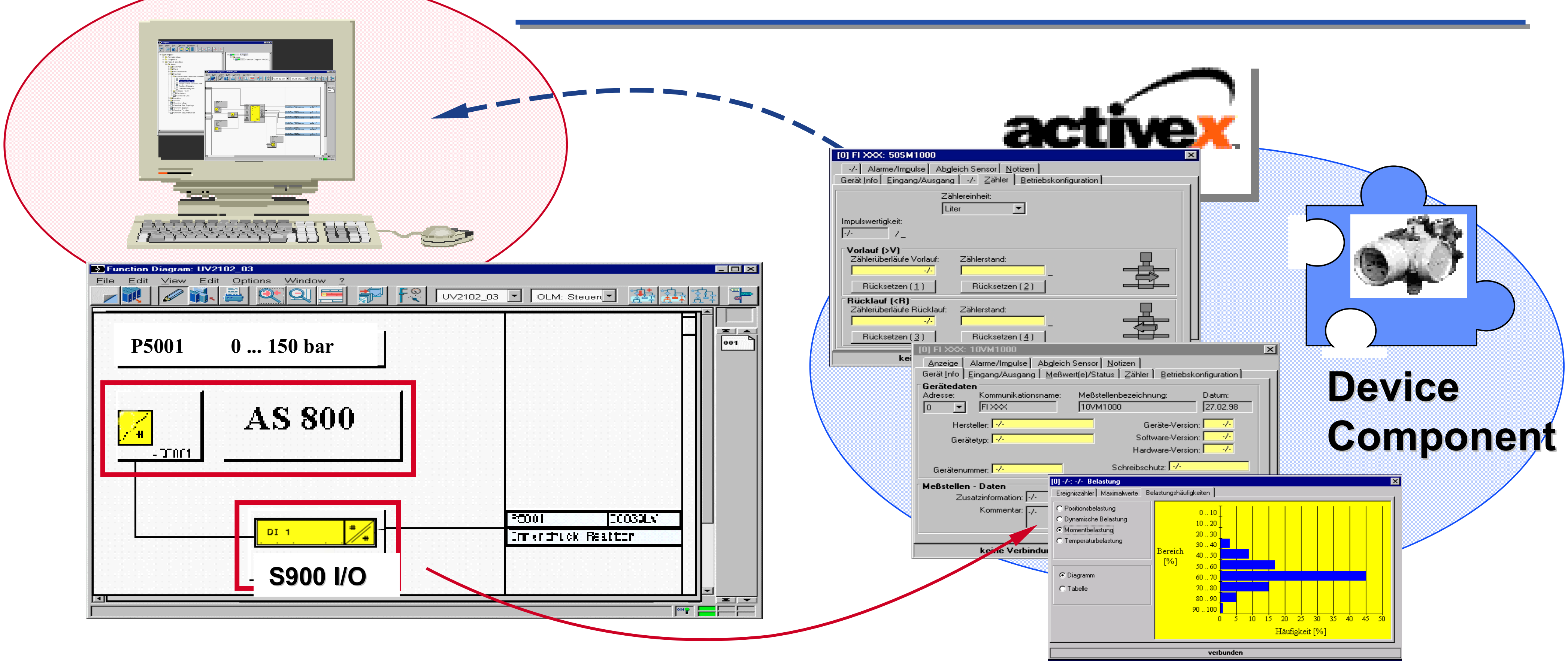


Component architecture



Device Component

Device Configuration and Maintenance



[0] FI XXX: 50SM1000

-/- | Alarme/Impulse | Abgleich Sensor | Notizen
Gerät Info | Eingang/Ausgang | -/- | Zähler | Betriebskonfiguration

Zählereinheit: Liter

Impulswertigkeit: -/- / _

Vorlauf (>V)

Zählerüberläufe Vorlauf: -/-
Zählerstand: _

Rücksetzen (1) Rücksetzen (2)

Rücklauf (<R)

Zählerüberläufe Rücklauf: -/-
Zählerstand: _

Rücksetzen (3) Rücksetzen (4)

[0] FI XXX: 10VM1000

Anzeige | Alarme/Impulse | Abgleich Sensor | Notizen
Gerät Info | Eingang/Ausgang | Meßwert(e)/Status | Zähler | Betriebskonfiguration

Gerätedaten

Adresse: 0
Kommunikationsname: FI XXX
Meßstellenbezeichnung: 10VM1000
Datum: 27.02.98

Hersteller: -/-
Gerätetyp: -/-
Geräte-Version: -/-
Software-Version: -/-
Hardware-Version: -/-
Gerätenummer: -/-
Schreibschutz: -/-

Meßstellen - Daten

Zusatzinformation: -/-
Kommentar: -/-

keine Verbindung

[0] -/-: -/- Belastung

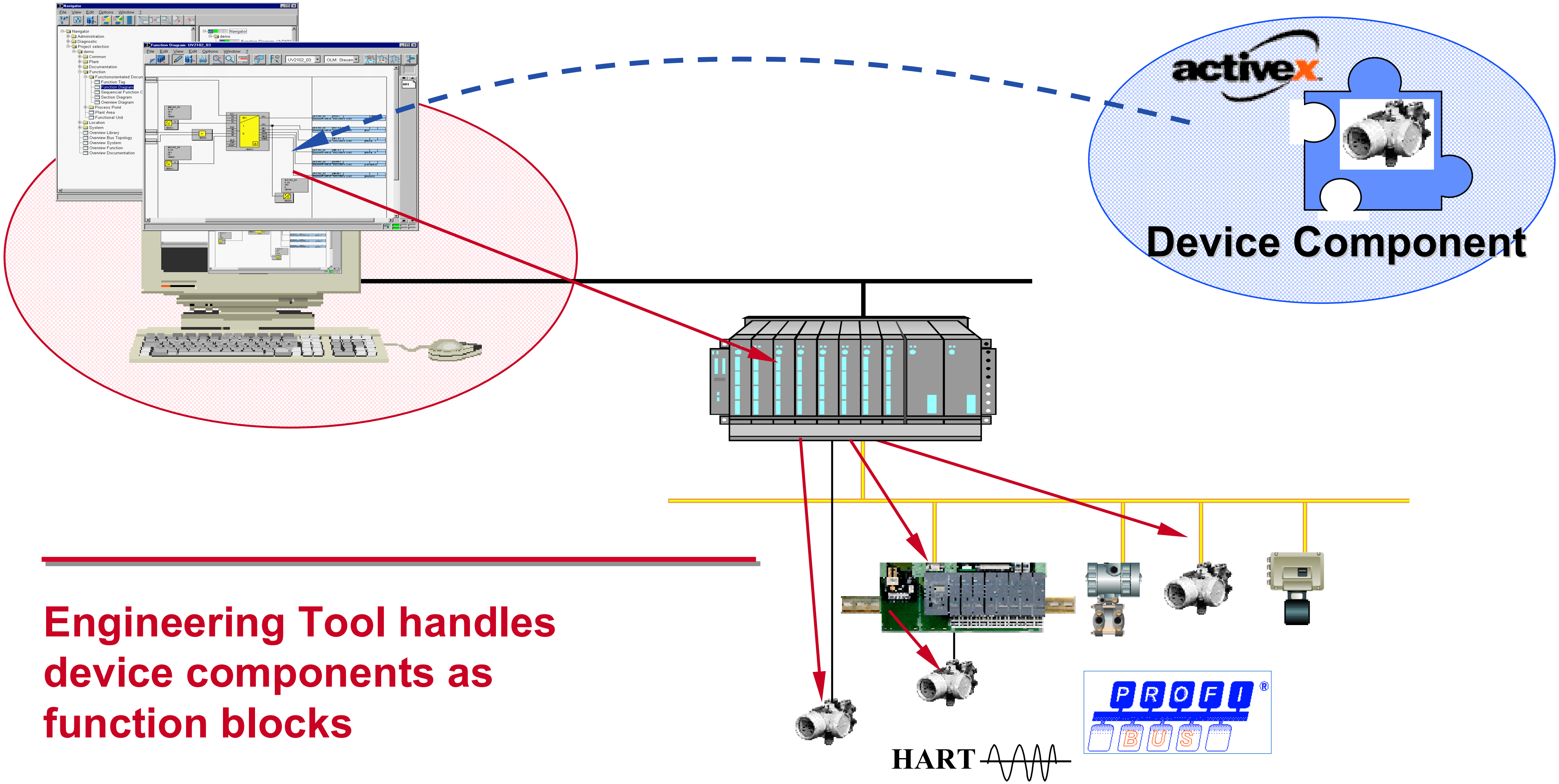
Ereigniszähler | Maximalwerte | Belastungshäufigkeiten

○ Positionsbelastung
○ Dynamische Belastung
● Momentbelastung
○ Temperaturbelastung

● Diagramm
○ Tabelle

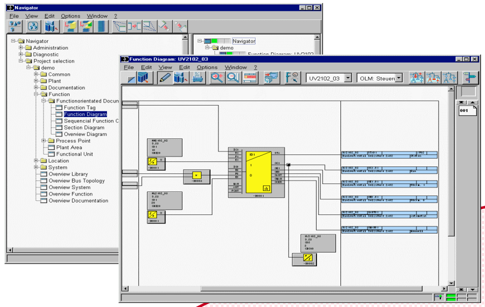
verbunden

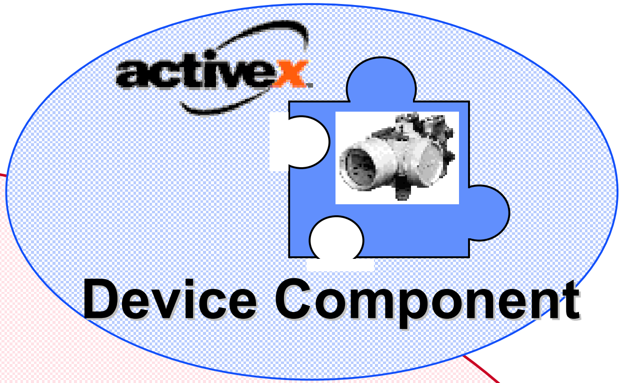
Engineering Environment for Components



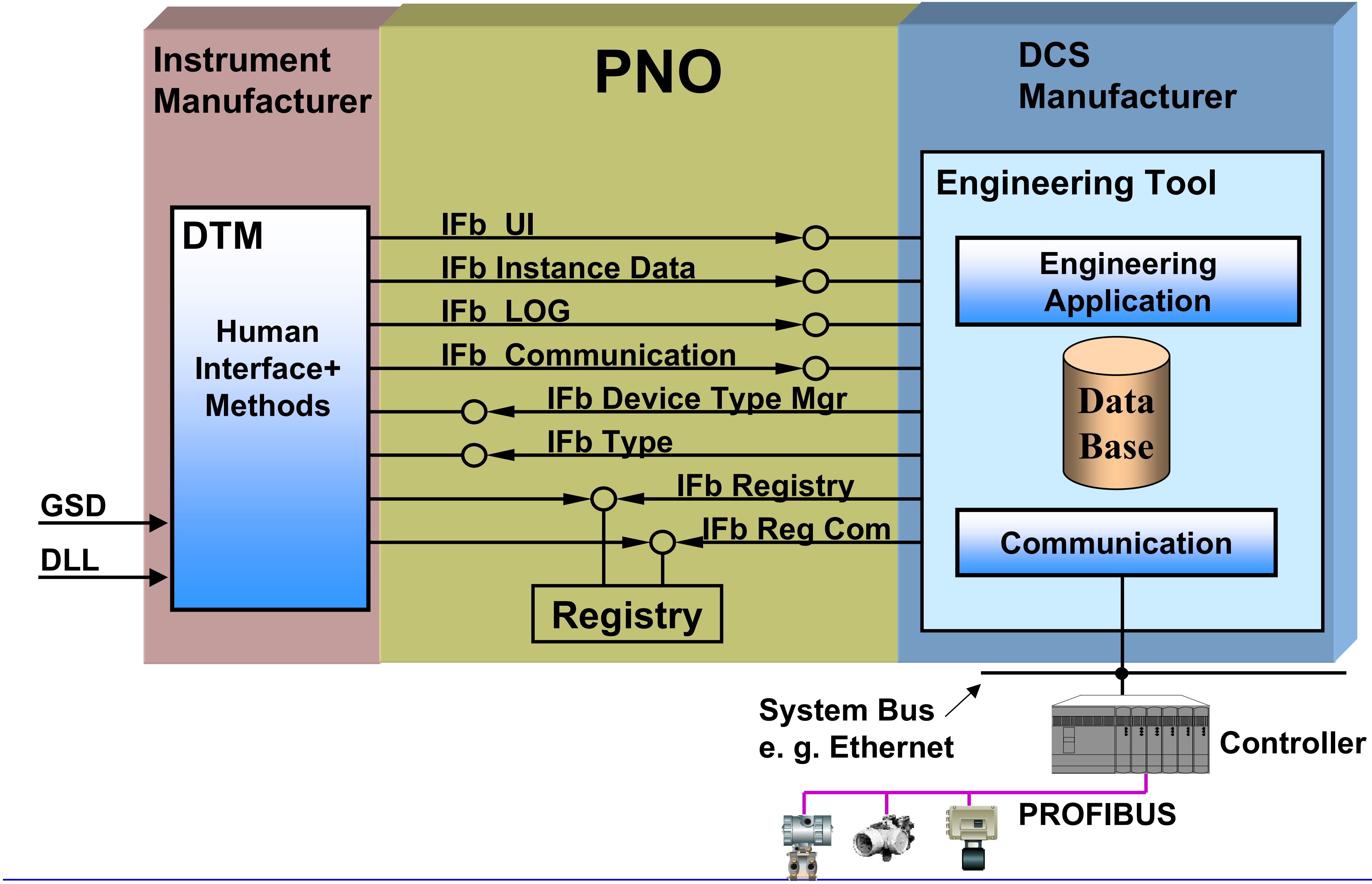
Engineering Tool handles device components as function blocks

Engineering Environment for Components





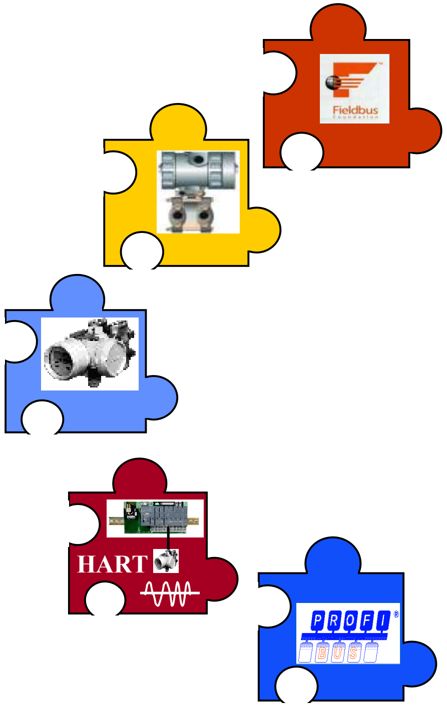
- no device specific knowledge required
- manages all device instances and stores all instance data
- offers individual communication and routing services via DCS-system.
- controls plant-wide consistent configuration
- offers data life cycle control for device parameters
- has multi user and server/client architecture


Instrument Manufacturer
PNO
DCS Manufacturer
Engineering Tool
DTM
Human Interface+ Methods
IFb UI
IFb Instance Data
IFb LOG
IFb Communication
IFb Device Type Mgr
IFb Type
IFb Registry
IFb Reg Com
Engineering Application
Data Base
Communication
GSD
DLL
Registry
System Bus
e. g. Ethernet
Controller
PROFIBUS

The Device Component ...

... belongs to a device

... is supplied with the device by the device manufacturer



... is no standalone tool

... is an ActiveX Component (COM/DCOM)

... has COM Interfaces as specified by PNO/ZVEI Working Group

... is called “ Device Type Manager “ (DTM)

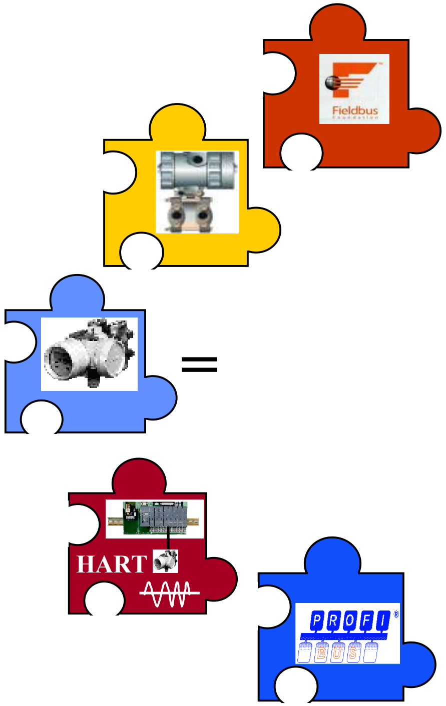
The Device Type Manager ...

... knows all business rules of the device

... contains all user dialogs

... does the device configuration and diagnosis

... generates the device specific documentation



... has no data storage capability

... has no direct device link to any device

... does not know anything about the engineering environment

The Device Type Manager ...

... supports one or several devices

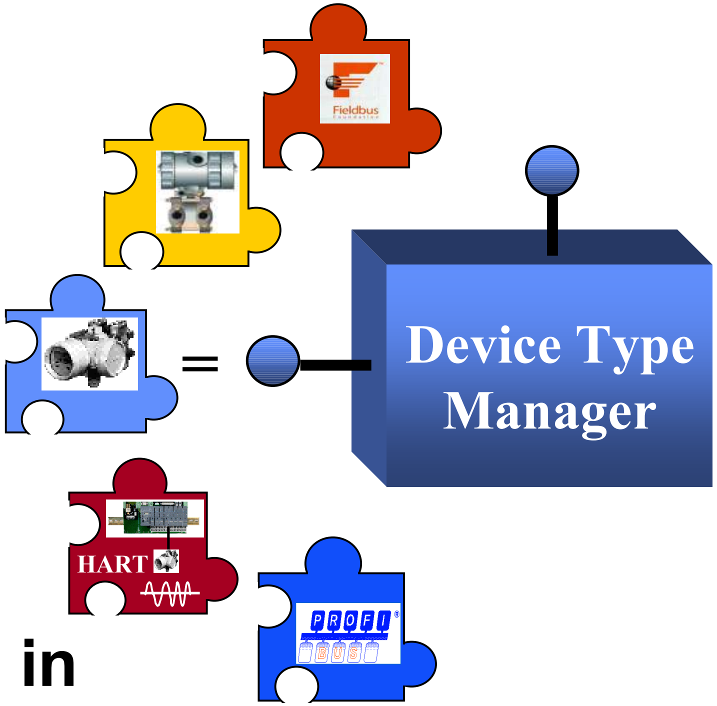
... could be created from existing device tools

... could be automatically compiled from Device Descriptions

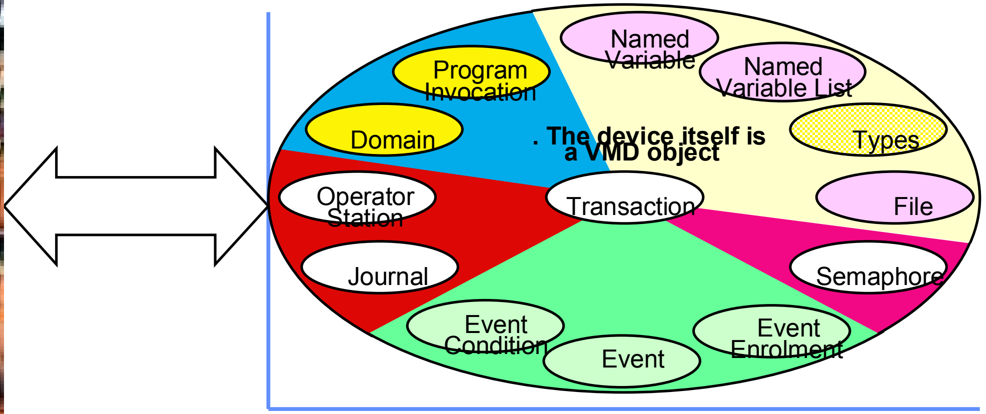
... could be developed from scratch

... should be an ActiveX Control for use in WebBrowser

... should be designed according to Microsoft Multi Layer Architecture



Industrial Automation
Automation Industrielle
Industrielle Automation



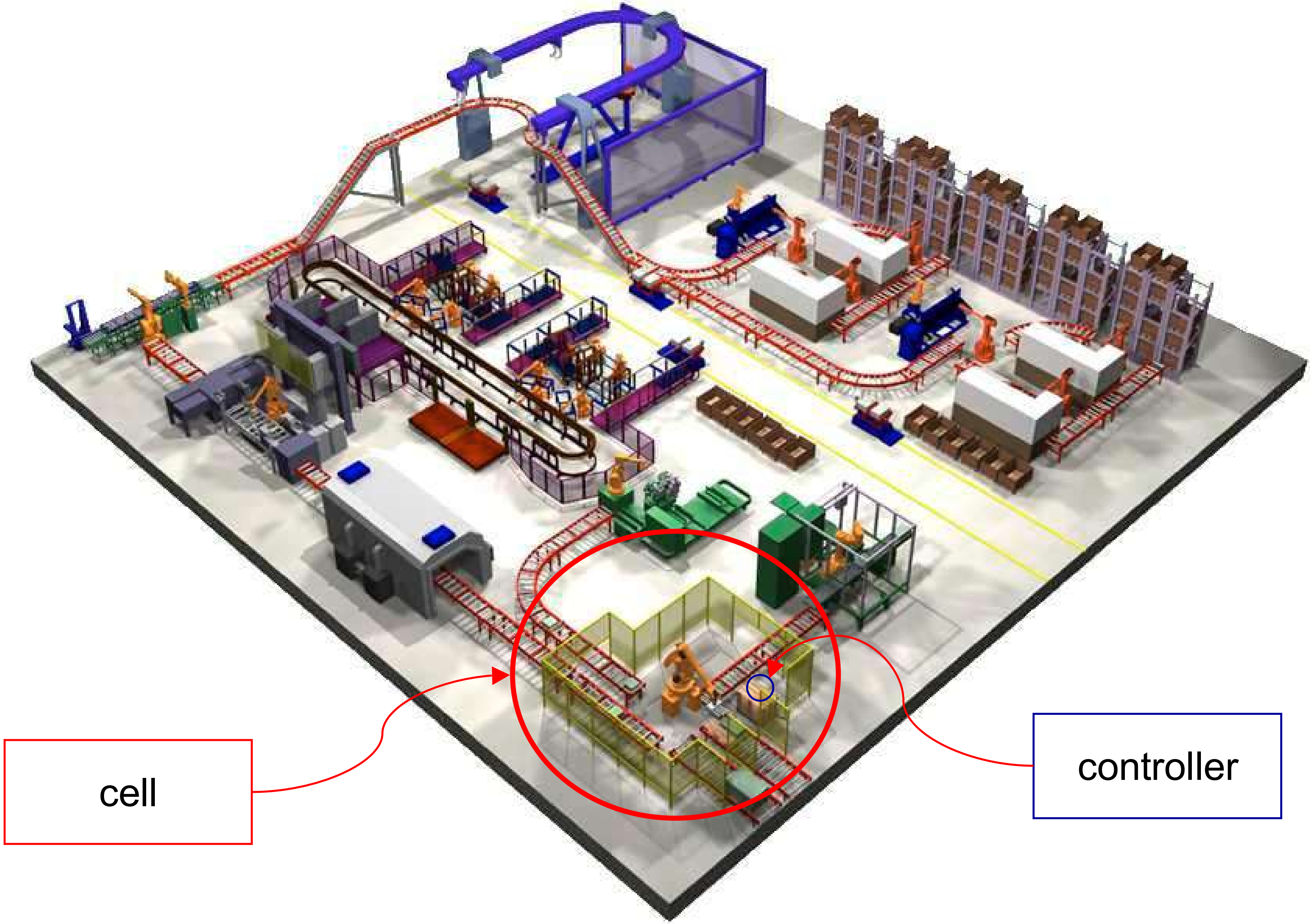
4. Device Management Protocols
Protocoles de gestion des appareils
Gerätezugangsprotokolle

4.2 MMS - Manufacturing Message Specifications

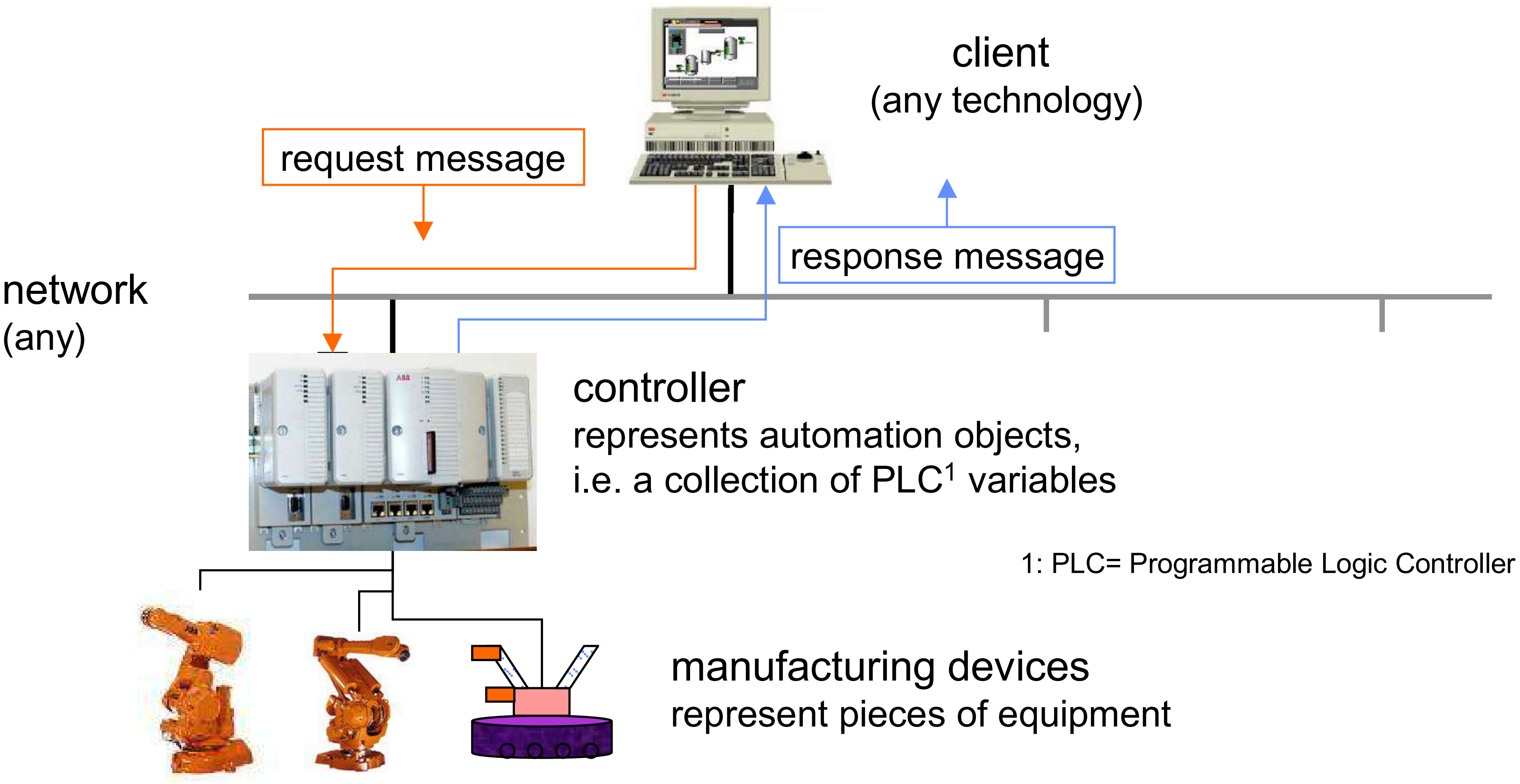
Prof. Dr. H. Kirrmann
ABB Research Center, Baden, Switzerland

2005 April, HK

MMS Application domain



A controller represents pieces of equipment



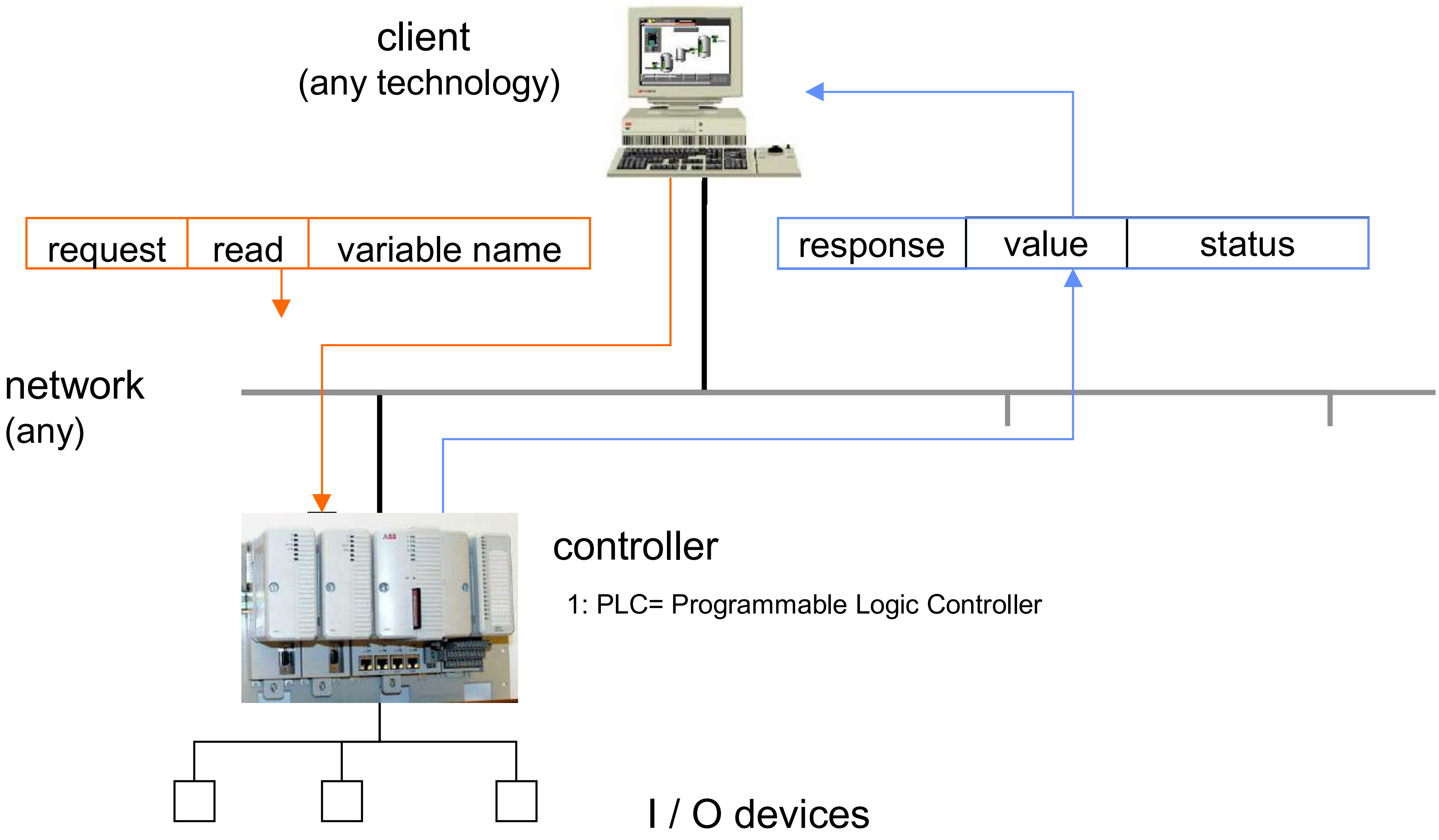
represents automation objects,
i.e. a collection of PLC[1] variables

represent pieces of equipment

[1] PLC= Programmable Logic Controller

Accessing variables that represent automation objects require
a standard model that defines the objects , called a virtual manufacturing device

The basic MMS idea



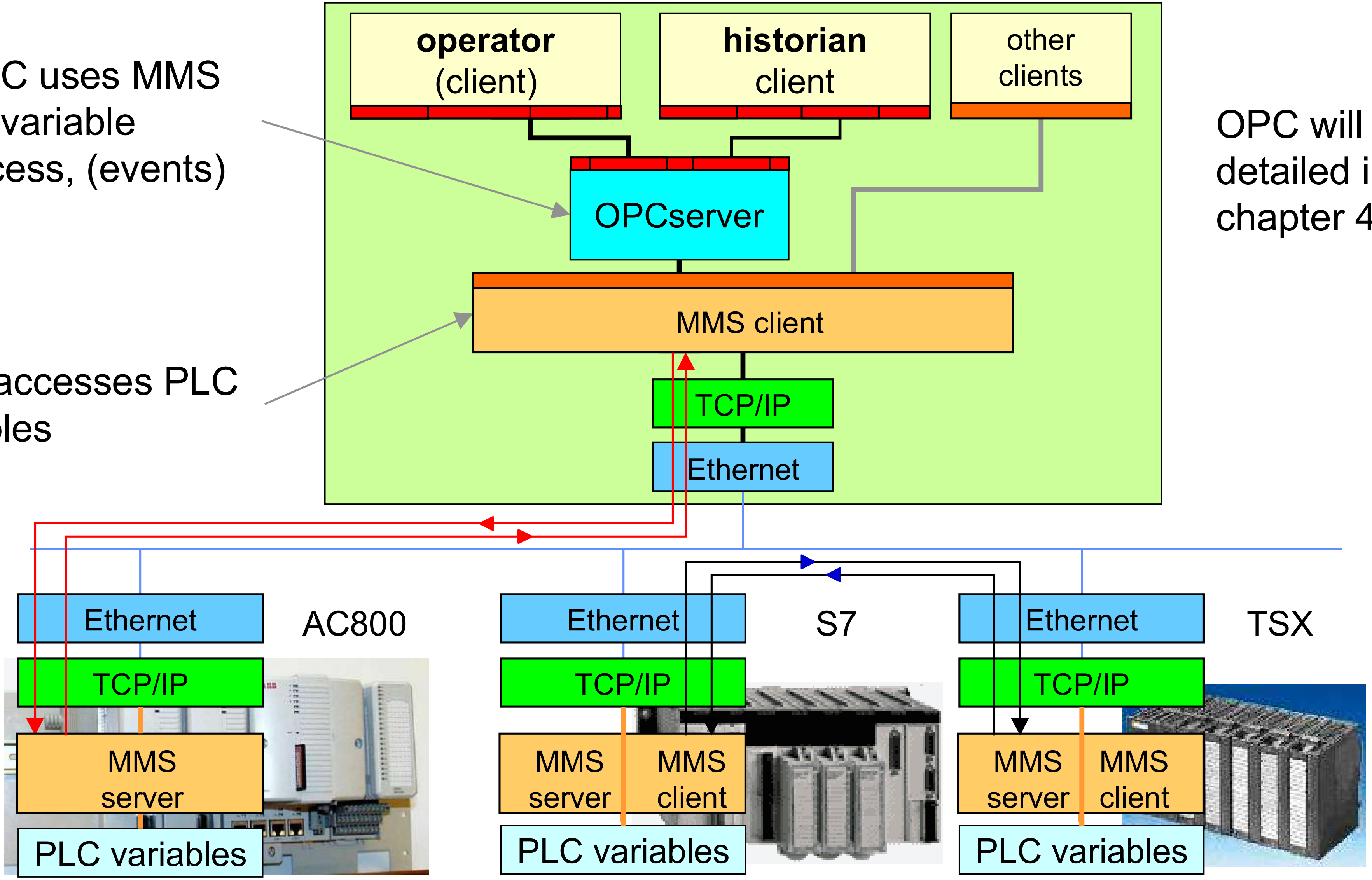
basic MMS idea: read and write equipment variables using standard messages.

Application: MMS for OPC

OPC uses MMS for variable access, (events)

MMS accesses PLC Variables

OPC will be detailed in chapter 4.3



intention: any PLC should be accessed that way (MMS as universal server)

MMS - Manufacturing Message Specification history

Developed 1980 (!) for the MAP project (General Motor’s flexible manufacturing initiative)

Originally unluckily tied to the OSI communication stack and Token Bus (IEEE 802.4)

Reputed for being heavy, complicated and costly due to poor implementations.

Boeing adopted MMS as TOPs (MMS on Ethernet) - a wise step.

Adopted by the automobile industry, aerospace industry, and PLC manufacturers:
Siemens, Schneider, Daimler, ABB.

Standardized since 1990 as:

[1] ISO/IEC 9506-1 (2003): Industrial Automation systems -
Manufacturing Message Specification -
Part 1: Service Definition

[2] ISO/IEC 9506-2 (2003): Industrial Automation systems -
Manufacturing Message Specification -
Part 2: Protocol Specification

MMS - Concept

MMS (Manufacturing Message Specifications) defines:

- A set of standard objects which must exist in every conformant device, on which operations can be executed (examples: read and write local variables, signal events...)

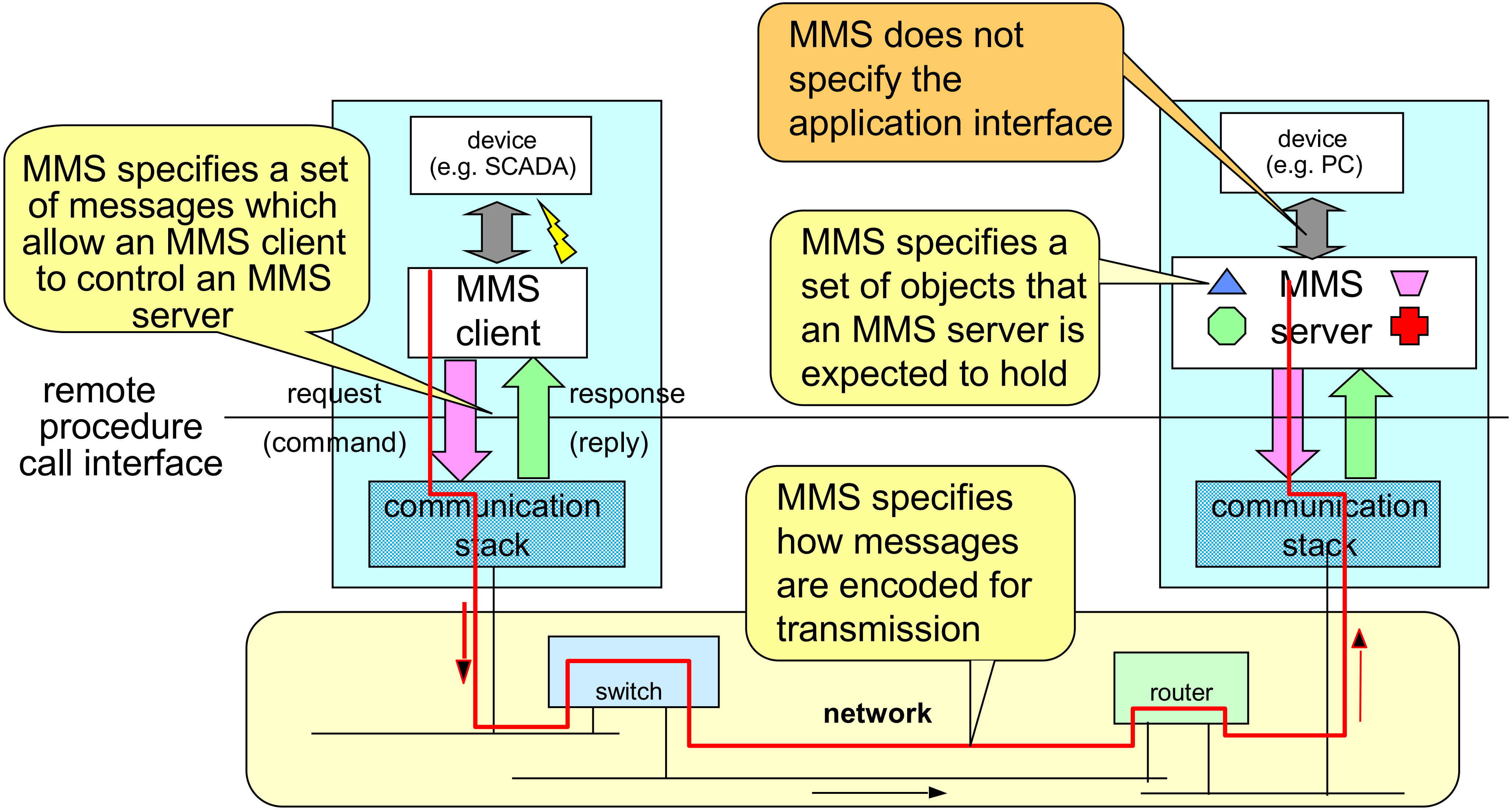
- A set of standard messages exchanged between a manager and an agent station for the purpose of controlling these objects

- A set of protocols (rules for exchanging messages between devices)

- A set of encoding rules for these messages (how values and parameters are mapped to bits and bytes when transmitted)

MMS does not specify application-specific operations (e.g. change motor speed).
This is covered by application-specific, “companion standards”
(e.g. flexible manufacturing, drives, remote meter reading, ...)

MMS - Communication model

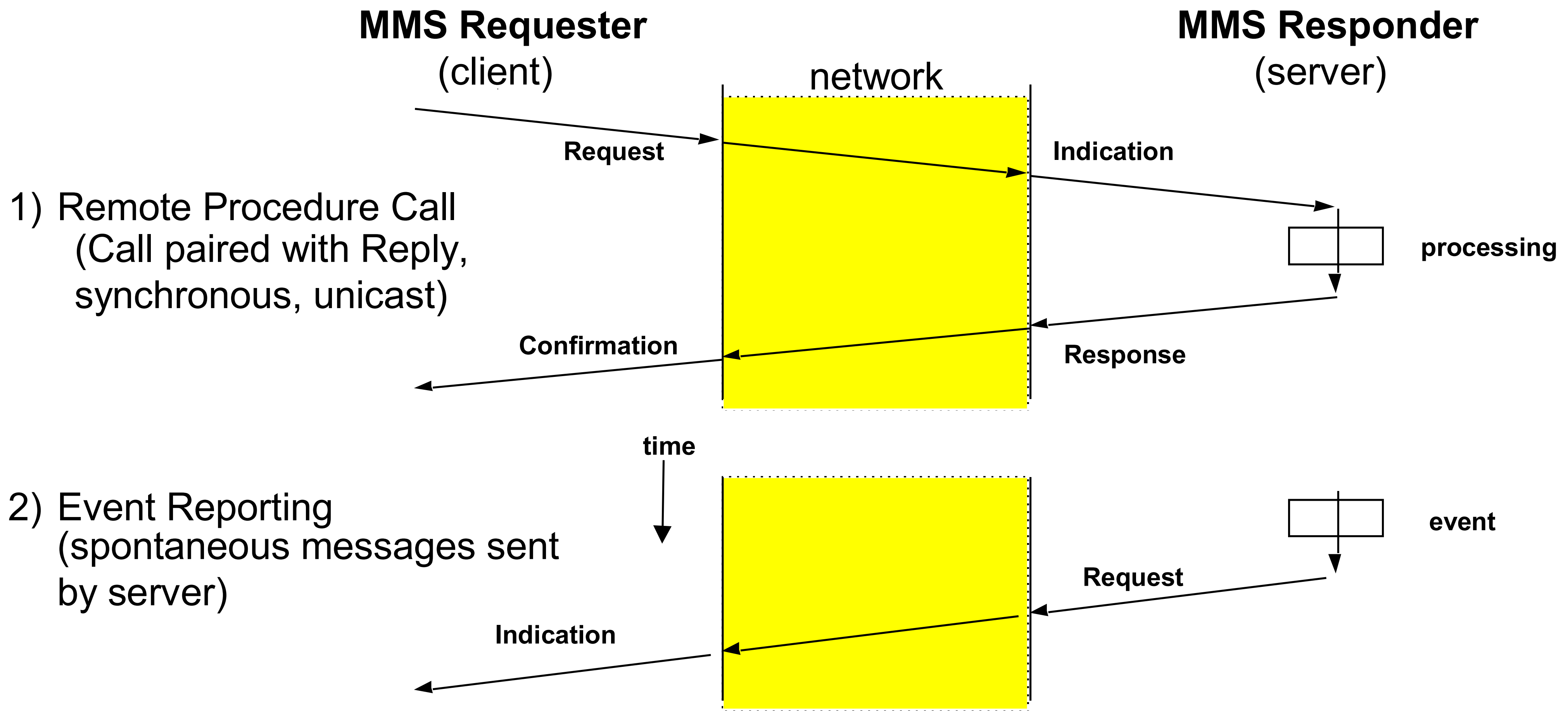

MMS specifies a set of messages which allow an MMS client to control an MMS server
MMS does not specify the application interface
MMS specifies a set of objects that an MMS server is expected to hold
MMS specifies how messages are encoded for transmission
device (e.g. SCADA)
MMS client
device (e.g. PC)
MMS server
remote procedure call interface
request
(command)
response
(reply)
communication stack
communication stack
switch
network
router

MMS mapping to communication

MMS is not by itself a communication protocol, it defines messages that have to be transported by an unspecified network

MMS - Underlying Communication Principles

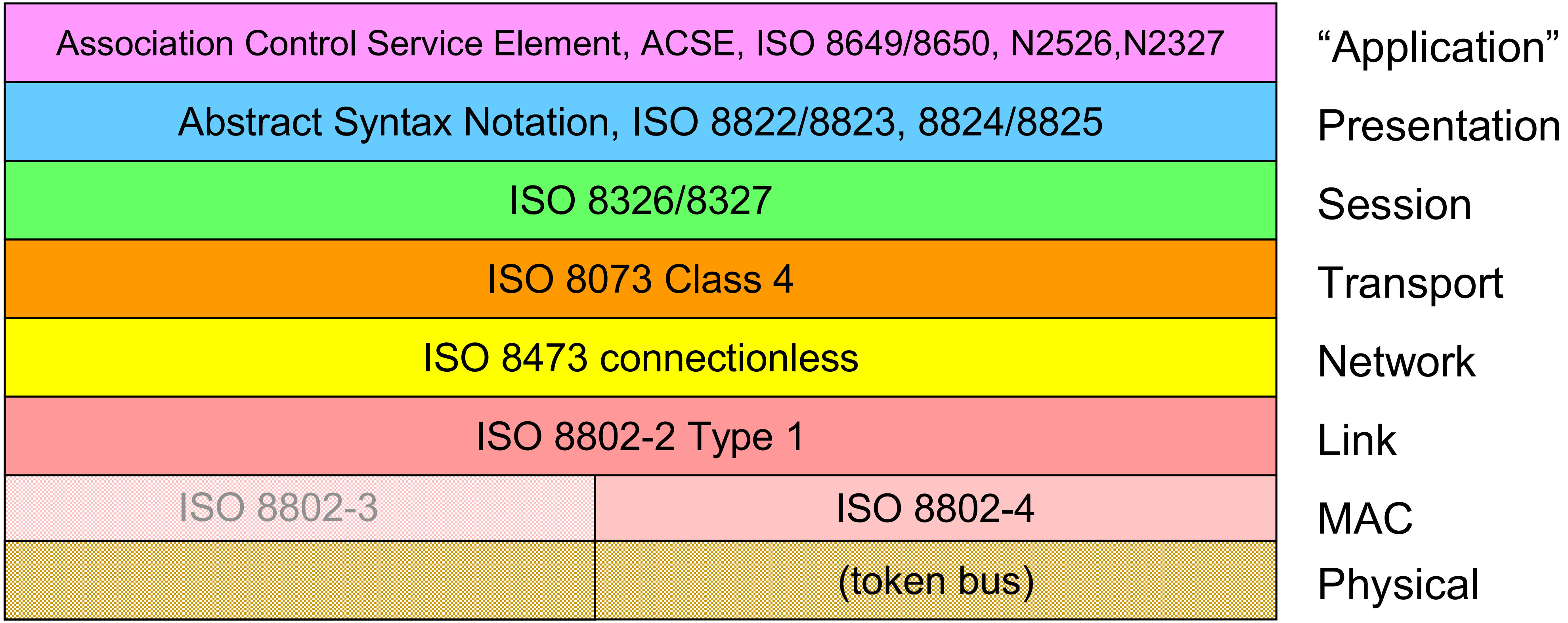
MMS is in principle independent from the communication stack.
MMS only requires that two types of communication services exist:



1) Remote Procedure Call (Call paired with Reply, synchronous, unicast)

2) Event Reporting (spontaneous messages sent by server)

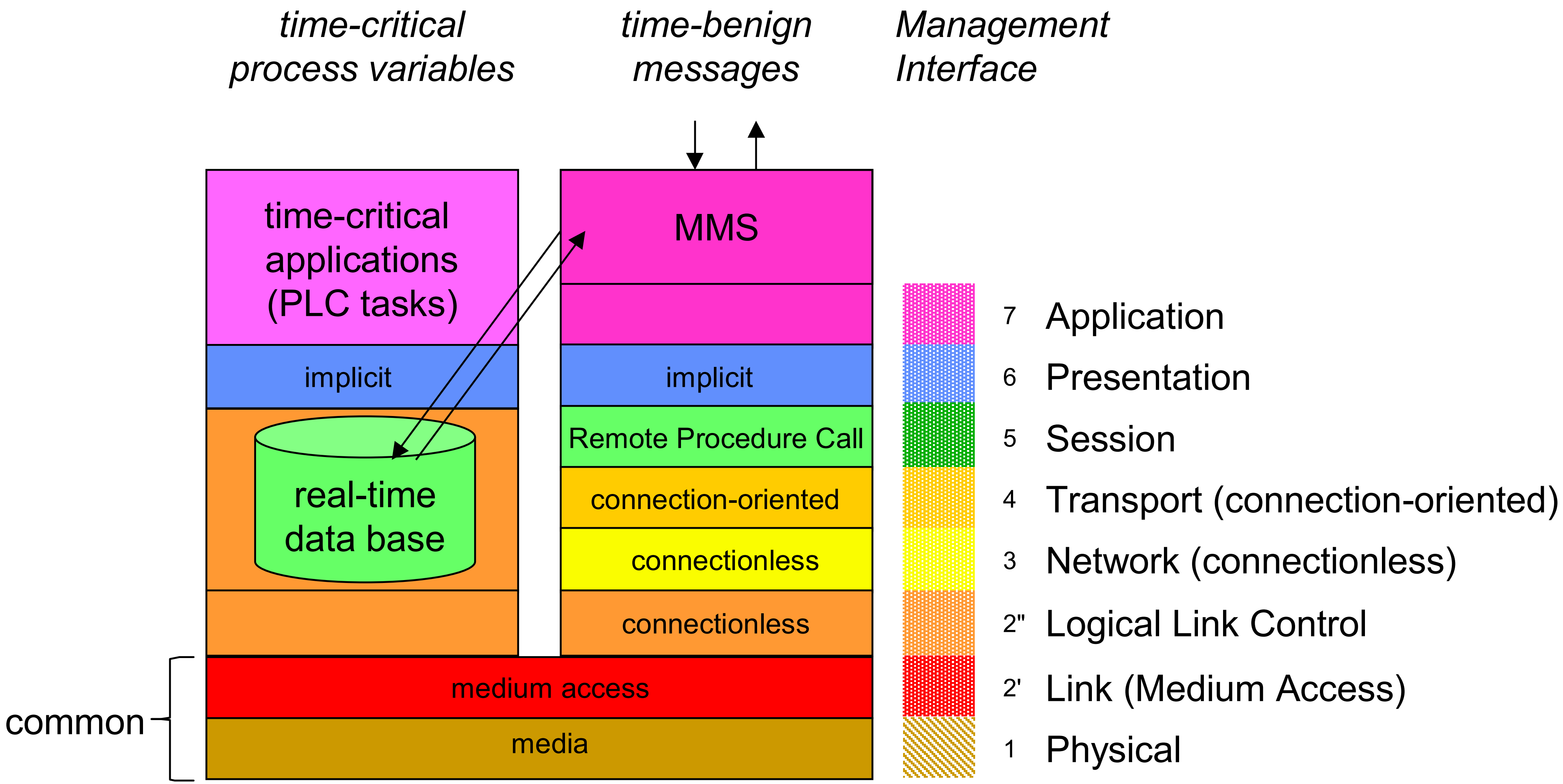
MMS - Original Communication Stack



Association Control Service Element, ACSE, ISO 8649/8650, N2526,N2327		"Application"
Abstract Syntax Notation, ISO 8822/8823, 8824/8825		Presentation
ISO 8326/8327		Session
ISO 8073 Class 4		Transport
ISO 8473 connectionless		Network
ISO 8802-2 Type 1		Link
ISO 8802-3	ISO 8802-4	MAC
	(token bus)	Physical

quite heavy… Boeing decided to drop ISO for TCP/IP, was not followed until 1999...

MMS in the fieldbus stack

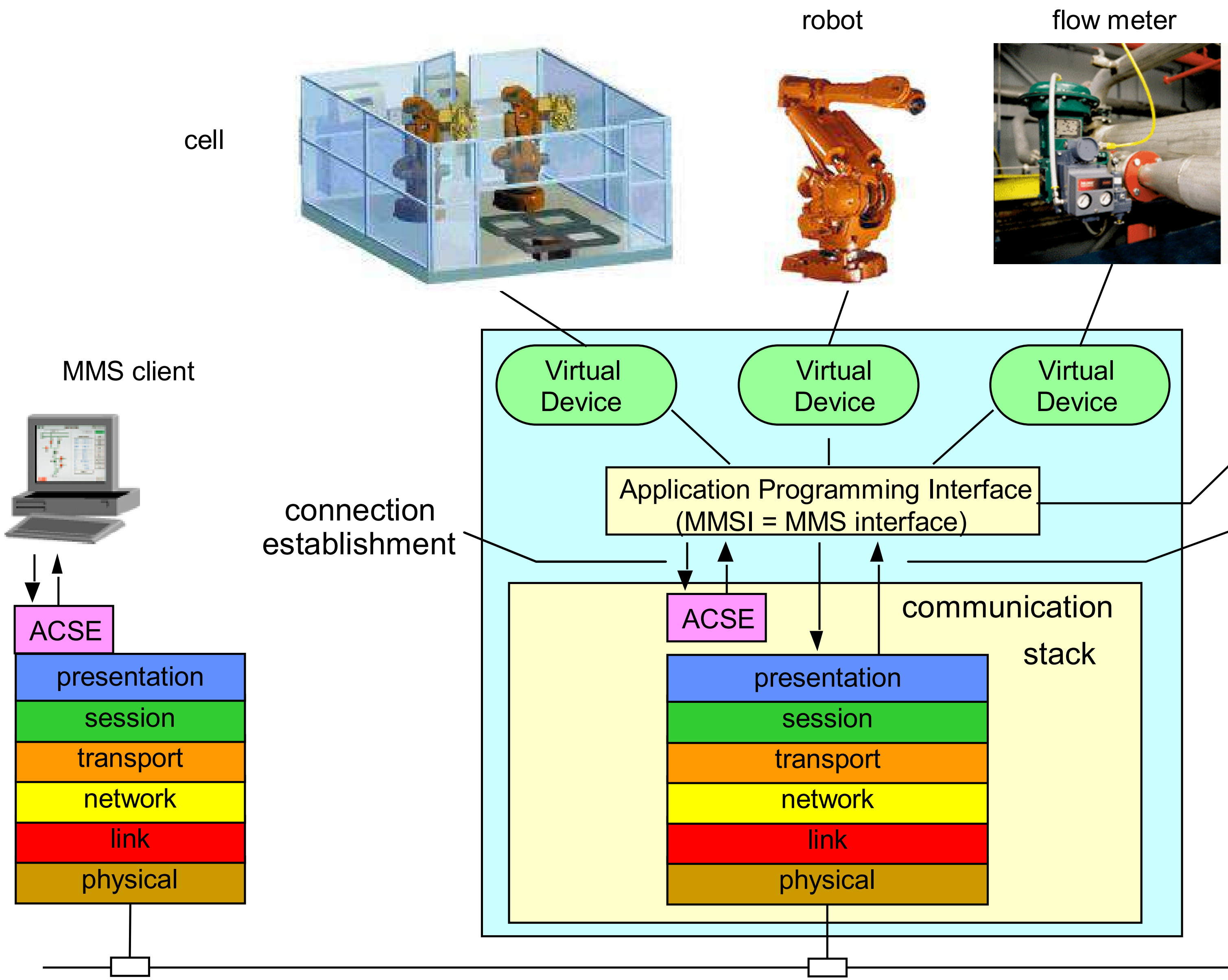


MMS is not for real-time communication, but it can access the real-time variables

MMS Objects

Each MMS server is expected to contain a number of standard objects

MMS - Concept of Virtual Manufacturing Device (VMD)



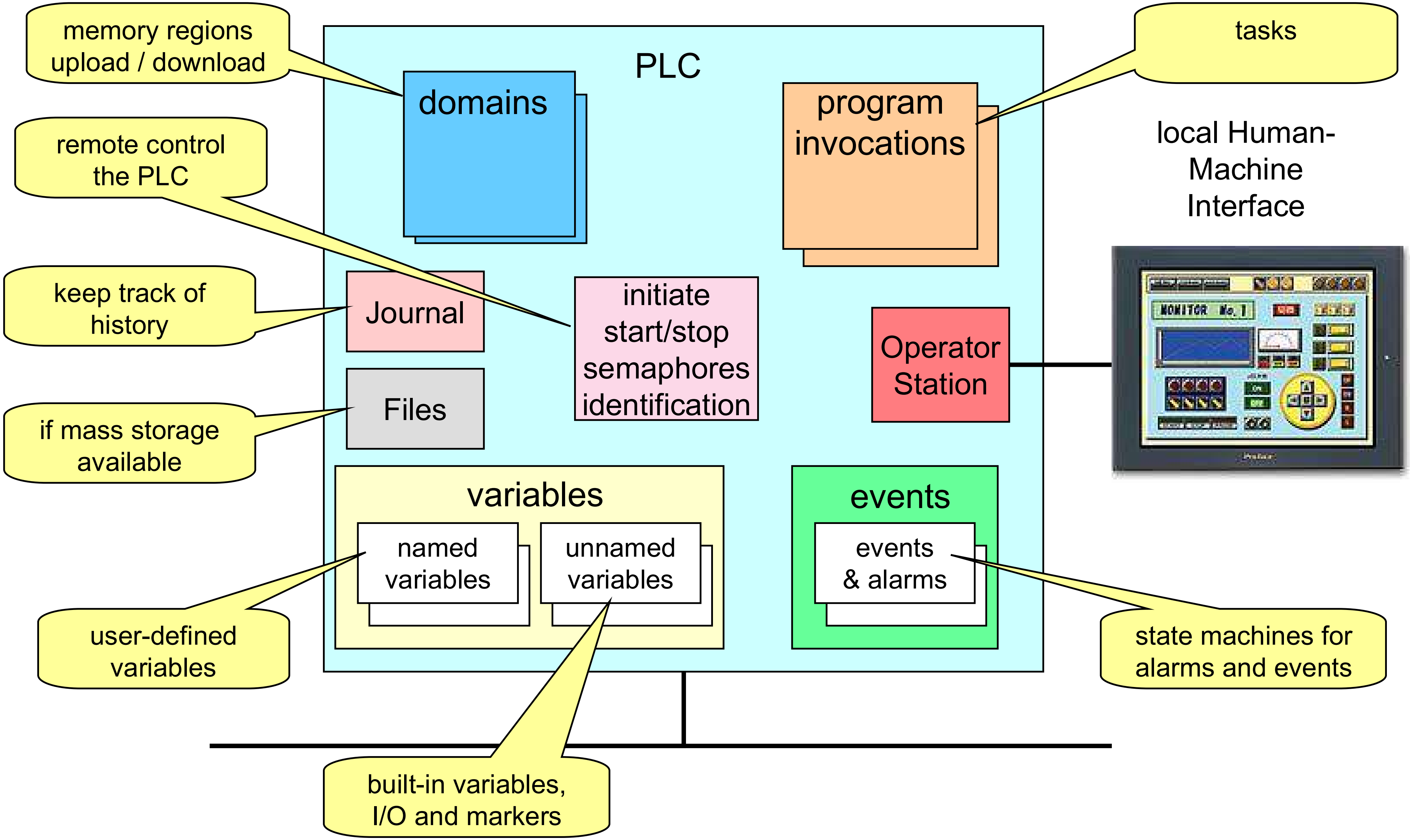
A virtual device represents a (complex) piece of equipment

Application-specific object models

MMS messages

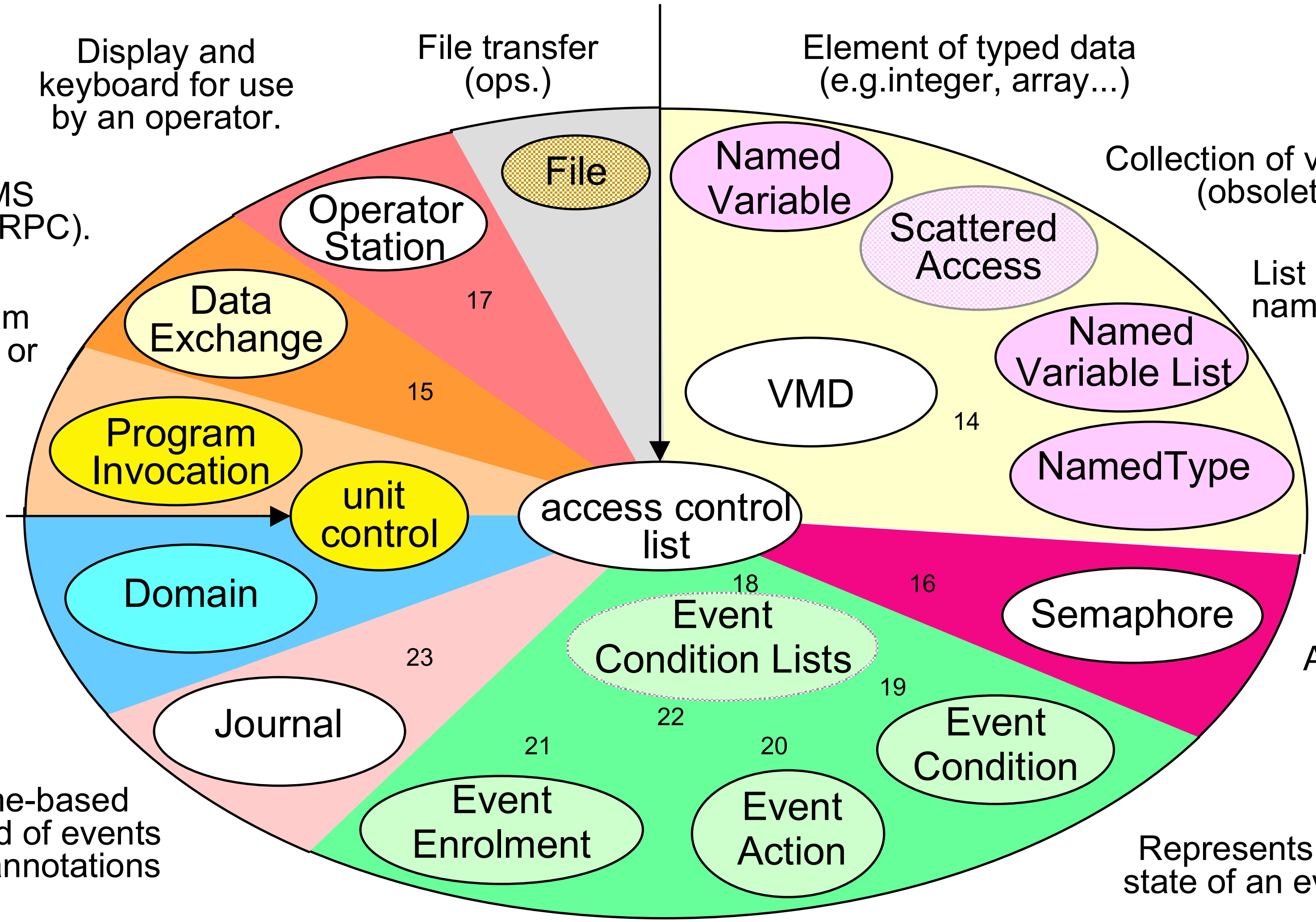
A physical device (PLC) may implement one or more virtual devices

MMS - Objects in a PLC device


memory regions upload / download
remote control the PLC
keep track of history
if mass storage available
user-defined variables
built-in variables, I/O and markers
tasks
state machines for alarms and events
PLC
domains
program invocations
Journal
Files
initiate start/stop semaphores identification
Operator Station
local Human-Machine Interface
variables
named variables
unnamed variables
events
events & alarms
MONITOR No. 1

MMS - Virtual Manufacturing Device (VMD) objects

Controls how to use objects



Display and keyboard for use by an operator.

File transfer (ops.)

Element of typed data (e.g.integer, array...)

Individual MMS service request (RPC).

Collection of variables (obsolete)

List of variables named as a list.

Runnable program consisting of one or more domains.

Description of the format of values

Object for uploading and downloading

Resource (e.g. memory region)

Access control to shared resources

Time-based record of events and annotations

Represents the state of an event.

Which network application to notify when an event condition changes state.

Action taken when an event condition changes state.

the numbers refer to parts of ISO 9506

MMS – Object Name

All objects (except unnamed variables) are identified by an **object name**, that may be

- VMD - specific persistent, pre-loaded, all clients see the same
“VMDstatus"

- domain -specific exists as long as the corresponding domain*
"e.g. Program1.List3"

- Application-Association specific exists as long as the client remains connected,
applies to non-persistent objects such as data sets that the client created
“@/MyDataSet”

The identifier itself is a "visible string“ (e.g. Call.Robot1.Joint3.Pos).

Access to all objects can be controlled by a special object, the *Access Control List* that tells which client can delete or modify the object.

The service GetNameList retrieves the name and type of all named objects in the VMD.
(this is the directory service)

* a domain is a (named) memory region that contains programs, variables, data

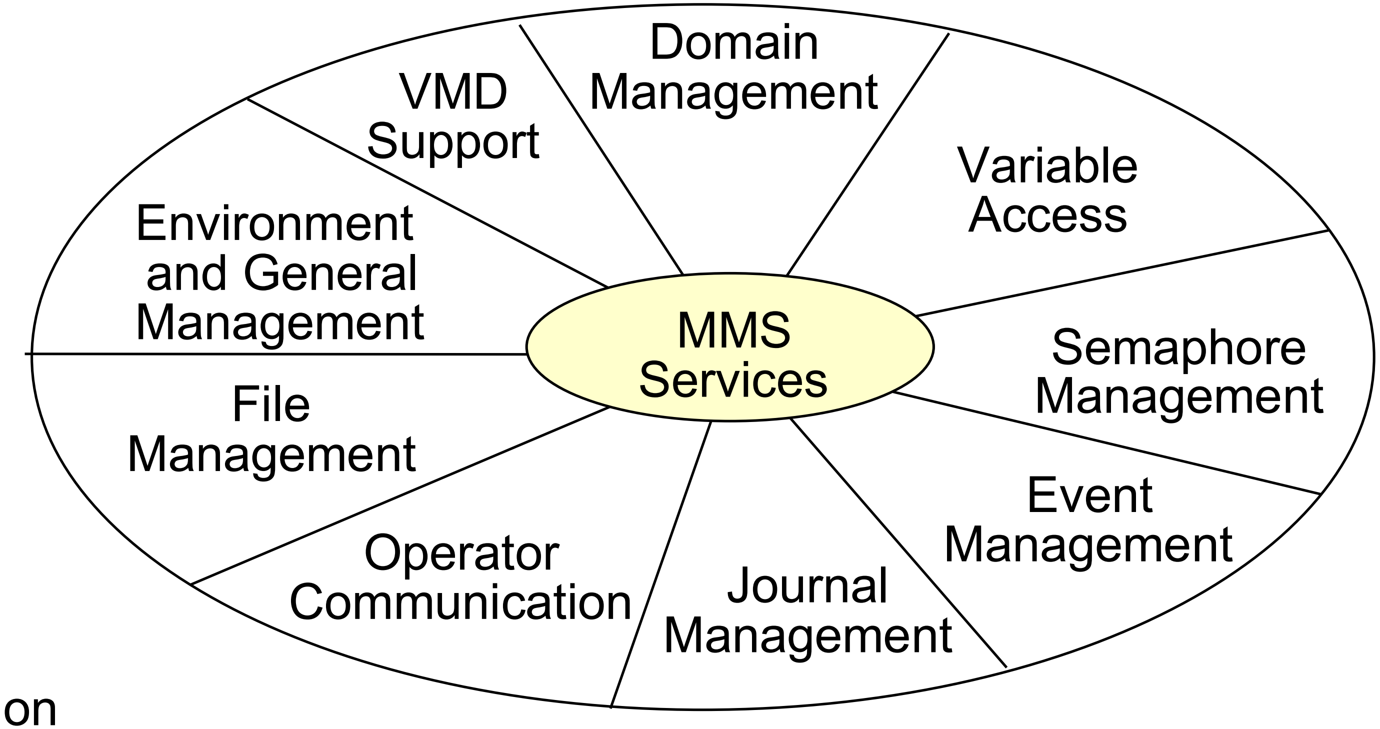
MMS - Data Types

MMS relies on the ASN.1 type (ISO 9988), but introduced new simple types:

TimeOfDay ::= OCTET STRING (SIZE(4|6))	-- First four octets are the milliseconds since midnight for the current date.
Identifier ::= VisibleString	-- up to 32 Uppercase and lowercase letters plus numbers, “$” and “_”.
Integer8 ::= INTEGER(-128..127)	-- range -128 <= i <= 127
Integer16 ::= INTEGER(-32768..32767)	-- range -32,768 <= i <= 32,767
Integer32 ::= INTEGER(-2147483648..2147483647)	-- range -2**31 <= i <= 2**31 - 1
Unsigned8 ::= INTEGER(0..127)	-- range 0 <= i <= 127
Unsigned16 ::= INTEGER(0..32767)	-- range 0 <= i <= 32767
Unsigned32 ::= INTEGER(0..2147483647)	-- range 0 <= i <= 2**31 – 1
FloatingPoint ::= OCTET STRING	-- according to IEEE 754 format
MMSString	Multilanguage string (VisibleString or ISO 10646)

These types map directly to MMS primitive types (they are a subrange of them), so there is no need to reserve additional primitive or constructed types for MMS.

MMS - 84 Services (methods) on the objects



1. Creation - Deletion
2. Read (Get, Report)
3. Modify (Alter)
4. Invoke (for domains)
5. Operate (Start, Stop,…)

MMS - Initialisation

An MMS client establishes first an Association (connection) with an MMS Server

A server may sustain several simultaneous associations with different clients
(to synchronize access, MMS provides semaphores)

At initialisation time, the client lists the capabilities it expects and the server responds with the capabilities it offers.

The capabilities are defined by Conformance Building Block parameters.
e.g. cto ∊ CBB means that the server agreed to provide an Access Control List

initialisation services:

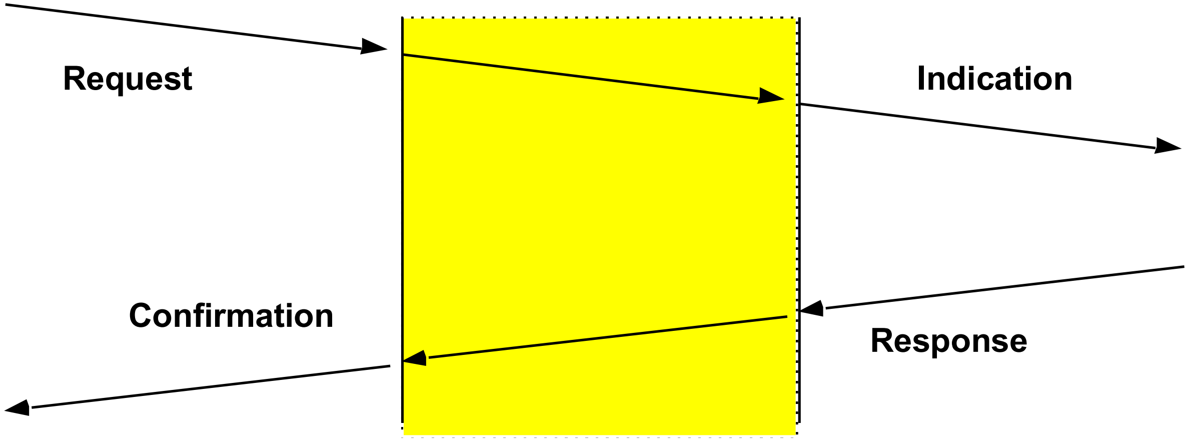
Initiate
ConcludeAbort
Reject
Cancel

Status
GetCapabilityList
GetNameList
Rename
Identify

MMS – Association establishment

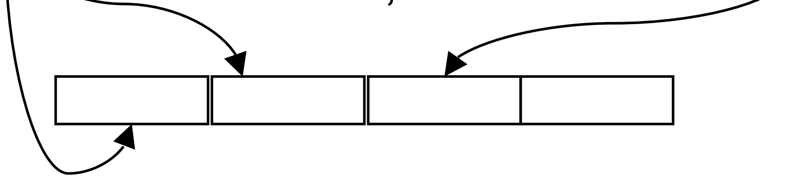
MMS Requester
(Requesting MMS-user)

MMS Responder
(Receiving MMS-user)



M_Associate_Req(
MMS_responder_address,
calling_application_reference,
called_application_reference,
communication_parameters,
authentication,..

M_Associate_Ind(
MMS_responder_address,
calling_application_reference,
called_application_reference,
communication_parameters,
authentication,..



this is no application interface, but a short way to describe the messages exchanged

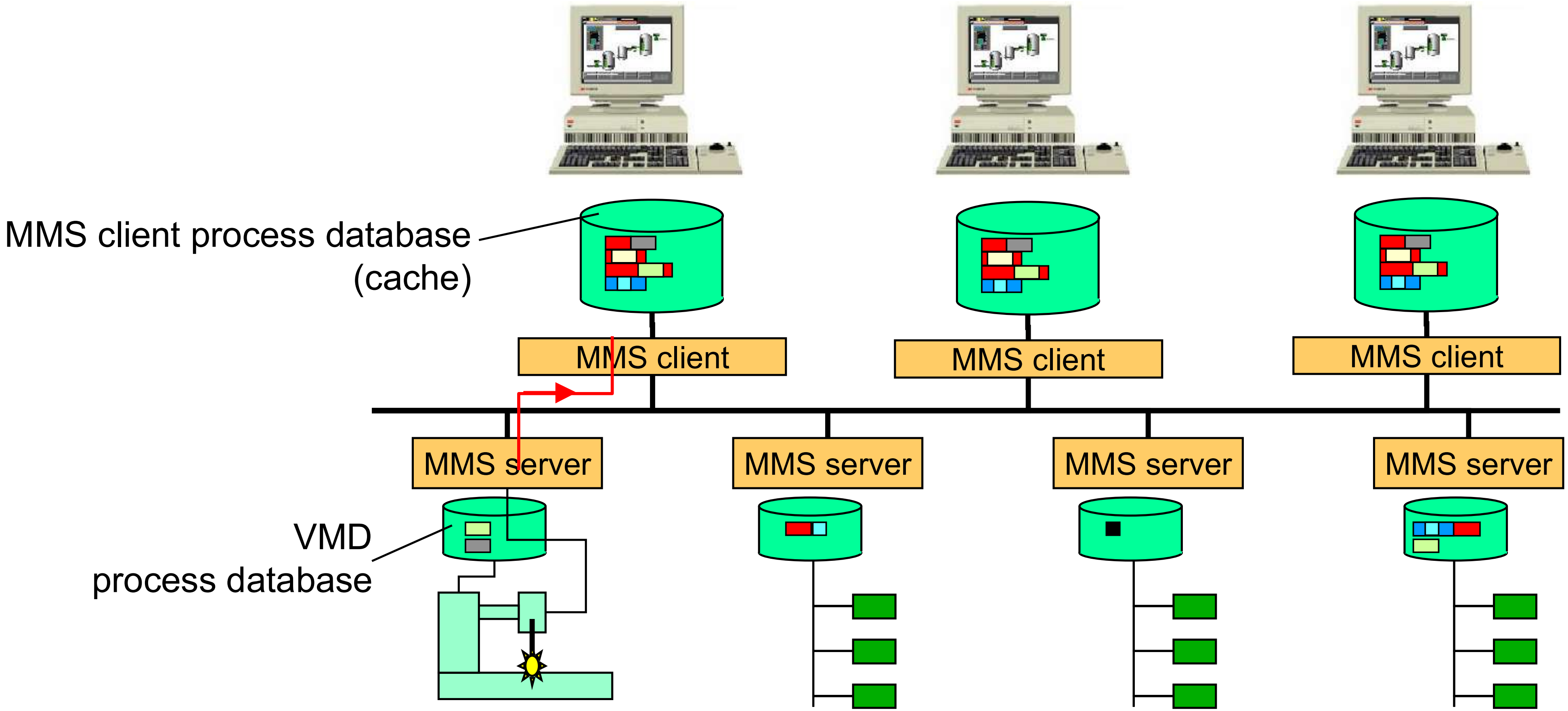
MMS - Addressing

MMS does not specify how to address clients and servers.

Messages contain only a communication reference
(number which identifies the connection)
obtained by unspecified means.

In practice, clients and servers are addressed by their IP address
and the MMS server uses port number 102.

MMS - Reading the variables



1) Polling:

a) the bus scans periodically the variables and actualises the local databases

b) the Operator Workstation polls cyclically the variables it is interested in

2) Events:

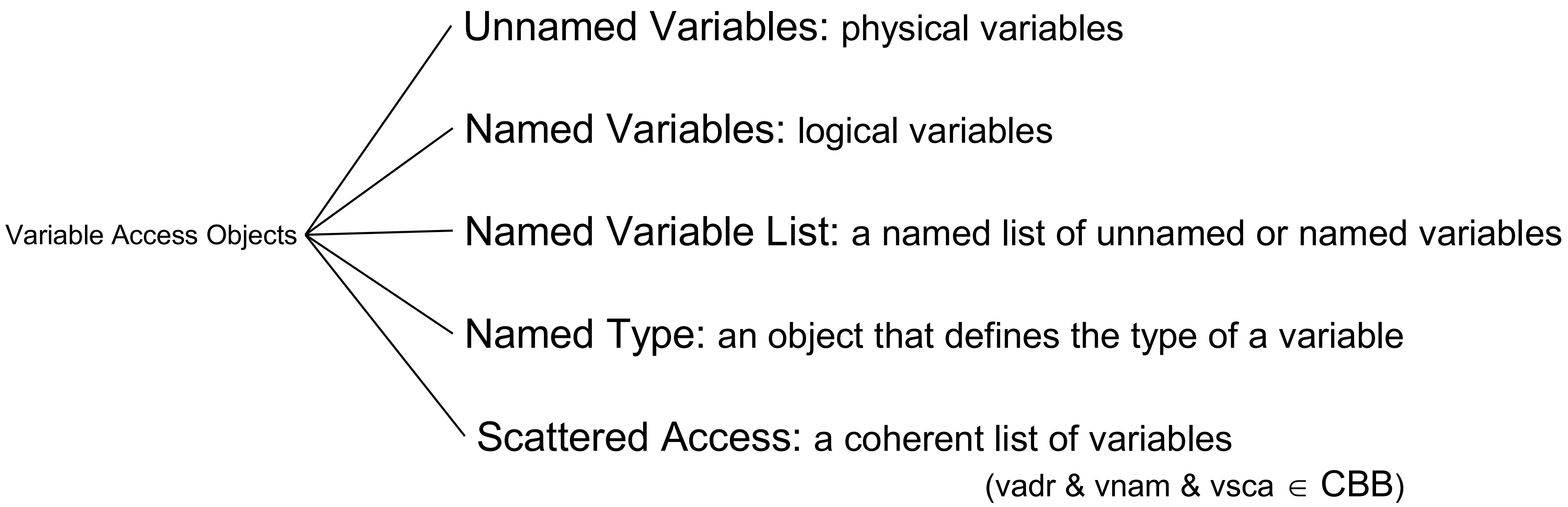
a) the Controllers signal predefined events and broadcasts the corresponding values

b) the Operator Workstation defines the relevant events and their destination(s)

MMS - Variables

Variables are the most important object type in MMS.
Through this service, a client can read and write local variables in a remote device.

Variables can be read or written as individual variables (not very efficient) or as lists.



Variable Access Objects

Unnamed Variables: physical variables

Named Variables: logical variables

Named Variable List: a named list of unnamed or named variables

Named Type: an object that defines the type of a variable

Scattered Access: a coherent list of variables

(vadr & vnam & vsca ∈ CBB)

MMS - Named and Unnamed Variables

Unnamed Variables (vadr ∈ CBB)

are identified by a **fixed physical address** in the VMD, expressed by either :

- numericAddress (an Unsigned32, e.g. 0xAF043BC0)
- symbolicAddress (a VisibleString, e.g. MW%1004)
- unconstrainedAddress (an OCTET STRING, e.g. 0x76AA)

Named variables (vnam ∈ CBB)

are identified by an **object name**

(a string of characters, VMD specific, domain specific or Association-specific)

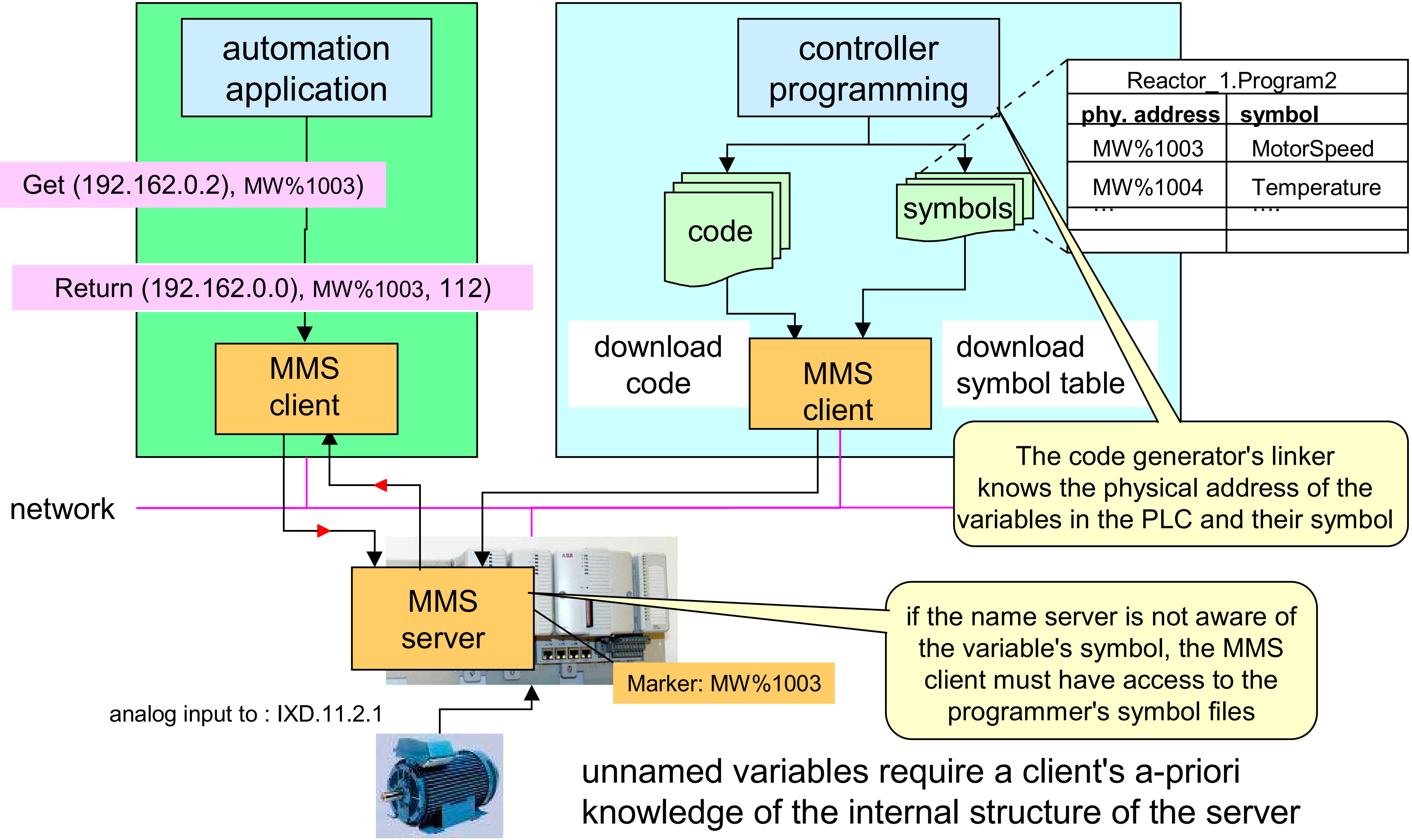
MMS supports two ways of structuring the variables space:

1) use the identifier string, separated by "$" signs

(e.g. Cell4$Robot1$Motor3$TemperatureOil)

2) define a variable with a complex type

MMS - Usage of Named and Unnamed variables



Reactor_1.Program2	
phy. address	**symbol**
MW%1003	MotorSpeed
MW%1004	Temperature
...	....

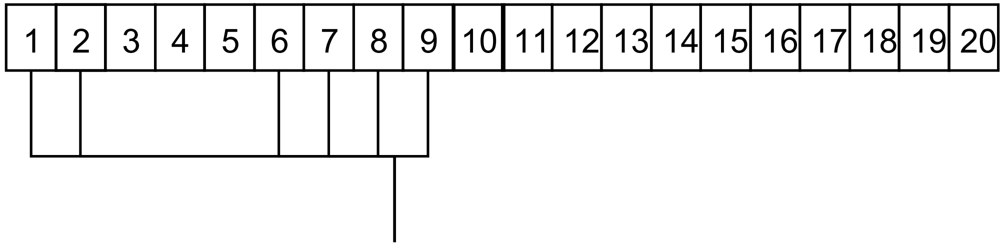
The code generator's linker knows the physical address of the variables in the PLC and their symbol

if the name server is not aware of the variable's symbol, the MMS client must have access to the programmer's symbol files

unnamed variables require a client's a-priori knowledge of the internal structure of the server

MMS - Variables Alternate Access

Alternate Access allows to transfer only certain elements in a (large) array or structure.



transmit: 1,2 6-9

This is useful if the data values are intertwined with static information such as descriptions.

Alternate access has been extended to specify a list of variables.

Structure of the “Data” type

Data in MMS belong to the following types:

```
Data ::= CHOICE {
                          -- context tag 0 is reserved for AccessResult
    array                 [1] IMPLICIT SEQUENCE OF Data, -- Nesting depth defined by nest ∈ CBB)
    structure             [2] IMPLICIT SEQUENCE OF Data, -- Possible if str1 ∈ CBB
    boolean               [3] IMPLICIT BOOLEAN,
    bit-string            [4] IMPLICIT BIT STRING,
    integer               [5] IMPLICIT INTEGER,
    unsigned              [6] IMPLICIT INTEGER,                  -- Shall not be negative
    floating-point        [7] IMPLICIT FloatingPoint,
    real                  [8] IMPLICIT REAL, -- obsolete
    octet-string          [9] IMPLICIT OCTET STRING,
    visible-string        [10] IMPLICIT VisibleString,
    generalized-time      [11] IMPLICIT GeneralizedTime,
    binary-time           [12] IMPLICIT TimeOfDay,
    bcd                   [13] IMPLICIT INTEGER,                 -- Shall not be negative
    booleanArray          [14] IMPLICIT BIT STRING,
    objId                 [15] IMPLICIT OBJECT IDENTIFIER,
    mMSString             [16] MMSString                         – Multilanguage string
}
```

MMS Variable Lists

MMS provides services to build a Data Set, a group of variables that is to be transmitted as a whole.

This is generally done for each client specifically (Application-Association specific)

MMS – Summary Variable Access Services

`Read`	read a remote variable
`Write`	write a remote variable
`InformationReport(optional)`	spontaneous send the value to a client
`GetVariableAccessAttributes`	get the attributes of the variable
`DefineNamedVariable`	assigns named variable to an unnamed & type
`DeleteVariableAccess`	

`DefineNamedVariableList`	defines lists of variables
`GetNamedVariableListAttributes`	
`(Read)`	for individual variables or lists
`(Write)`	
`(Information Report)`	
`DeleteNamedVariableList`	

`DefineNamedType`	defines the types
`GetNamedTypeAttributes`	
`DeleteNamedType`	

`DefineScatteredAccess`	defines variables group treated as a whole
`GetScatteredAccessAttributes`	`(obsolete, but useful)`

MMS - Domains

Domains are **named** memory regions, for the purpose of downloading and uploading large unstructured data such as program code.

Domain loading / uploading requires a special protocol because it can involve the MMS driver itself or even the communication stack, and storing to stable storage.

Typically, a domain is loaded by segments of a size chosen by the receiver.

When a domain is loaded, it may be saved to EPROM (typical PLC programming).

Domains may be erased.

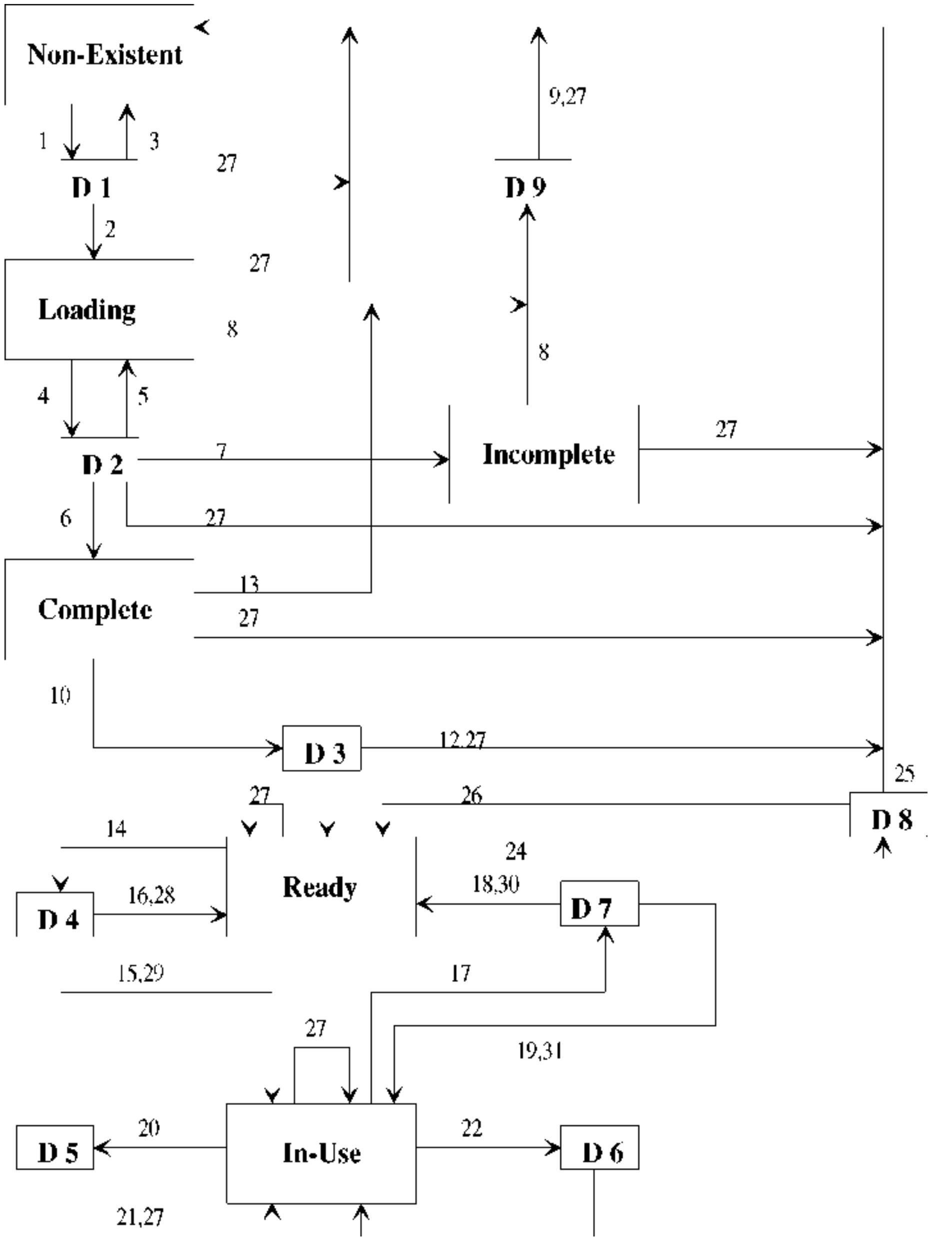
Objects (Variables, Events, Program invocations,..) may be tied to a domain.

MMS Domain State Diagram

Each domain is controlled by a state machine in MMS.

This is necessary since a domain is large and often needs to be loaded in several steps.

Also, it may be necessary to write the domain into a non-volatile memory and that needs a tighter control.



Summary: MMS Operations on Domains

Operations on domains:

InitiateDownloadSequence	Download
DownloadSegment	
TerminateDownloadSequence	
RequestDomainDownload	

InitiateUploadSequence	Upload
UploadSegment	
TerminateUploadSequence	
RequestDomainUpload	

LoadDomainContent	
StoreDomainContent	e.g. to EPROM
DeleteDomain	erase
GetDomainAttributes	

Program Invocations

Program invocations are tasks running in the VMD.

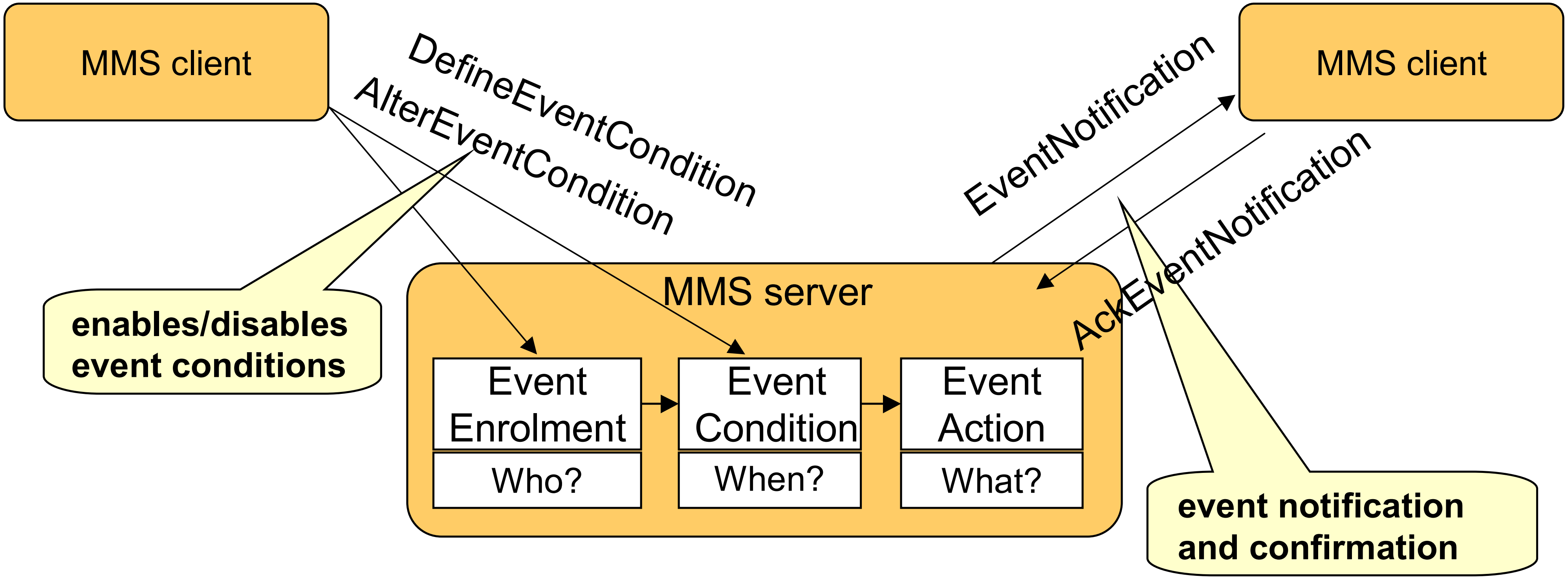
Programs are tied to domains.

For instance, all tied programs are stopped before loading a domain containing them

```
CreateProgramInvocation
DeleteProgramInvocation
Start
Stop
Resume
Reset
Kill
GetProgramInvocationAttributes
Select
AlterProgramInvocationAttributes
ReconfigureProgramInvocation
```

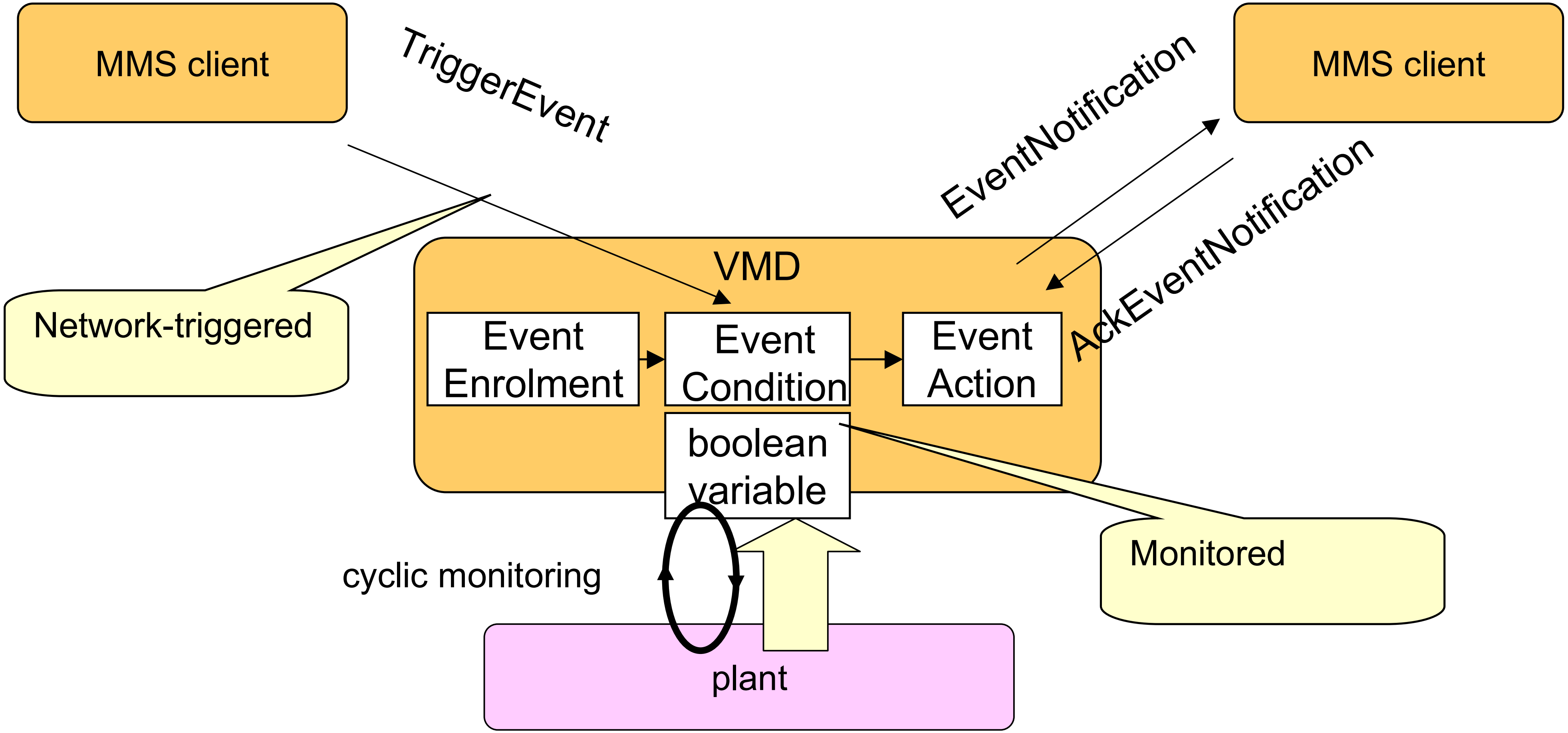
MMS - Event services

MMS provides services to:
- Event Condition (define the Boolean condition that triggers an event and its priority)
- Event Enrolment (define the MMS client(s) to notify when an event is triggered)
- Event Action (define the MMS confirmed service to be executed when the event occurs)



Events are the most complicated part of MMS

MMS - Event triggering



events are triggered by a change in a boolean variable in the server (monitored event) or by an MMS client (trigger event) as an invitation procedure.

Event Services

The Event services are the most complicated part of MMS.
However, the event mechanism in a SCADA system is complex in nature.

Event Management

```
TriggerEvent
EventNotification
AcknowledgeEventNotification
GetAlarmSummary
GetAlarmEnrollmentSummary
```

Event Conditions

```
DefineEventCondition
DeleteEventCondition
GetEventConditionAttributes
ReportEventConditionStatus
AlterEventConditionMonitoring
```

Event Actions

```
DefineEventAction
DeleteEventAction
GetEventActionAttributes
ReportEventActionStatus
```

Event Conditions Lists

```
DefineEventConditionList
DeleteEventConditionList
AddEventConditionListReference
RemoveEventConditionListReference
GetEventConditionListAttributes
ReportEventConditionListStatus
AlterEventConditionListMonitoring
```

Event Enrollment

```
DefineEventEnrollment
DeleteEventEnrollment
GetEventEnrollmentAttributes
ReportEventEnrollmentStatus
AlterEventEnrollment service
```

Other MMS services

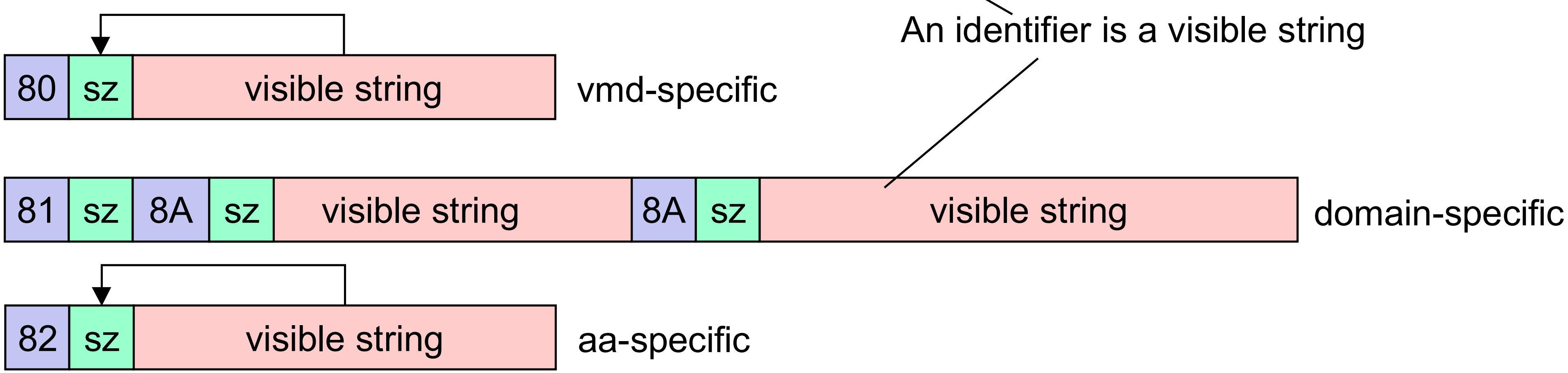
The most important services are:
-Variables,
-Events,
-Domains

for the other services, see the ISO / IEC standard 9506

MMS - PDU Notation

MMS uses ASN.1 (ISO 8824) to describe the network messages (PDUs).
MMS specifies the use of BER (Basic Encoding Rules, ISO 8825) of ASN.1
(in principle only for connection establishment, but in practice for all PDUs)

```
ObjectName ::= CHOICE {
        vmd-specific                [0] Identifier,
        domain-specific             [1] SEQUENCE {
                domainId            Identifier,
                itemId              Identifier
                },
         aa-specific                 [2] Identifier
         }
```



This notation is quite heavy for simple variable transport (24 bits for one Boolean value) but decoding costs must be weighted against communication costs.

MMS - Importance

MMS is becoming (after 15 years of existence) a reference model for industry rather than an actual implementation.

Its high complexity makes it very general, but difficult to implement

It gave rise to several other "simpler" models (DLMS, BacNet, FMS....)

It is the base of the Utility Communication Architecture (UCA), an EPRI*-sponsored standardization of data exchange between control centers.

http://www.epri.com/uca/iccp.html

For more information, see:

http://lamspeople.epfl.ch/kirrmann/mms/

http://www.nettedautomation.com/qanda/mms/#OPC/MMS

EPRI = USA electrical power research institute

MMS companion standards

MMS does not define the meaning of the exchanged information.
For this, companion standards exist, such as:

IEC/ISO 9506-5 Industrial automation systems - Manufacturing message specification -
Part 3: Companion Standard for Robots (1992)
Part 4: Companion Standard for Numeric Control (1993)
Part 5: Companion Standard for Programmable Controllers (1997)
Part 6: Companion Standard for Process Control (1994)

One standard which emerged in direct line from MMS is IEC 61850
„Communication networks and systems in substations“

IEC 61850 defines an MMS implementation based on Ethernet / TCP-IP
and elaborates on the object model.

It is currently being developed at ABB, Siemens and Areva for substation automation.

Conclusion

Although MMS itself had little success (it is complicated), the concepts behind MMS have inspired numerous other standards.

Industrial Communication protocols require a large bandwidth and a lot of processing power at the servers, which is incompatible with low-cost, decentralized periphery, but fully in line with the concept “Ethernet in the factory floor”.

While most field busses are able to connect relatively simple devices, the same is not true for MMS and its derivatives.

The MMS concept is being challenged by COM/DCOM (OPC) and by Web Services, but these services will have to rediscover the semantics of MMS.

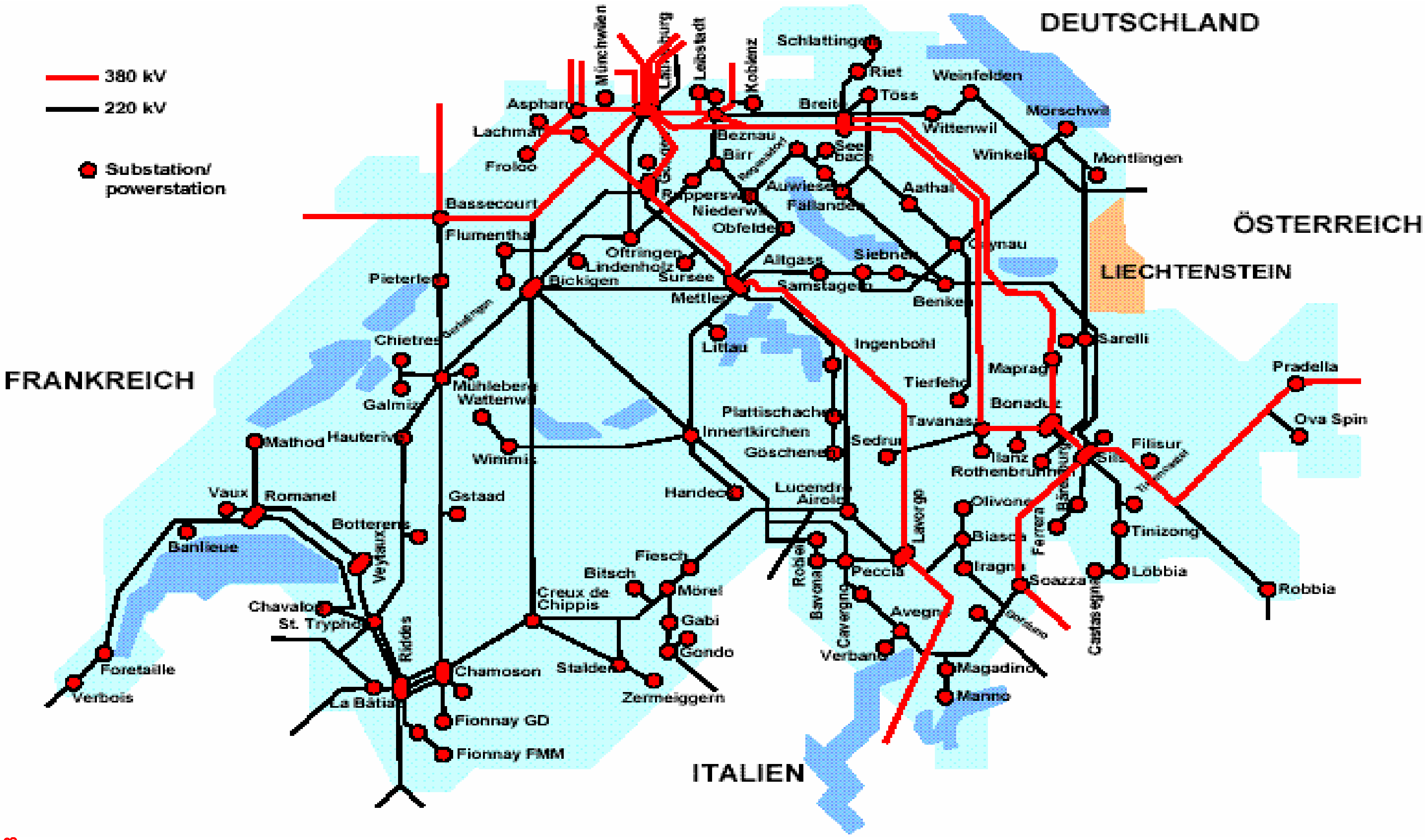
Hubert Kirrmann

ABBCH-RD.C1

Introduction to IEC 61850 substation communication standard

Project Red61850, CRID 30254, C1-069-03/00

Electrical Substations: nodes of the grid



Swiss transmission network

Air isolated substation

Substation objects

Station (Unterstation, *Sous-station*)
- Complete node in the power network (= substation) *or*
- Station buss bar

Bay (*départ*, Abgang)
Part of the substation with local functionality, e.g. related to an
- incoming line (“feeder”)
- connection between the buss bars

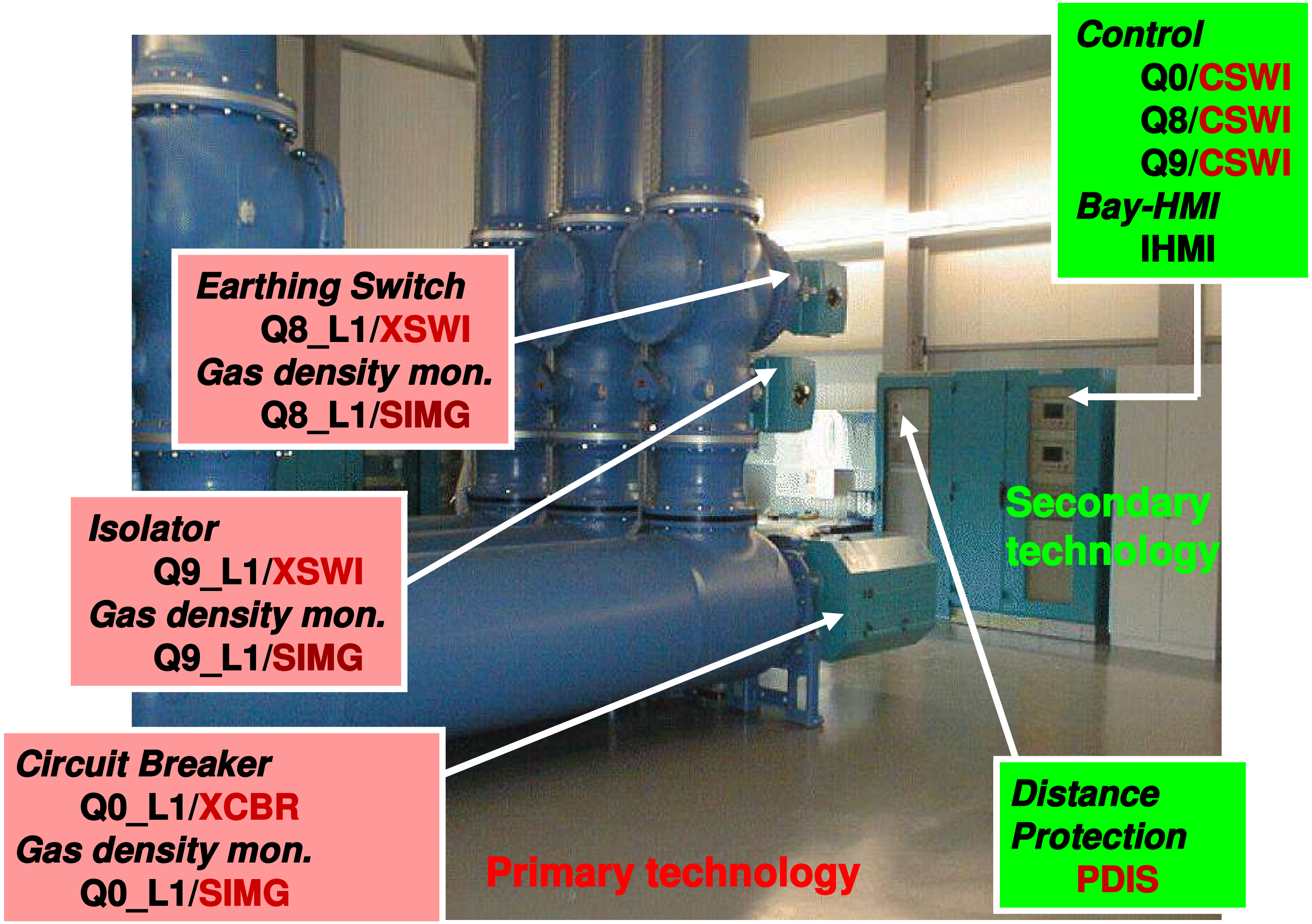
etc.

Process objects (switchyard)
- breaker
- transformer

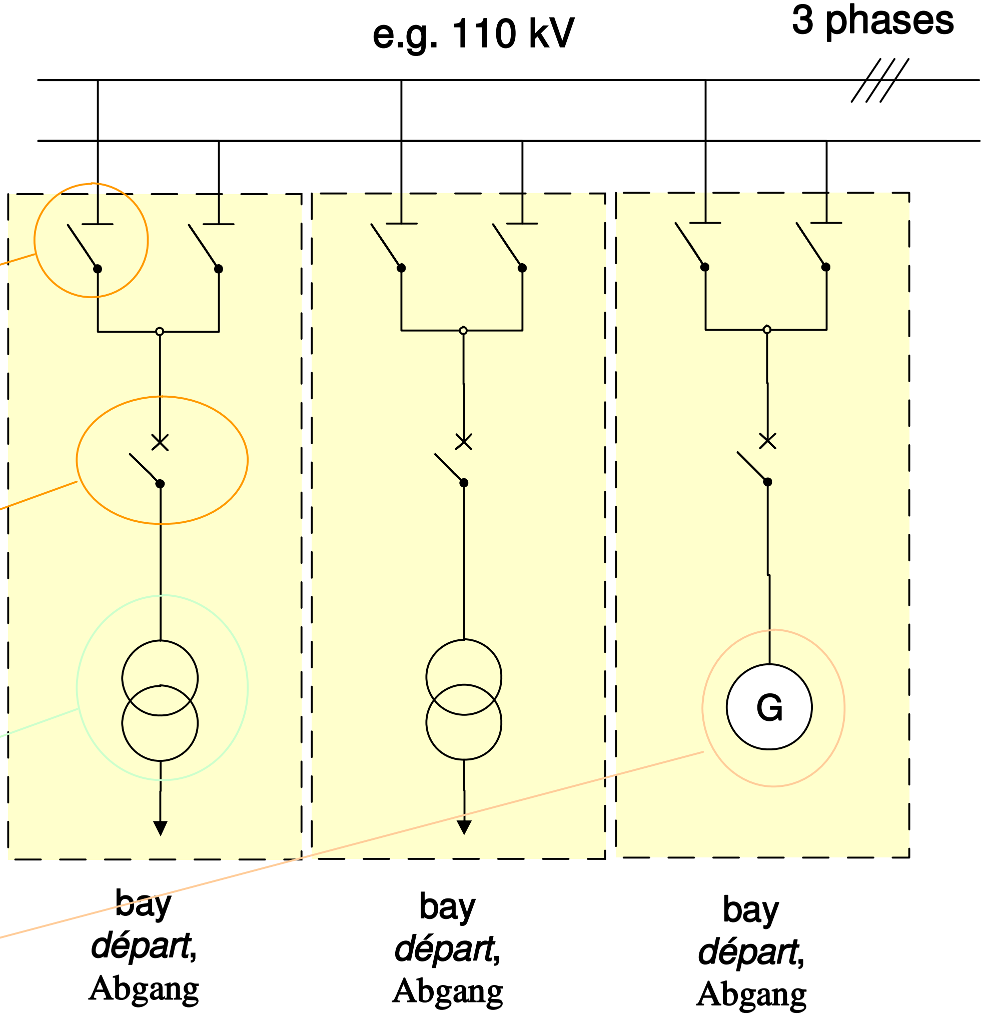
etc.

Primary equipment = switchyard hardware
Secondary equipment = control, monitoring and protection devices

Gas Isolated Substation



Principle substation: single line diagram



two buss bars

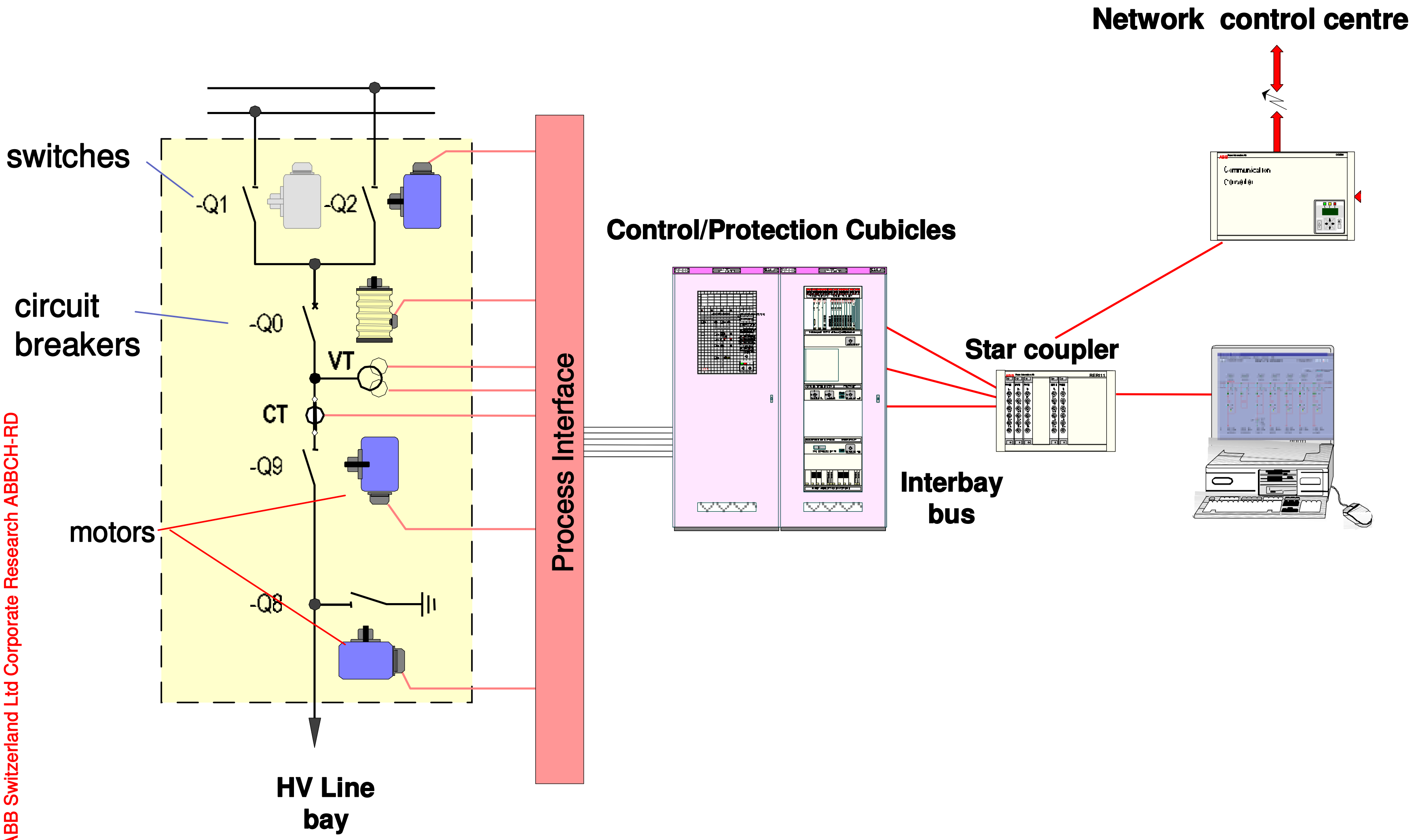
switches
(isolators,*interrupteurs*,
Trenner)
cannot be switched
under power

circuit breaker
(*disjoncteur*,
Leistungsschalter)
can switch fault current

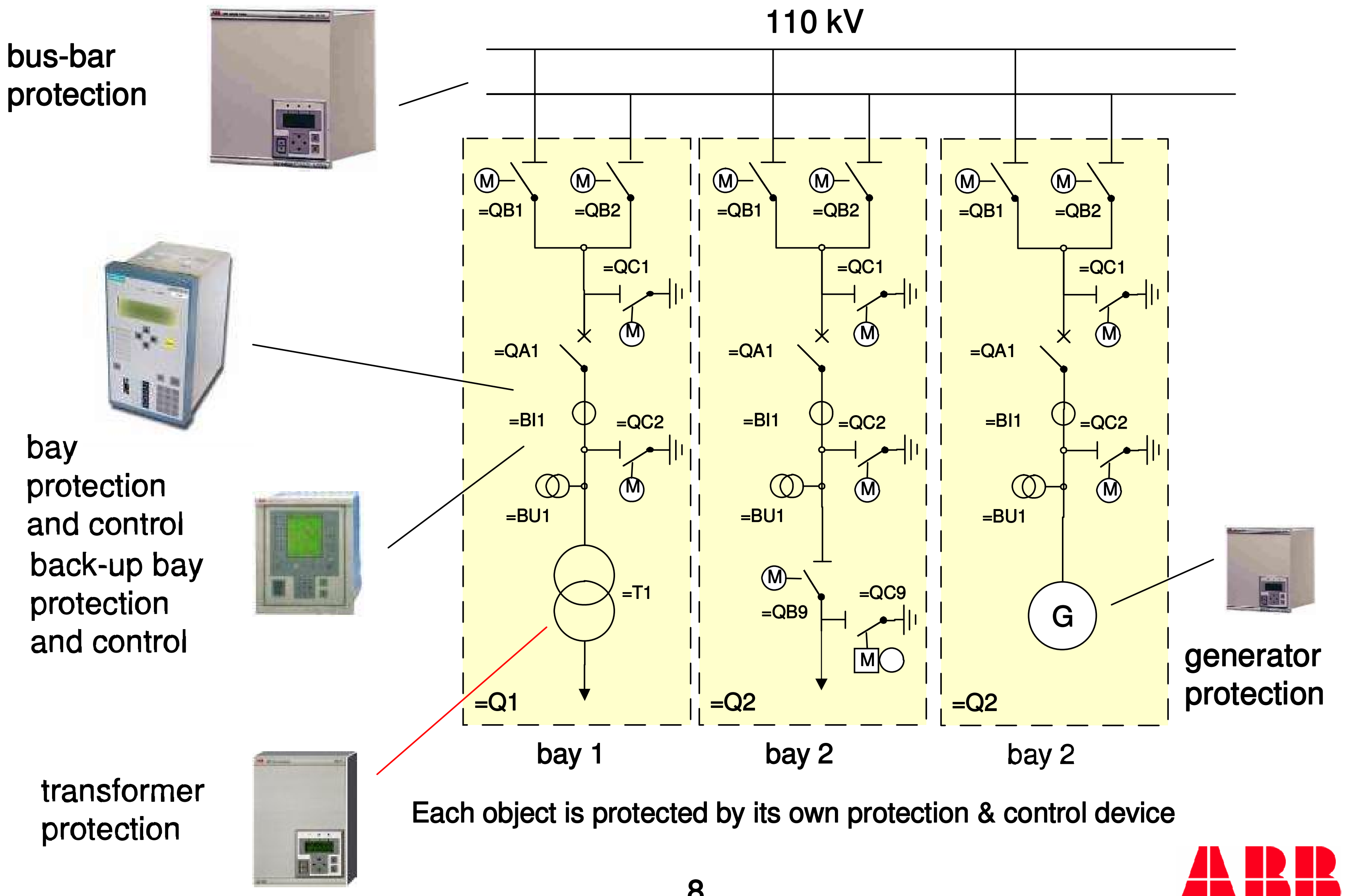
transformer
(*transformateur*,
Trafo)

generator
(*générateur*,
Generator)

Substation: One bay

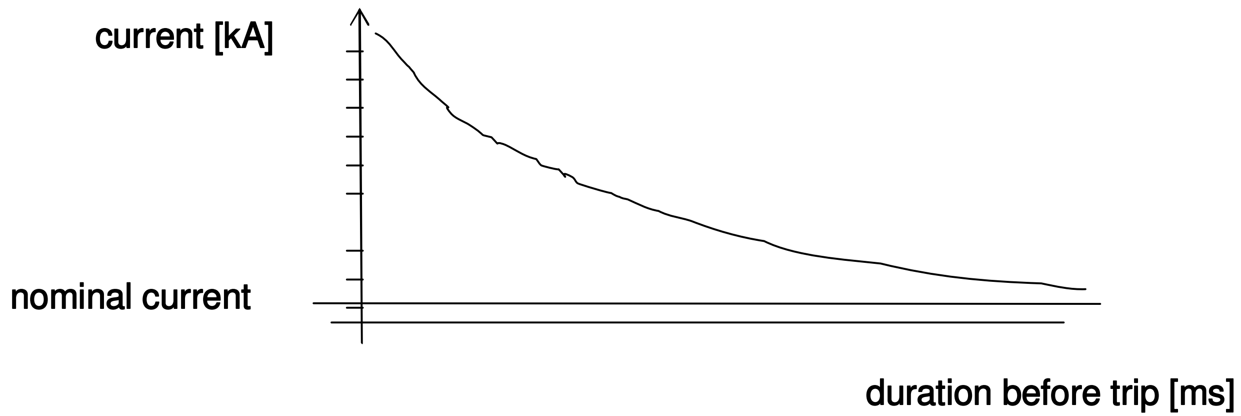


Physical Devices (IEDs)



Each object is protected by its own protection & control device

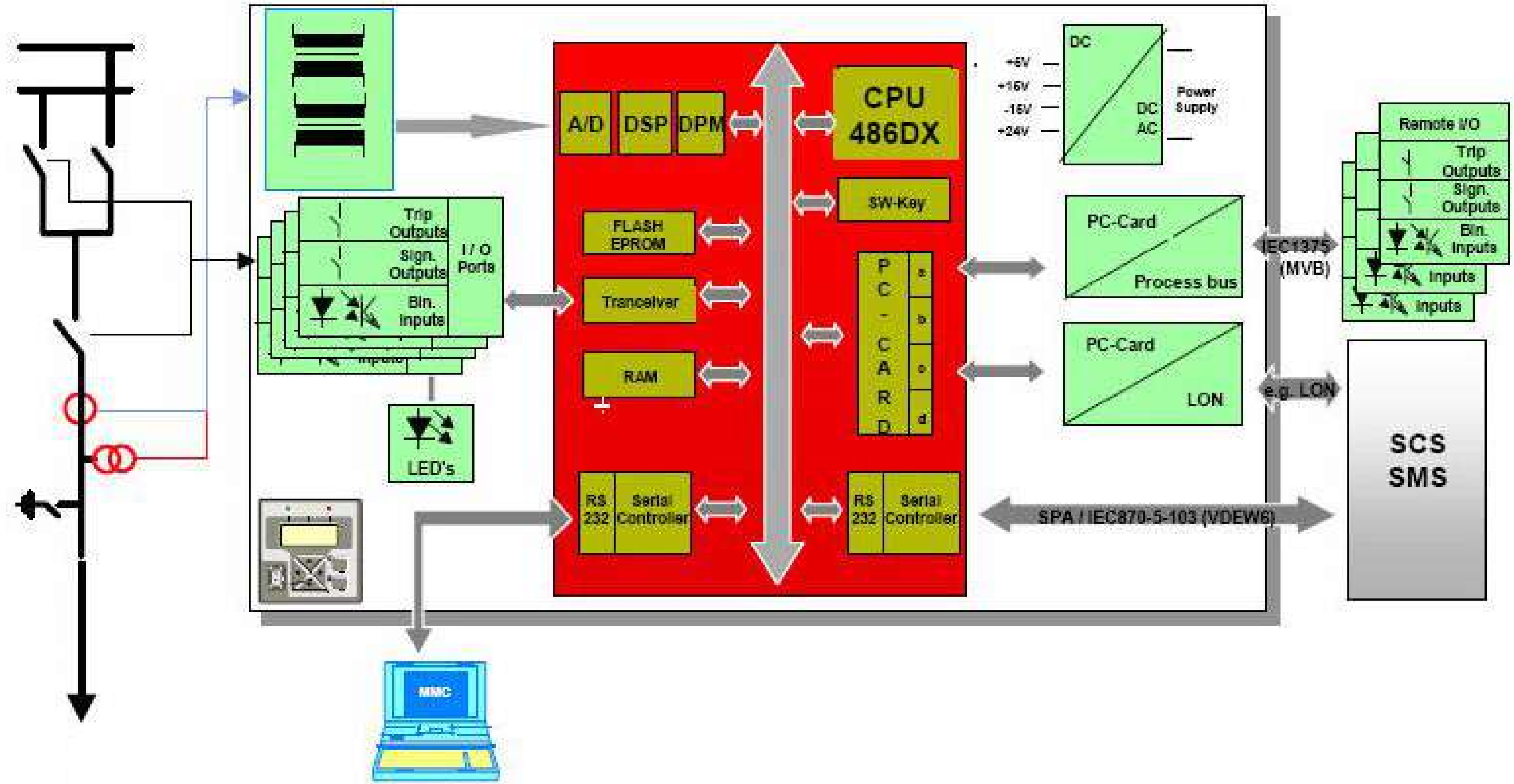
Example of protection function: overcurrent



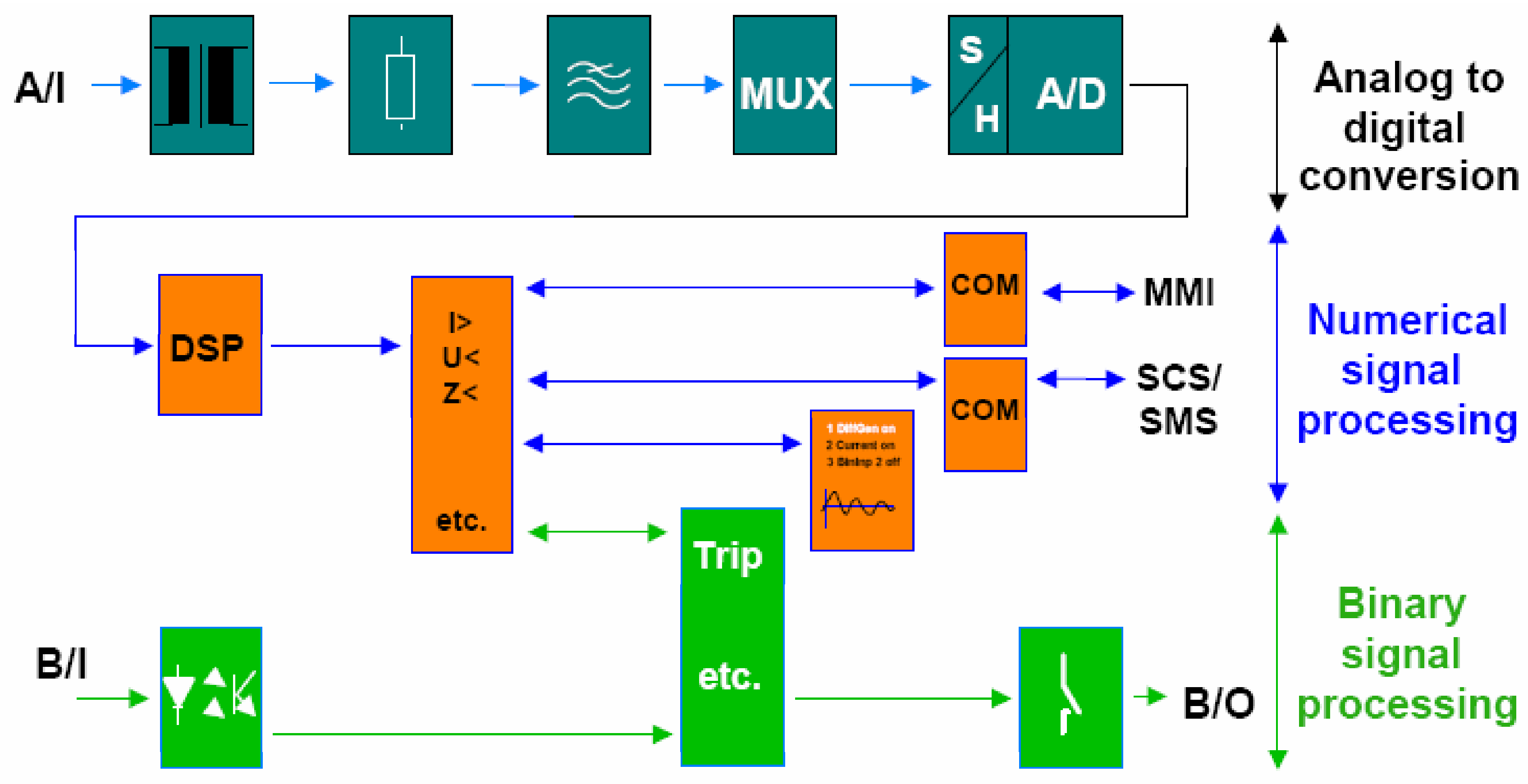
The protection function is adjusted with a number of parameters that are tuned for a specific substation and bay, the is called a *setting*.

Protection function have usually different settings, that are used depending on the situation.

Internals of an IED

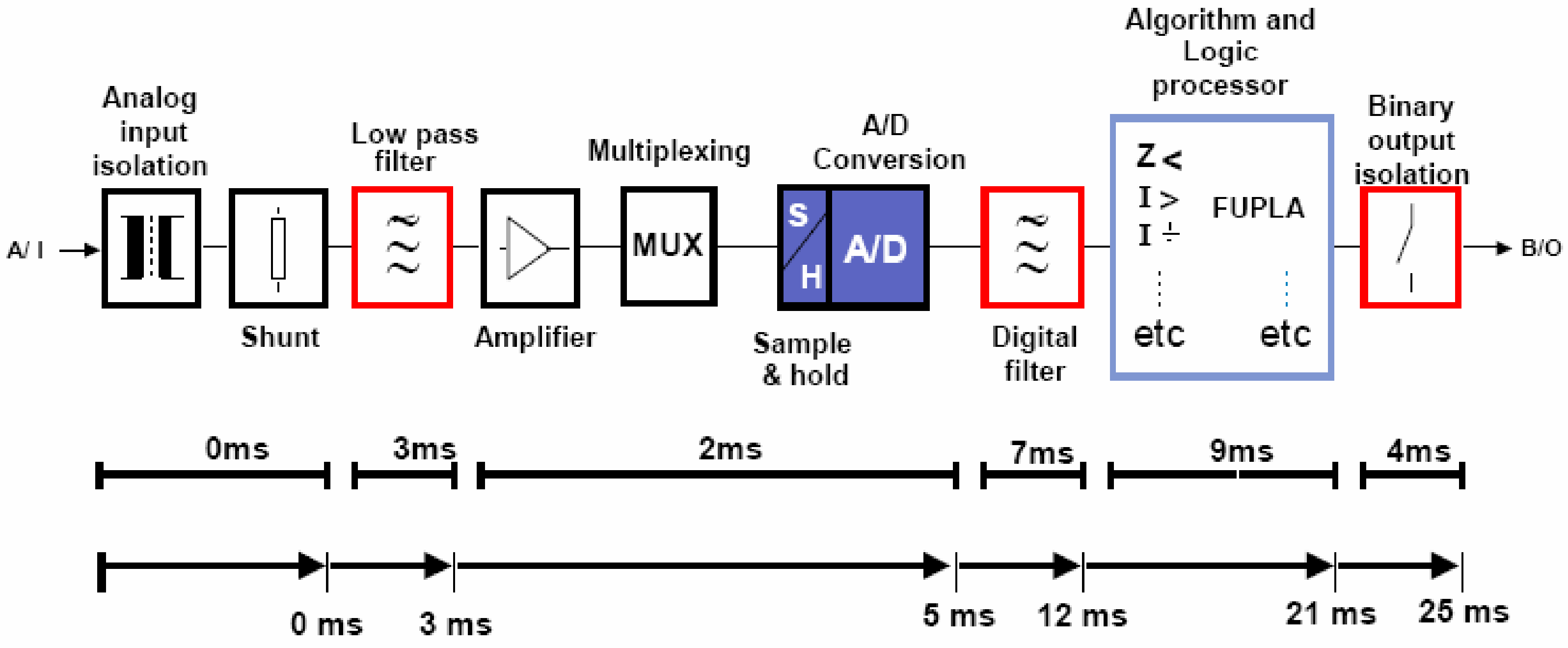


Signal flow in an IED



IEC 61850
MMS Applied

Time delays in an IED



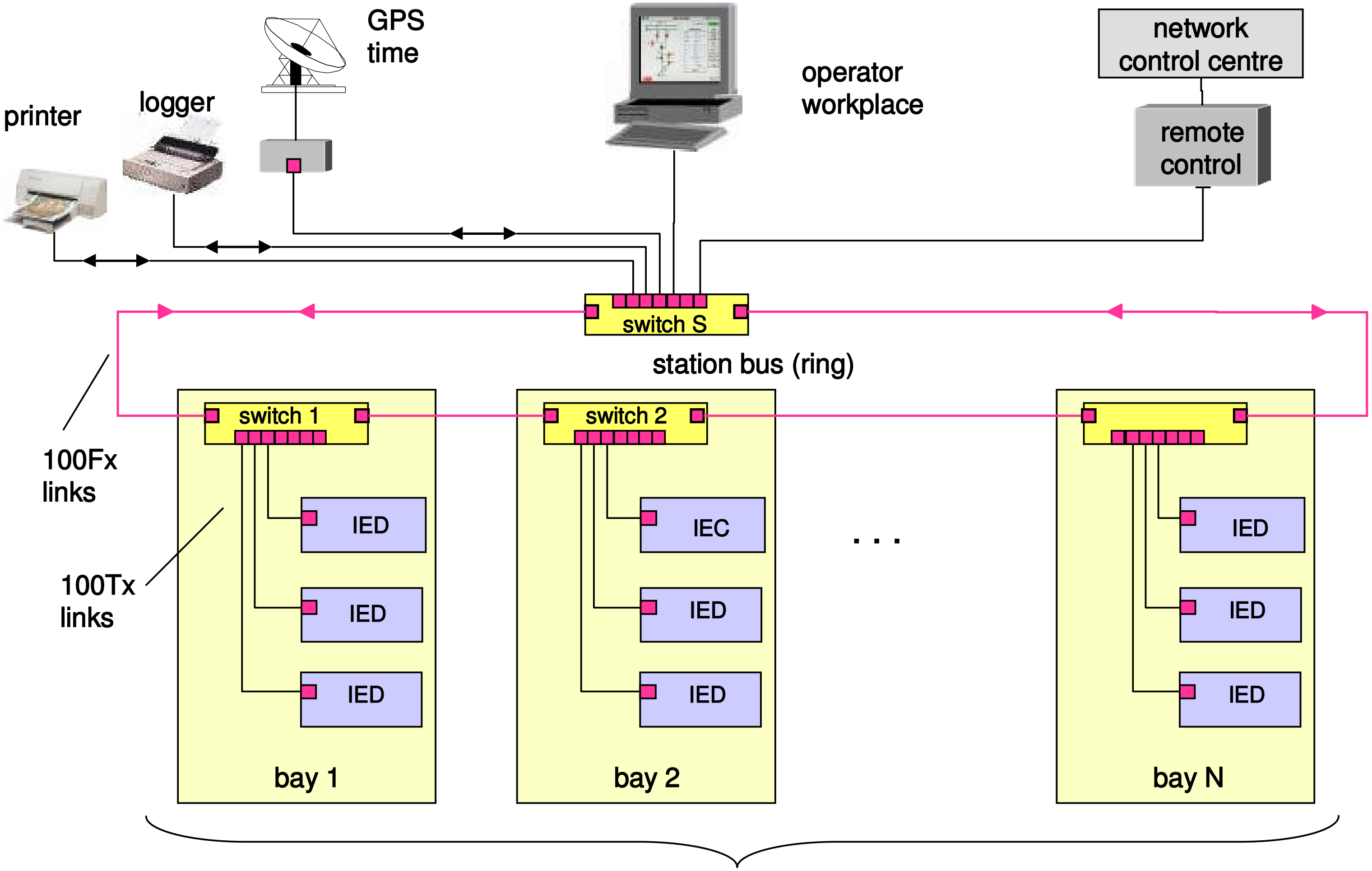
Functions

Protection functions
prevent hazard to people, damage to power network components (devices) and breakdown of the power network.
performed autonomously within some 10 ms .

Monitoring functions
supervise the status of the primary and/or secondary equipment, disturbance recorder, sequence of events with resolution 1 ms

Control functions
allow a local or remote operator to operate the power
(response time about of 1 s).
Automatic sequence functions execute sequence of operations, such as switch from one buss bar to the other ((order of 100 ms)

An IEC 61850 network

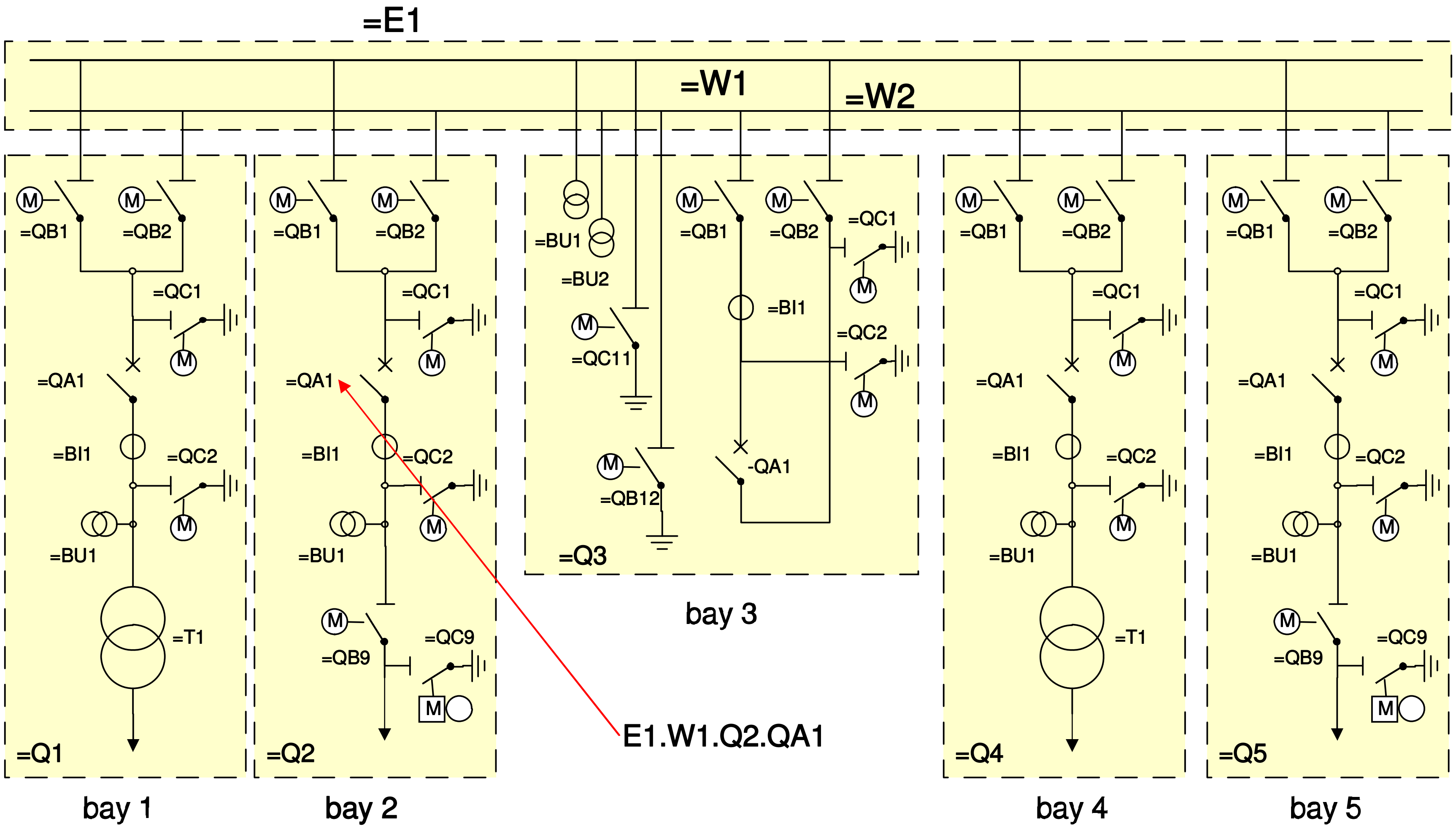


the structure of the network reflects the structure of the substation

The IEC 61850 object model

Although IEC 61850 is defined as a
“communication structure for substation and feeder equipment”
its main contribution is the definition of an object model for all substation objects

IEC 61346: Naming of substation elements

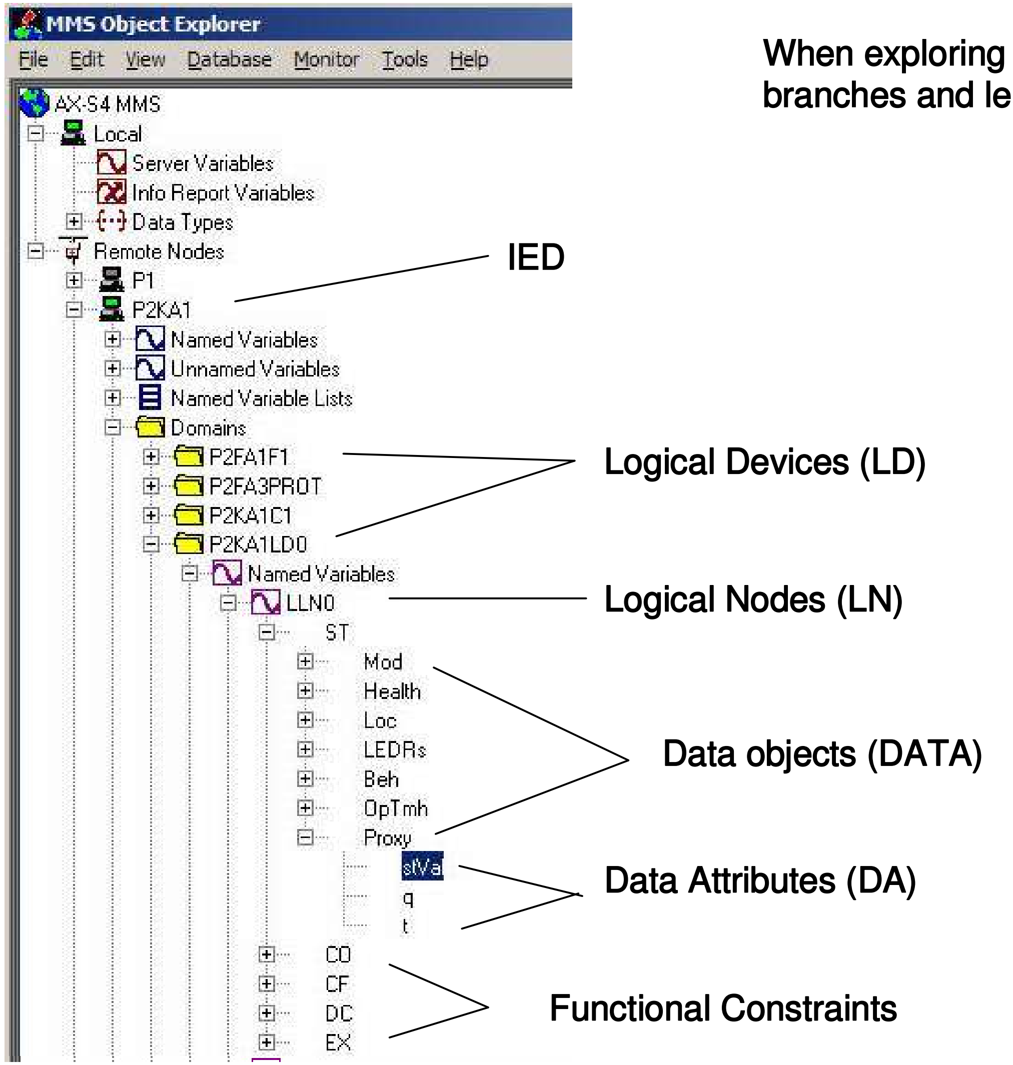


The IEC 61346 standard defines how substation elements should be named.
Customers define their own names, e.g. Q1 is “City_Broadway”

IEC 61850
MMS Applied

Exploring an IED

When exploring an IED, one finds a hierarchy of branches and leaves



```
AX-S4 MMS
  Local
    Server Variables
    Info Report Variables
    Data Types
  Remote Nodes
    P1
    P2KA1
      Named Variables
      Unnamed Variables
      Named Variable Lists
      Domains
        P2FA1F1
        P2FA3PROT
        P2KA1C1
        P2KA1LD0
          Named Variables
            LLN0
              ST
                Mod
                Health
                Loc
                LEDRs
                Beh
                OpTmh
                Proxy
                  stVal
                  q
                  t
              CO
              CF
              DC
              EX
```

IED

Logical Devices (LD)

Logical Nodes (LN)

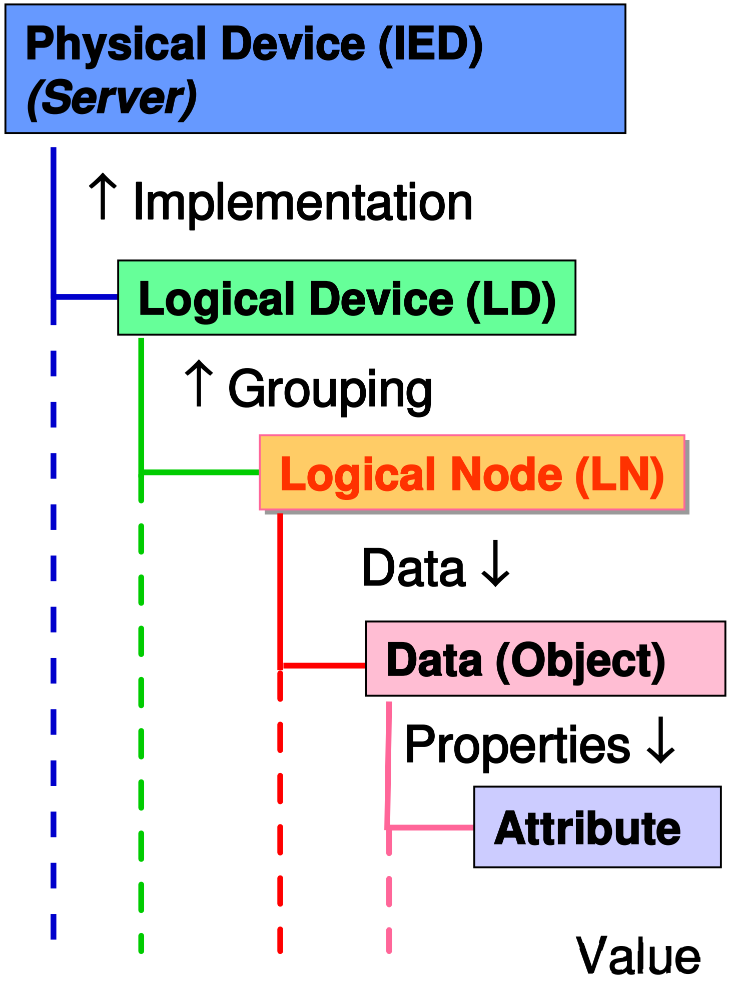
Data objects (DATA)

Data Attributes (DA)

Functional Constraints

IEC 61850
MMS Applied

Hierarchy within a protection & control device



Physical Device (IED)
(Server)

↑ Implementation

Logical Device (LD)

↑ Grouping

Logical Node (LN)

Data ↓

Data (Object)

Properties ↓

Attribute

Value

example

Bay Unit

Control

CSWI Switch Control

Position

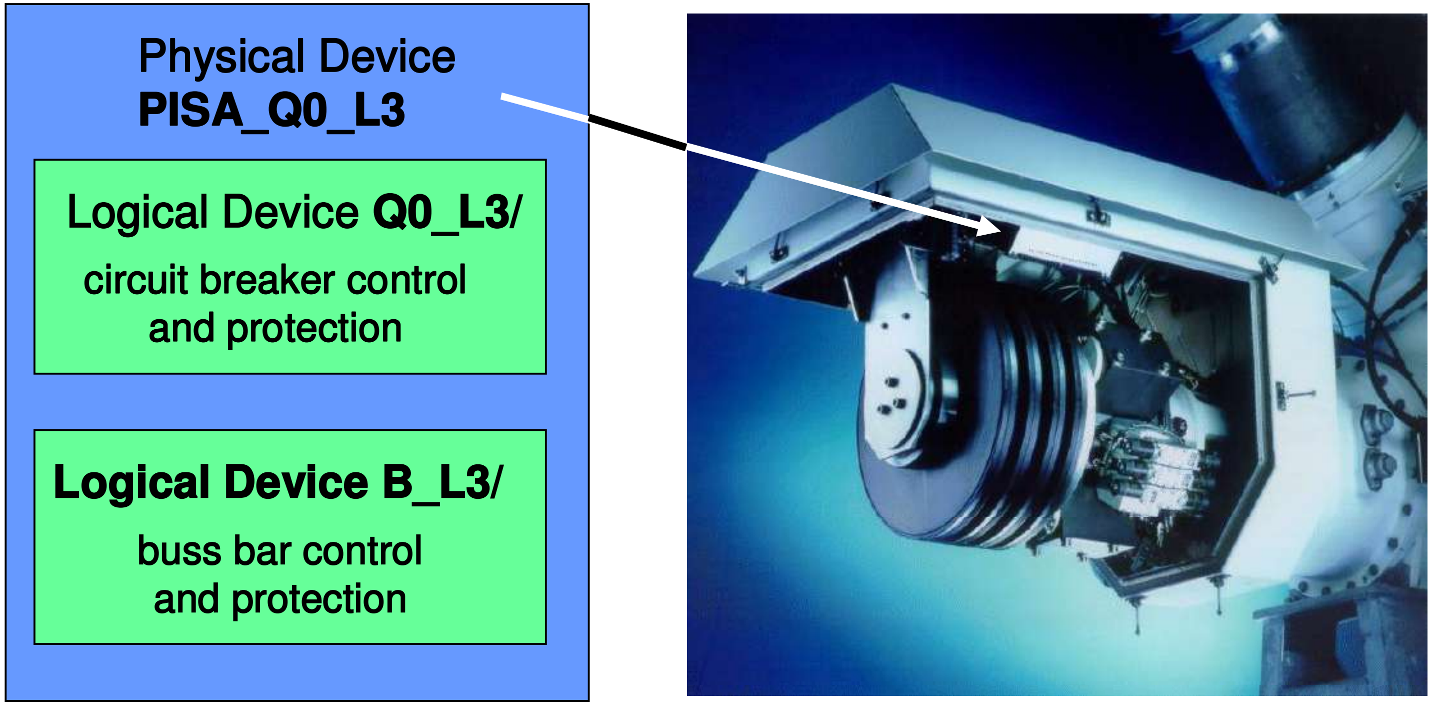
Control Value
Status Value

ON/OFF

Logical device

Each physical device (called an IED) can perform functions that was formerly performed by different protection or control devices.

Those former devices are represented by Logical Devices within the physical device.



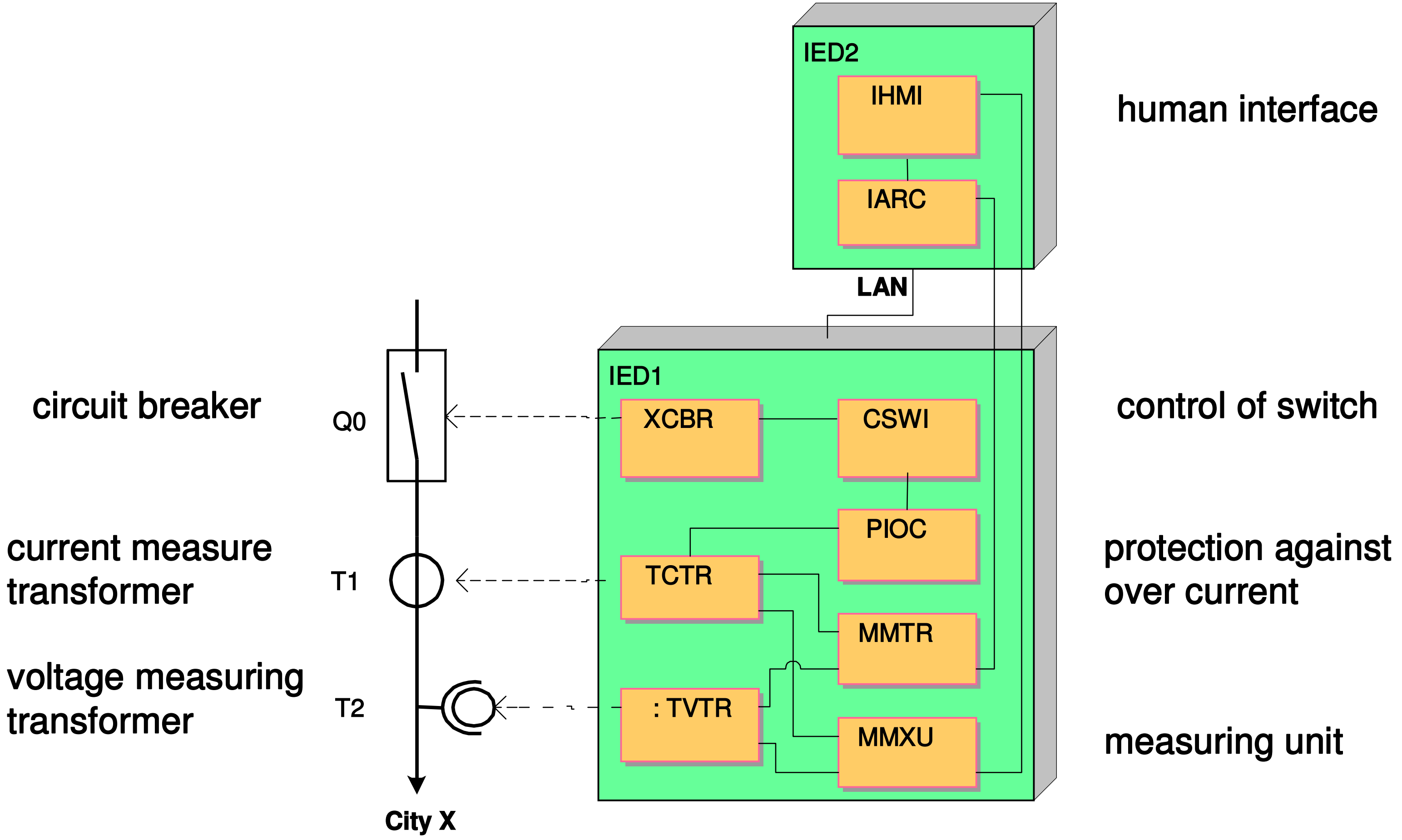
Physical Device
PISA_Q0_L3

Logical Device **Q0_L3/**
circuit breaker control
and protection

Logical Device B_L3/
buss bar control
and protection

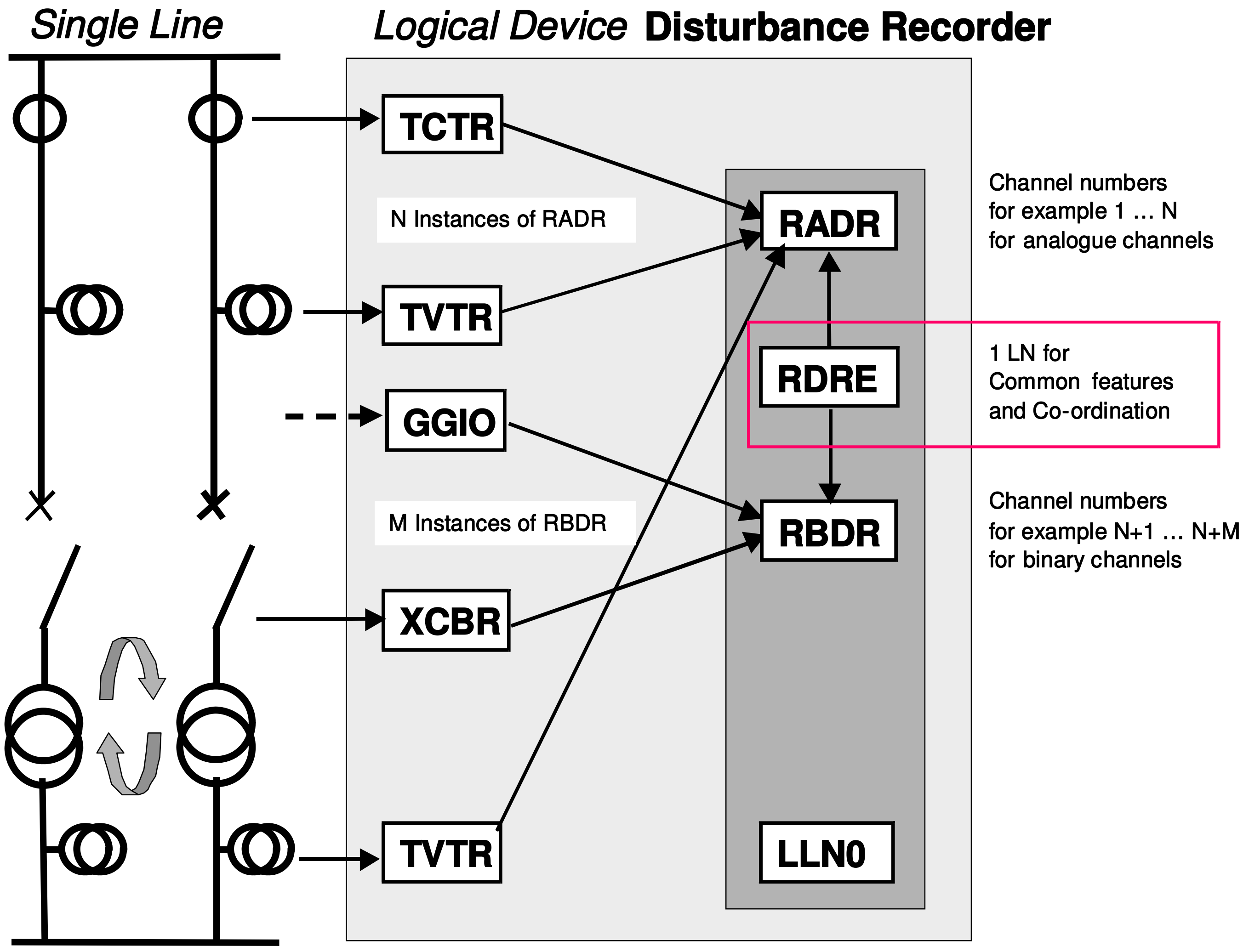
Logical Nodes

IEC 61850 assigns to each function within a substation equipment (transformer, circuit breaker, protection function...) a logical node (LN).



IEC 61850
MMS Applied

Interaction between logical nodes



Channel numbers
for example 1 … N
for analogue channels

1 LN for
Common features
and Co-ordination

Channel numbers
for example N+1 … N+M
for binary channels

The interaction is an application issue, not defined in the standard

Logical Nodes Groups defined in IEC 61850-7-4

IEC 61850-7-4 defines **91** Logical nodes divided into **13** Logical Groups
The first letter of the Logical Node identifies the group.

Logical Group	Name	Number of Logical Nodes
L	System LN	2
P	Protection	28
R	Protection related	10
C	Control	5
G	Generic	3
I	Interfacing and archiving	4
A	Automatic control	4
M	Metering and measurement	8
S	Sensor and monitoring	4
X	Switchgear	2
T	Instrument transformers	2
Y	Power transformers	4
Z	Further power system equipment	15

The X-Group: Logical nodes for switchgear

LNname	Function	
XCBR	Circuit breaker	a high-power switch capable of switching off or on under full load current (Schalter, Interrupteur)
XSWI	Circuit switch	a switching device capable of electrically isolating a line, but which may only be operated when essentially no current is flowing

Logical Nodes Groups defined in IEC 61850-7-4

IEC 61850-7-4 defines **91** Logical nodes divided into **13** Logical Groups
The first letter of the Logical Node identifies the group.

Logical Group	Name	Number of Logical Nodes
L	System LN	2
P	Protection	28
R	Protection related	10
C	Control	5
G	Generic	3
I	Interfacing and archiving	4
A	Automatic control	4
M	Metering and measurement	8
S	Sensor and monitoring	4
X	Switchgear	2
T	Instrument transformers	2
Y	Power transformers	4
Z	Further power system equipment	15

The P-group, with 28 protection logical nodes

LNname	IEEE protection function(s)	Protection Function
PDIF	87,87P,87L,87N,87T,87B, 87M, 87G	Differential
PDIR	87B	Direction comparison
PDIS	21	Distance protection
PDOP	32	Directional Overpower
PDUP	32,37,40	Directional Underpower
PFRC	81	Rate of change of frequency
PHAR	87T	Harmonic restraint
PHIZ	64	Ground detector
PIOC	50	Instantaneous overcurrent
PMRI	49R,66,48,51LR	Motor restart inhibition
PMSS		Motor starting supervision
POPF	55	Over power factor
PPAM		Phase angle measuring
PSCH	21,85	Protection scheme
PSDE		Sensitive directional earth fault
PTEF		Transient earth fault
PTOC	46,51,60,64R,64S,64W,67,67N,76	Time overcurrent
PTOF	81	Overfrequency
PTOV	47,59,59DC,60	Overvoltage
PTRC		
PTTR	49,49R,49S	Thermal overload
PTUC	37	Undercurrent
PTUV	27	Undervoltage
PTUF		Underfrequency
PUPF	55	Under power factor
PVOC	51V	Voltage controlled time overcurrent
PVPH	24	Volt per Hertz
PZSU	14	Zero speed or underspeed

(Protection functions standardized by IEEE)

21	Distance protection
24	Volts to Hertz = Overfluxing protection
25AR	Voltage- and synchro-check for autoreclosure
25C	Voltage- and synchro-check for control
49	Thermal overload
49D	Supervision of through current in diameter, current of phase A only
50BF	Breaker fail protection
50EndF	End-fault protection, fast overcurrent for faults between open CB and current-sensor
50GTert	Non-delayed overcurrent in ground connection of the tertiary
50Stub	T-zone protection
	Alternative 1: In case there is no sensor for protection in the feeder, a non-delayed feeder overcurrent function is part of the distance protection. This overcurrent function is released if the feeder disconnector is open and 21 blocked. Alternative 2: In case the feeder is equipped with sensors for protection, a dedicated differential protection is applied (additional zone of REB500)
51	Time overcurrent phase
51G	Ground back-up overcurrent
51N	Overcurrent protection measuring the transformer neutral current
59	Overvoltage
64Tert	Zero-sequence overvoltage as ground protection of an ungrounded tertiary circuit
67G	DEF = directional ground fault with communication to opposite line end
79	Autoreclosure
87B	Busbar Protection
87B1	Protection of bus 1
87B2	Protection of bus 2
87REF	Restricted earth fault, restricted to one winding
87REFMainWdg/ph	Biased differential for main winding, phase-segregated. Used as ground fault protection in case of transformer groups with one tank per phase
87L	Line differential protection
87T	Transformer differential protection
87T	overall Biased differential covering all windings

Logical Nodes Groups defined in IEC 61850-7-4

IEC 61850-7-4 defines **91** Logical nodes divided into **13** Logical Groups
The first letter of the Logical Node identifies the group.

Logical Group	Name	Number of Logical Nodes
L	System LN	2
P	Protection	28
R	Protection related	10
C	Control	5
G	Generic	3
I	Interfacing and archiving	4
A	Automatic control	4
M	Metering and measurement	8
S	Sensor and monitoring	4
X	Switchgear	2
T	Instrument transformers	2
Y	Power transformers	4
Z	Further power system equipment	15

The L-group: Logical Node Zero and LPHD

LNname	Function	
LLN0	Logical Node Zero	Special LN that administrates the virtual device it is part of. It defines in particular the communication objects and the log of the virtual device.
LPHD1	Physical Device Logical Node	represents the physical device, and in particular its communication properties, that are identical for all Logical Devices

Logical Node Model

All logical nodes are constructed according to the template:

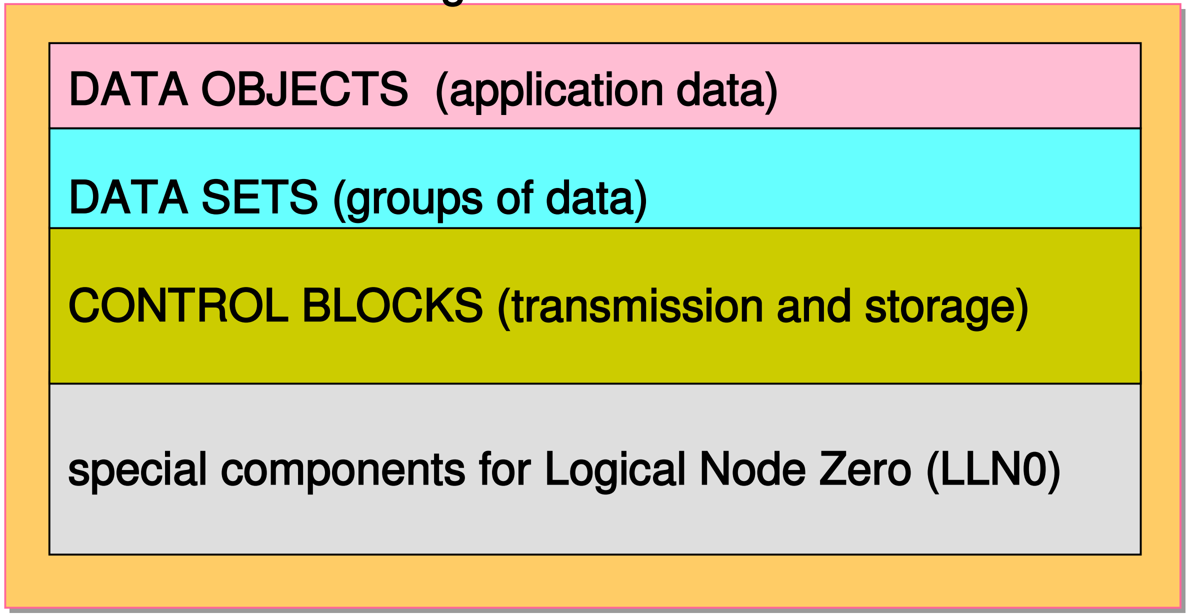
Logical-Node class

Attribute Name	Attribute type	Explanation
LNName LNRef	ObjectName ObjectReference	String of characters, e.d XCBR1 Location string, e.g. Q1B1W1/XCBR1
Data [1..n]	DATA	Data Objects, will be detailed
DataSet [0..n]	DATA-SET	Named groups of Data Objects and attributes
BufferedReportControlBlock [0..n]	BRCB	Control block for events
UnbufferedReportControlBlock [0..n]	BRCB	
LogControlBlocks [0..n]	LCB	Control block for history
Only for LLN0		
SettingGroupsControlBlock [0..1] Log [0..1] GOOSEControlBlock [0..n] GSSEControlBlock [0..n] MulticastSampledValues [0..n] UnicastSampledValues [0..n]	SGCB LOG GoCB GsCB MSVCB USVCB	Control block for settings

Services
GetLogicalNodeDirectory
GetAllDataValues

Logical Node components

The attributes of logical nodes are divided into:



DATA OBJECTS (application data)

DATA SETS (groups of data)

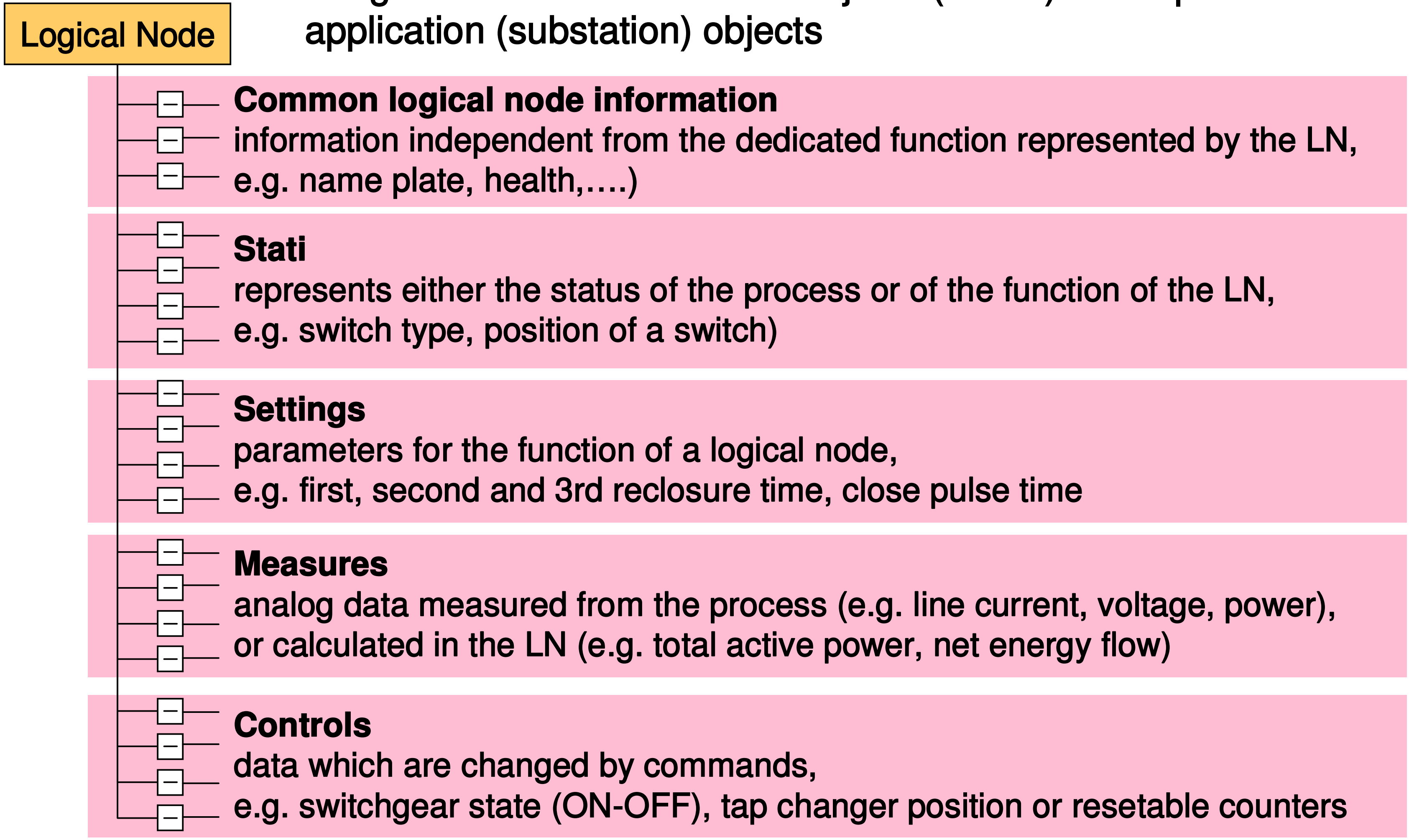
CONTROL BLOCKS (transmission and storage)

special components for Logical Node Zero (LLN0)

Let’s start with Data Objects

Modeling the Data Objects (DATA) in the Logical nodes

A logical node contains Data Objects (DATA) that represent application (substation) objects



Logical Node

Common logical node information
information independent from the dedicated function represented by the LN, e.g. name plate, health,....)

Stati
represents either the status of the process or of the function of the LN, e.g. switch type, position of a switch)

Settings
parameters for the function of a logical node, e.g. first, second and 3rd reclosure time, close pulse time

Measures
analog data measured from the process (e.g. line current, voltage, power), or calculated in the LN (e.g. total active power, net energy flow)

Controls
data which are changed by commands, e.g. switchgear state (ON-OFF), tap changer position or resetable counters

Example of DATA in a Logical Node: XCBR

XCBR			
Data Object	**Explanation**	**CDC**	**Mandatory**
	Basic LN		
Mod	Mode	INC	M
Beh	Behavior	INS	M
Health	Health	INS	M
NamePlt	Name Plate	LPL	
Loc	Local operation, not remote	SPS	
EEHealth	External equipment health	INS	
EEName	External equipment name plate	DPL	
NamPlt	Name Plate	LPL	
OpCnt	Operation counter	INS	M
	Controls		
Pos	Switch position	DPC	M
BlkOpn	Block opening	SPC	M
BlkCls	Block closing	SPC	M
ChaMotEna	Charger motor enable	SPC	
	Measures		
SumSwARs	Sum of switched amperes, resetable	BCR	
	Status		
CBOpCap	Circuit breaker operating capability	INS	M
POWCap	Point on wave switching capability	INS	
MaxOpCap	Operating capability when fully charged	INS	

common to all logical nodes

Pos is a DATA of Logical Node XCBR

A Data Object consists of Data Attributes

Each attribute of a DATA consists of a number of Data Attributes,
with a Data Attribute Type (DAType) that belong to Functional Constraints (FC)

DATA “Pos”		
Attribute Name	**Attribute Type**	**Functional Constraint**
stVal q t	BOOLEAN Quality TimeStamp	Status (ST)
d	Visible String255	Description (DC)
subEna subVal subQ subID	BOOLEAN BOOLEAN Quality Visible String64	Substitution (SV)

CDC = DPC

Basic Type

Common data attribute type

only needed when substitution is possible

Common Data Classes in IEC 61850-7-4

Many Logical Nodes have Data Objects with the same Data Attributes.

For instance, all binary input variables need the Data Attributes
<status>
<quality>
<timestamp>
<description>

To simplify engineering, IEC 61850 defined standard groups of Data Attributes, called CDC ("Common Data Classes")

("Classes" is not related to classes in object-oriented languages, a class is similar to a "struct" in "C".

Each Data Object of a logical node belongs to a CDC.

The 30 Common Data Classes of 61850-7-3

Status information (binary, integer):

SPS:	Single Point Status
DPS:	Double Point Status
INS:	Integer Status
ACT:	Protection Activation info
ACD:	Activation Info Directional Protection
SEC:	Security Violation Counting
BCR:	Binary Counter Reading

Measurand information:

MV:	Measurement Value
CMV:	Complex Measured Variable
SAV:	Sampled Value
WYE:	Phase to Ground
DEL:	Phase to Phase
SEQ:	Sequence
HMV:	Harmonic Value
HWYE:	Harmonic Value for WYE
HDEL:	Harmonic Value for DEL

Controllable status:

SPC	Single Point Control
DPC	Double Point Control
INC	Integer Status Control
BSC	Binary Controlled Step Position Info
ISC	Integer Controlled Step Position Info

Controllable Analog:

APC	(fc=SP, set point)

Status settings:

SPG	Single Point Setting
ING	(fc = SG, SE or SP)

Analog settings:

ASG,
CURVE (fc= SG, SE or SP)

Descriptive information:

DPL	Device Name Plate
LPL	Logical Node Name Plate
CSD	Curve Shape Description

These are all the possible types for Data Objects

CDC: Single Point Setting (SPS)

Single Point Setting (SPS) class					
Attribute	**Attribute Type**	**FC**	**TrgOp**	**Value/Value Range**	**M/O/C**
			status		
stVal	BOOLEAN	ST	dchg	TRUE | FALSE	M
q	Quality	ST	qchg		M
t	TimeStamp	ST			M
			substitution		
subEna	BOOLEAN	SV			PICS_SUBST
subVal	BOOLEAN	SV		TRUE | FALSE	PICS_SUBST
subQ	Quality	SV			PICS_SUBST
subID	VISIBLE STRING64	SV			PICS_SUBST
			configuration, description and extension		
d	VISIBLE STRING255	DC	Text		O
dU	UNICODE STRING255	DC			O
cdcNs	VISIBLE STRING255	EX			AC_DLNDA_M
cdcName	VISIBLE STRING255	EX			AC_DLNDA_M
dataNs	VISIBLE STRING255	EX			AC_DLN_M

CDC: Common Measurement Value (CMV)

Common Measurement Value					
Attribute	**Attribute Type**	**FC**	**TrgOp**	**Value/Value Range**	**M/O/C**
		measured attributes			
instCVal	Vector	MX			O
cVal	Vector	MX	dchg		M
range	ENUMERATED	MX	dchg	normal|high|low|high-high|low-low|...	O
q	Quality	MX	qchg		M
t	TimeStamp	MX			M
		substitution			
subEna	BOOLEAN	SV			PICS_SUBST
subCVal	Vector	SV			PICS_SUBST
subQ	Quality	SV			PICS_SUBST
subID	VISIBLE STRING64	SV			PICS_SUBST
		configuration, description and extension			
units	Unit	CF		see Annex A	O
db	INT32U	CF		0 … 100 000	O
zeroDb	INT32U	CF		0 … 100 000	O
rangeC	RangeConfig	CF			GC_CON
magSVC	ScaledValueConfig	CF			AC_SCAV
angSVC	ScaledValueConfig	CF			AC_SCAV
angRef	ENUMERATED	CF		V | A | other …	O
smpRate	INT32U	CF			O
d	VISIBLE STRING255	DC	Text		O
dU	UNICODE STRING255	DC			O
cdcNs	VISIBLE STRING255	EX			AC_DLNDA_M
cdcName	VISIBLE STRING255	EX			AC_DLNDA_M
dataNs	VISIBLE STRING255	EX			AC_DLN_M

CDC Controllable Double Point CDP (e.g. Pos in XCBR)

Attribute	Attribute Type	FC	TrgOp	Value/Value Range	M/O/C
control and status					
ctlVal	BOOLEAN	CO		off (FALSE) | on (TRUE)	AC_CO_M
operTm	TimeStamp	CO			AC_CO_O
origin	Originator	CO, ST			AC_CO_O
ctlNum	INT8U	CO, ST		0..255	AC_CO_O
stVal	CODED ENUM	ST	dchg	intermediate | off | on | bad	M
q	Quality	ST	qchg		M
t	TimeStamp	ST			M
stSeld	BOOLEAN	ST	dchg		AC_CO_O
substitution					
subEna	BOOLEAN	SV			PICS_SUBST
subVal	CODED ENUM	SV		intermediate | off | on | bad	PICS_SUBST
subQ	Quality	SV			PICS_SUBST
subID	VISIBLE STRING64	SV			PICS_SUBST
configuration, description and extension					
pulseConfig	PulseConfig	CF			AC_CO_O
ctlModel	CtlModels	CF			M
sboTimeout	INT32U	CF			AC_CO_O
sboClass	SboClasse	CF			AC_CO_O
d	VISIBLE STRING255	DC		Text	O
dU	UNICODE STRING255	DC			O
cdcNs	VISIBLE STRING255	EX			AC_DLNDA_M
cdcName	VISIBLE STRING255	EX			AC_DLNDA_M
dataNs	VISIBLE STRING255	EX			AC_DLN_M

M = mandatory, O = optional, AC_CO_M: mandatory when AC_CO option slected, ….

Data Attribute Types

Data Attributes may be of

- primitive (a simple type, e.g. BOOLEAN)

- composite (constructed, e.g. Vector) in which case they consist of Attributes Components

Vector Type Definition			
Attribute Name	**Attribute Type**	**Value/Value Range**	**M/O/C**
mag	AnalogueValue		M
ang	AnalogueValue		O

AnalogueValue itself is defined as:

AnalogueValue Type Definition			
Attribute Name	**Attribute Type**	**Value/Value Range**	**M/O/C**
I	INT32		integer value GC_1
f	FLOAT32		floating point value GC_1

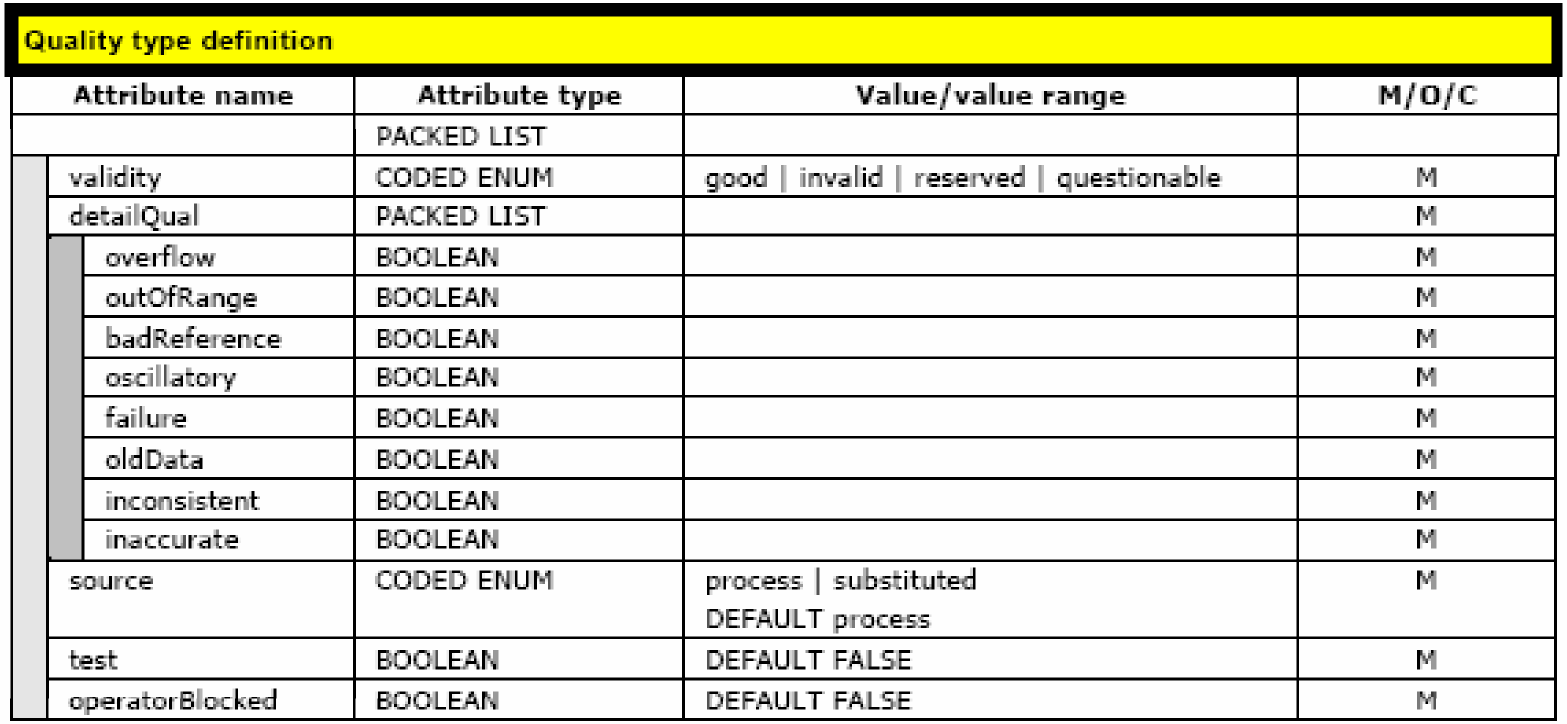
e.g. **PhaseVoltage.mag.f** is the magnitude of the phase voltage as a floating point number

Common Data Attributes

IEC 61850-7-3.6 defines 12 Common Data Attributes (CDA)

Quality
Analogue value
Configuration of analogue value
Range configuration
Step position with transient indication
Pulse configuration
Originator
Unit definition
Vector definition
Point definition
CtlModels definition
SboClasses definition (Select Before Operate)

Common Data Attributes: e.g. Quality



Quality type definition

Attribute name	Attribute type	Value/value range	M/O/C
	PACKED LIST		
validity	CODED ENUM	good | invalid | reserved | questionable	M
detailQual	PACKED LIST		M
overflow	BOOLEAN		M
outOfRange	BOOLEAN		M
badReference	BOOLEAN		M
oscillatory	BOOLEAN		M
failure	BOOLEAN		M
oldData	BOOLEAN		M
inconsistent	BOOLEAN		M
inaccurate	BOOLEAN		M
source	CODED ENUM	process | substituted DEFAULT process	M
test	BOOLEAN	DEFAULT FALSE	M
operatorBlocked	BOOLEAN	DEFAULT FALSE	M

Functional Constraints

Variables are of different relevance and time criticality.

e.g. the position variable “Pos” of a circuit breaker is of class CDP,
it contains variables of different urgency:
-the actual position of the switch (XCBR2.StVal) and
-the description (XCBR2.d).

To retrieve information from an IED selectively, each leave has an associated a functional constrain, that becomes part of its name.

The functional constraints apply to each data attribute.

A leaf can belong to more than one Function Constraint, although this occurs seldom.

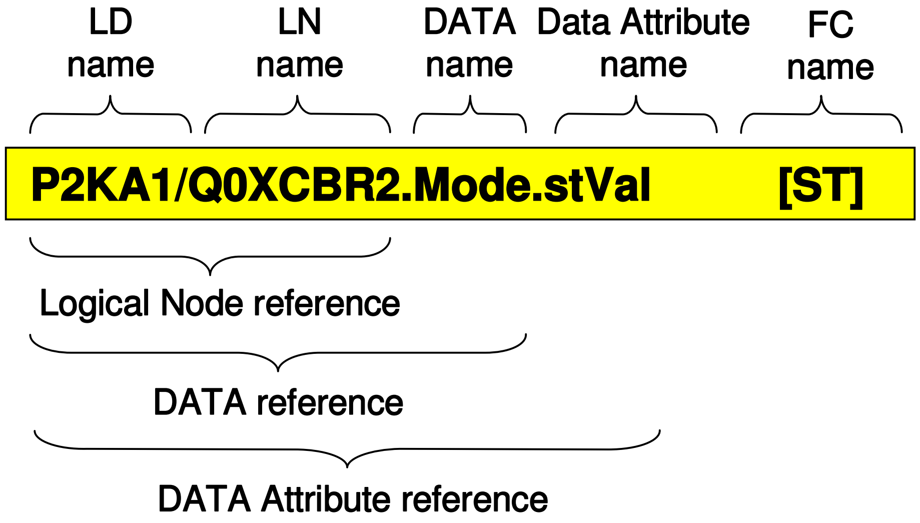
Functional Constraints

FC	Meaning	Services
ST MX	Process values: Status, Measurand	Read, substitute, report, log
CO SP	Process commands; binary, analog (Set Points)	Operate
SV	Substitution related	Substitute (read, write)
CF DC	Configuration, description	Read, Write (report, log)
SG SE	Parameters, in setting groups (SG: the active, SE: the editable value)	GetSGValue, SetSGValue
CB related	Each CB type	GetxxxCBValues, SetxxxCBValues
SP	Parameter (outside SG)	Read, write
EX	Name space definition	Read

Naming scheme (LNName and LNRef)

The name of the logical node is that of an instance of the standard logical nodes, unique in the Logical Device e.g. XCBR2

The Object reference is the full path of the object, completed with the Functional Constraint:



LD name

LN name

DATA name

Data Attribute name

FC name

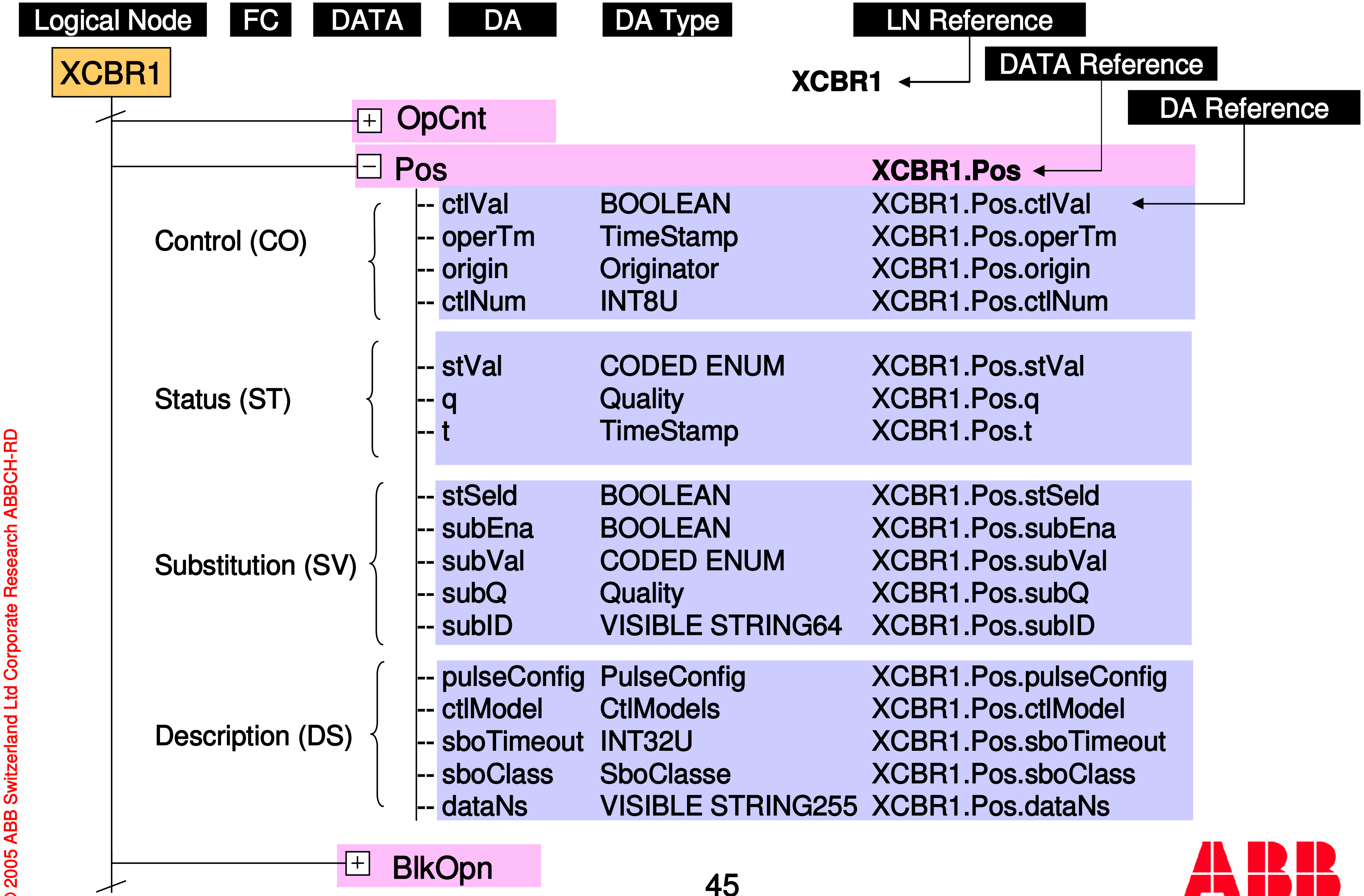
P2KA1/Q0XCBR2.Mode.stVal [ST]

Logical Node reference

DATA reference

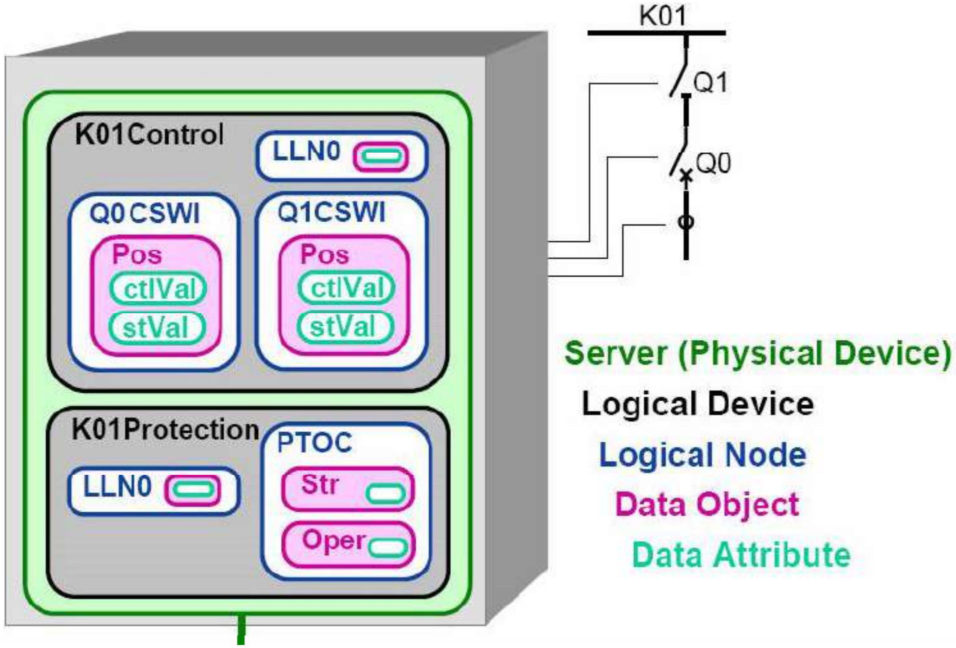
DATA Attribute reference

Naming a Data Attribute (“Pos” in an “XCBR”)



Logical Node	FC	DATA	DA	DA Type	LN Reference / DATA Reference / DA Reference
XCBR1					XCBR1
		⊞ OpCnt			
		⊟ Pos			XCBR1.Pos
	Control (CO)		-- ctlVal	BOOLEAN	XCBR1.Pos.ctlVal
			-- operTm	TimeStamp	XCBR1.Pos.operTm
			-- origin	Originator	XCBR1.Pos.origin
			-- ctlNum	INT8U	XCBR1.Pos.ctlNum
	Status (ST)		-- stVal	CODED ENUM	XCBR1.Pos.stVal
			-- q	Quality	XCBR1.Pos.q
			-- t	TimeStamp	XCBR1.Pos.t
	Substitution (SV)		-- stSeld	BOOLEAN	XCBR1.Pos.stSeld
			-- subEna	BOOLEAN	XCBR1.Pos.subEna
			-- subVal	CODED ENUM	XCBR1.Pos.subVal
			-- subQ	Quality	XCBR1.Pos.subQ
			-- subID	VISIBLE STRING64	XCBR1.Pos.subID
	Description (DS)		-- pulseConfig	PulseConfig	XCBR1.Pos.pulseConfig
			-- ctlModel	CtlModels	XCBR1.Pos.ctlModel
			-- sboTimeout	INT32U	XCBR1.Pos.sboTimeout
			-- sboClass	SboClasse	XCBR1.Pos.sboClass
			-- dataNs	VISIBLE STRING255	XCBR1.Pos.dataNs
		⊞ BlkOpn			

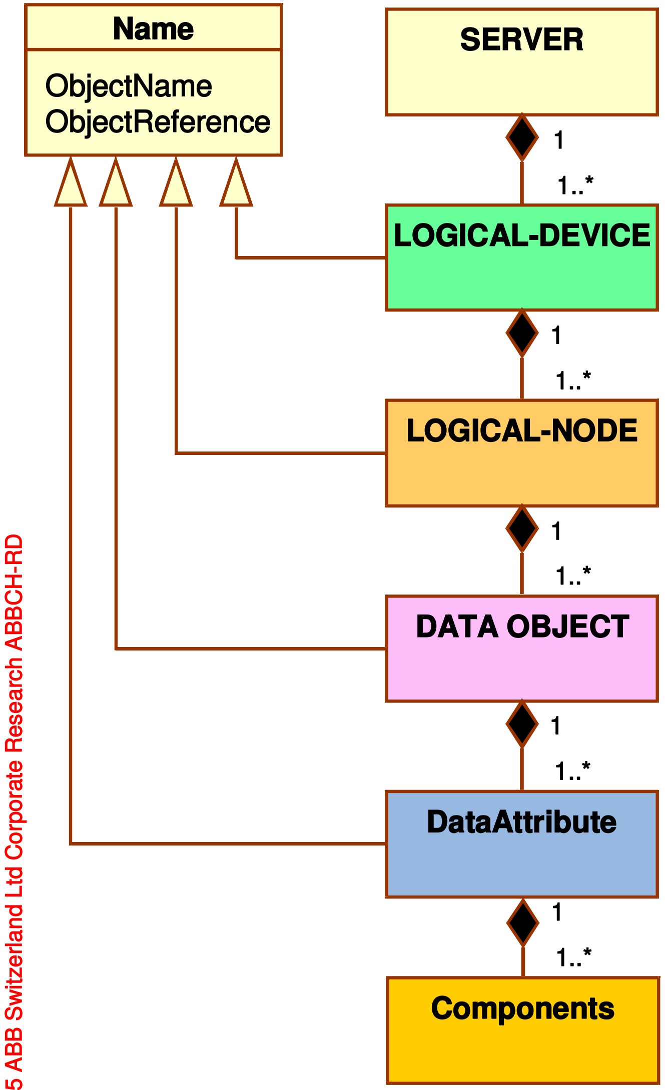
Resumé



IEC 61850-7-4 defines 90 Logical Nodes, divided into 13 groups (L,P,R,C,G,…)

Each LN consists of Data Object (DATA) grouped in 5 categories
general, settings, status, command and measure
Each Data Object consists of Data Attributes (DA) that belong to one of 30
CDC (common data classes) defined in IEC 61850-3.
Each CDC consists of other CDC or of components.

Simplified IEC 61850 object model



Physical Device:
access by network address

Aggregates data from multiple devices into a single physical device

Represents the protection and control functions within a device, and the device itself (90LN)

Variables of a Logical Node represented as a collection of Common Data Classes (30 CDC)

Elements of an Common Data Class (value, time-stamp,..) belonging to Functional Constraints (12 FC)

Basic or composite Data Attribute types, e.g. Common Data Attributes (12 CDA)

The IEC 61850 data exchange model

The IEC 61850 supports two kinds of traffic:

1) real-time traffic based directly on communication layer 2, GOOSE or Sampled Values. Encoding of these data is simplified.

2) sporadic traffic over TCP/IP – MMS using ASN.1 / BER encoding.

3) The sporadic traffic supports the object model described

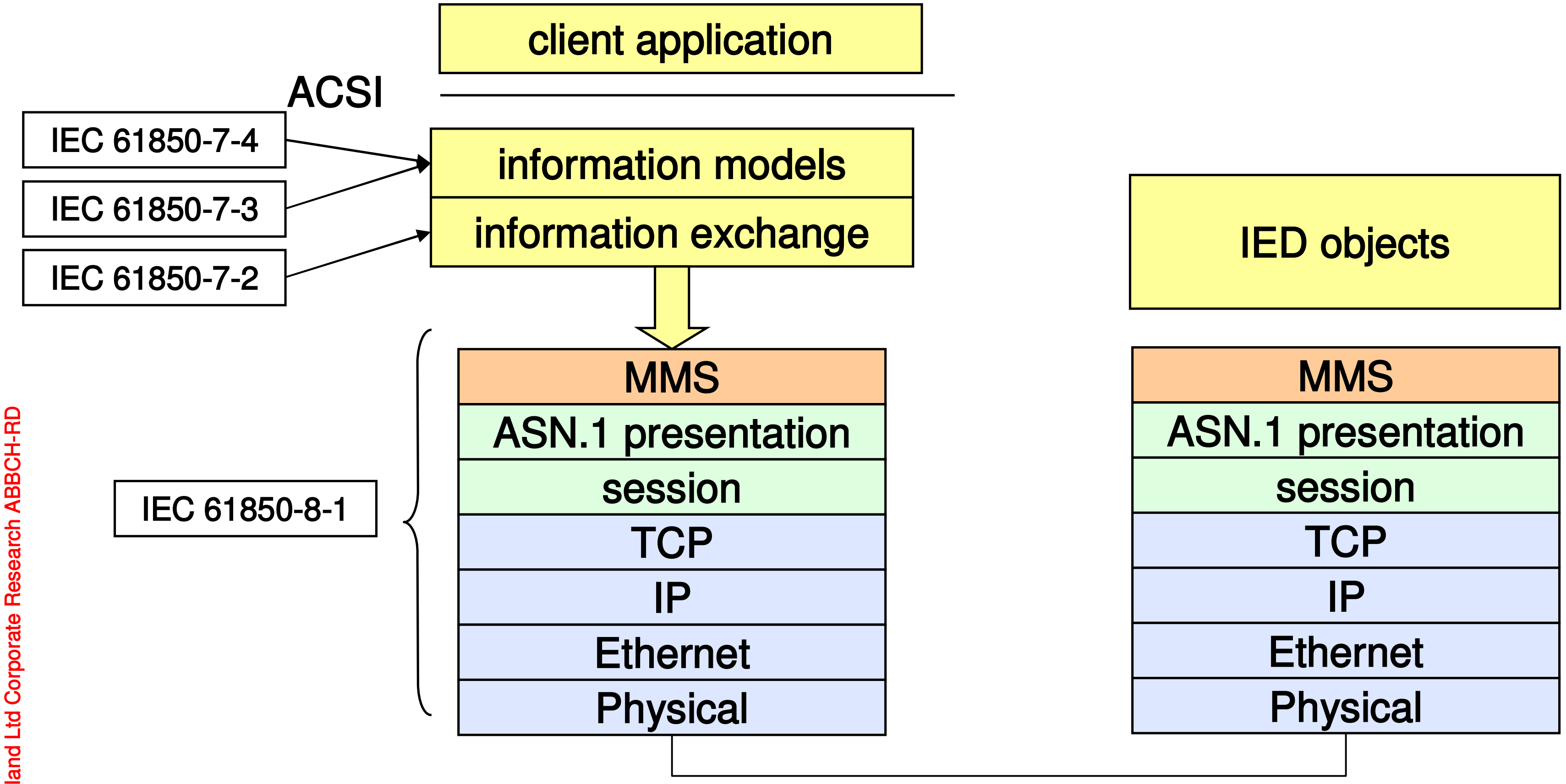
Datasets

Datasets are lists of data attributes that are handled as a whole.

For instance, all Switch positions can be put into a dataset.

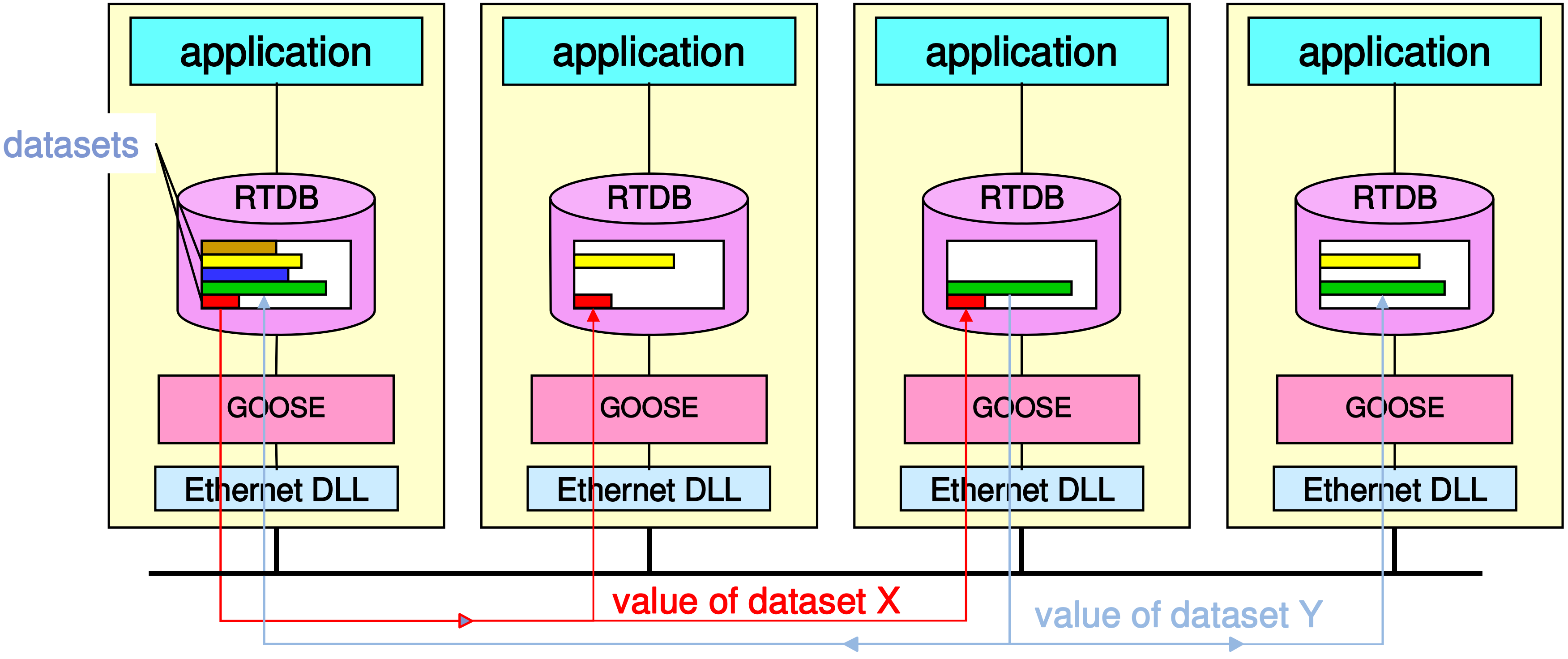
A dataset can be defined

ACSI


client application
ACSI
IEC 61850-7-4
IEC 61850-7-3
IEC 61850-7-2
information models
information exchange
IED objects
IEC 61850-8-1
MMS
ASN.1 presentation
session
TCP
IP
Ethernet
Physical
MMS
ASN.1 presentation
session
TCP
IP
Ethernet
Physical

GOOSE traffic

GOOSE exchanges real-time data on the publisher / subscriber principle:
An application reads and writes its real-time database, that is organized as datasets.
Each real-time database contains a subset of all datasets on the network.
Reading or writing the datasets causes no immediate network traffic.
The GOOSE protocol refreshes the data bases by broadcasting the dataset values that changed, several times in sequence.



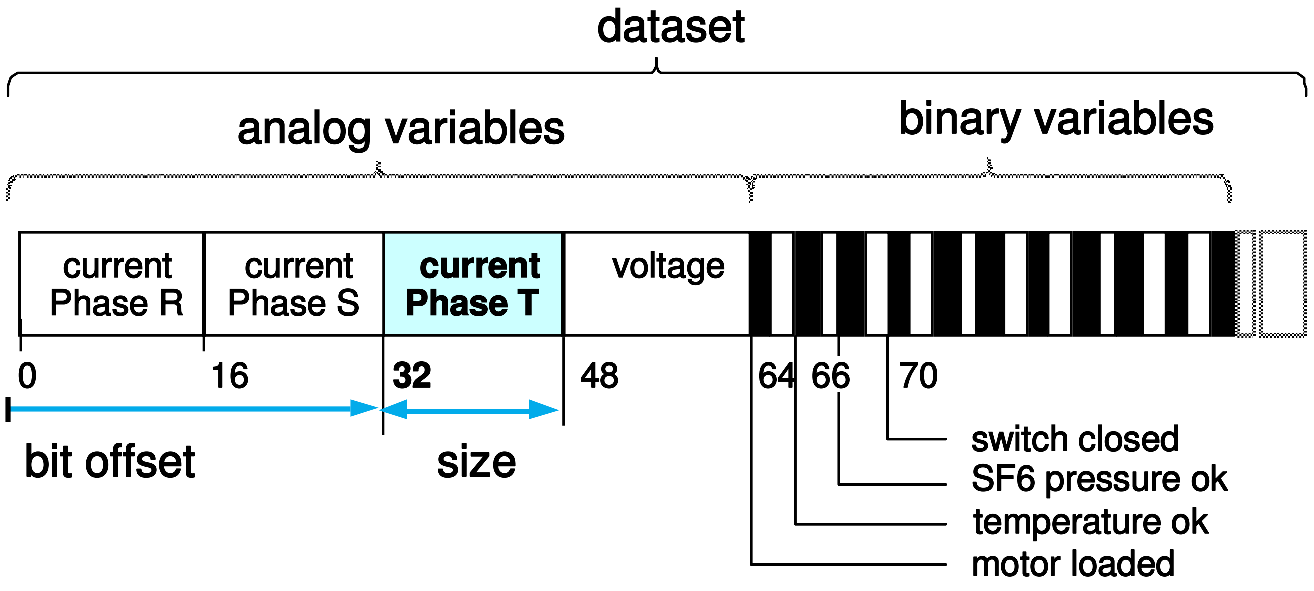
Datasets

It is economical to transport several variables in the same frame as a dataset.

A dataset is treated as a whole for communication and access.

A variable is identified within a dataset by its offset and its size

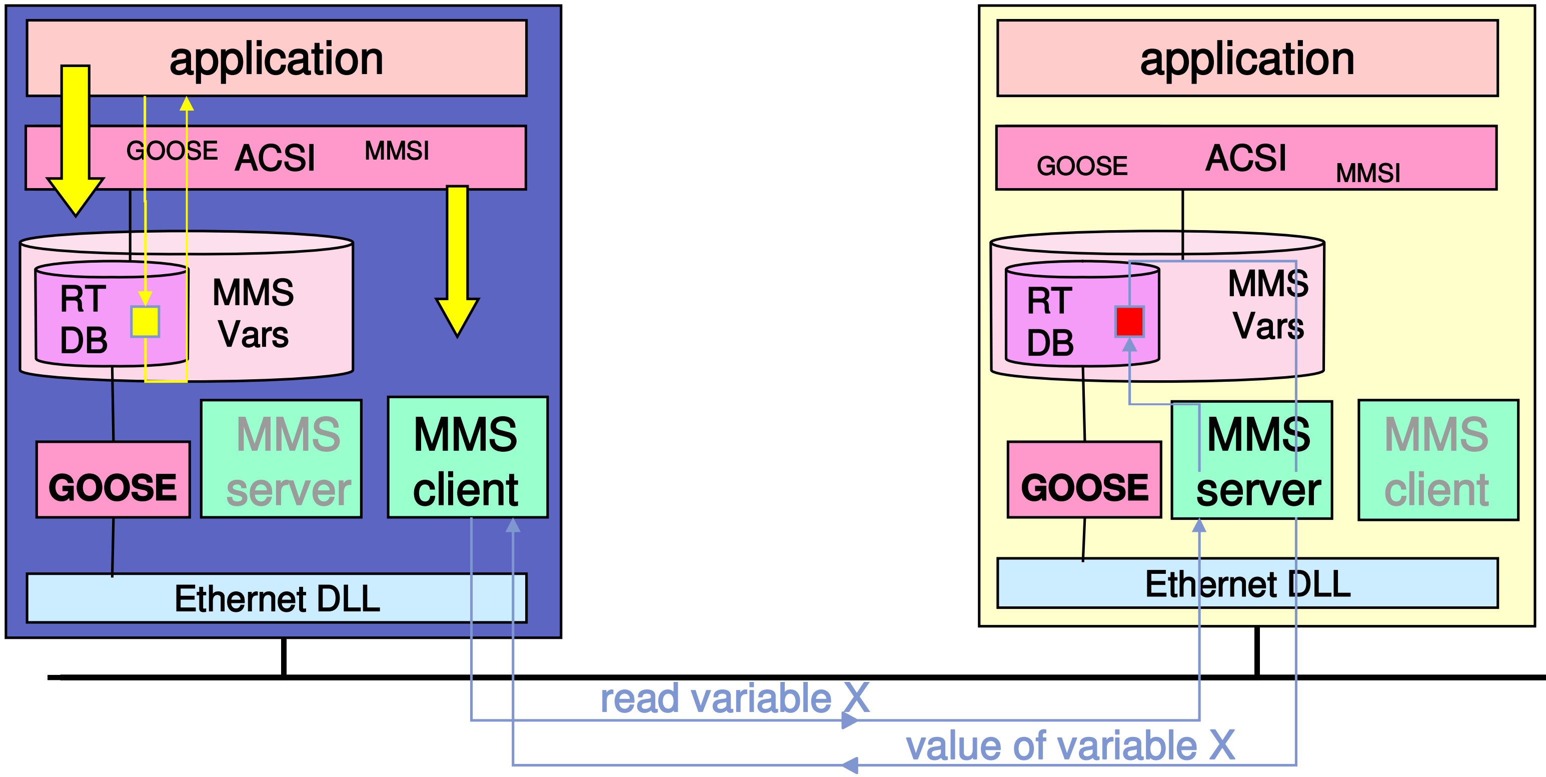
Variables may be of different types, types can be mixed.



MMS access to remote variables in 61850

An application can access a remote variable:
- through its MMS client (somewhat slow)
- through its local copy in the GOOSE RTDB.

An application can access a remote GOOSE RTDB only through its MMS client



MMS services in IEC 61850

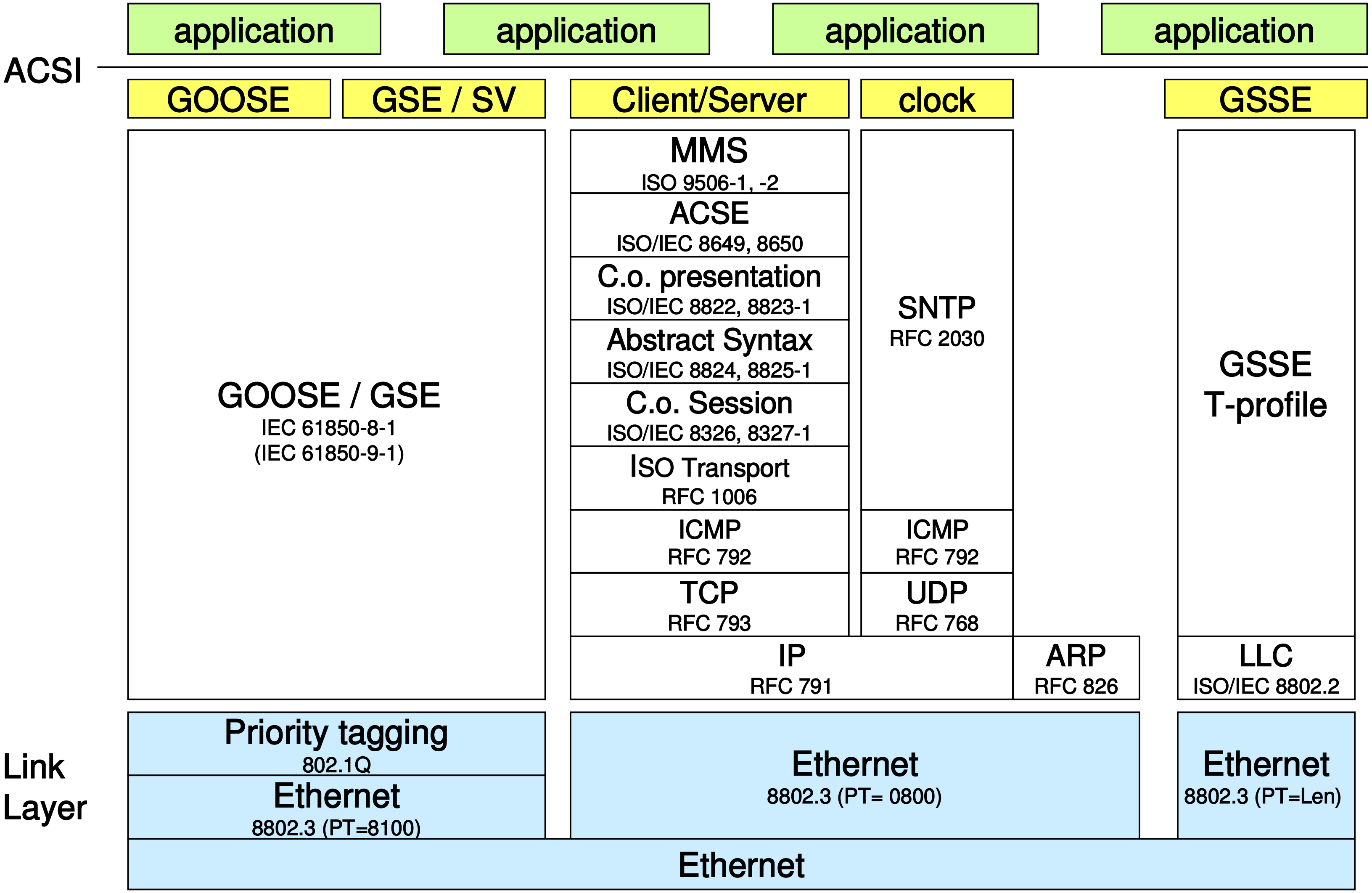
IEC 61850 Object	MMS Object	MMS Services
Server	Application Process VMD	Initiate Conclude Abort Reject Cancel Identify
Logical Nodes and Data	Named Variable Objects	Read Write InformationReport GetVariableAccessAttribute GetNameList
Data Sets	Named Variable List Objects	GetNamedVariableListAttributes GetNameList DefineNamedVariableList DeleteNamedVariableList GetNameList Read Write InformationReport
Logs	Journal Objects	ReadJournal InitializeJournal GetNameList
Logical Devices	Domain Objects	GetNameList GetDomainAttributes StoreDomainContents
Files	Files	FileOpen FileRead ObtainFile FileClose FileDirectory FileDelete

Control Blocks

Control blocks define, how and when the data is transferred

Reports: at data / quality change, changed data only, or periodically
GSE: immediate at some change, else periodically; always whole set
Sampled Values (SV, SMV): periodically
Log: stored at change, changed data only; fetched when needed

IEC 61850 stack (detail)



application

application

application

application

ACSI

GOOSE

GSE / SV

Client/Server

clock

GSSE

GOOSE / GSE
IEC 61850-8-1
(IEC 61850-9-1)

MMS
ISO 9506-1, -2

ACSE
ISO/IEC 8649, 8650

C.o. presentation
ISO/IEC 8822, 8823-1

Abstract Syntax
ISO/IEC 8824, 8825-1

C.o. Session
ISO/IEC 8326, 8327-1

ISO Transport
RFC 1006

ICMP
RFC 792

TCP
RFC 793

SNTP
RFC 2030

ICMP
RFC 792

UDP
RFC 768

IP
RFC 791

ARP
RFC 826

GSSE
T-profile

LLC
ISO/IEC 8802.2

Link
Layer

Priority tagging
802.1Q

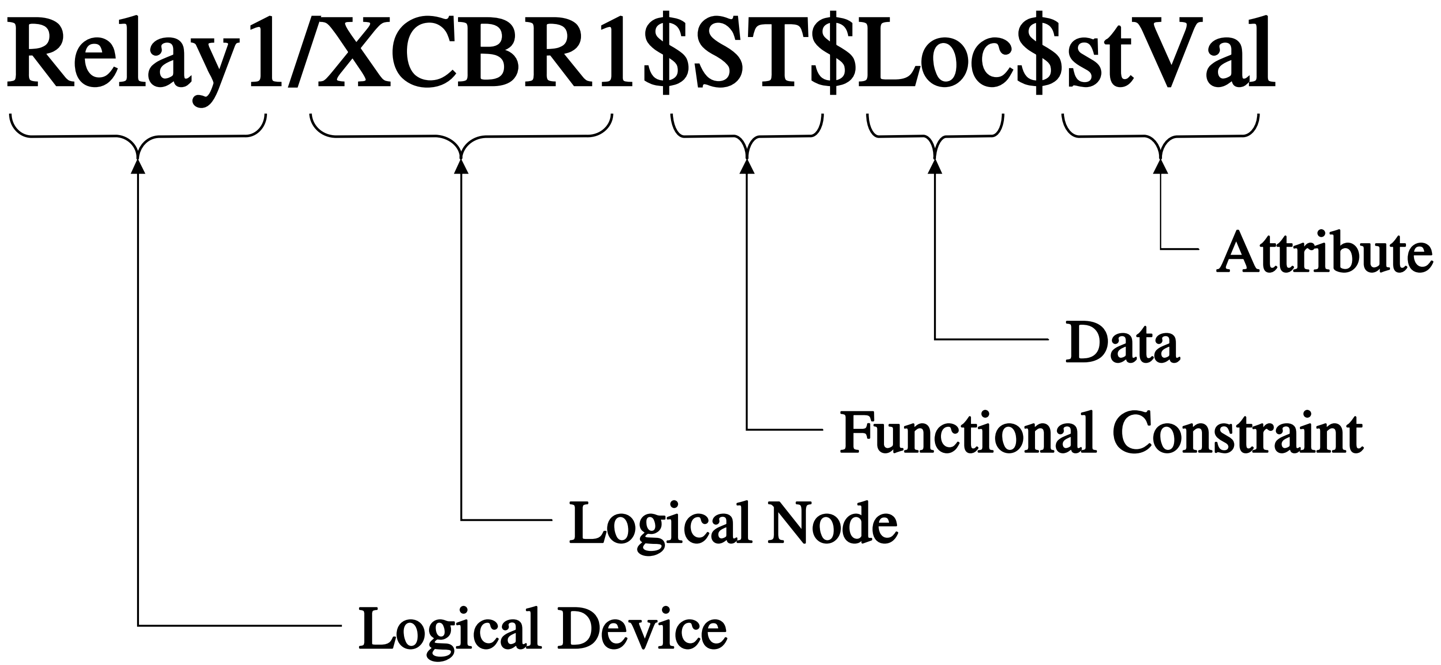
Ethernet
8802.3 (PT=8100)

Ethernet
8802.3 (PT= 0800)

Ethernet
8802.3 (PT=Len)

Ethernet

Object Name structure in MMS



IEC 61850 MMS Objects - Example for Breaker Control

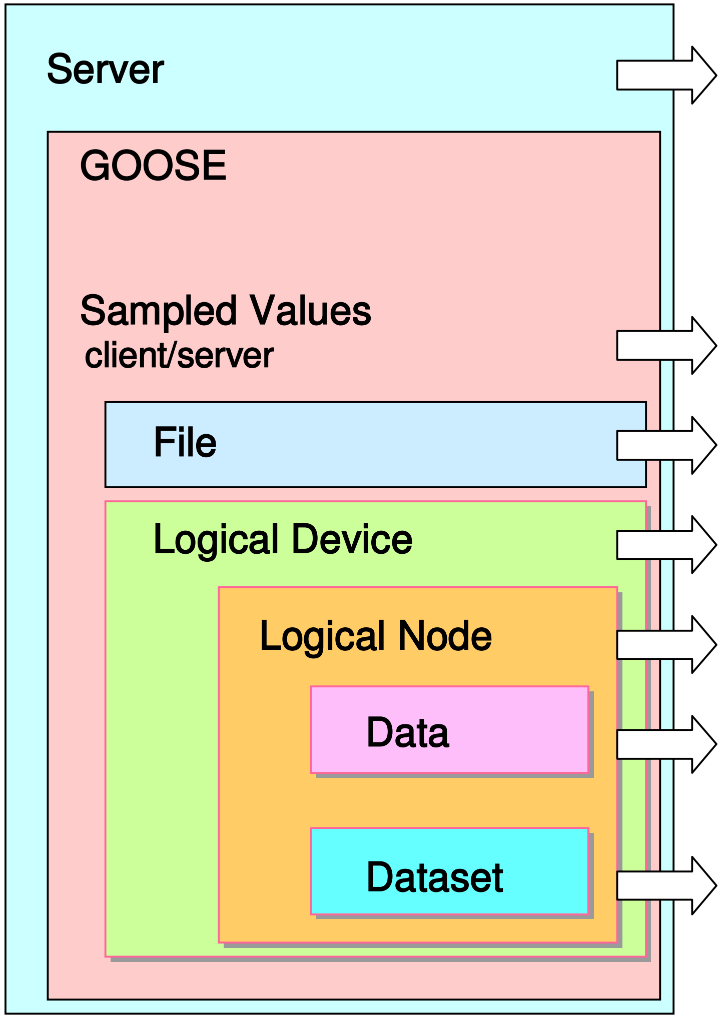
K03/Q0CSWI
K03/Q0CSWI$ST
K03/Q0CSWI$ST$Pos
K03/Q0CSWI$ST$Pos$stVal
K03/Q0CSWI$ST$Pos$q
K03/Q0CSWI$ST$Pos$t
K03/Q0CSWI$ST$Pos$origin$orCat
K03/Q0CSWI$ST$Pos$origin$orIdent
K03/Q0CSWI$SV
K03/Q0CSWI$SV$Pos
K03/Q0CSWI$SV$Pos$subEna
K03/Q0CSWI$SV$Pos$subVal
K03/Q0CSWI$SV$Pos$subQ
K03/Q0CSWI$SV$Pos$subID
K03/Q0CSWI$CO
K03/Q0CSWI$CO$Pos
K03/Q0CSWI$CO$Pos$ctlVal
K03/Q0CSWI$CO$Pos$origin$orCat
K03/Q0CSWI$CO$Pos$origin$orIdent
K03/Q0CSWI$CO$Pos$T
K03/Q0CSWI$CO$Pos$Test
K03/Q0CSWI$CO$Pos$Check
K03/Q0CSWI$CO$Pos$InvokeID
K03/Q0CSWI$CO$Pos$SID
K03/Q0CSWI$CF
K03/Q0CSWI$CF$Pos
K03/Q0CSWI$CF$Pos$ctlModel = sbo-with-enhanced-security

MMS AA-Specific Named Variable for negative 7-2 control responses

@/**LastApplError**
@/LastApplError$CntrlObj
@/LastApplError$Error
@/LastApplError$InvokeID
@/LastApplError$AddCause

defined in 7-2
defined in 8-1
no predefined names

Mapping to MMS



MMS Virtual Manufacturing Device (VMD)

Service Access Points

MMS file object

single MMS domain, same name

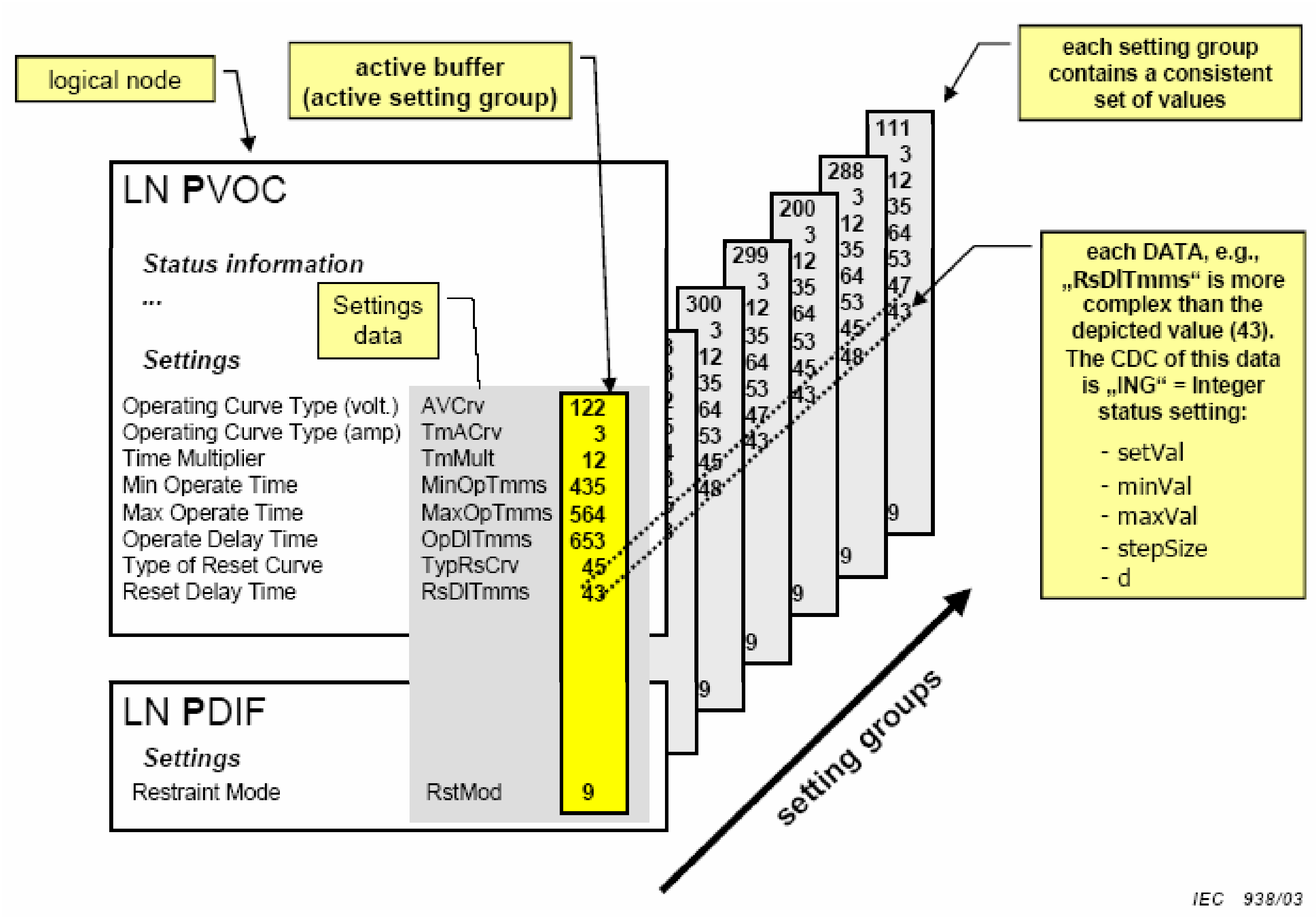
single MMS NameVariable (structured)

named components within MMS Type Description

MMS Named Variable List

IEC 61850
MMS Applied

Setting groups



logical node

active buffer
(active setting group)

each setting group contains a consistent set of values

each DATA, e.g., „RsDITmms“ is more complex than the depicted value (43). The CDC of this data is „ING“ = Integer status setting:

- setVal
- minVal
- maxVal
- stepSize
- d

LN PVOC

Status information

...

Settings data

Settings

Operating Curve Type (volt.)	AVCrv	122
Operating Curve Type (amp)	TmACrv	3
Time Multiplier	TmMult	12
Min Operate Time	MinOpTmms	435
Max Operate Time	MaxOpTmms	564
Operate Delay Time	OpDlTmms	653
Type of Reset Curve	TypRsCrv	45
Reset Delay Time	RsDlTmms	43

LN PDIF

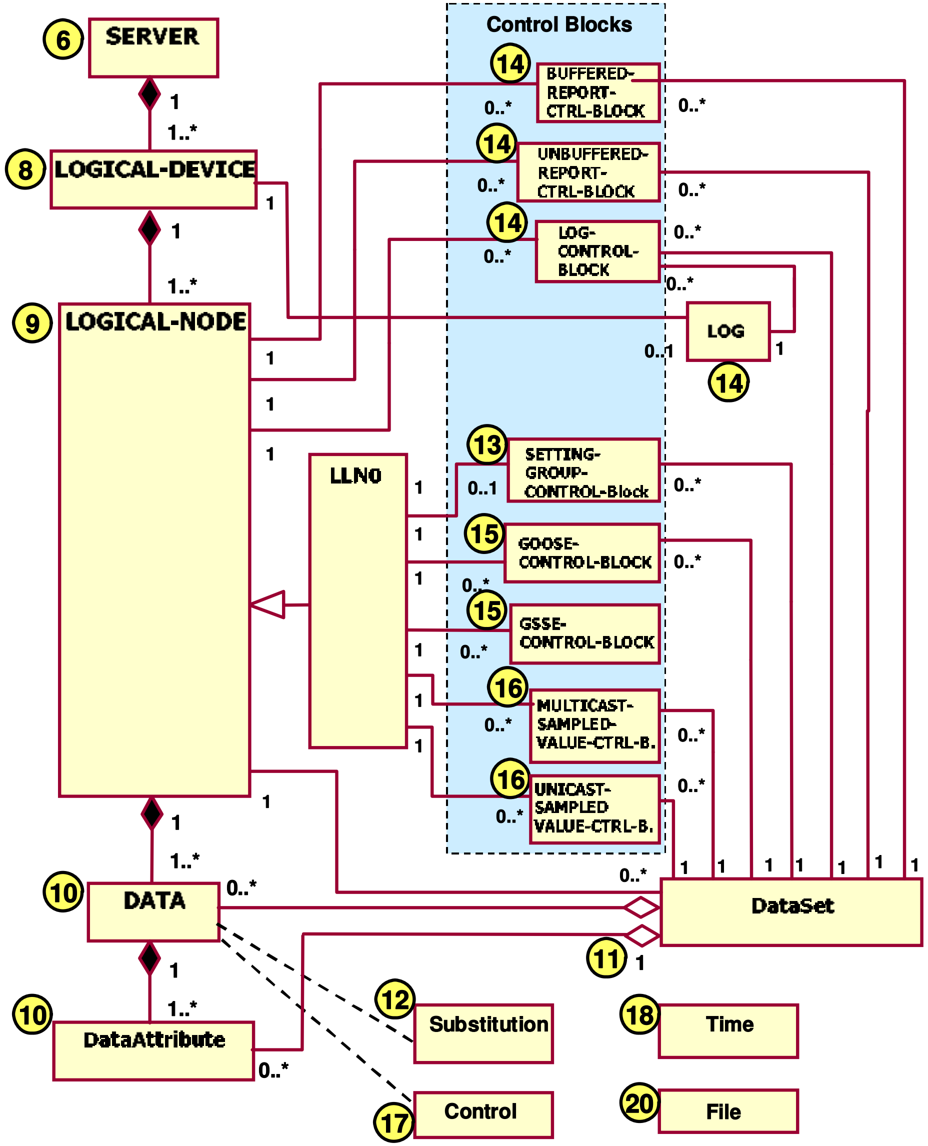
Settings

Restraint Mode	RstMod	9

A setting group is a set of parameters treated as a whole that can be edited and applied to a logical node.

IEC 61850
MMS Applied

Complete Object Model



Industrial Automation
Automation Industrielle
Industrielle Automation

OPC
OLE for Process Control

4 Access to devices

4.3 OLE for Process Control (OPC)

4.3.1 Common elements

Prof. Dr. H. Kirrmann

ABB Research Centre, Baden, Switzerland

2005 May, HK

Executive Summary

OPC is a set of standard commands collected in a software library (DLL) that can be called by client applications, written in Visual Basic, C# or other Microsoft programming languages, that allow to access automation devices (PLCs) in a uniform way, independently from their built or manufacturer.

To that effect, the particularities of the automation devices are hidden by an OPC server running either on the same machine as the client program or on another machine, by using DCOM. The OPC Servers are supplied by the manufacturer of the PLC or by 3rd parties and can manage several PLCs of the same type. Several servers can run in parallel.

The OPC library allows in particular to read and write process variables, read alarms and events and acknowledge alarms, and retrieve historical data from data bases according to several criteria.

Automation platforms such as ABB's 800XA platform act as OPC clients to collect data from PLCs or databases through third-party OPC servers. Several automation platforms act themselves as an OPC server to publish their data, events and historical data.

OPC is the preferred connectivity for 78% of MES, 75% of HMI / SCADA, 68% of DCS / PLC and 53% or ERP /Enterprise system level applications (according to Arc Advisory Group, 2004)"

keep on reading even if you are not an executive....

OPC Common Overview

OPC Common

- **Overview: usage and specifications**
- OPC as an integration tool
- Clients and Servers: configuration
- Automation and Custom Interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification

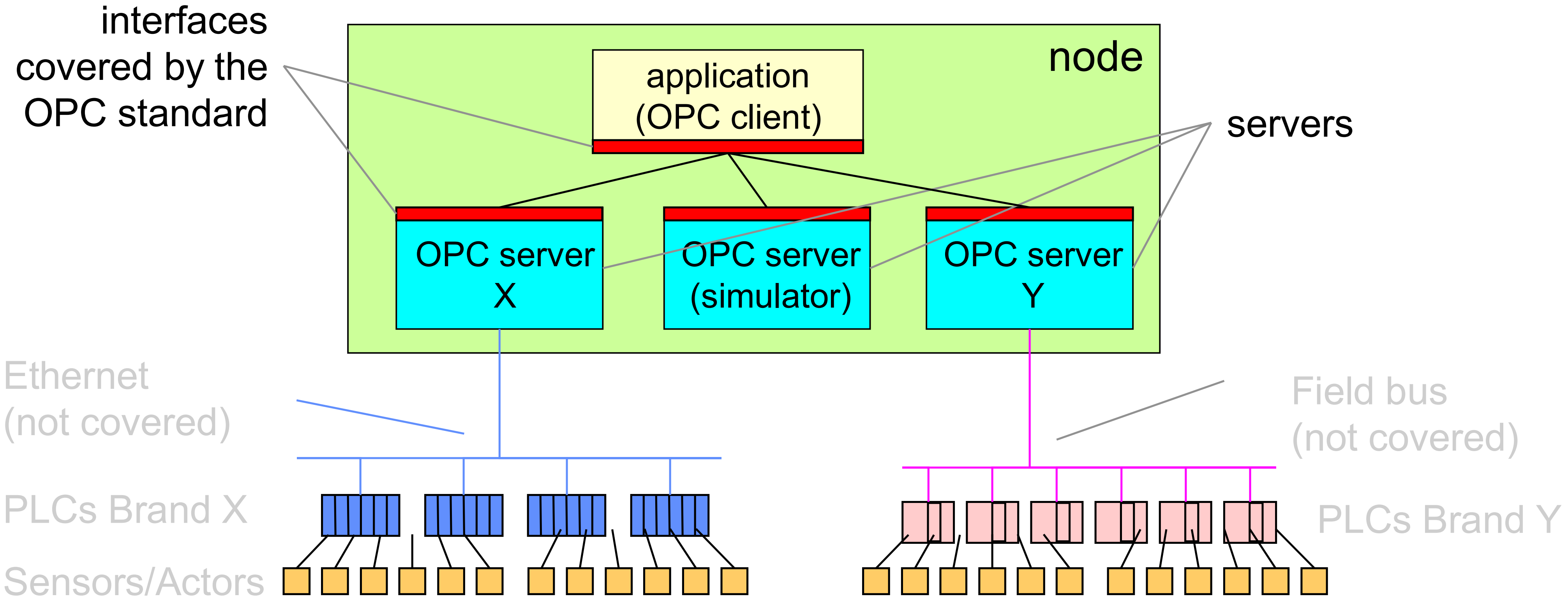
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

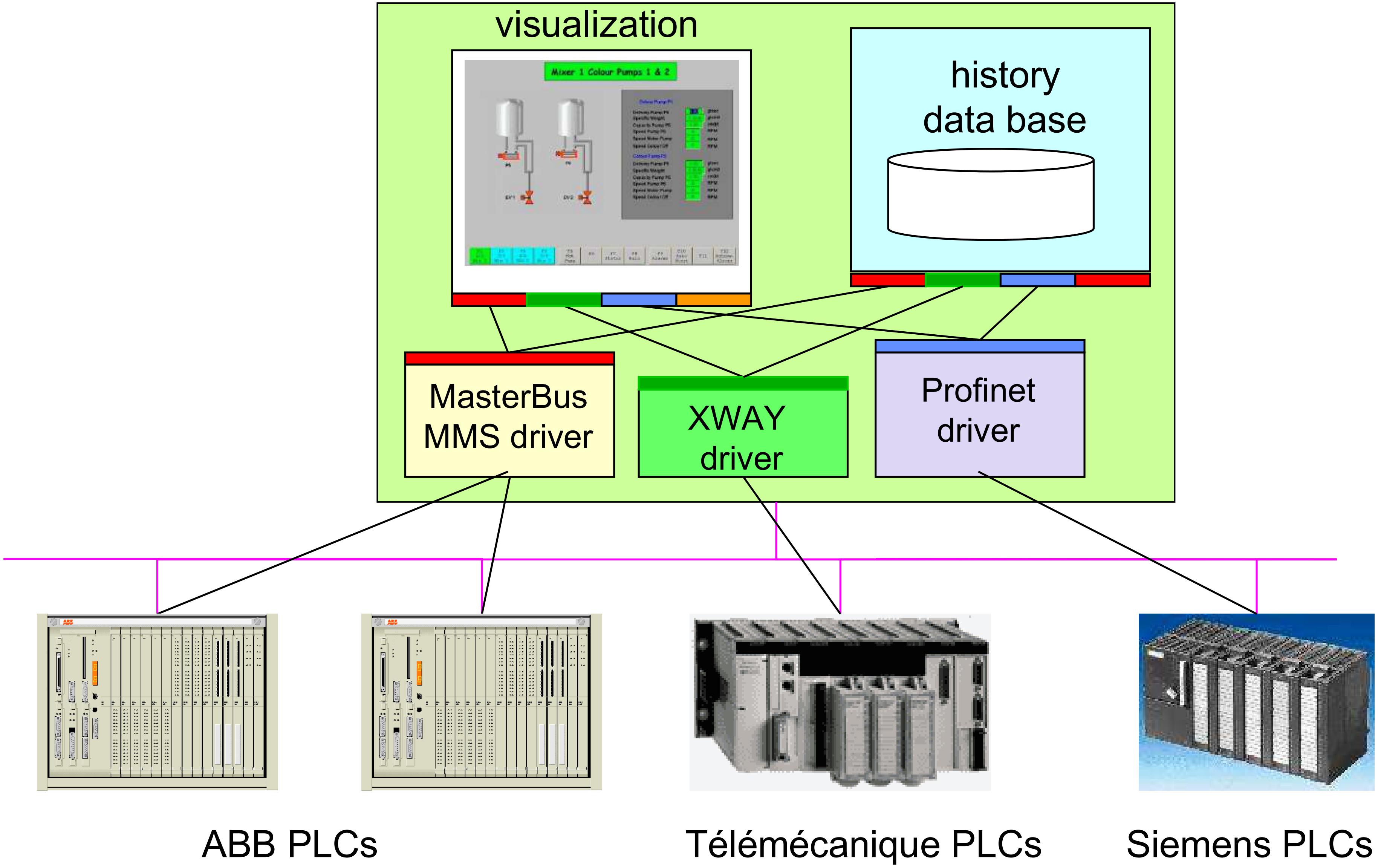
What is OPC ?

OPC (formerly: "OLE[1] for Process Control", now: "Open Process Control") is an industry standard set up by the *OPC Foundation* specifying the software interface (objects, methods) to a server that collects data produced by field devices and programmable logic controllers.

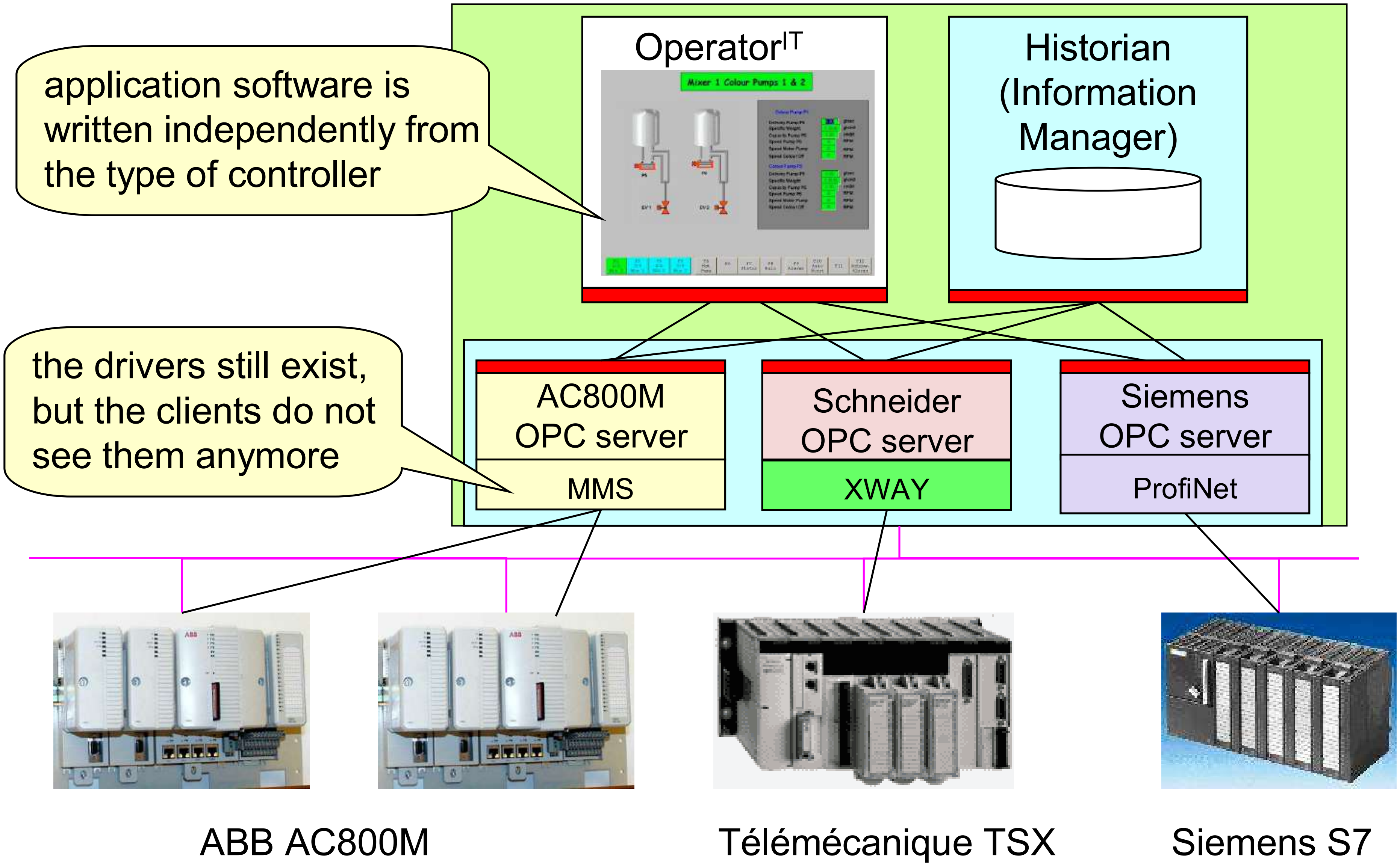


[1] OLE (Object Linking and Embedding) is a Microsoft technology for connecting software components. It has since been extended by the COM / DCOM technology. It corresponds to Java Beans.

Before OPC


visualization
Mixer 1 Colour Pumps 1 & 2
history
data base
MasterBus
MMS driver
XWAY
driver
Profinet
driver
ABB PLCs
Télémécanique PLCs
Siemens PLCs

With OPC: ABB Operator Workplace Connection



application software is written independently from the type of controller

the drivers still exist, but the clients do not see them anymore

Importance

OPC is the greatest improvement in automation since IEC 61131.

OPC is supported by the OPC foundation (http://www.opcfoundation.org/)

More than 150 vendors offer OPC servers to connect their PLCs, field bus devices, displays and visualization systems.

OPC is also used for data exchange between applications and for accessing databases

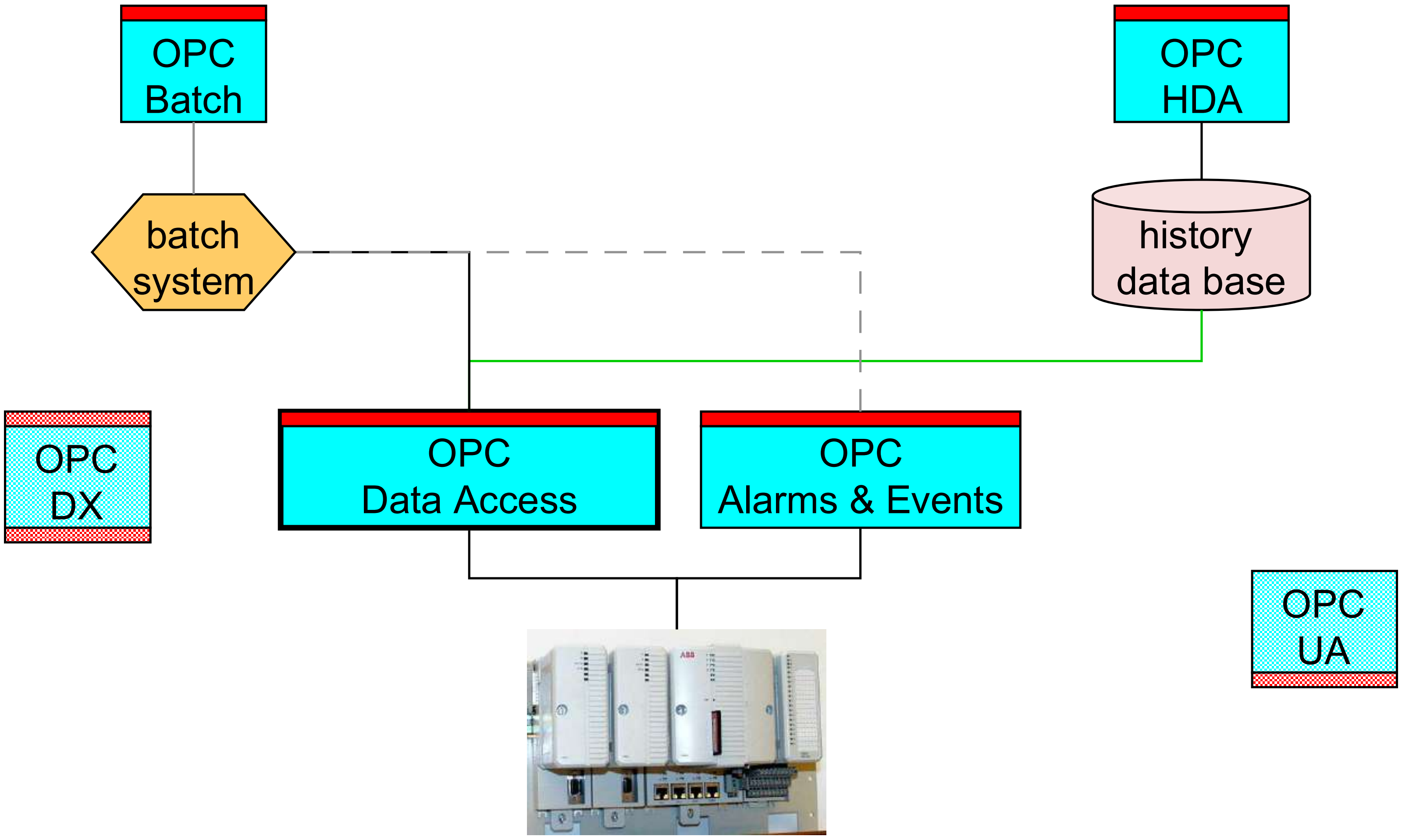
OPC is available as DLL for Automation Interface (Visual Basic,..) and Custom (C++,..)

OPC consists of three major components:

1) OPC - DA = Data Access (widespread, mature)
2) OPC - AE = Alarms and Events (not yet much used)
3) OPC - HDA = Historical Data Access (seldom used)

… and some profiles (batch,…)

The main OPC Specifications

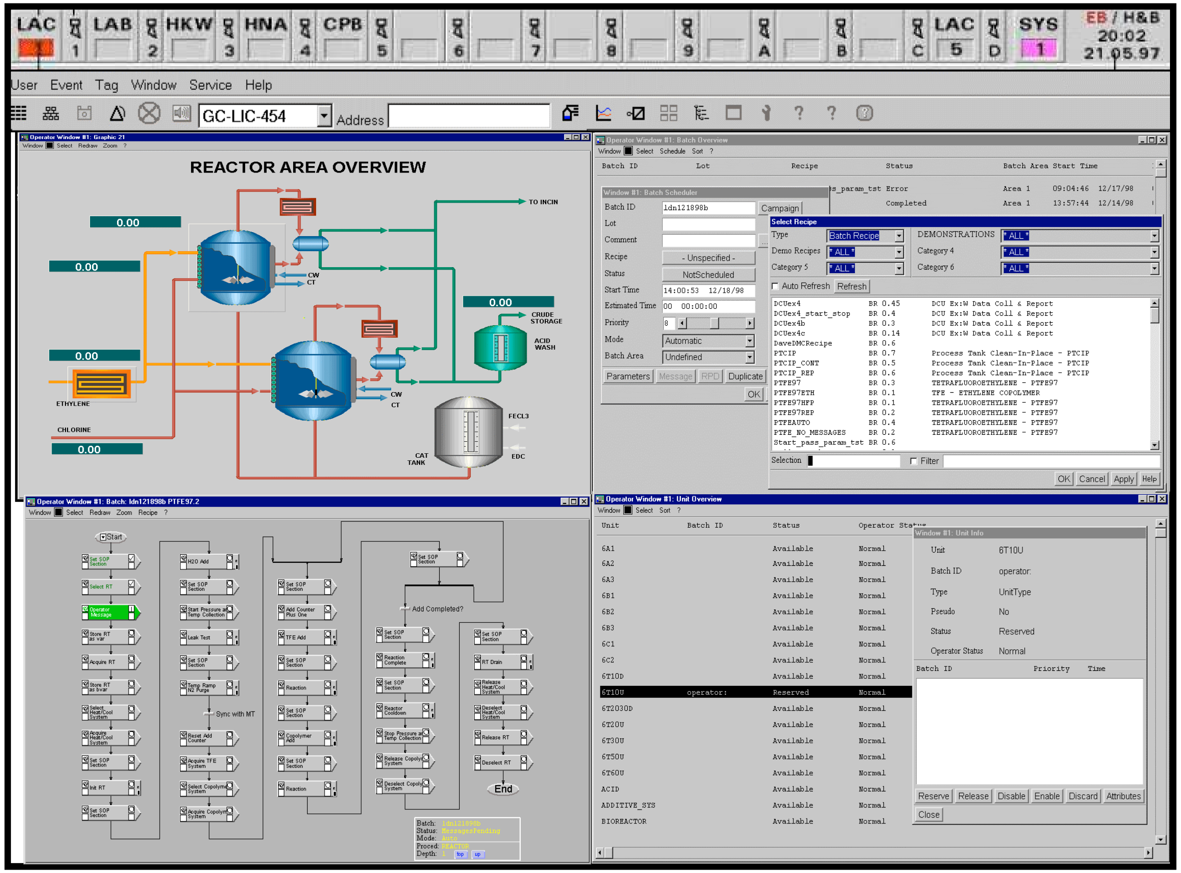


Specification 1: OPC DA for Data Access

Process variables describe the plant's <u>state</u>, they are generated by the sensors or calculated in the programmable logic controllers (PLCs).

Process variables can be sent upon a change, on demand or when a given time elapsed.

The OPC DA (Data Access) specification addresses collecting Process Variables.
The main clients of OPC DA are visualization and (soft-) control.



Batch: ldn121898b
Status: MessagesPending
Mode: Auto
Proced: REACTOR
Depth: 1

```
Batch ID          Lot           Recipe          Status          Batch Area Start Time
                                s_param_tst     Error           Area 1     09:04:46  12/17/98
                                                Completed       Area 1     13:57:44  12/14/98
```

Batch ID: ldn121898b
Lot:
Comment:
Recipe: - Unspecified -
Status: NotScheduled
Start Time: 14:00:53 12/18/98
Estimated Time: 00 00:00:00
Priority: 8
Mode: Automatic
Batch Area: Undefined

Type: Batch Recipe
DEMONSTRATIONS: * ALL *
Demo Recipes: * ALL *
Category 4: * ALL *
Category 5: * ALL *
Category 6: * ALL *
☐ Auto Refresh

```
DCUex4                BR 0.45    DCU Ex:W Data Coll & Report
DCUex4_start_stop     BR 0.4     DCU Ex:W Data Coll & Report
DCUex4b               BR 0.3     DCU Ex:W Data Coll & Report
DCUex4c               BR 0.14    DCU Ex:W Data Coll & Report
DaveDMCRecipe         BR 0.6
PTCIP                 BR 0.7     Process Tank Clean-In-Place - PTCIP
PTCIP_CONT            BR 0.5     Process Tank Clean-In-Place - PTCIP
PTCIP_REP             BR 0.6     Process Tank Clean-In-Place - PTCIP
PTFE97                BR 0.3     TETRAFLUOROETHYLENE - PTFE97
PTFE97ETH             BR 0.1     TFE - ETHYLENE COPOLYMER
PTFE97HFP             BR 0.1     TETRAFLUOROETHYLENE - PTFE97
PTFE97REP             BR 0.2     TETRAFLUOROETHYLENE - PTFE97
PTFEAUTO              BR 0.4     TETRAFLUOROETHYLENE - PTFE97
PTFE_NO_MESSAGES      BR 0.2     TETRAFLUOROETHYLENE - PTFE97
Start_pass_param_tst  BR 0.6
```

Selection:
☐ Filter

Unit	Batch ID	Status	Operator Status
6A1		Available	Normal
6A2		Available	Normal
6A3		Available	Normal
6B1		Available	Normal
6B2		Available	Normal
6B3		Available	Normal
6C1		Available	Normal
6C2		Available	Normal
6T10D		Available	Normal
6T10U	operator:	Reserved	Normal
6T2030D		Available	Normal
6T20U		Available	Normal
6T30U		Available	Normal
6T50U		Available	Normal
6T60U		Available	Normal
ACID		Available	Normal
ADDITIVE_SYS		Available	Normal
BIOREACTOR		Available	Normal

Unit: 6T10U
Batch ID: operator:
Type: UnitType
Pseudo: No
Status: Reserved
Operator Status: Normal

Batch ID	Priority	Time

Specification 2: OPC AE for Alarms and Events

Events are <u>changes</u> in the process that need to be logged, such as "production start"
Alarms are abnormal <u>states</u> in the process that require attention, such as "low oil pressure"

OPC AE (Alarms and Events) specifies how alarms and events are subscribed, under which conditions they are filtered and sent with their associated messages.
The main clients of OPC AE are the Alarms and Event loggers.

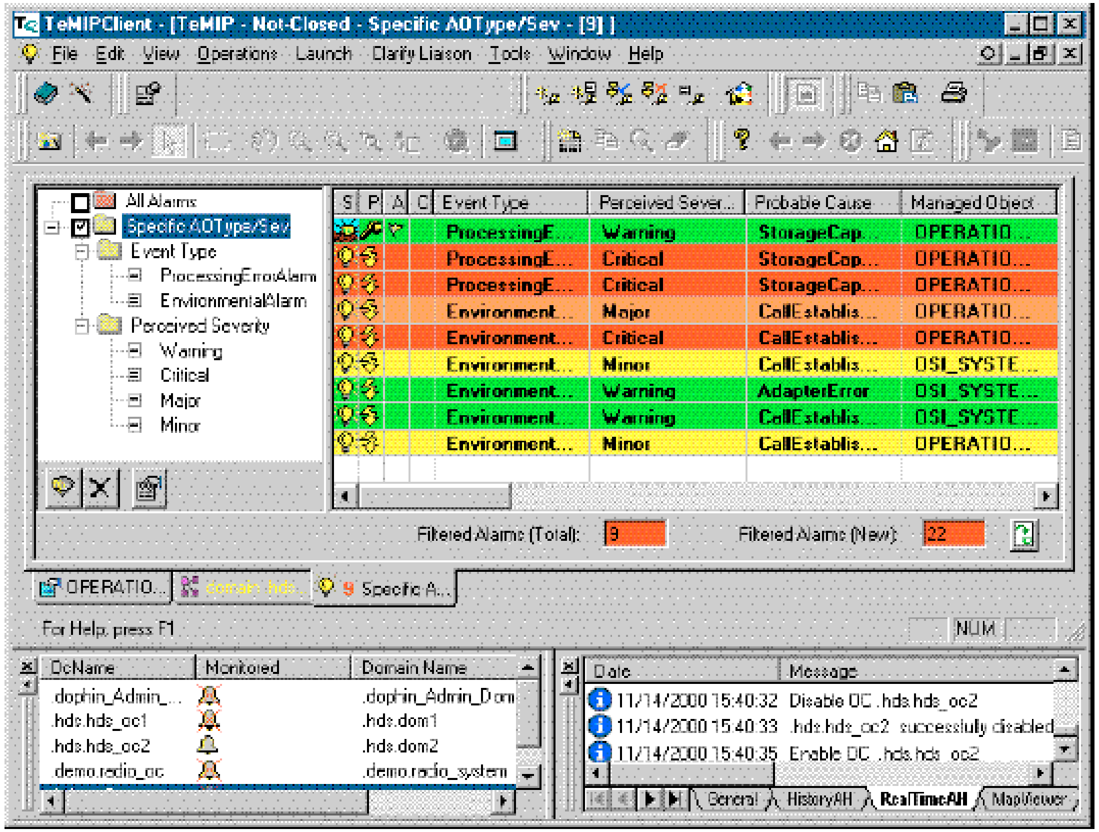
determine the exact time of change (time stamping)

categorize by priorities

log for further use

acknowledge alarms
(events are not acknowledged)

link to clear text explanation



S	P	A	C	Event Type	Perceived Sever...	Probable Cause	Managed Object
				ProcessingE...	Warning	StorageCap...	OPERATIO...
				ProcessingE...	Critical	StorageCap...	OPERATIO...
				ProcessingE...	Critical	StorageCap...	OPERATIO...
				Environment...	Major	CallEstablis...	OPERATIO...
				Environment...	Critical	CallEstablis...	OPERATIO...
				Environment...	Minor	CallEstablis...	OSI_SYSTE...
				Environment...	Warning	AdapterError	OSI_SYSTE...
				Environment...	Warning	CallEstablis...	OSI_SYSTE...
				Environment...	Minor	CallEstablis...	OPERATIO...

Filtered Alarms (Total): 9   Filtered Alarms (New): 22

OcName	Monitored	Domain Name
.dophin_Admin_...		.dophin_Admin_Dom
.hds.hds_oc1		.hds.dom1
.hds.hds_oc2		.hds.dom2
.demo.radio_oc		.demo.radio_system

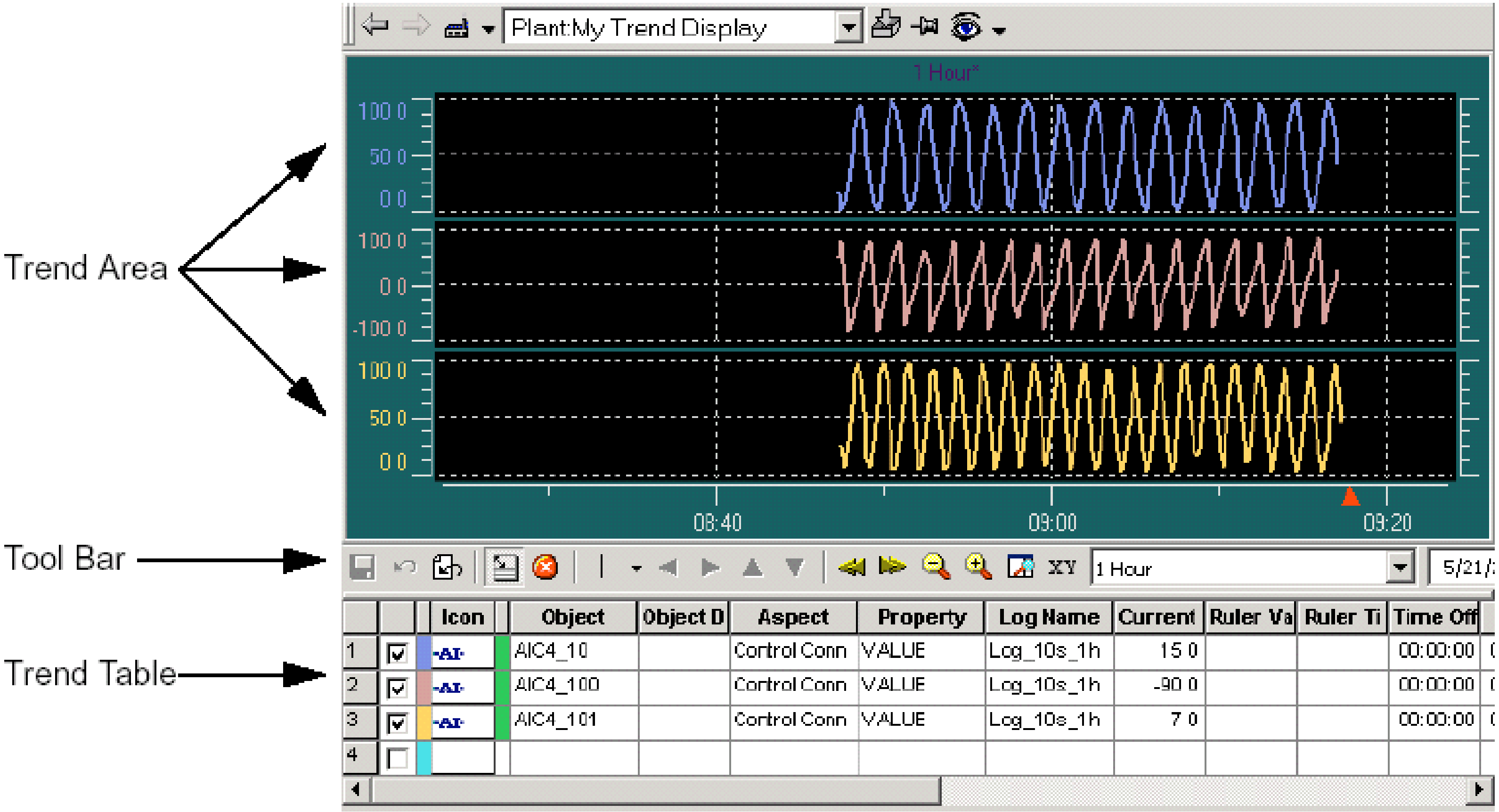
Date	Message
11/14/2000 15:40:32	Disable OC .hds.hds_oc2
11/14/2000 15:40:33	.hds.hds_oc2 successfuly disabled
11/14/2000 15:40:35	Enable OC .hds.hds_oc2

Specification 3: HDA for Historical Data Access

Historical Data are process states and events such as: process variables, operator actions, recorded alarms,... that are stored as logs in a long-term storage for later analysis.

OPC HDA (Historical Data Access) specifies how historical data are retrieved from the logs in the long-term storage, filtered and aggregated (e.g. compute averages, peaks).

The main client of OPC HDA are Trend Displays and Historians.



Plant:My Trend Display

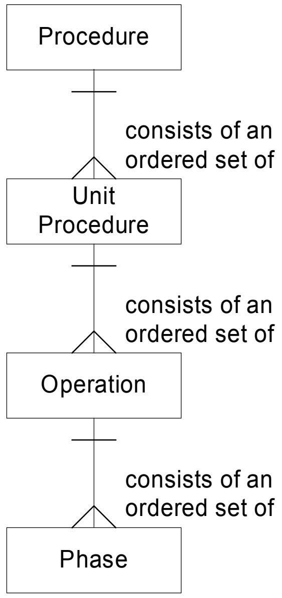
1 Hour

5/21/

		Icon	Object	Object D	Aspect	Property	Log Name	Current	Ruler Va	Ruler Ti	Time Off
1	☑	-AI-	AIC4_10		Control Conn	VALUE	Log_10s_1h	15 0			00:00:00
2	☑	-AI-	AIC4_100		Control Conn	VALUE	Log_10s_1h	-90 0			00:00:00
3	☑	-AI-	AIC4_101		Control Conn	VALUE	Log_10s_1h	7 0			00:00:00
4	□										

Specification 4: OPC Batch

based on:
IEC 61512-1 Batch Control – Part 1: Models and Terminology
(ANSI/ISA S88.01 1995)
ISA-dS88.02-2000 draft 17 of May 2000


Procedure
consists of an ordered set of
Unit Procedure
consists of an ordered set of
Operation
consists of an ordered set of
Phase

allows to access:

- equipment capabilities,
- current operating conditions,
- historical and
- recipe contents

Beyond Microsoft: OPC UA

In a move to get more independence from Microsoft and use web technology, a new specification called " Unified Architecture" (formerly. OPC XML) that uses web services for all kinds of transactions: query, read, write, subscribe,...

The classical OPC DA, AE and HDA are implemented with XML / SOAP /WSDL this allows encryption and authentication of process data.

This does not only standardize the interfaces, but also the transmitted data.

OPC as an integration tool

OPC Common

- Overview: usage and specifications
- **OPC as an integration tool**
- Clients and Servers: configuration
- Automation and Custom Interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification

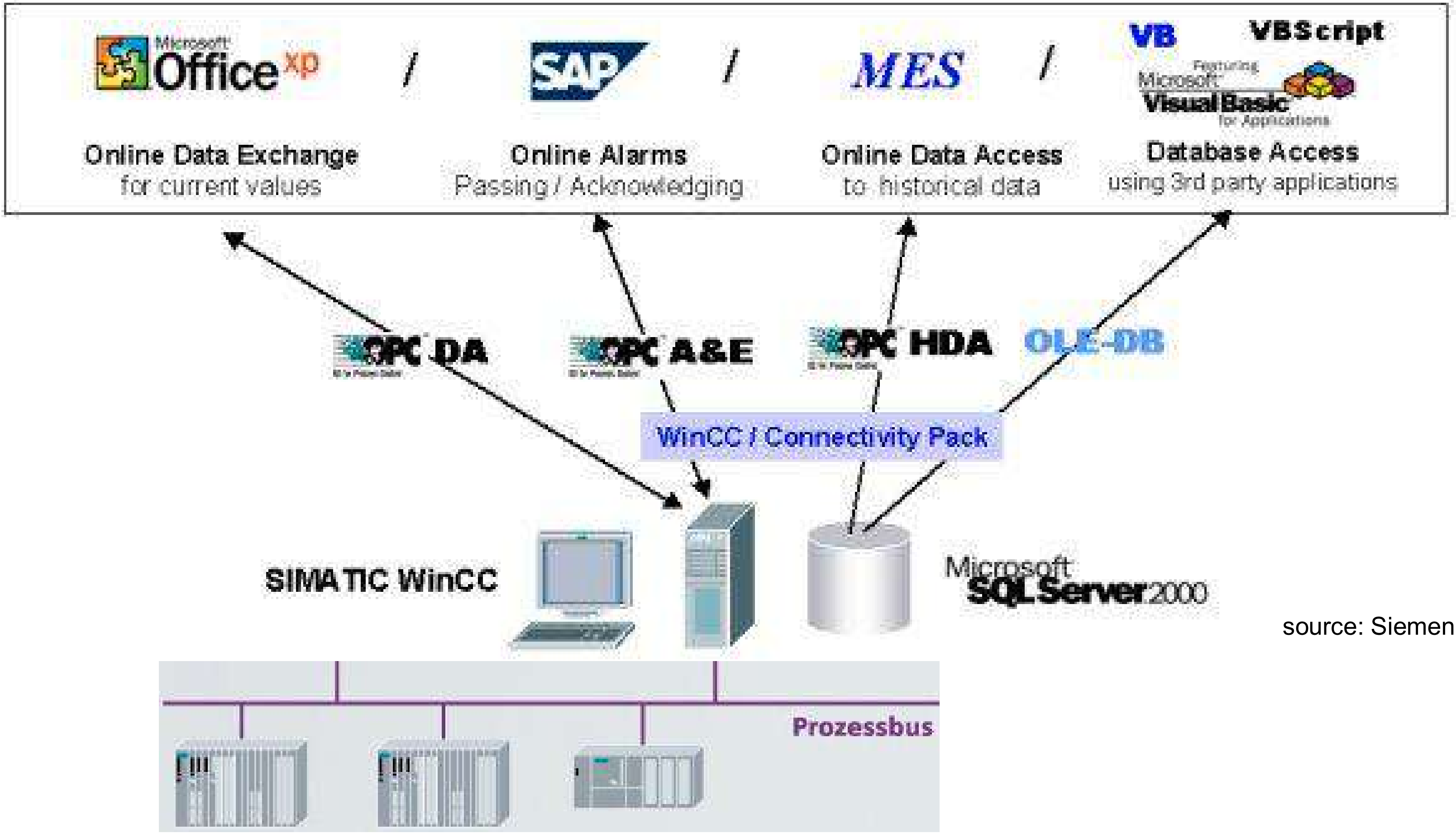
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

OPC as a hub

OPC variables is also a convenient way to exchange data between applications on the same machine. OPC data can be easily read in any Microsoft Office application



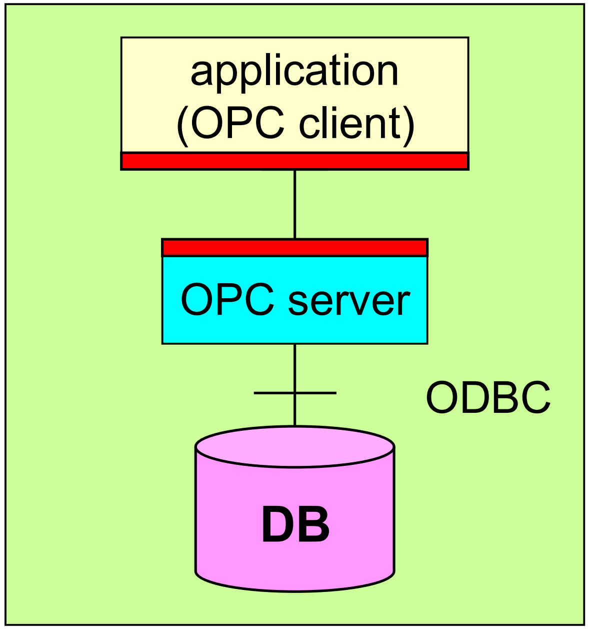
source: Siemens WinCC

OPC connection to databases

Tools such as LifeWire's allow to build an OPC DA interface to any ODBC - equipped database.

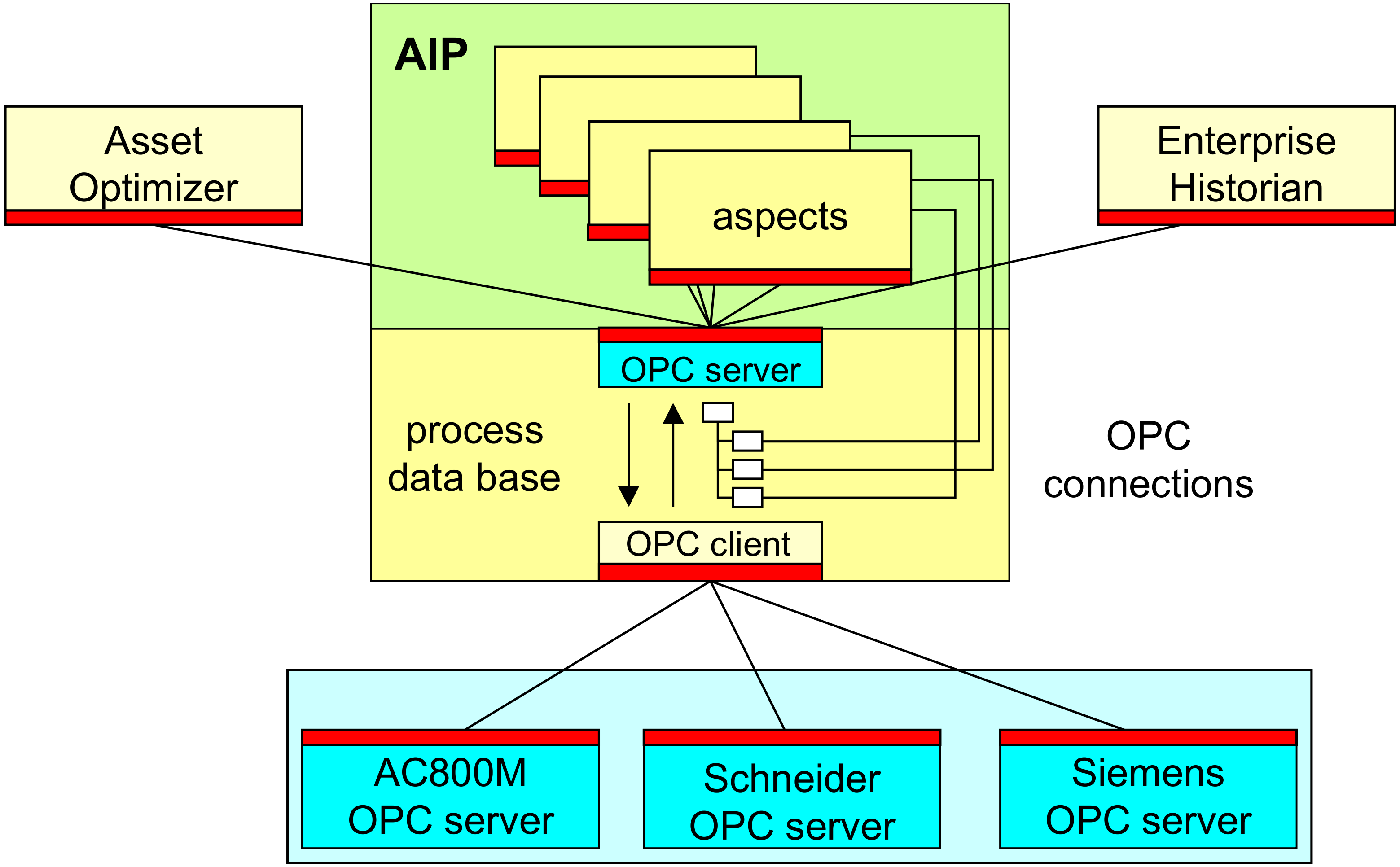
The database internal structure (exposed through queries) is reflected as a hierarchy of OPC items.

This allows to give a unified access to simple items.



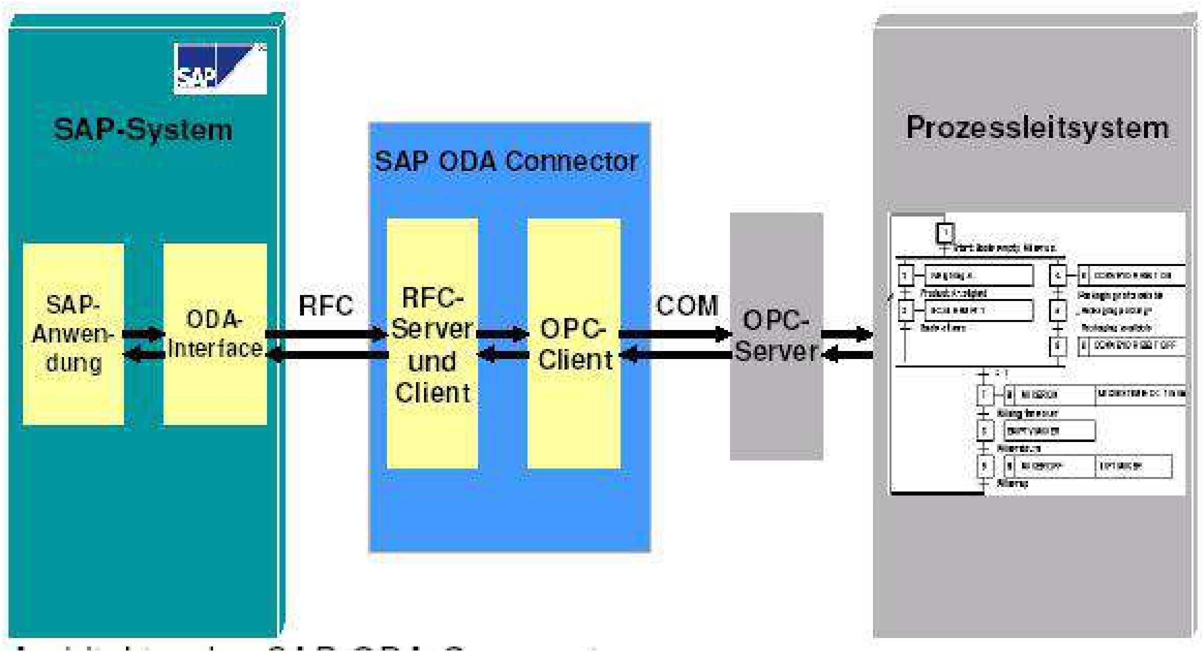
OPC for internal communication: AIP as example

ABB's Integration Platform (AIP) is at the same time an OPC server and an OPC client.
Components (aspects) within AIP expose their properties as OPC objects.
Internal (within AIP) and external communication takes place over OPC.



OPC Connection to Enterprise Resource Planning

Direct connection to SAP (BAPI) is provided by tools such as Matrikon's or Intellution's



Simulators and Explorer: which helps are available

Explorer:

Several tools are available on the market to browse OPC servers, especially:

- Matrikon OPC Explorer (no source code)
- TopServer Client (source code in VB available)

Simulator:

OPC data should be simulated before commissioning the real plant.

To this effect, commercial simulation servers allow to create, observe and change variables by hand or according to time functions (ramp, random,…).

Most PLC servers have also a simulation mode.

Freeware servers such as Matrikon have only limited number of variables

These explorers and simulators work with OPC DA, AE is yet seldom.

Client and Servers

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- Automation and Custom Interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

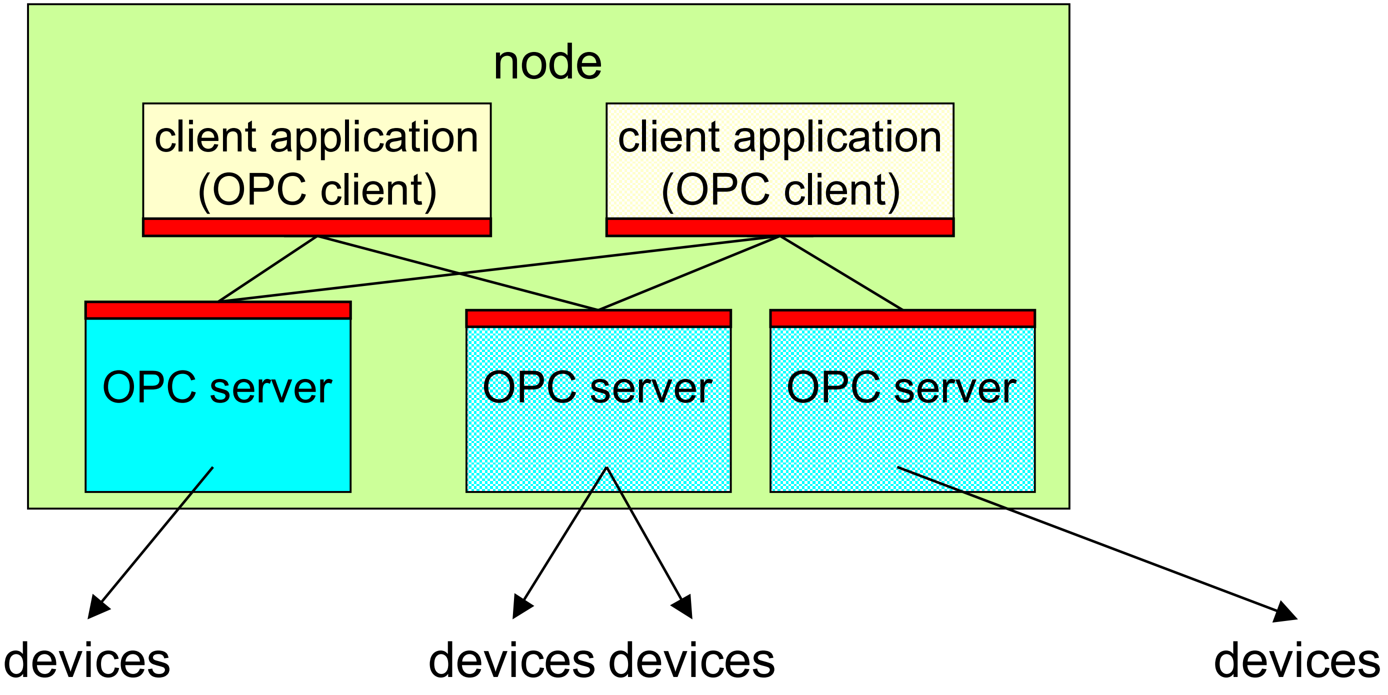
OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

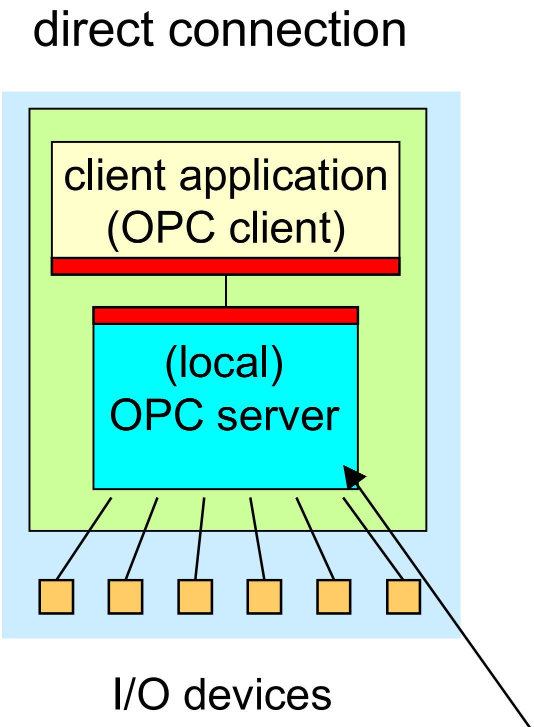
Server(s) and Client(s) in the same node



Clients and servers run as parallel processes

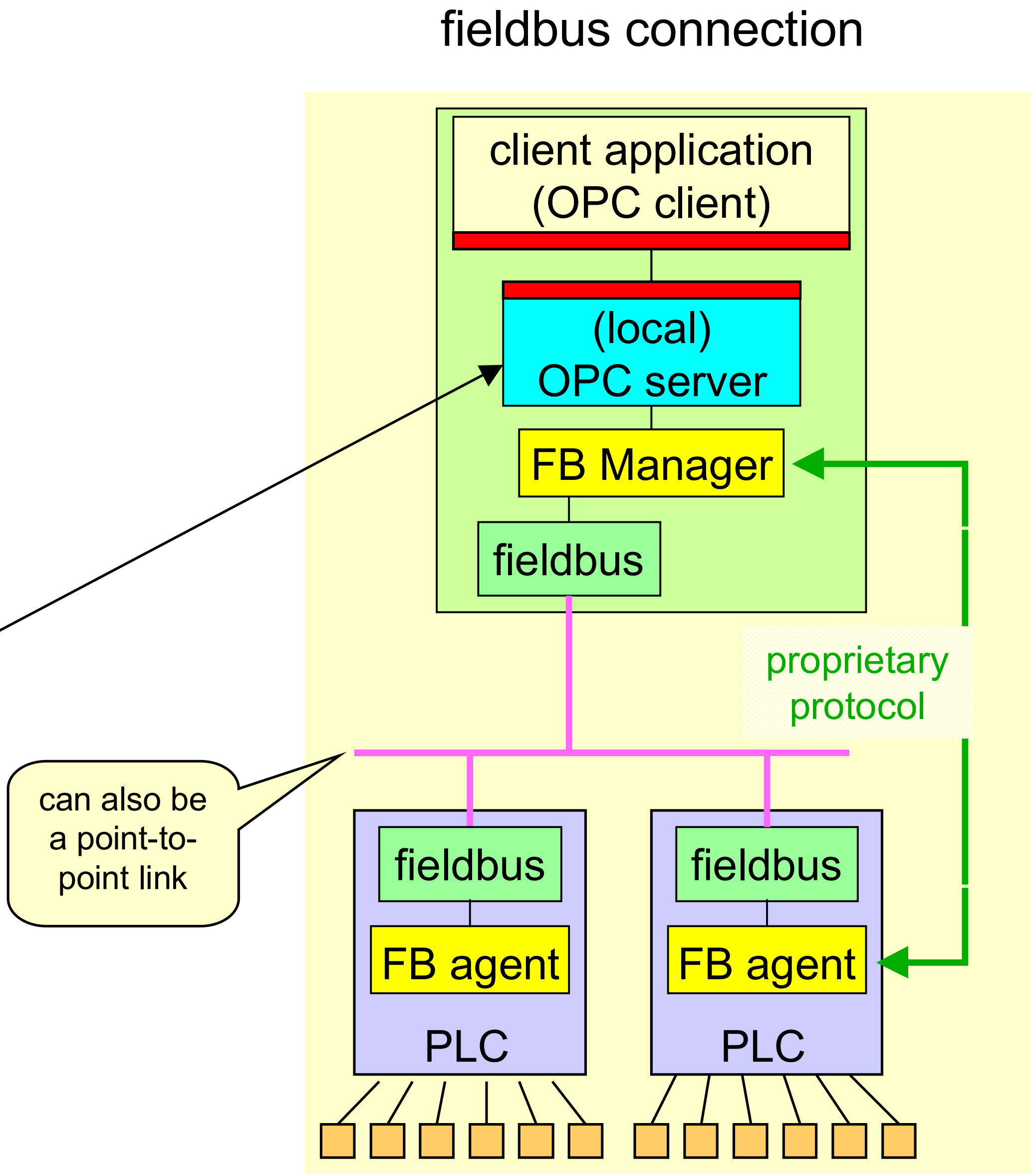
The OPC specification defines the interface between client and server in the form of objects and methods.

Direct and Fieldbus access

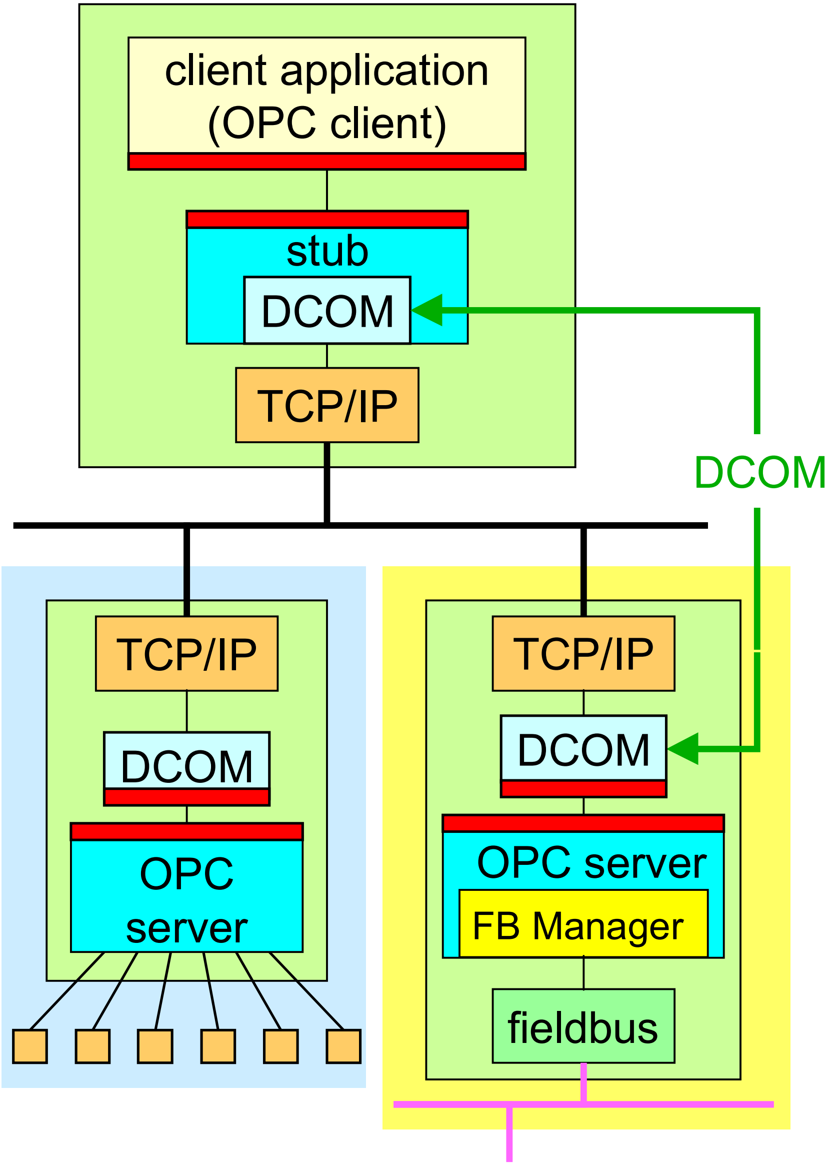


clients and servers run as parallel processes

The OPC server is running all the time, as soon as at least one client is present

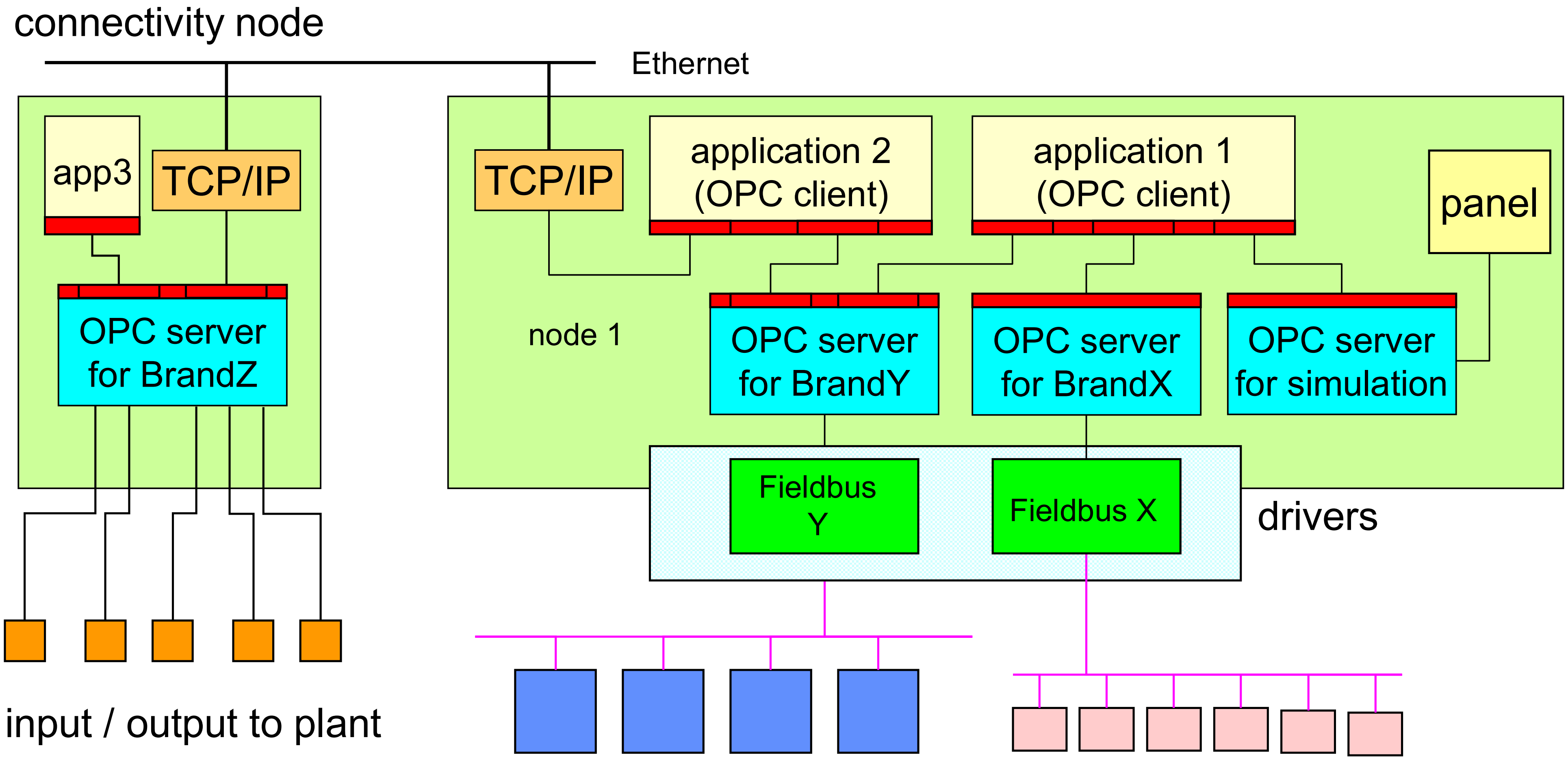


Accessing a server in another node



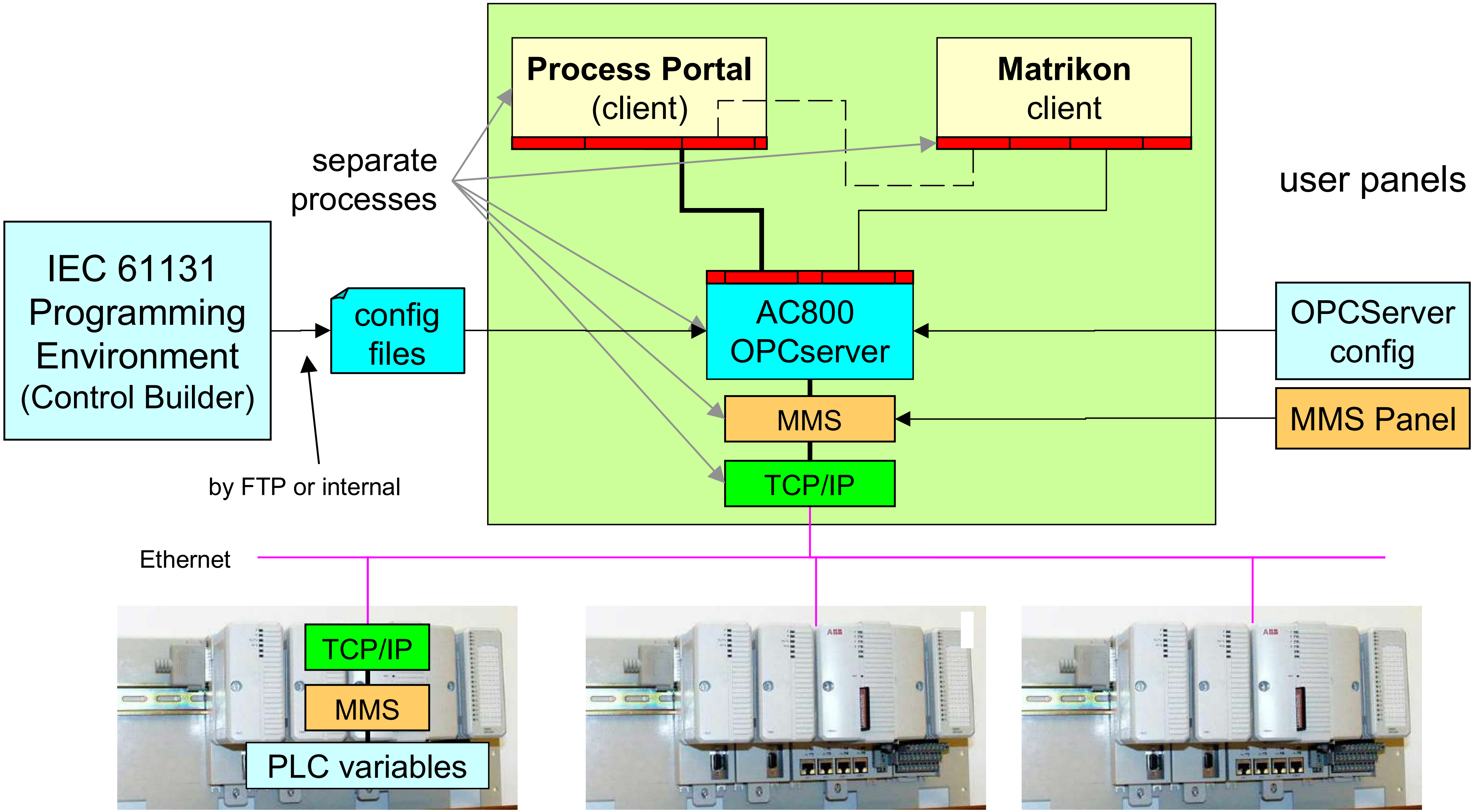
Limitation:
does not work over firewalls.
Solution:
OPC XML (see later)

Full-fledged COM/DCOM across multiple nodes



The OPC servers supports multiple clients and servers on the same, or on remote nodes.
they run as separate processes (as soon as at least one client is requesting them)

Example: ABB AC800 OPC Server



The variables are defined in the server, not in the PLC.

OPC Technology

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

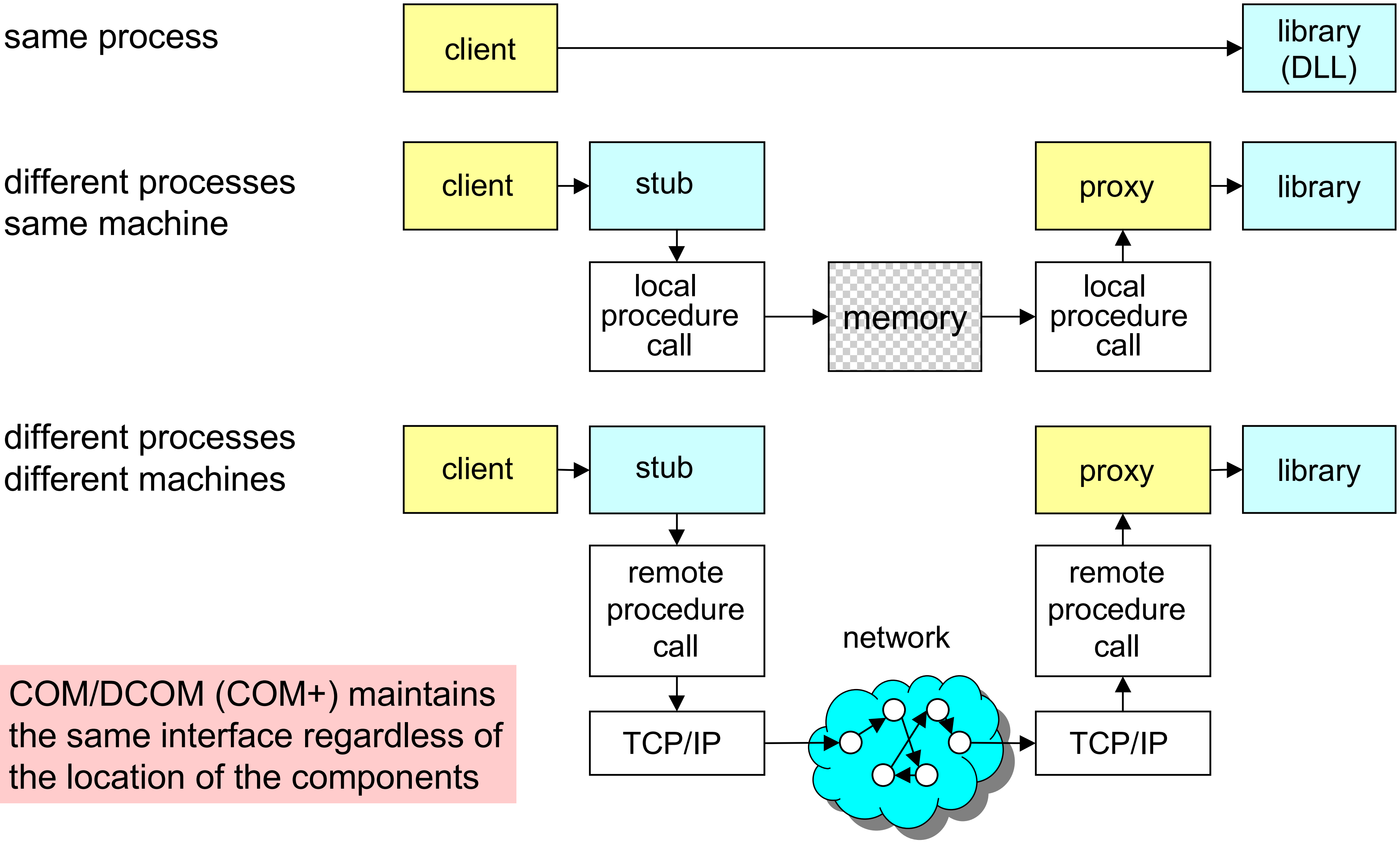
OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

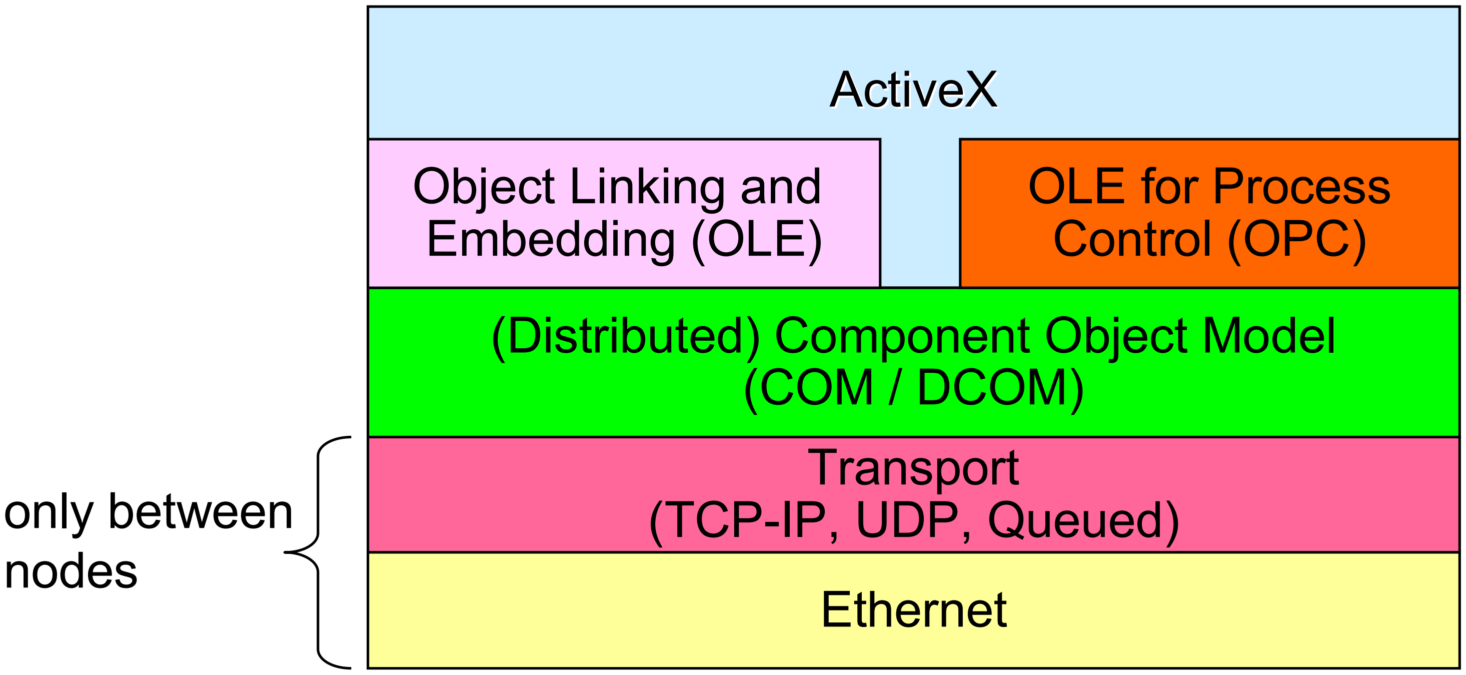
- Overview

COM/DCOM quick intro



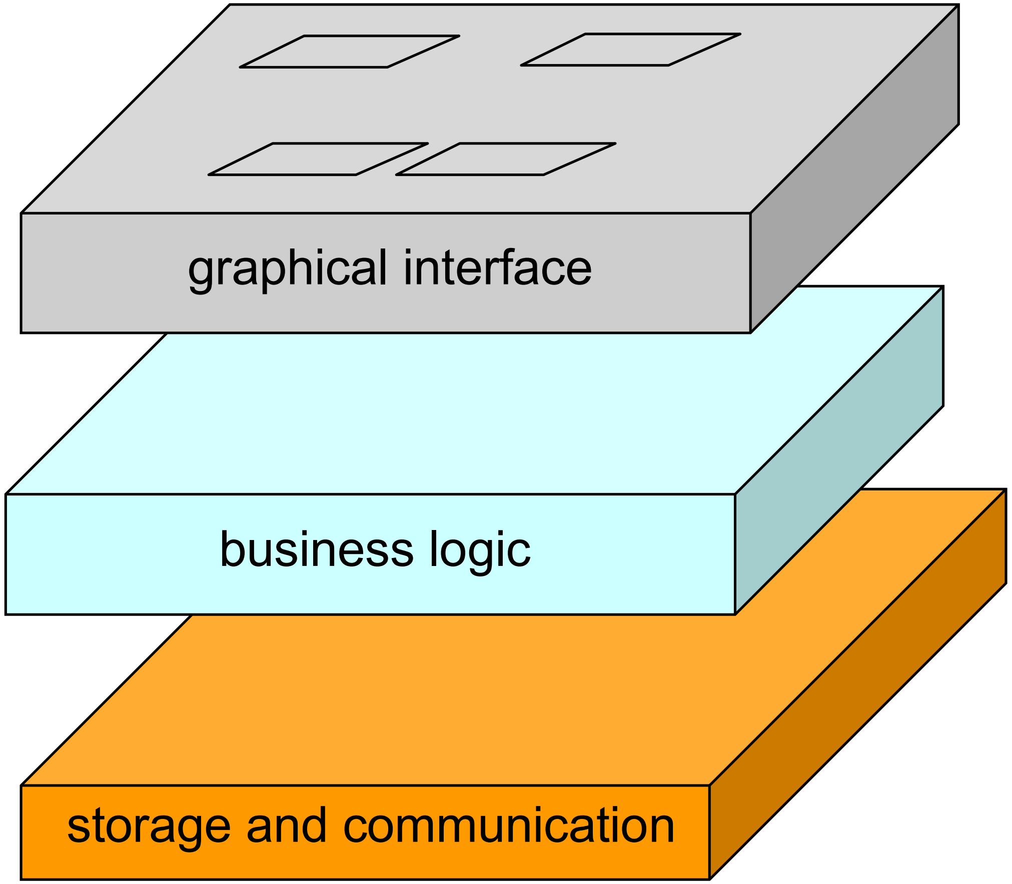
COM/DCOM (COM+) maintains the same interface regardless of the location of the components

OPC technologies

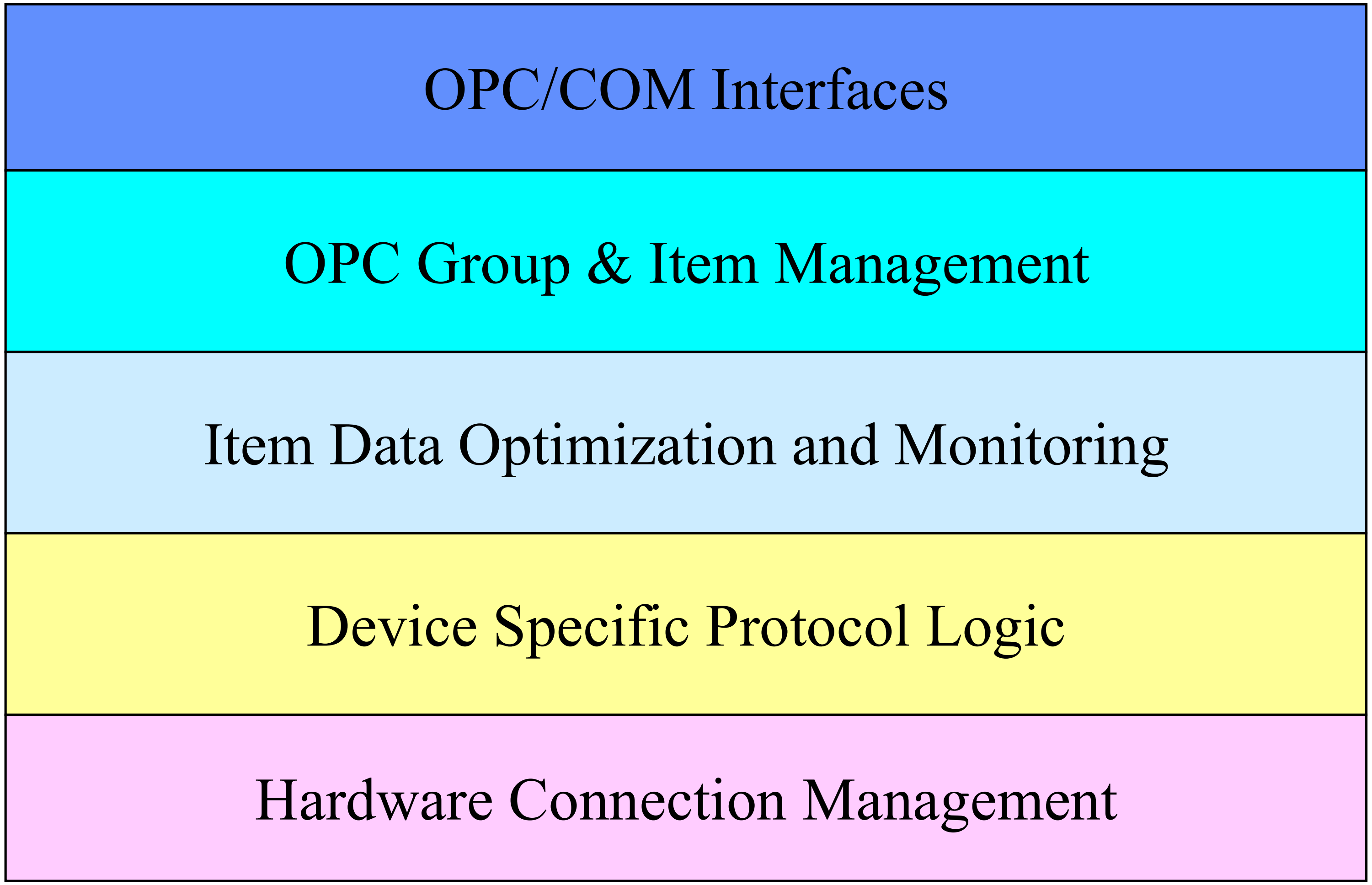

ActiveX
Object Linking and Embedding (OLE)
OLE for Process Control (OPC)
(Distributed) Component Object Model (COM / DCOM)
Transport (TCP-IP, UDP, Queued)
Ethernet
only between nodes

OPC bases on Microsoft's COM/DCOM technology (i.e. it only works on Windows platforms).
Effort to port it to other platforms (Linux) and web transport protocols (XML) are in progress.
Advantages are the direct integration into all applications, such as Excel.

Three-tiers Active-X components



Structure of an OPC server



OPC/COM Interfaces

OPC Group & Item Management

Item Data Optimization and Monitoring

Device Specific Protocol Logic

Hardware Connection Management

“Automation” vs. “Custom” interface

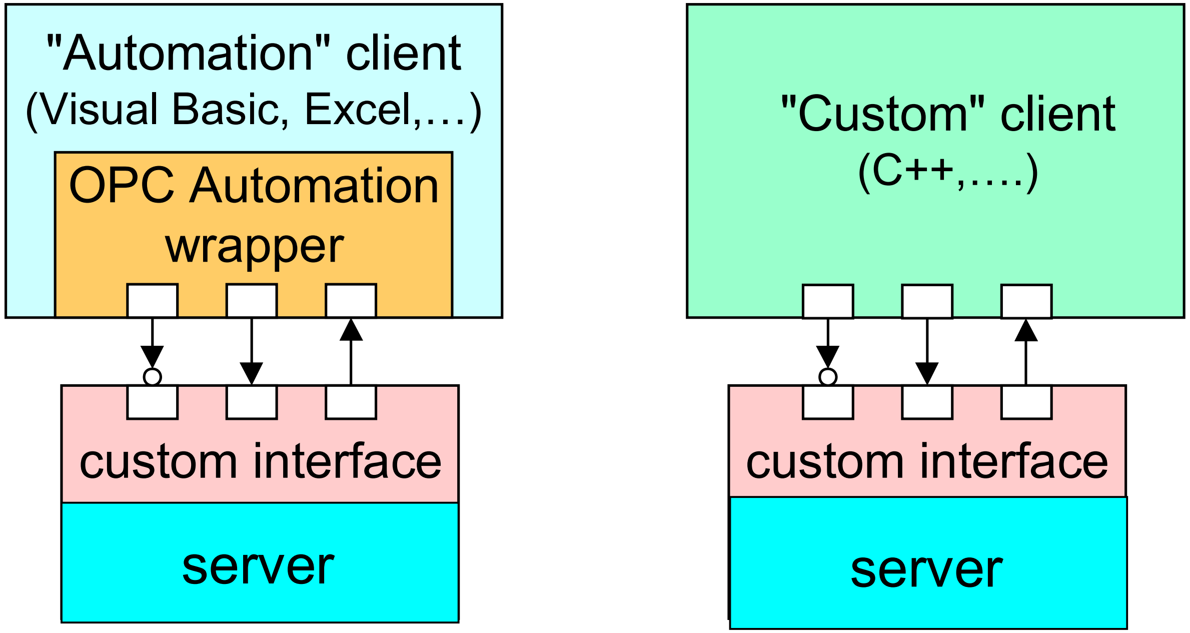
The OPC specifications define two interfaces: "custom" and "automation".

“custom” is the native C++ interface of COM.

“automation” is the interface offered in Visual Basic, used in Word, Excel,…..

The interface is defined by a Type Library (distributed by the OPC Foundation)

Functionality is roughly the same in both models, “automation” is easier to use, but "custom" gives a more extended control.



Assessment Common

What is the objective of OPC ?

On which technology does OPC rely ?

What is an OPC Server ?

Which are the main OPC specifications ?

What are the components of the OPC DA Automation Interface ?

How does an automation platform use the OPC interfaces ?

Industrial Automation
Automation Industrielle
Industrielle Automation

OPC
OLE for Process Control

4 Access to devices

4.3 OLE for Process Control (OPC)

4.3.1 Common elements

Prof. Dr. H. Kirrmann

ABB Research Centre, Baden, Switzerland

Executive Summary

OPC is a set of standard commands collected in a software library (DLL) that can be called by client applications, written in Visual Basic, C# or other Microsoft programming languages, that allow to access automation devices (PLCs) in a uniform way, independently from their built or manufacturer.

To that effect, the particularities of the automation devices are hidden by an OPC server running either on the same machine as the client program or on another machine, by using DCOM. The OPC Servers are supplied by the manufacturer of the PLC or by 3rd parties and can manage several PLCs of the same type. Several servers can run in parallel.

The OPC library allows in particular to read and write process variables, read alarms and events and acknowledge alarms, and retrieve historical data from data bases according to several criteria.

Automation platforms such as ABB's 800XA platform act as OPC clients to collect data from PLCs or databases through third-party OPC servers. Several automation platforms act themselves as an OPC server to publish their data, events and historical data.

OPC is the preferred connectivity for 78% of MES, 75% of HMI / SCADA, 68% of DCS / PLC and 53% or ERP /Enterprise system level applications (according to Arc Advisory Group, 2004)"

keep on reading even if you are not an executive....

OPC Common Overview

OPC Common

- **Overview: usage and specifications**
- OPC as an integration tool
- Clients and Servers: configuration
- Automation and Custom Interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification

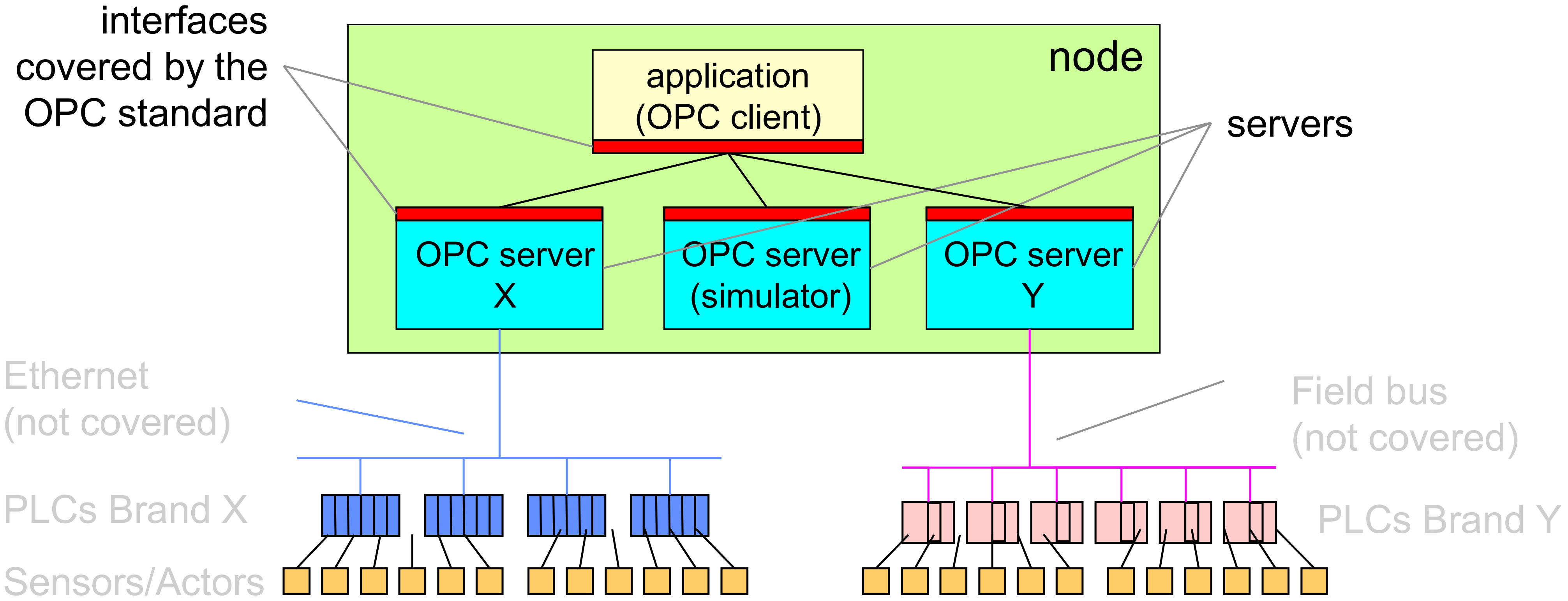
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

What is OPC ?

OPC (formerly: "OLE[1] for Process Control", now: "Open Process Control") is an industry standard set up by the *OPC Foundation* specifying the software interface (objects, methods) to a server that collects data produced by field devices and programmable logic controllers.



[1] OLE (Object Linking and Embedding) is a Microsoft technology for connecting software components. It has since been extended by the COM / DCOM technology. It corresponds to Java Beans.

Before OPC

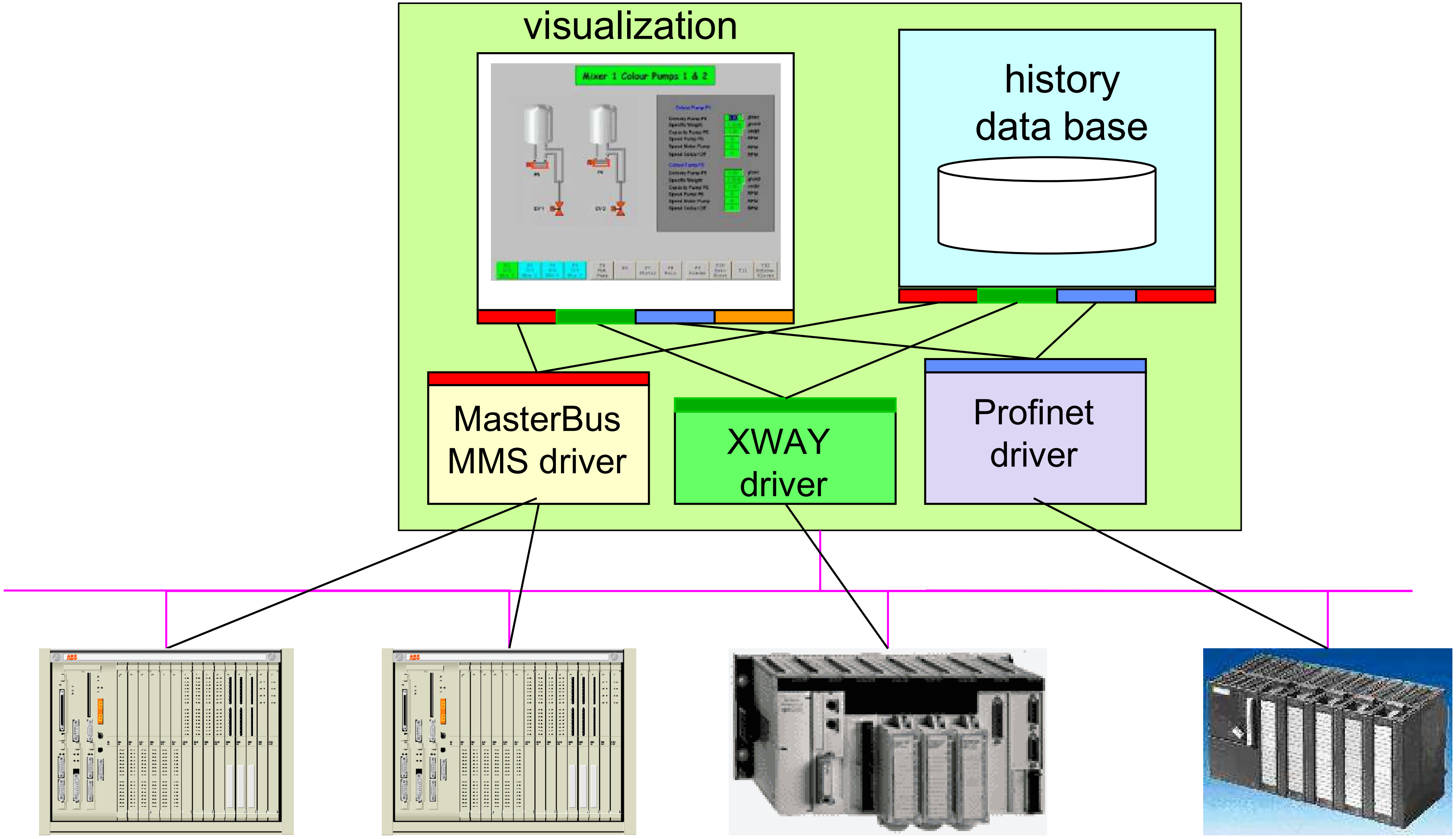


ABB PLCs Télémécanique PLCs Siemens PLCs

With OPC: ABB Operator Workplace Connection

application software is written independently from the type of controller

the drivers still exist, but the clients do not see them anymore

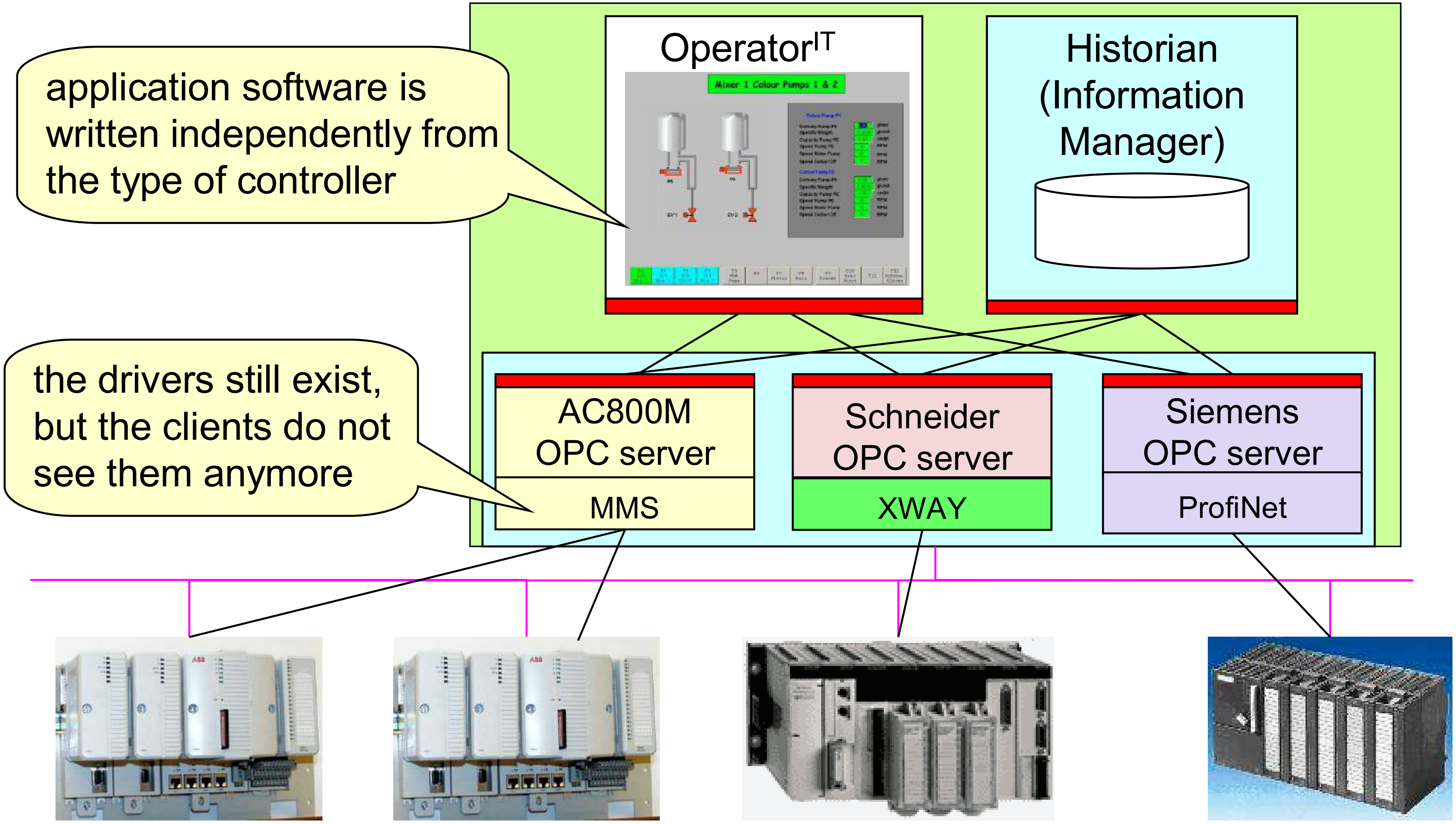


ABB AC800M

Télémécanique TSX

Siemens S7

Importance

OPC is the greatest improvement in automation since IEC 61131.

OPC is supported by the OPC foundation (http://www.opcfoundation.org/)

More than 150 vendors offer OPC servers to connect their PLCs, field bus devices, displays and visualization systems.

OPC is also used for data exchange between applications and for accessing databases

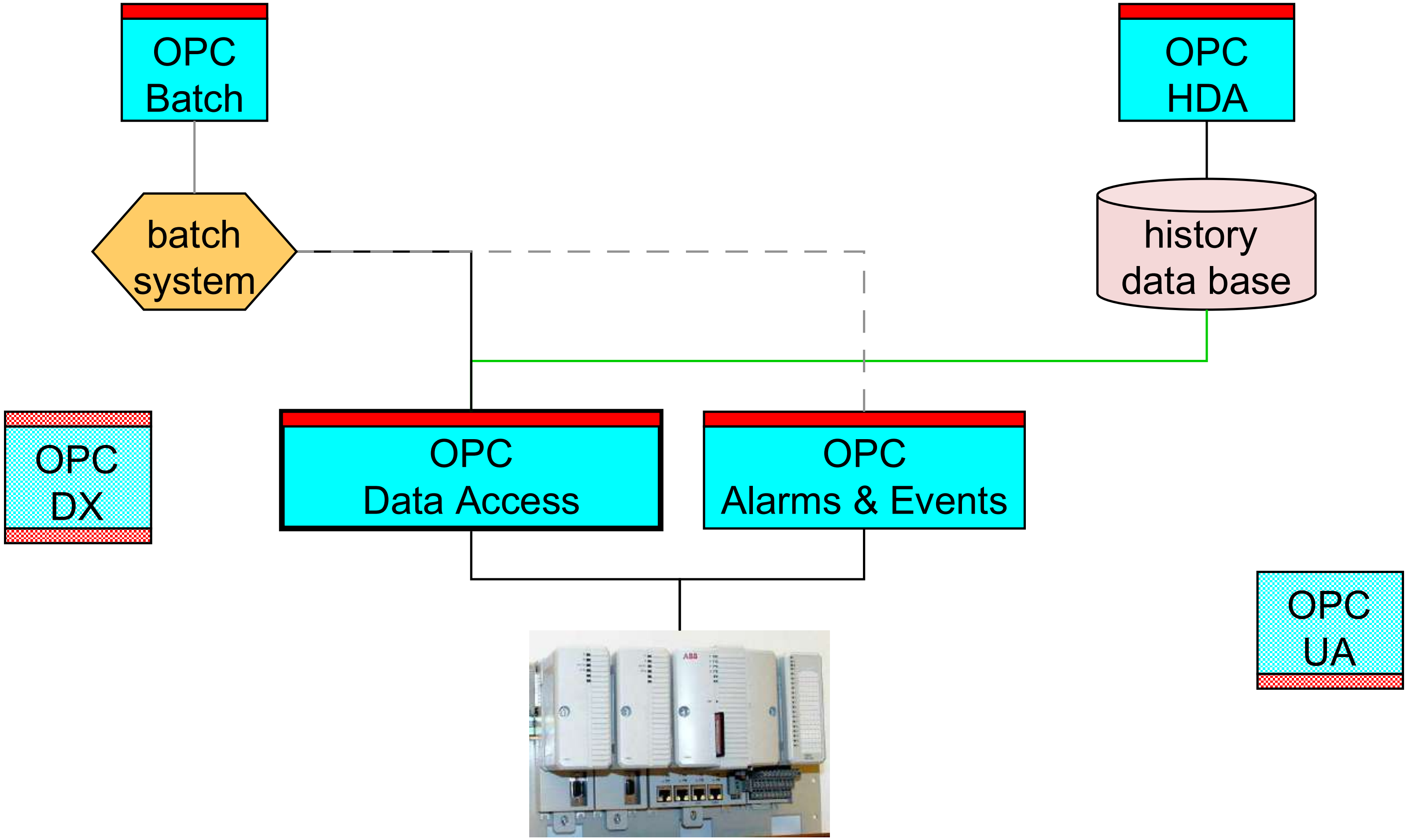
OPC is available as DLL for Automation Interface (Visual Basic,..) and Custom (C++,..)

OPC consists of three major components:

1) OPC - DA = Data Access (widespread, mature)
2) OPC - AE = Alarms and Events (not yet much used)
3) OPC - HDA = Historical Data Access (seldom used)

… and some profiles (batch,…)

The main OPC Specifications

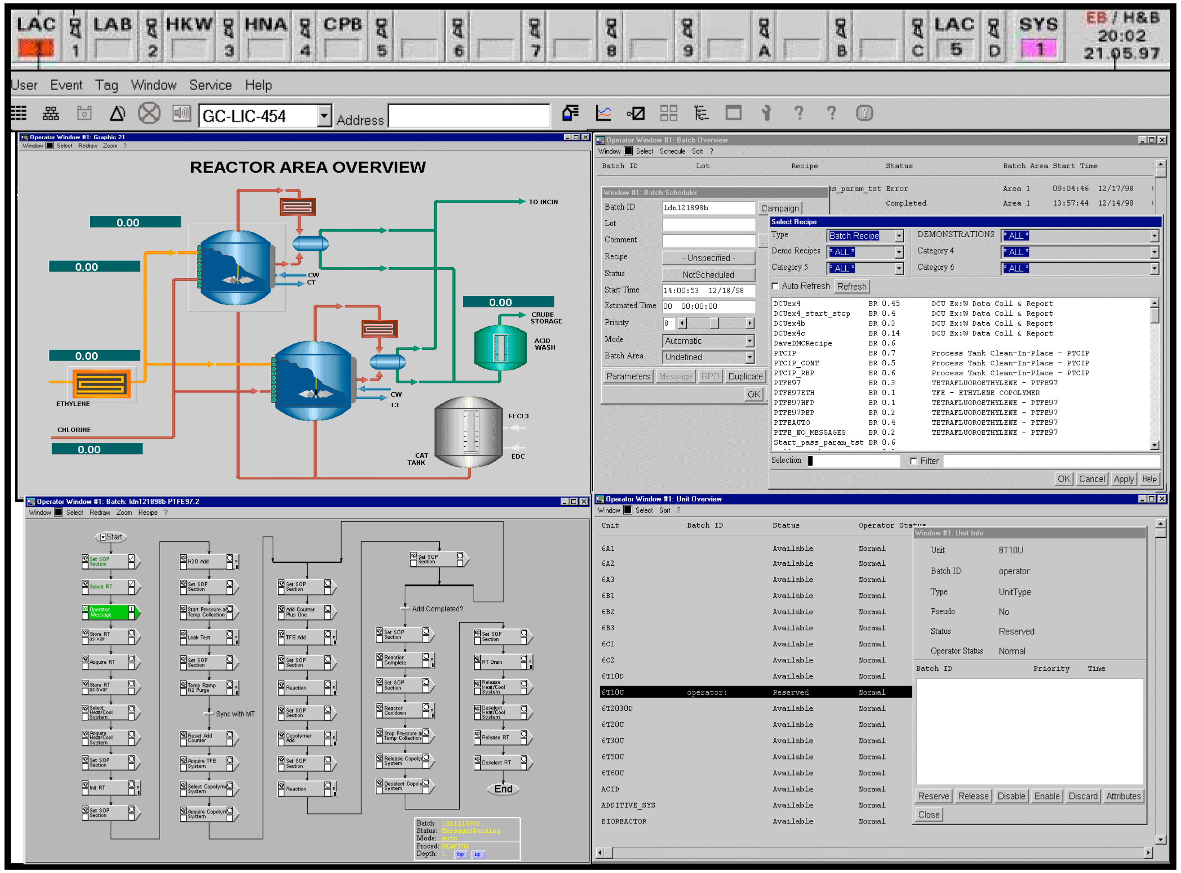


Specification 1: OPC DA for Data Access

Process variables describe the plant's state, they are generated by the sensors or calculated in the programmable logic controllers (PLCs).

Process variables can be sent upon a change, on demand or when a given time elapsed.

The OPC DA (Data Access) specification addresses collecting Process Variables.
The main clients of OPC DA are visualization and (soft-) control.



Batch: ldn121898b
Status: MessagesPending
Mode: Auto
Proced: REACTOR
Depth: 1

Operator Window #1: Batch Overview

Window Select Schedule Sort ?

Batch ID	Lot	Recipe	Status	Batch Area	Start Time
		s_param_tst	Error	Area 1	09:04:46 12/17/98
			Completed	Area 1	13:57:44 12/14/98

Window #1: Batch Scheduler

Batch ID	ldn121898b
Lot	
Comment	
Recipe	- Unspecified -
Status	NotScheduled
Start Time	14:00:53 12/18/98
Estimated Time	00 00:00:00
Priority	8
Mode	Automatic
Batch Area	Undefined

Campaign

Parameters Message RPD Duplicate

OK

Select Recipe

Type	Batch Recipe	DEMONSTRATIONS	* ALL *
Demo Recipes	* ALL *	Category 4	* ALL *
Category 5	* ALL *	Category 6	* ALL *

☐ Auto Refresh Refresh

```
DCUex4              BR 0.45      DCU Ex:W Data Coll & Report
DCUex4_start_stop   BR 0.4       DCU Ex:W Data Coll & Report
DCUex4b             BR 0.3       DCU Ex:W Data Coll & Report
DCUex4c             BR 0.14      DCU Ex:W Data Coll & Report
DaveDMCRecipe       BR 0.6
PTCIP               BR 0.7       Process Tank Clean-In-Place - PTCIP
PTCIP_CONT          BR 0.5       Process Tank Clean-In-Place - PTCIP
PTCIP_REP           BR 0.6       Process Tank Clean-In-Place - PTCIP
PTFE97              BR 0.3       TETRAFLUOROETHYLENE - PTFE97
PTFE97ETH           BR 0.1       TFE - ETHYLENE COPOLYMER
PTFE97HFP           BR 0.1       TETRAFLUOROETHYLENE - PTFE97
PTFE97REP           BR 0.2       TETRAFLUOROETHYLENE - PTFE97
PTFEAUTO            BR 0.4       TETRAFLUOROETHYLENE - PTFE97
PTFE_NO_MESSAGES    BR 0.2       TETRAFLUOROETHYLENE - PTFE97
Start_pass_param_tst BR 0.6
```

Selection ☐ Filter

OK Cancel Apply Help

Operator Window #1: Unit Overview

Window Select Sort ?

Unit	Batch ID	Status	Operator Status
6A1		Available	Normal
6A2		Available	Normal
6A3		Available	Normal
6B1		Available	Normal
6B2		Available	Normal
6B3		Available	Normal
6C1		Available	Normal
6C2		Available	Normal
6T10D		Available	Normal
6T10U	operator:	Reserved	Normal
6T2030D		Available	Normal
6T20U		Available	Normal
6T30U		Available	Normal
6T50U		Available	Normal
6T60U		Available	Normal
ACID		Available	Normal
ADDITIVE_SYS		Available	Normal
BIOREACTOR		Available	Normal

Window #1: Unit Info

Unit	6T10U
Batch ID	operator:
Type	UnitType
Pseudo	No
Status	Reserved
Operator Status	Normal

Batch ID	Priority	Time

Reserve Release Disable Enable Discard Attributes

Close

Specification 2: OPC AE for Alarms and Events

Events are <u>changes</u> in the process that need to be logged, such as "production start"
Alarms are abnormal <u>states</u> in the process that require attention, such as "low oil pressure"

OPC AE (Alarms and Events) specifies how alarms and events are subscribed, under which conditions they are filtered and sent with their associated messages.
The main clients of OPC AE are the Alarms and Event loggers.

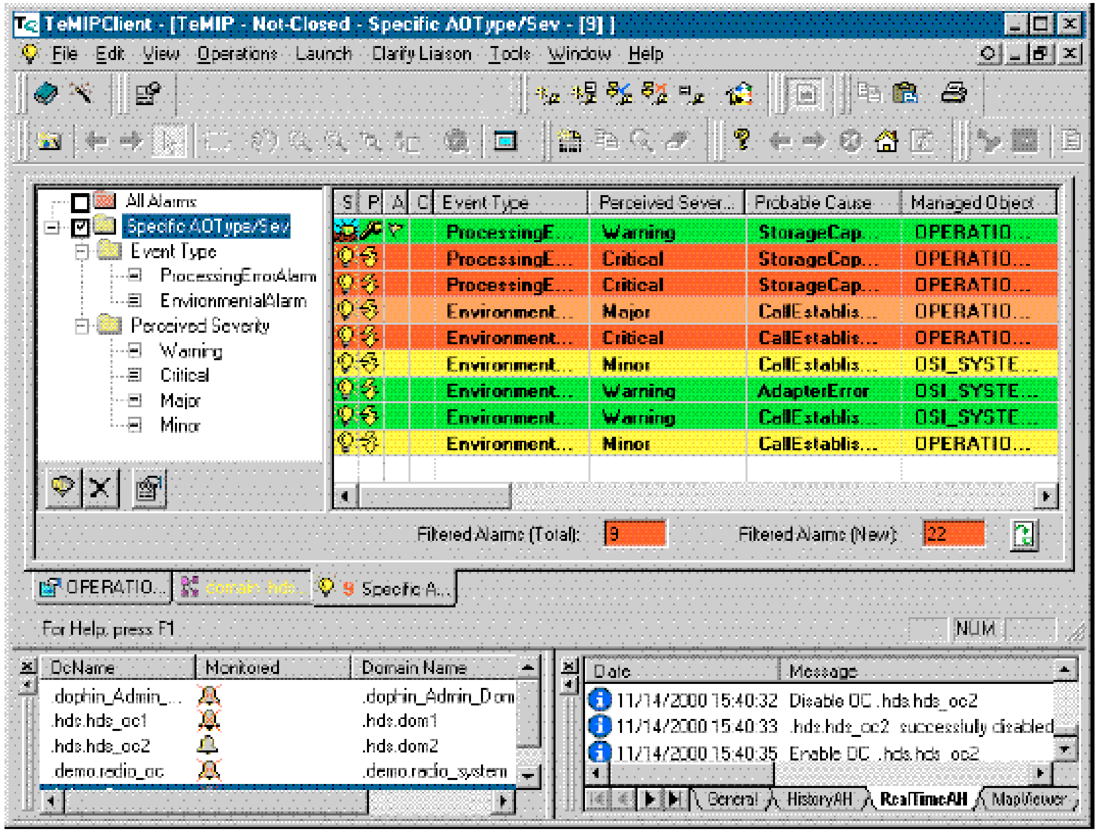
determine the exact time of change (time stamping)

categorize by priorities

log for further use

acknowledge alarms
(events are not acknowledged)

link to clear text explanation



S	P	A	C	Event Type	Perceived Sever...	Probable Cause	Managed Object
				ProcessingE...	Warning	StorageCap...	OPERATIO...
				ProcessingE...	Critical	StorageCap...	OPERATIO...
				ProcessingE...	Critical	StorageCap...	OPERATIO...
				Environment...	Major	CallEstablis...	OPERATIO...
				Environment...	Critical	CallEstablis...	OPERATIO...
				Environment...	Minor	CallEstablis...	OSI_SYSTE...
				Environment...	Warning	AdapterError	OSI_SYSTE...
				Environment...	Warning	CallEstablis...	OSI_SYSTE...
				Environment...	Minor	CallEstablis...	OPERATIO...

Filtered Alarms (Total): 9 Filtered Alarms (New): 22

DcName	Monitored	Domain Name
.dophin_Admin_...		.dophin_Admin_Dom
.hds.hds_oc1		.hds.dom1
.hds.hds_oc2		.hds.dom2
.demo.radio_oc		.demo.radio_system

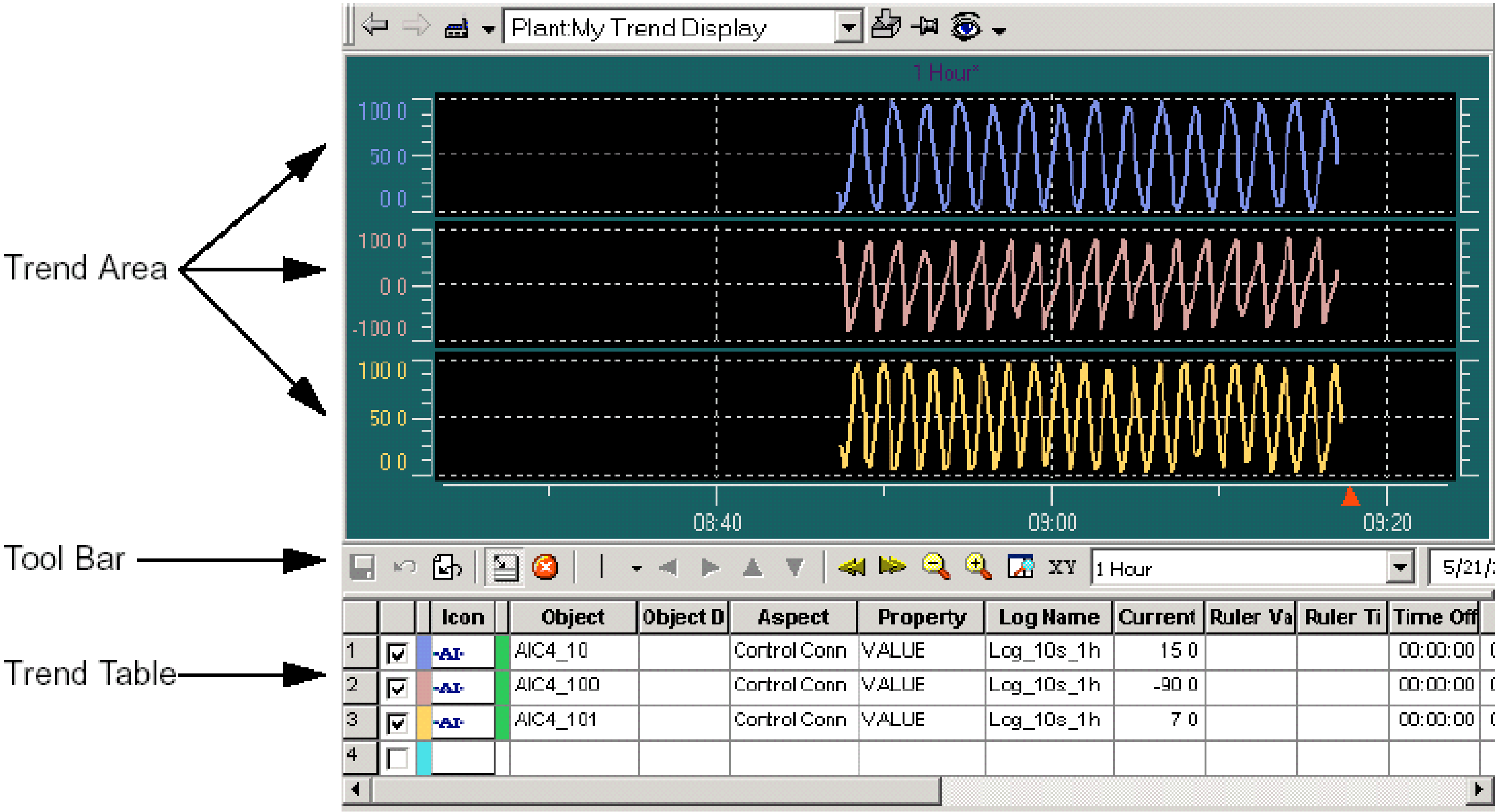
Date	Message
11/14/2000 15:40:32	Disable OC .hds.hds_oc2
11/14/2000 15:40:33	.hds.hds_oc2 successfuly disabled
11/14/2000 15:40:35	Enable OC .hds.hds_oc2

Specification 3: HDA for Historical Data Access

Historical Data are process states and events such as: process variables, operator actions, recorded alarms,... that are stored as logs in a long-term storage for later analysis.

OPC HDA (Historical Data Access) specifies how historical data are retrieved from the logs in the long-term storage, filtered and aggregated (e.g. compute averages, peaks).

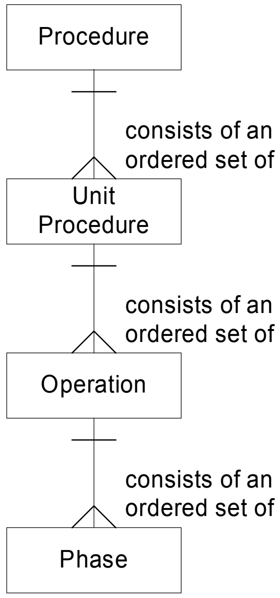
The main client of OPC HDA are Trend Displays and Historians.



		Icon	Object	Object D	Aspect	Property	Log Name	Current	Ruler Va	Ruler Ti	Time Off
1	☑	-AI-	AIC4_10		Control Conn	VALUE	Log_10s_1h	15 0			00:00:00
2	☑	-AI-	AIC4_100		Control Conn	VALUE	Log_10s_1h	-90 0			00:00:00
3	☑	-AI-	AIC4_101		Control Conn	VALUE	Log_10s_1h	7 0			00:00:00
4	□										

Specification 4: OPC Batch

based on:
IEC 61512-1 Batch Control – Part 1: Models and Terminology
(ANSI/ISA S88.01 1995)
ISA-dS88.02-2000 draft 17 of May 2000


Procedure
consists of an ordered set of
Unit Procedure
consists of an ordered set of
Operation
consists of an ordered set of
Phase

allows to access:

- equipment capabilities,
- current operating conditions,
- historical and
- recipe contents

Beyond Microsoft: OPC UA

In a move to get more independence from Microsoft and use web technology,
a new specification called " Unified Architecture" (formerly. OPC XML) that uses
web services for all kinds of transactions: query, read, write, subscribe,...

The classical OPC DA, AE and HDA are implemented with XML / SOAP /WSDL
this allows encryption and authentication of process data.

This does not only standardize the interfaces, but also the transmitted data.

OPC as an integration tool

OPC Common

- Overview: usage and specifications
- **OPC as an integration tool**
- Clients and Servers: configuration
- Automation and Custom Interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification

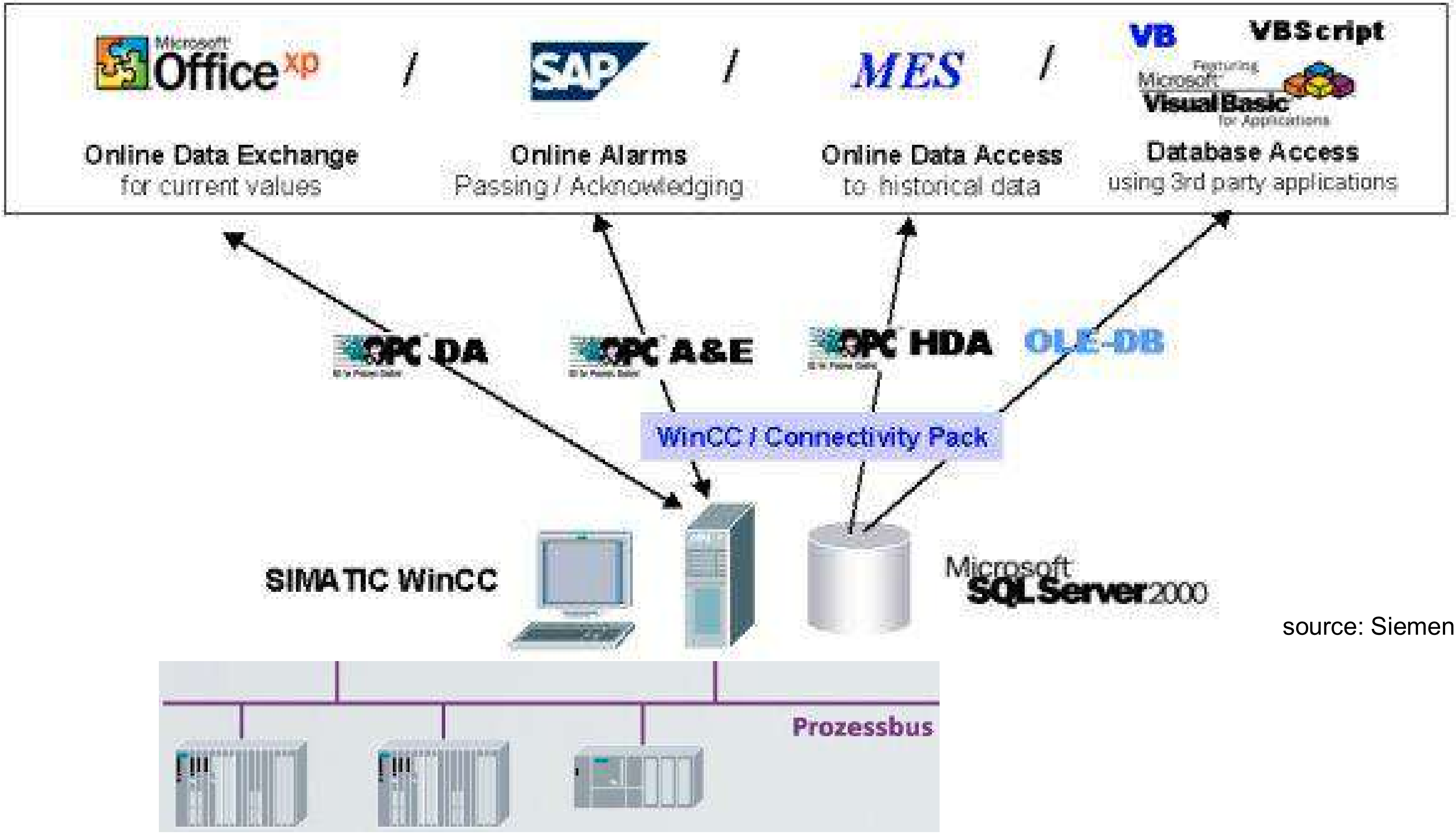
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

OPC as a hub

OPC variables is also a convenient way to exchange data between applications on the same machine. OPC data can be easily read in any Microsoft Office application



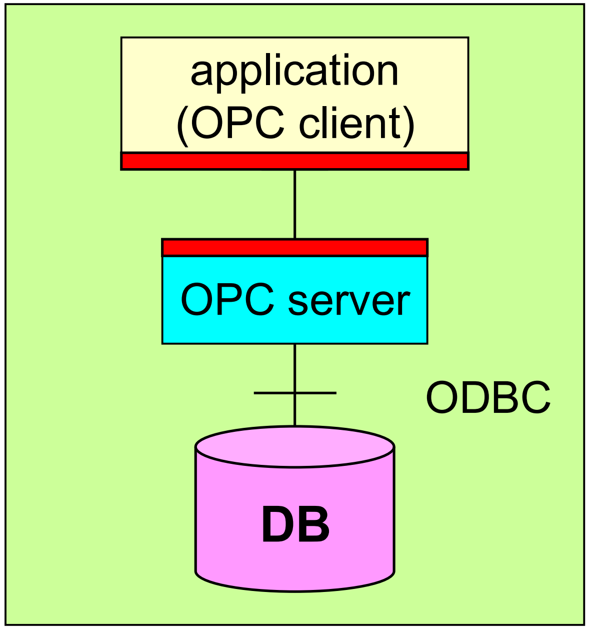
source: Siemens WinCC

OPC connection to databases

Tools such as LifeWire’s allow to build an OPC DA interface to any ODBC - equipped database.

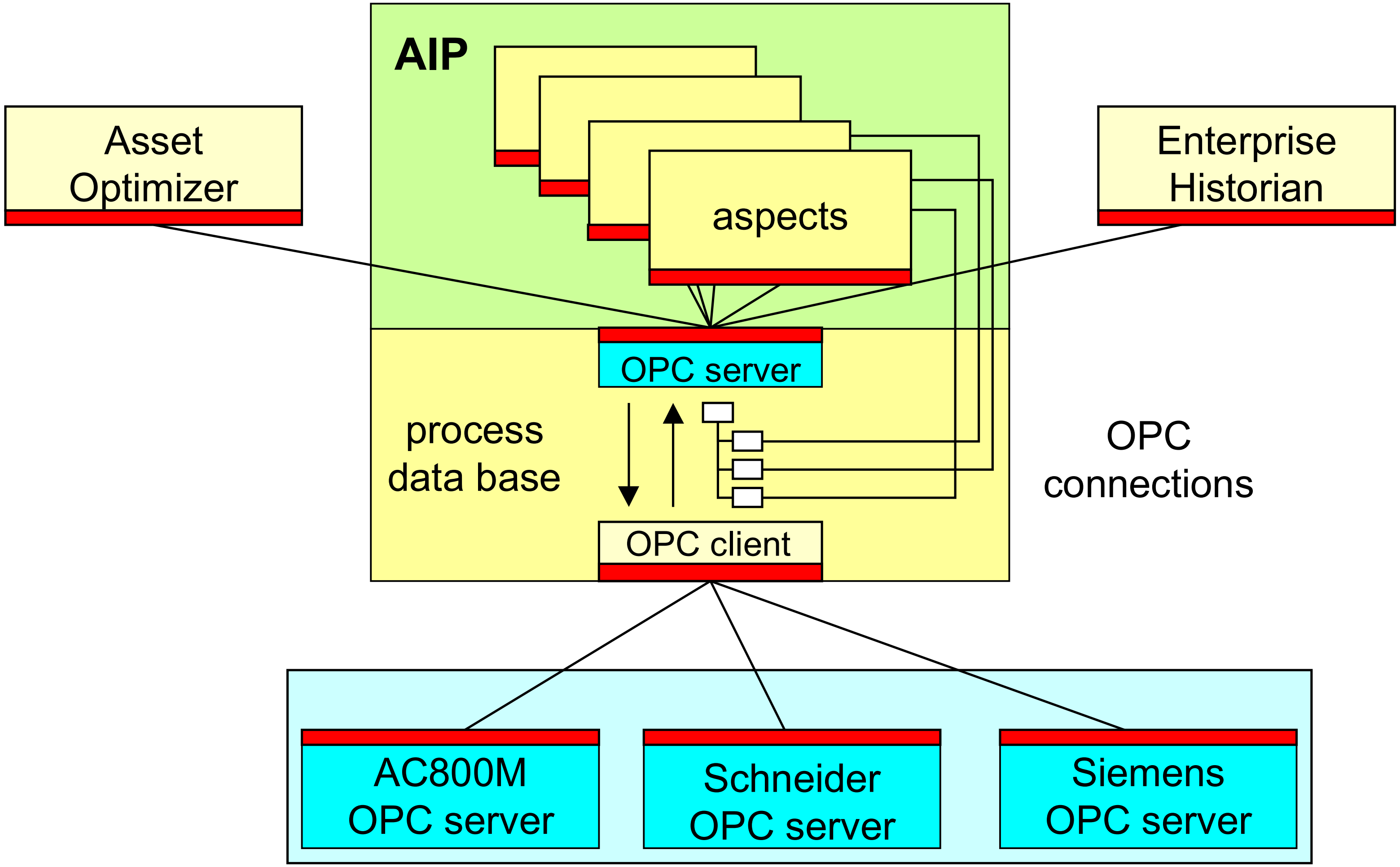
The database internal structure (exposed through queries) is reflected as a hierarchy of OPC items.

This allows to give a unified access to simple items.



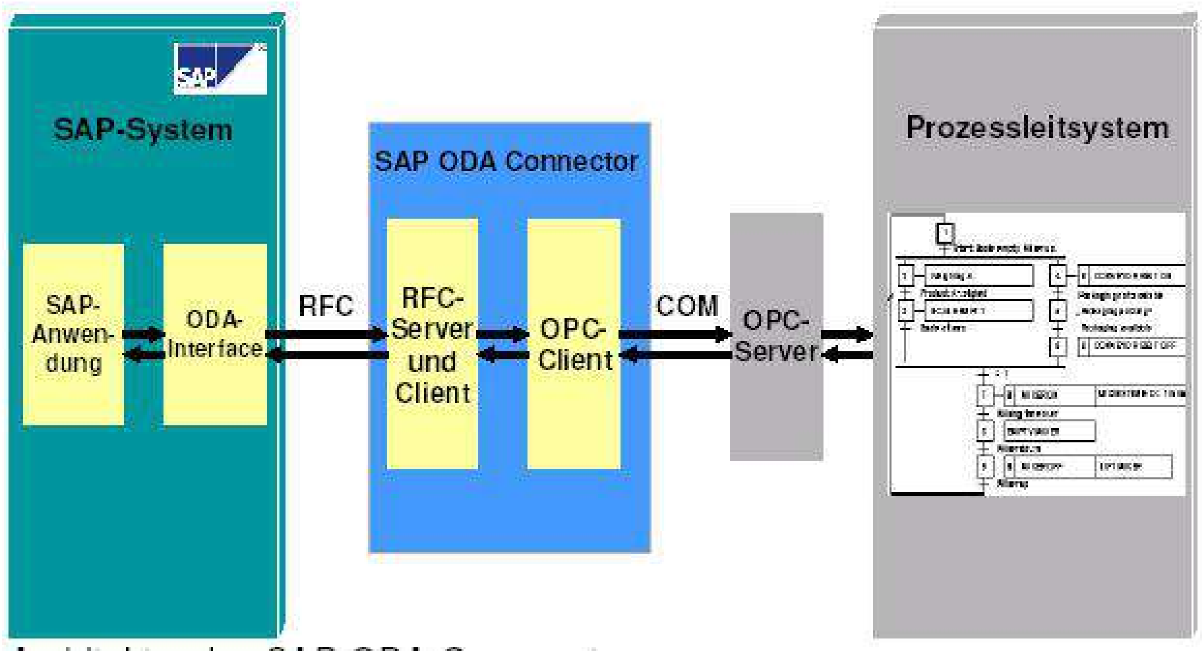
OPC for internal communication: AIP as example

ABB's Integration Platform (AIP) is at the same time an OPC server and an OPC client.
Components (aspects) within AIP expose their properties as OPC objects.
Internal (within AIP) and external communication takes place over OPC.



OPC Connection to Enterprise Resource Planning

Direct connection to SAP (BAPI) is provided by tools such as Matrikon's or Intellution's



Simulators and Explorer: which helps are available

Explorer:

Several tools are available on the market to browse OPC servers, especially:

- Matrikon OPC Explorer (no source code)
- TopServer Client (source code in VB available)

Simulator:

OPC data should be simulated before commissioning the real plant.

To this effect, commercial simulation servers allow to create, observe and change variables by hand or according to time functions (ramp, random,…).

Most PLC servers have also a simulation mode.

Freeware servers such as Matrikon have only limited number of variables

These explorers and simulators work with OPC DA, AE is yet seldom.

Client and Servers

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- Automation and Custom Interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

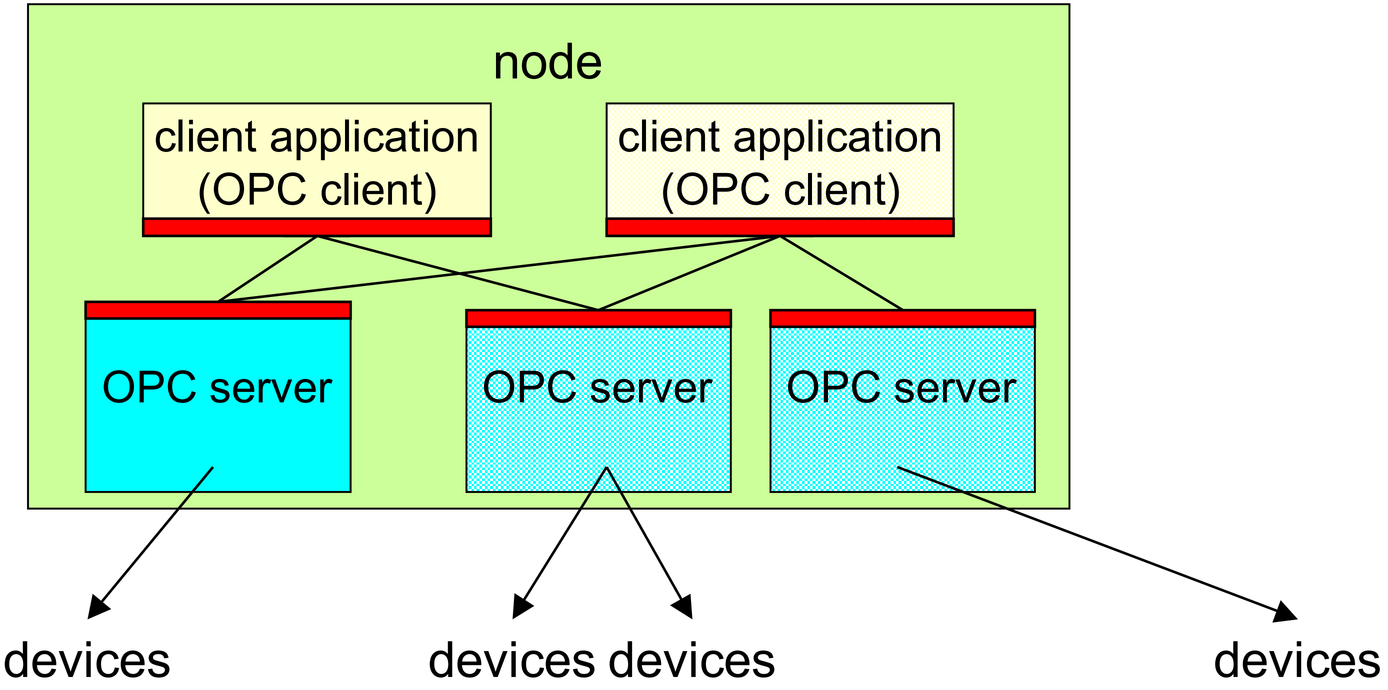
OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

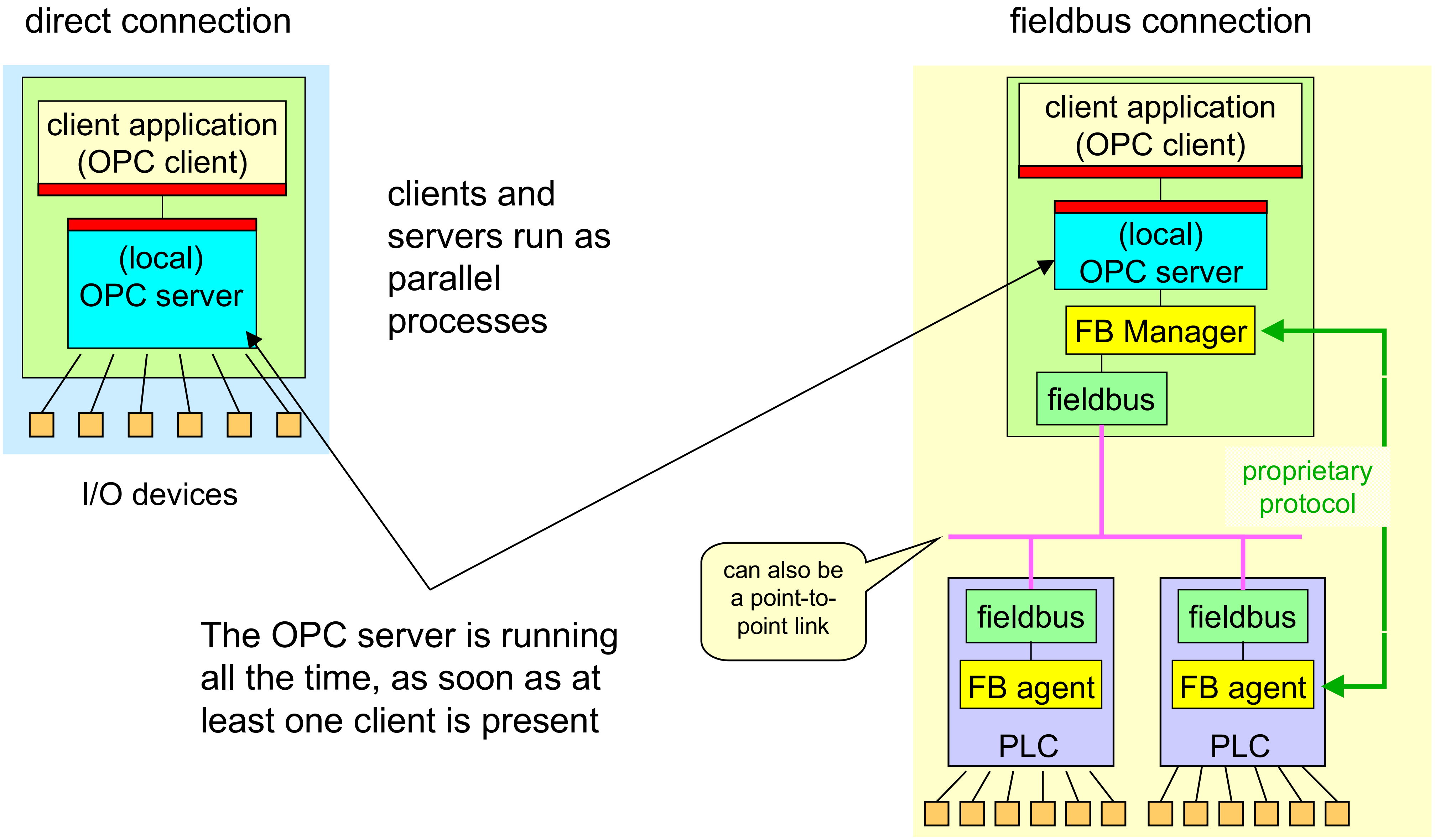
Server(s) and Client(s) in the same node



Clients and servers run as parallel processes

The OPC specification defines the interface between client and server in the form of objects and methods.

Direct and Fieldbus access



direct connection

client application (OPC client)

(local) OPC server

I/O devices

clients and servers run as parallel processes

The OPC server is running all the time, as soon as at least one client is present

fieldbus connection

client application (OPC client)

(local) OPC server

FB Manager

fieldbus

proprietary protocol

can also be a point-to-point link

fieldbus

FB agent

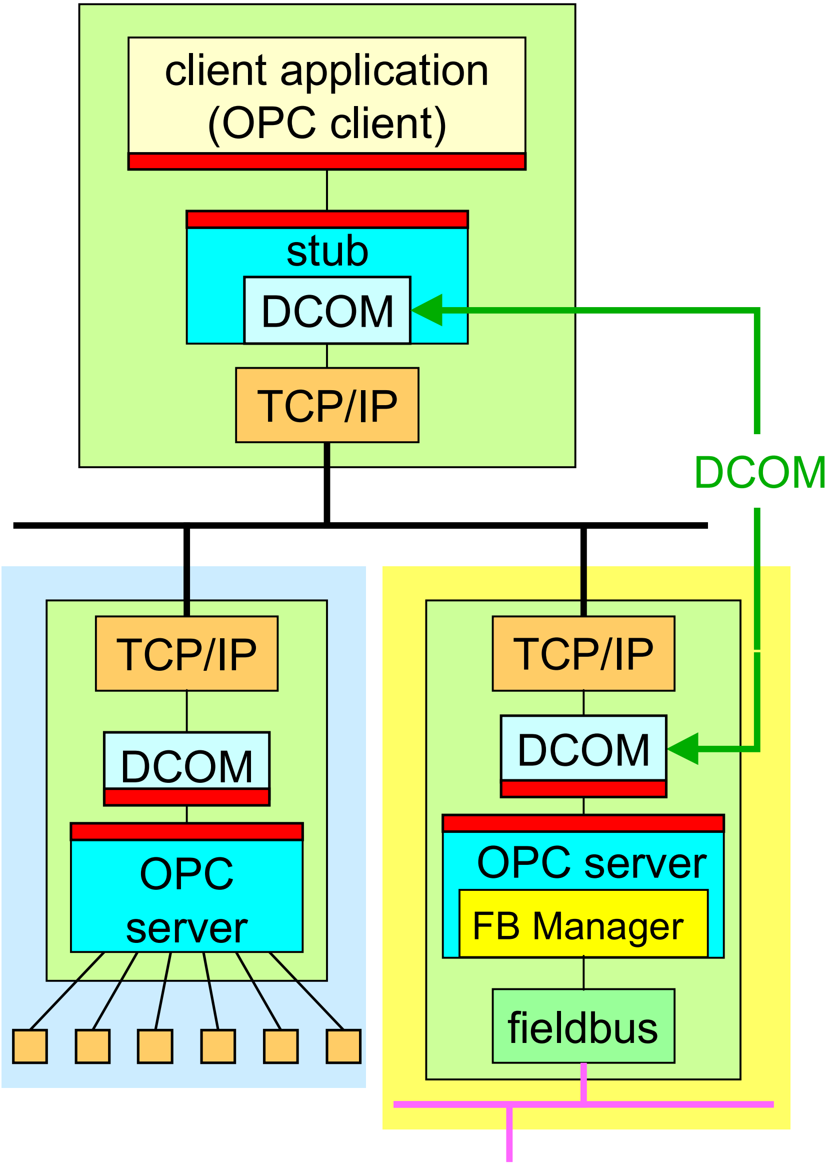
PLC

fieldbus

FB agent

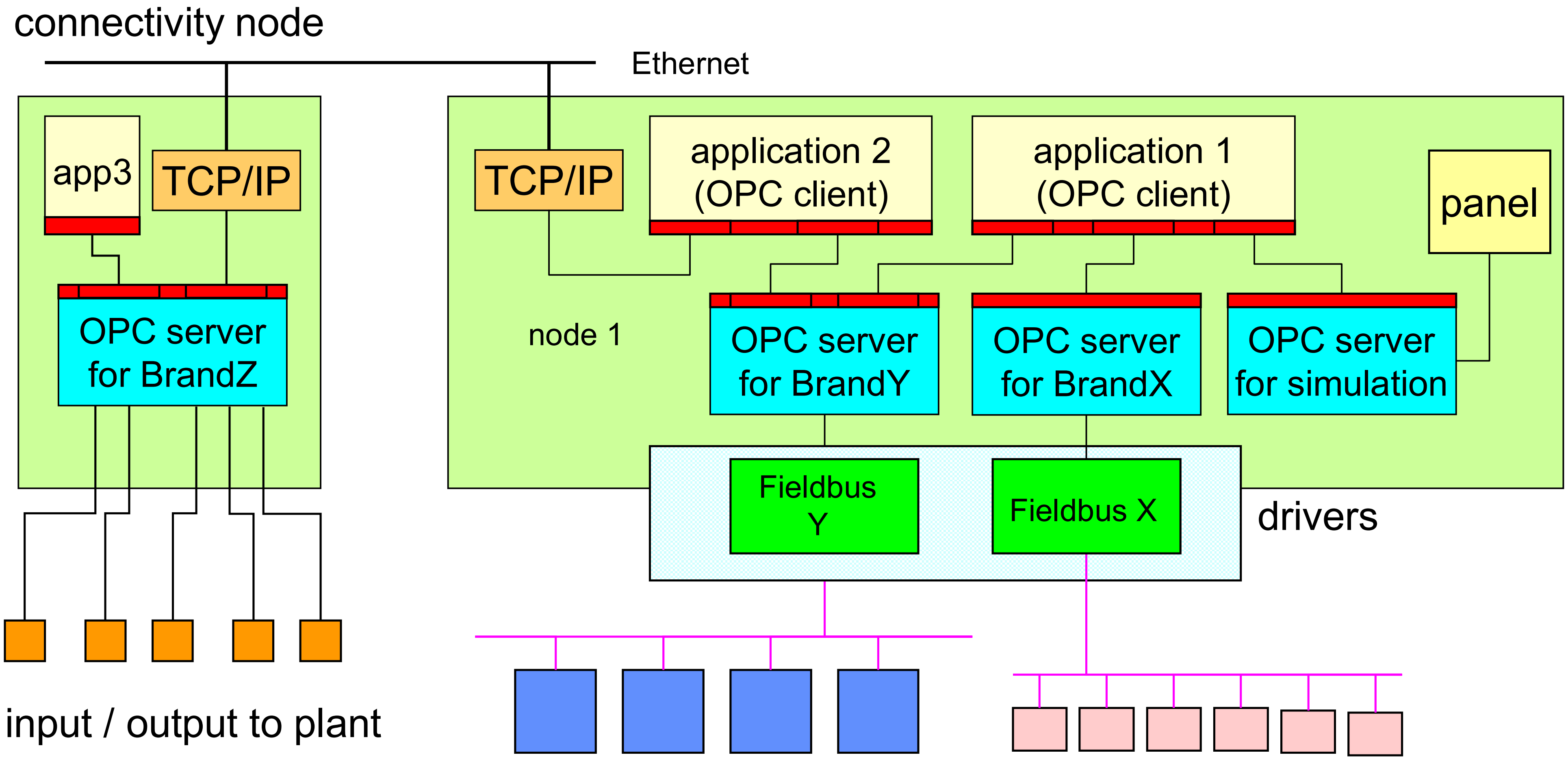
PLC

Accessing a server in another node



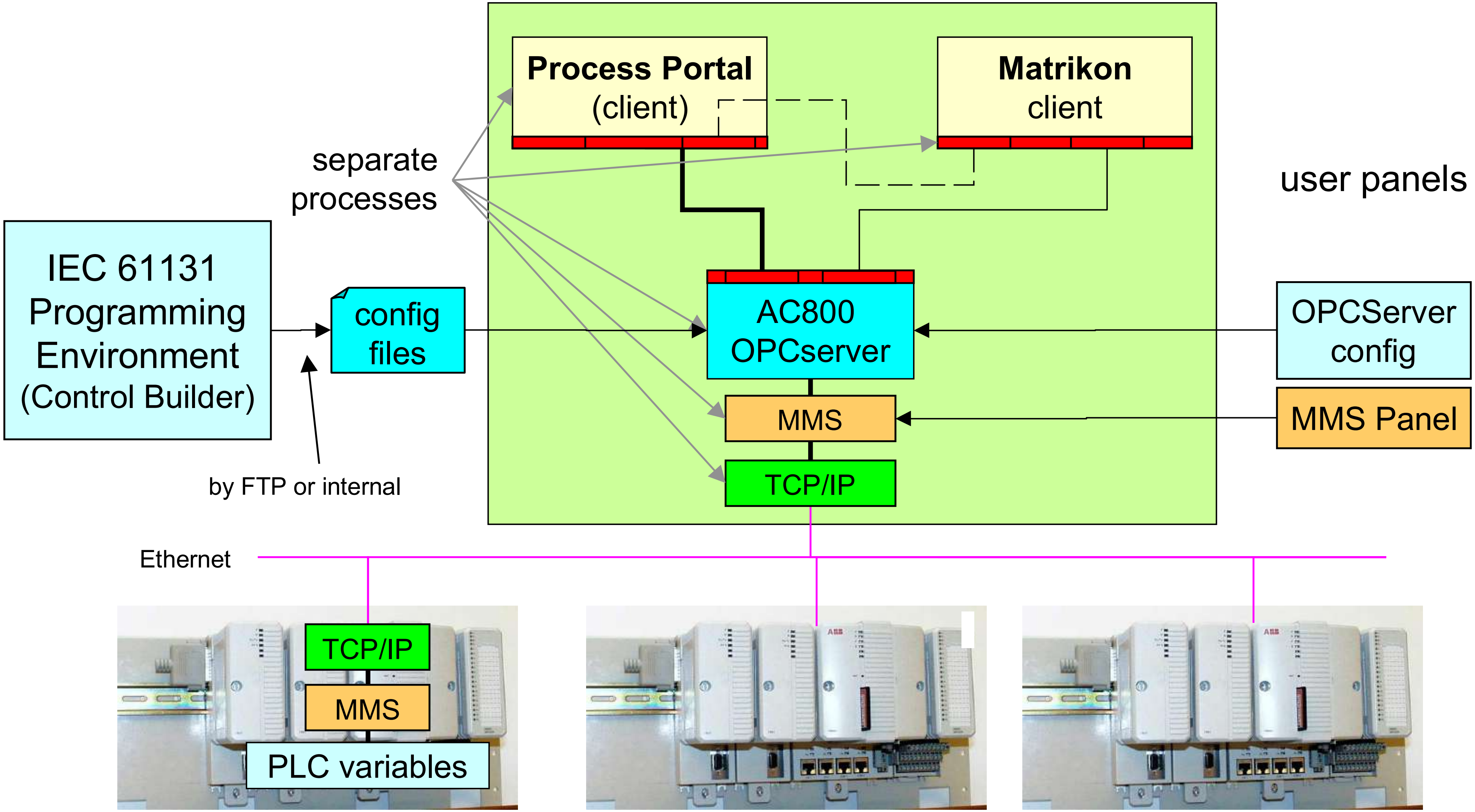
Limitation:
does not work over firewalls.
Solution:
OPC XML (see later)

Full-fledged COM/DCOM across multiple nodes



The OPC servers supports multiple clients and servers on the same, or on remote nodes. they run as separate processes (as soon as at least one client is requesting them)

Example: ABB AC800 OPC Server



The variables are defined in the server, not in the PLC.

OPC Technology

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: Browsing the server
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

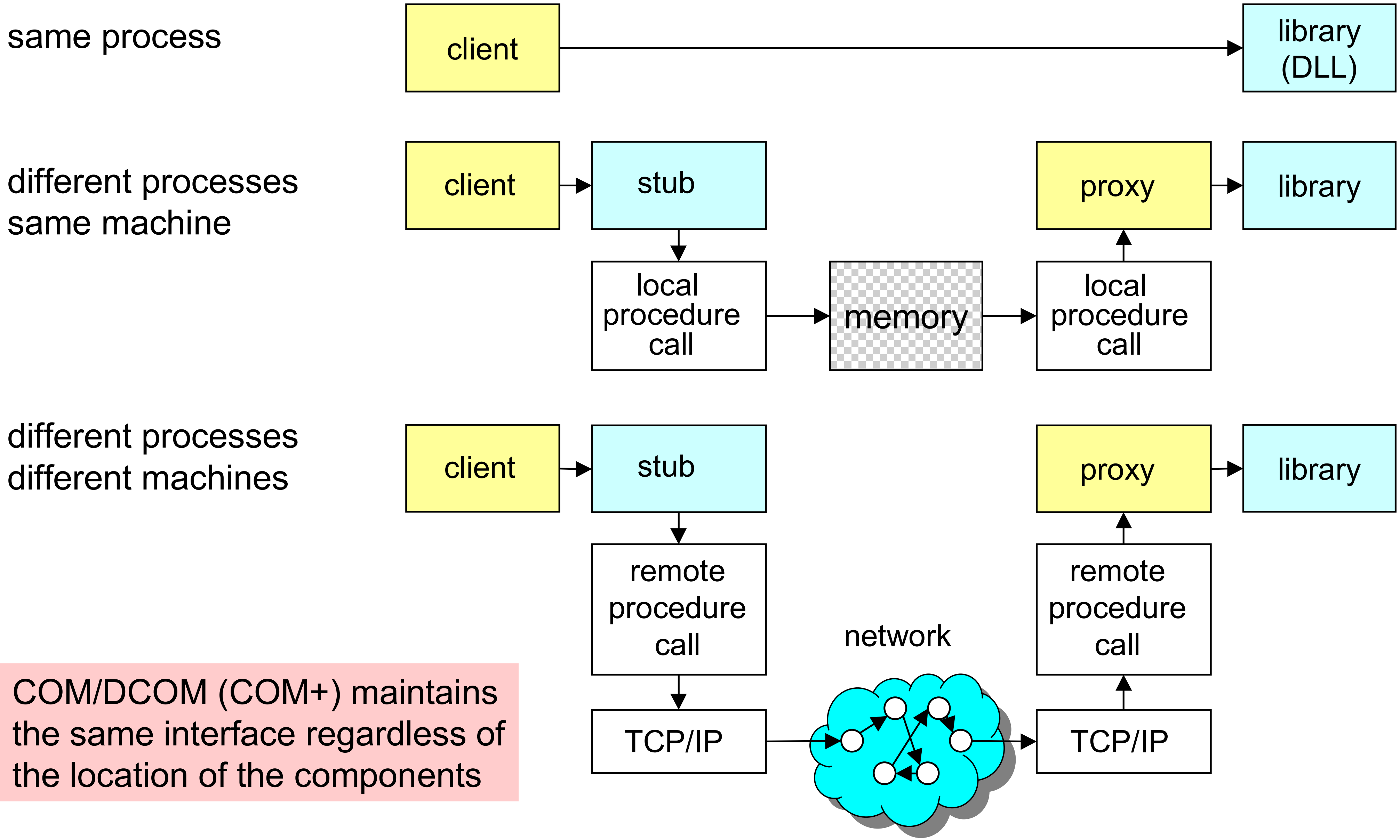
OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

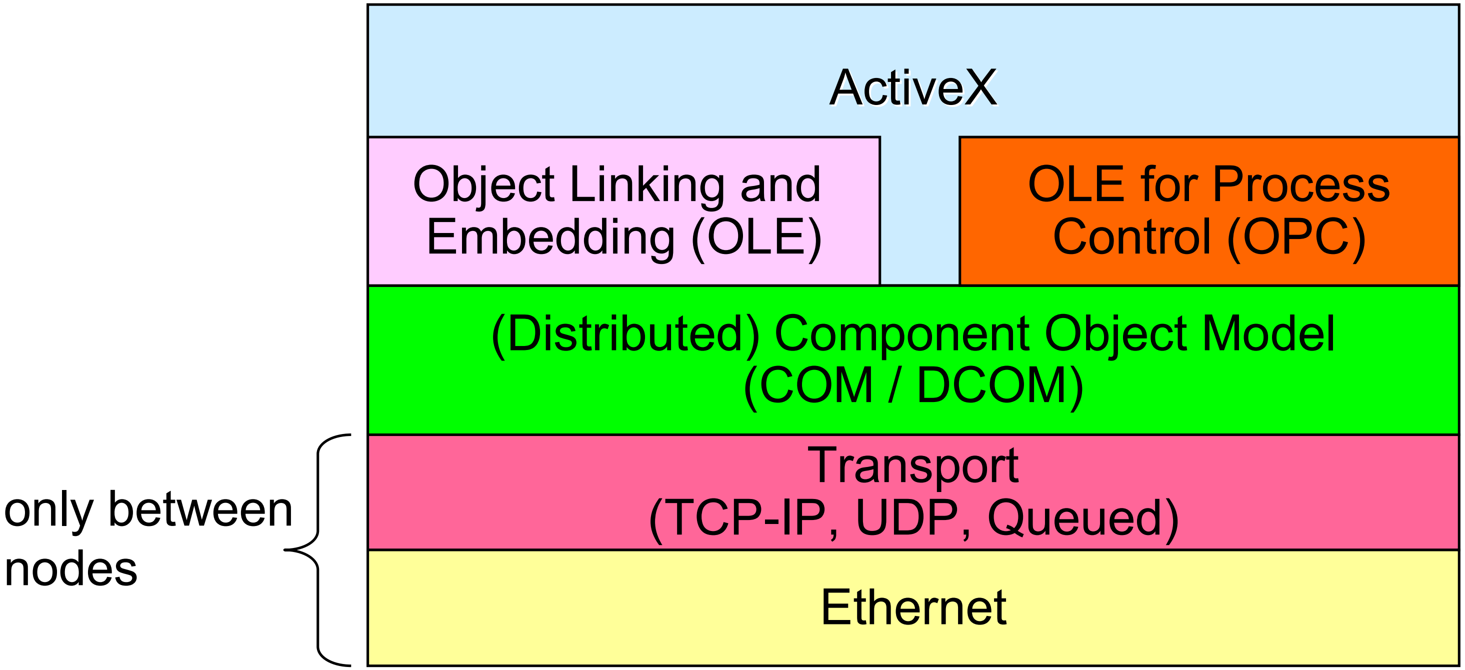
- Overview

COM/DCOM quick intro



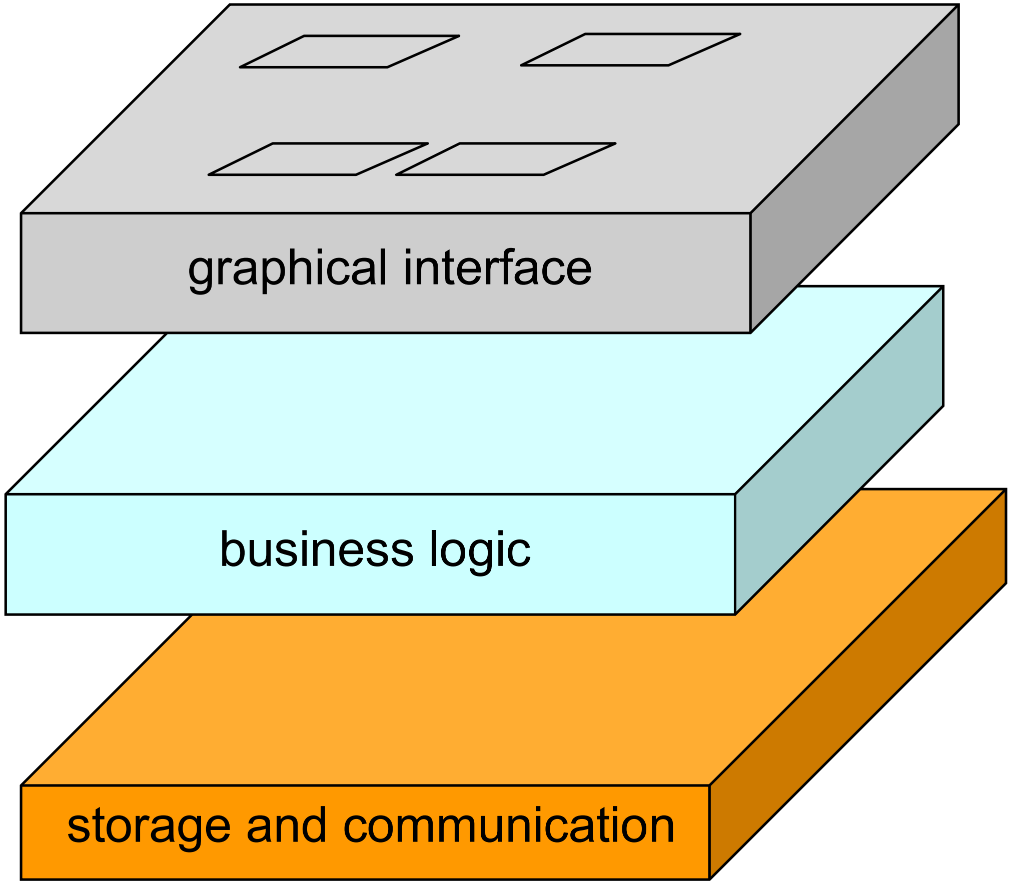
COM/DCOM (COM+) maintains the same interface regardless of the location of the components

OPC technologies

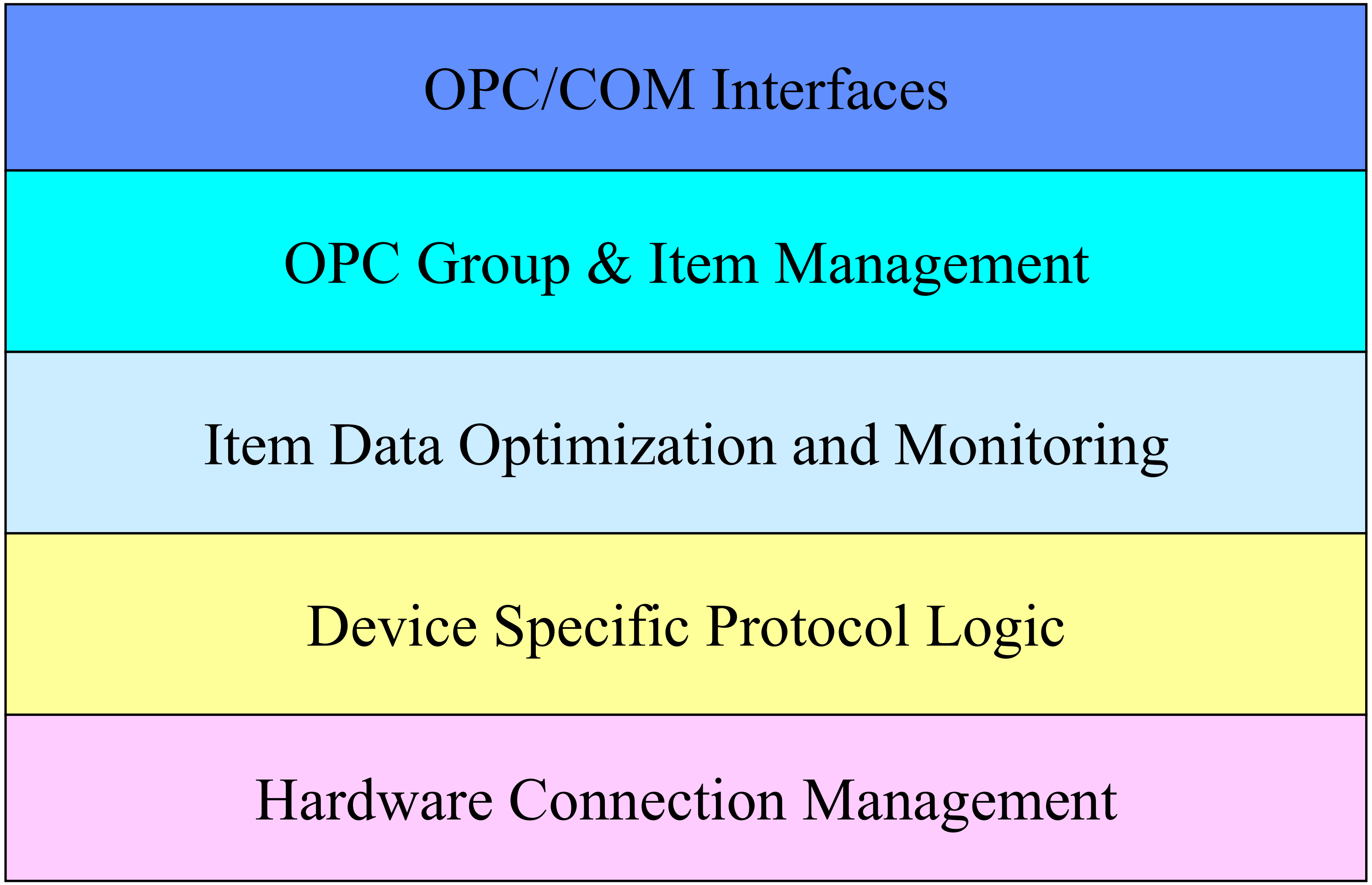

ActiveX
Object Linking and Embedding (OLE)
OLE for Process Control (OPC)
(Distributed) Component Object Model (COM / DCOM)
Transport (TCP-IP, UDP, Queued)
Ethernet
only between nodes

OPC bases on Microsoft's COM/DCOM technology (i.e. it only works on Windows platforms). Effort to port it to other platforms (Linux) and web transport protocols (XML) are in progress. Advantages are the direct integration into all applications, such as Excel.

Three-tiers Active-X components



Structure of an OPC server



OPC/COM Interfaces

OPC Group & Item Management

Item Data Optimization and Monitoring

Device Specific Protocol Logic

Hardware Connection Management

“Automation” vs. “Custom” interface

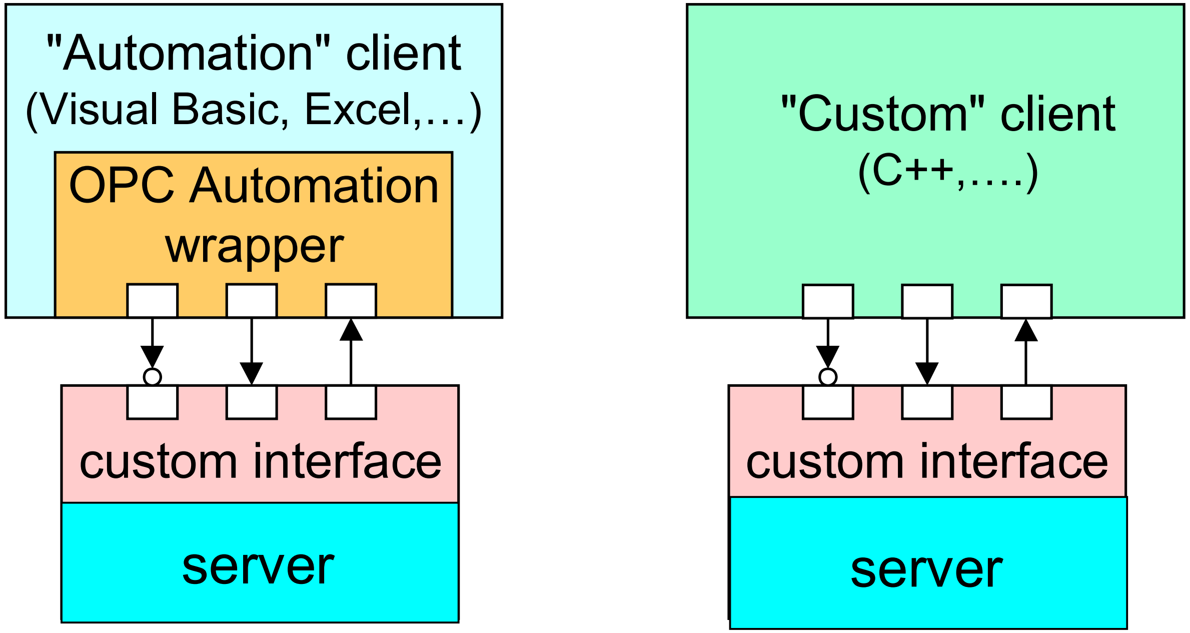
The OPC specifications define two interfaces: "custom" and "automation".

“custom” is the native C++ interface of COM.

“automation” is the interface offered in Visual Basic, used in Word, Excel,…..

The interface is defined by a Type Library (distributed by the OPC Foundation)

Functionality is roughly the same in both models, “automation” is easier to use, but "custom" gives a more extended control.



Assessment Common

What is the objective of OPC ?

On which technology does OPC rely ?

What is an OPC Server ?

Which are the main OPC specifications ?

What are the components of the OPC DA Automation Interface ?

How does an automation platform use the OPC interfaces ?

Industrial Automation
Automation Industrielle
Industrielle Automation

OPC
OLE for Process Control

4 Access to devices

4.3 OLE for Process Control (OPC)

4.3.2 Data Access Specification

Prof. Dr. H. Kirrmann
ABB Research Centre, Baden, Switzerland

2005 May, HK

OPC DA: Overview

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- **Overview: Browsing the server**
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

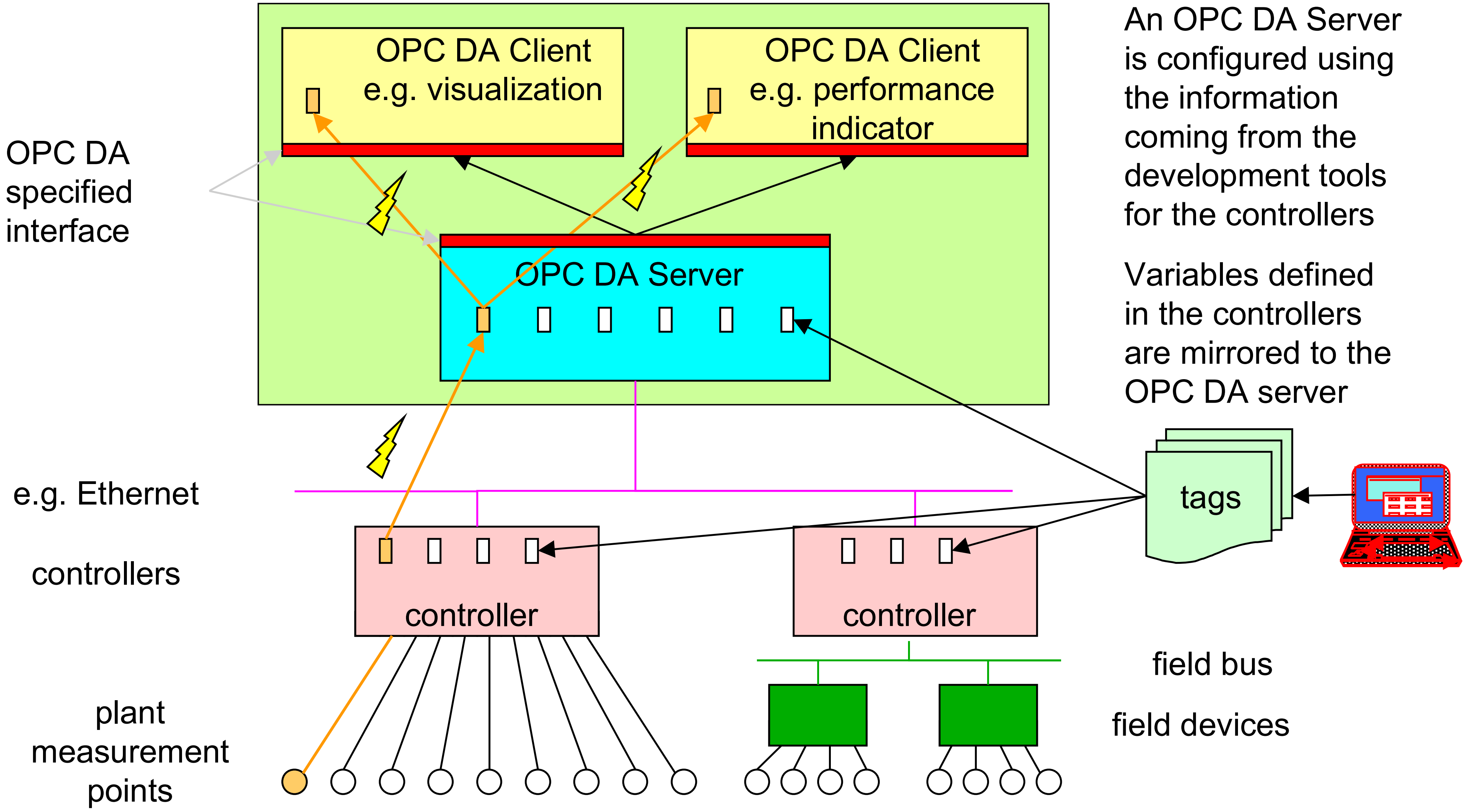
OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

OPC DA: Scope of specification



OPC DA Client
e.g. visualization

OPC DA Client
e.g. performance
indicator

OPC DA
specified
interface

OPC DA Server

e.g. Ethernet

controllers

controller

controller

tags

field bus

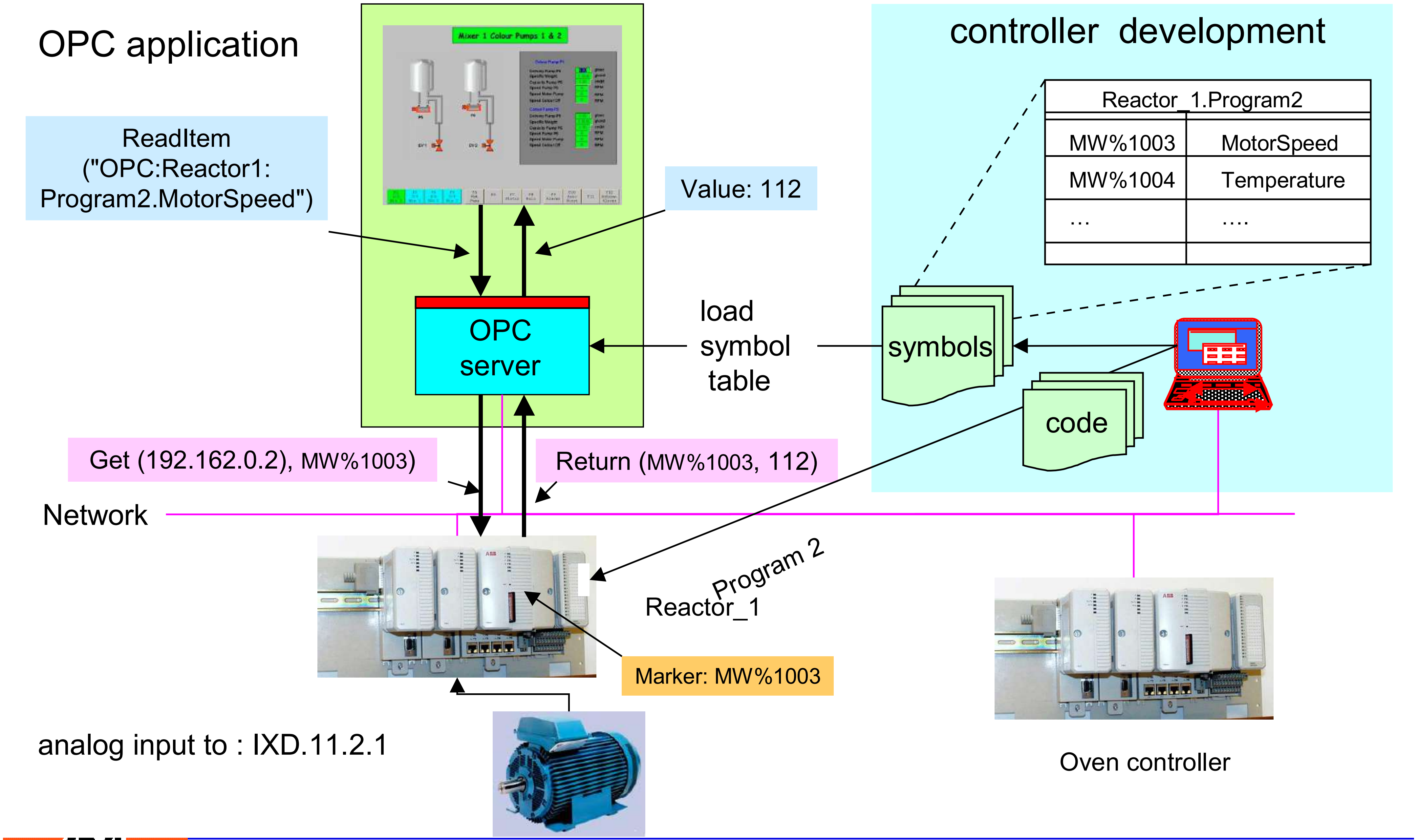
field devices

plant
measurement
points

An OPC DA Server is configured using the information coming from the development tools for the controllers

Variables defined in the controllers are mirrored to the OPC DA server

OPC DA: Example of access to a variable



OPC application

ReadItem ("OPC:Reactor1: Program2.MotorSpeed")

Value: 112

OPC server

load symbol table

controller development

Reactor_1.Program2	
MW%1003	MotorSpeed
MW%1004	Temperature
…	….

symbols

code

Get (192.162.0.2), MW%1003)

Return (MW%1003, 112)

Network

Program 2

Reactor_1

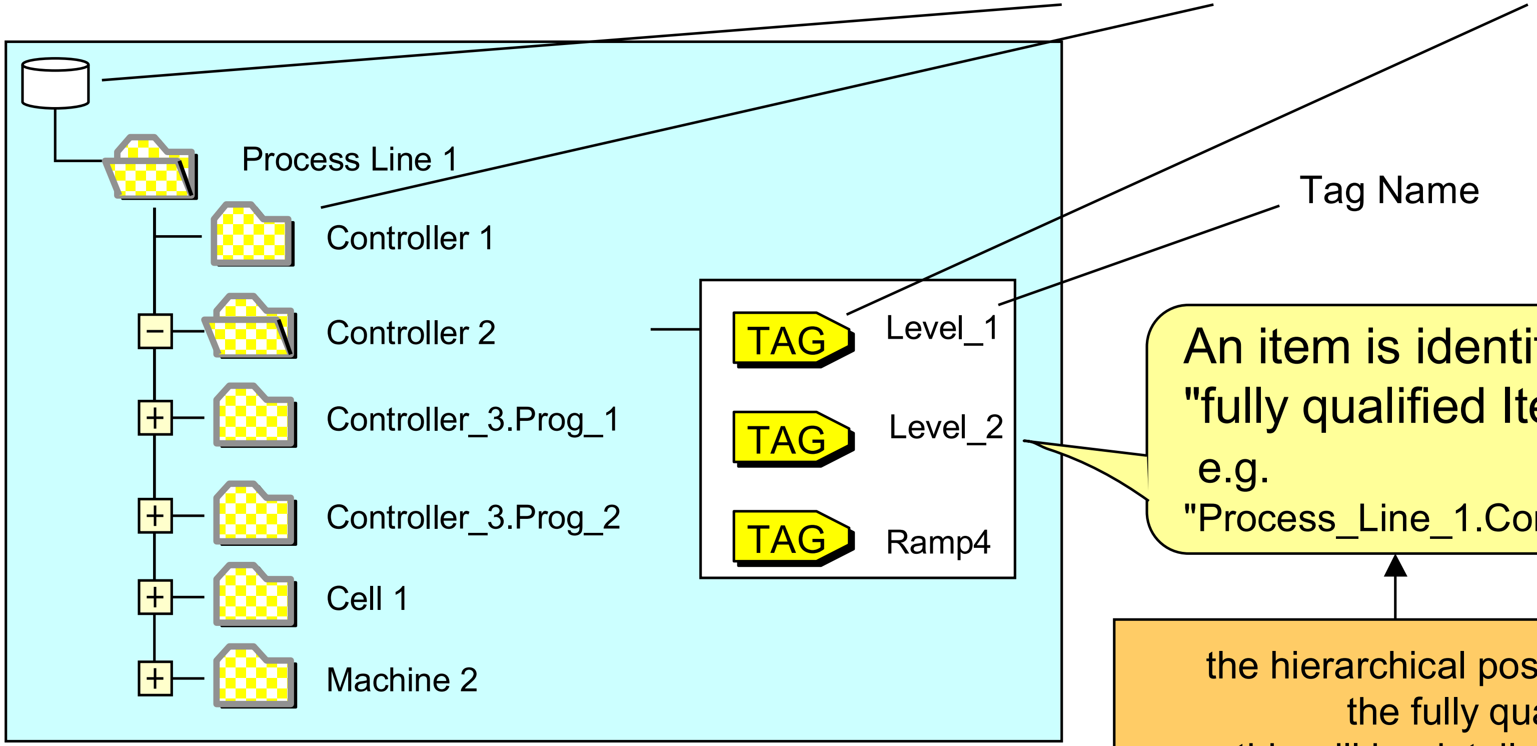
Marker: MW%1003

analog input to : IXD.11.2.1

Oven controller

OPC DA: Objects as viewed by the OPC server

An OPC server is structured as a directory with root, branches and leaves (items)



An item is identified by its "fully qualified ItemID", e.g. "Process_Line_1.Controller_2.Level_2"

the hierarchical position may differ from the fully qualified ItemID
this will be detailed under „browsing“

Branches may contain other branches and items
The structure may also be flat instead of hierarchical
This structure is defined during engineering of the attached devices and sensor/actors.
(Intelligent servers could configure themselves by reading the attached devices)

OPC DA: Browsing - methods

An OPC DA server presents an interface that allows the client to explore its structure, with the methods:

```
MoveDown
MoveUp
MoveToRoot


showBranches
showLeafes *
```

`GetItemID`: retrieves the fully qualified item ID (see later)

Optional:
`GetAccessPath`: retrieves the access path for items that can be accessed over different ways.

The Access Path is an optional information that the client may provide regarding how to get to the data, where several possibilities exist. Its use is highly server specific. Do not confound with hierarchical path.

(*the English error is unfortunate)

OPC DA: Browsing: Fully Qualified ItemID and hierarchy

A server has internally two ways to access the items:

1) the path shown when exploring the tree, and
2) the „fully qualified ItemID“, which is the internal name used by the server.

Example:

an item reached as:	Root.SimulatedItems.UserDefined.Ramp.Ramp1
needs to be accessed internally as:	UserDefined!Ramp.Ramp1

Clients usually look for an item though the hierarchical way.

They position the browser on the corresponding branch and retrieve the fully qualified item ID, which is the name of the item as the server understands it.

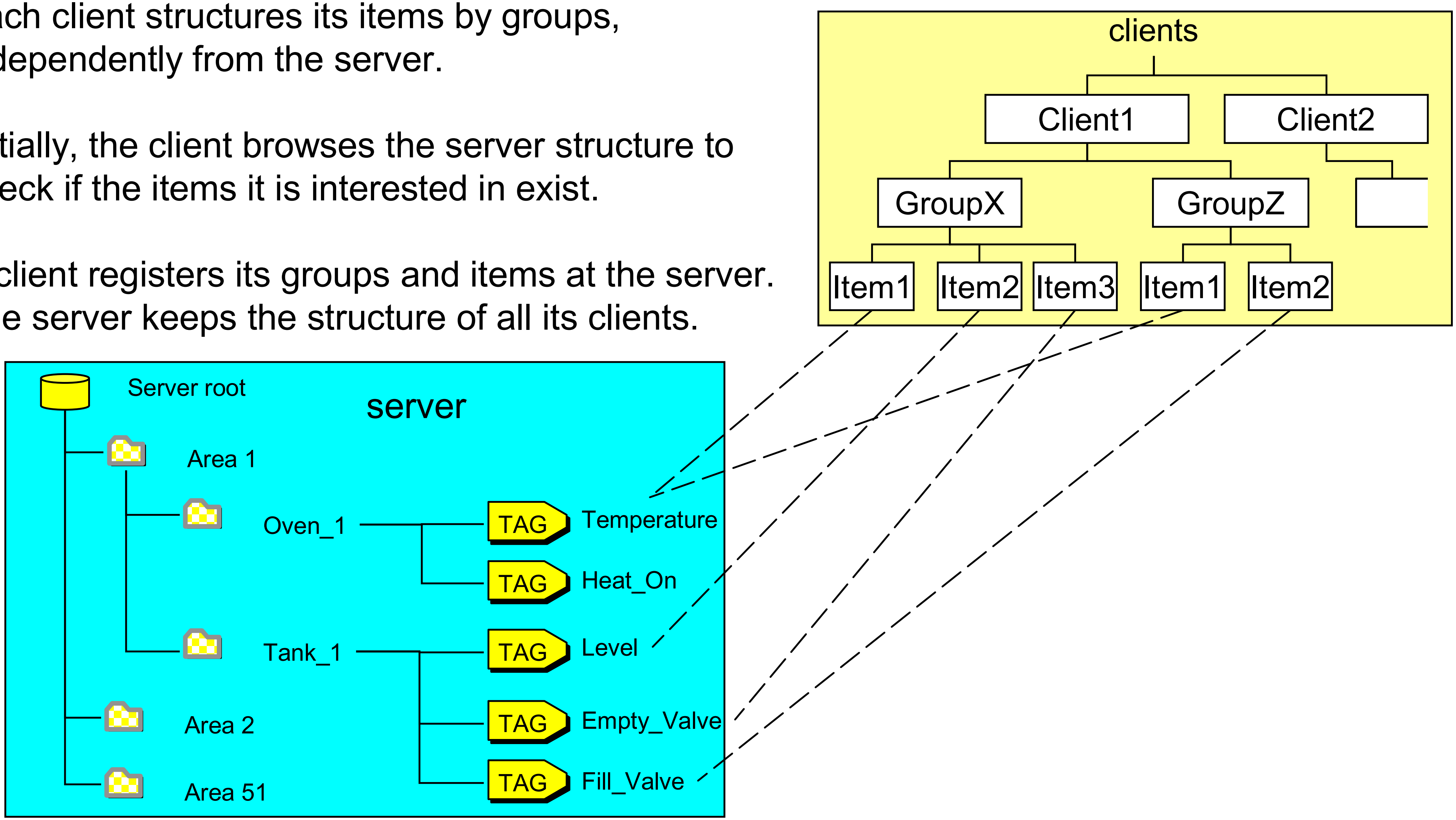
The fully qualified name is only used at configuration time, afterwards, objects are accessed over client handles and server handles (see later)

OPC DA: Objects as viewed by the OPC client

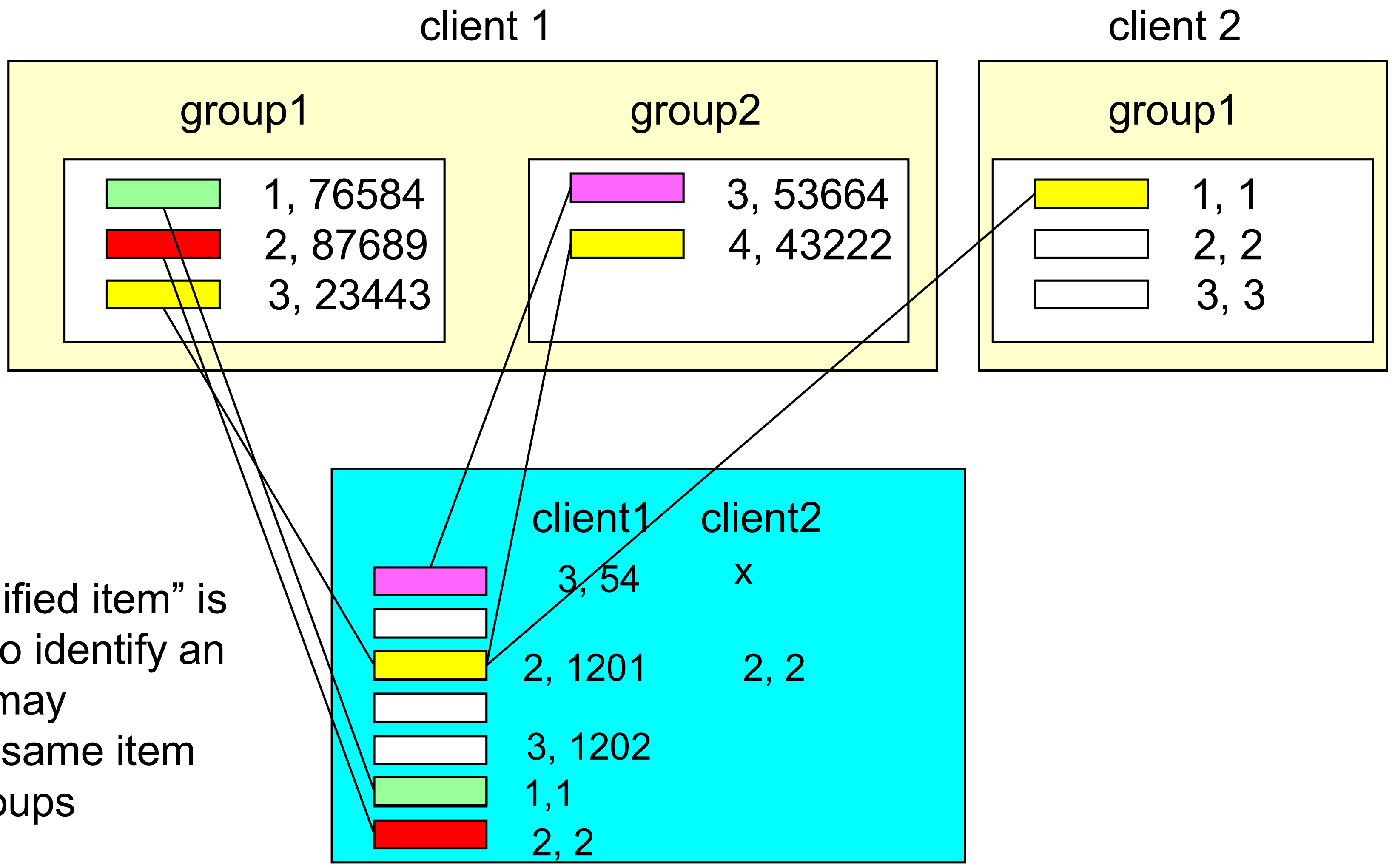
Each client structures its items by groups, independently from the server.

Initially, the client browses the server structure to check if the items it is interested in exist.

A client registers its groups and items at the server. The server keeps the structure of all its clients.



OPC DA: Client Handle and Server handle



The “fully qualified item” is not sufficient to identify an item, a client may subscribe the same item in different groups

The pair { ClientHandle, ServerHandle } uniquely identifies an item.

OPC DA: Object Types and properties

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- **Objects, types and properties**
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

OPC DA: Item properties

The process data are represented by three dynamic properties of an item:

value: numerical or text

time-stamp: the time at which this data was transmitted from the PLC to the server
this time is UTC (Greenwich Winter time), not local time.

quality: validity of the reading (not readable, dubious data, o.k.)

(when writing, only the value is used)

OPC DA: Item types

Each item value has a type:

Boolean,
Character,
Byte, (1 byte)
Word, (2 bytes)
Double Word, (4 bytes)
Short Integer (2 bytes)
Integer (4 bytes)
Long Integer:
Long Unsigned Integer
Single Float (4 bytes)
Double Float (8 bytes)
Currency,
Date,
String,
Array of "the above"

When accessing an item, the client may request that it is returned with a specific type, which could be different from the server's type.

(The server's type is returned by browsing)

Type conversion is left to the server, there are no rules whether and how a server does the conversion.
(use with caution)

Care must be taken that the data types in the programming language or in the database match those of the OPC Server.

Items also may have engineering units, but this option is not often used.

OPC DA: Communication Model

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- Objects, types and properties
- **Communication model**
- Simple Programming Example
- Standard and components

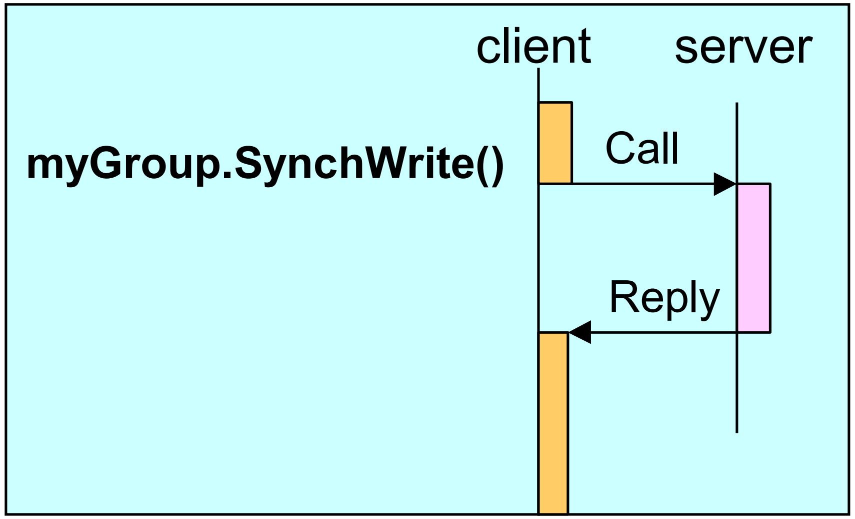
OPC Alarms and Events Specification

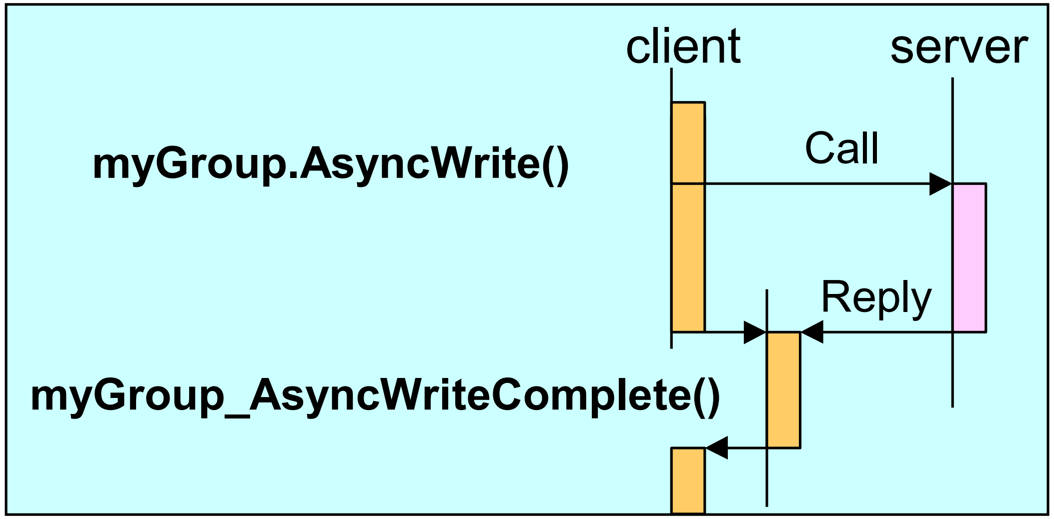
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

OPC DA: Write Communication Models (per group)



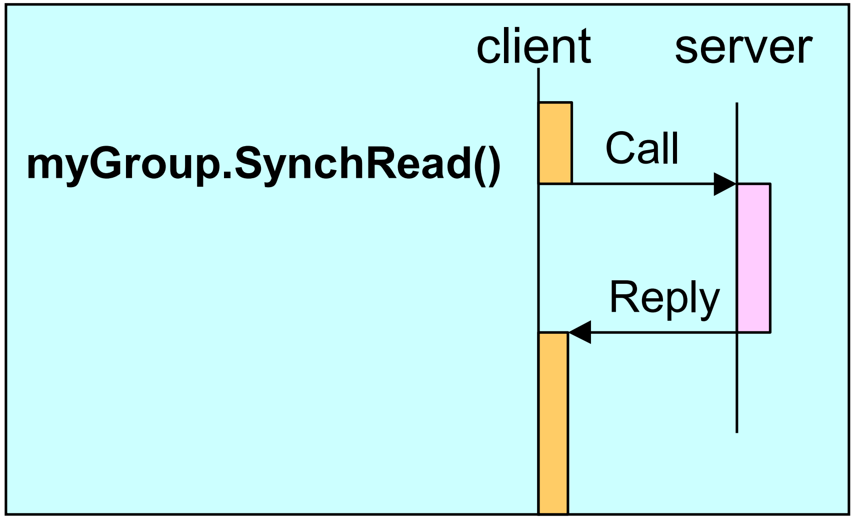


The OPC interface accesses only groups, not individual items.

However, the "automation interface" allows to access individual items, but this does not give rise to a communication

OPC DA: Read Communication Models (per group)

synchronous



client

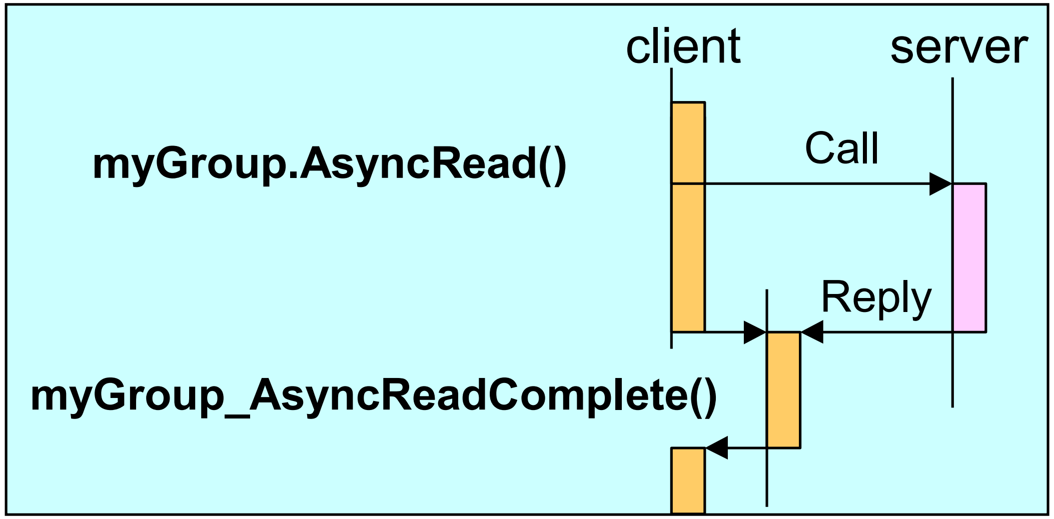
server

myGroup.SynchRead()

Call

Reply

asynchronous



client

server

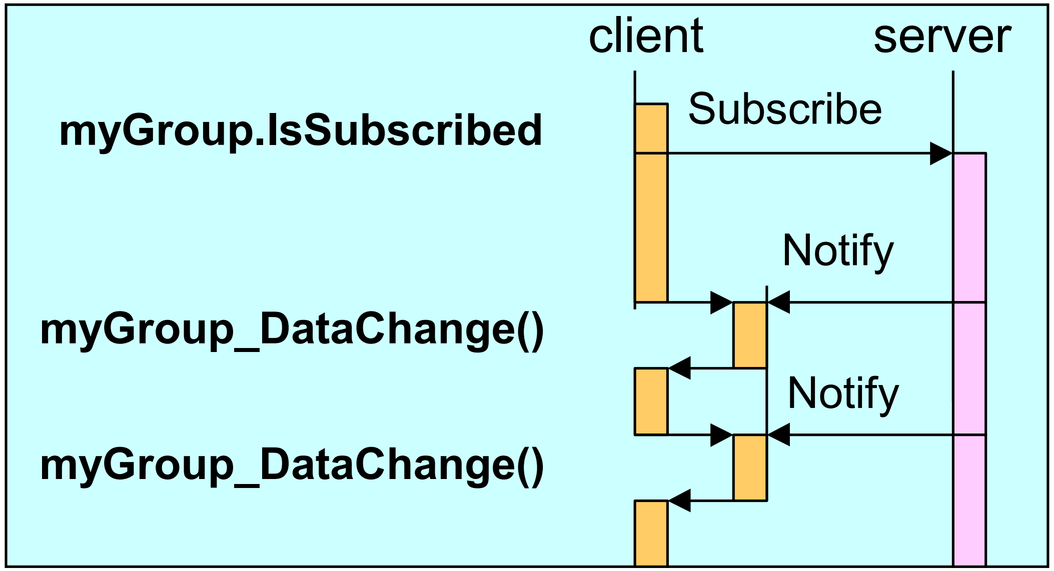
myGroup.AsyncRead()

Call

Reply

myGroup_AsyncReadComplete()

on change
("subscription-based")



client

server

myGroup.IsSubscribed

Subscribe

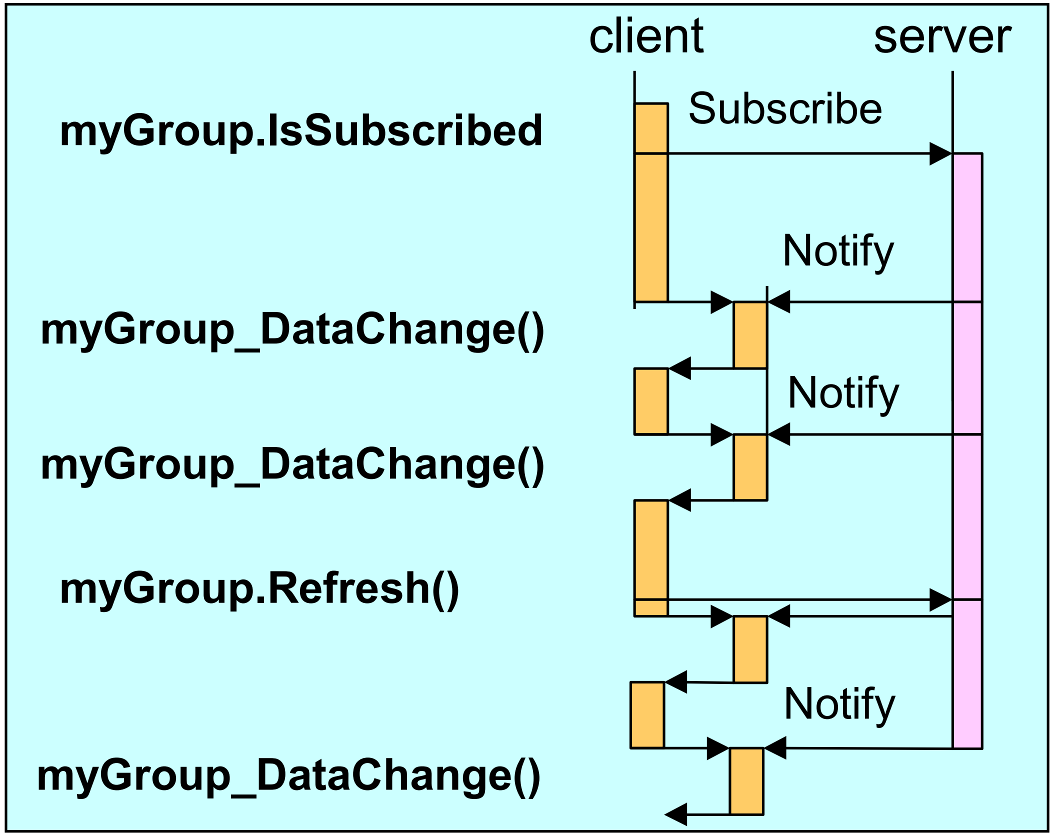
Notify

myGroup_DataChange()

Notify

myGroup_DataChange()

OPC DA: Transmission by subscription (events)



The server notifies the client if an item changed

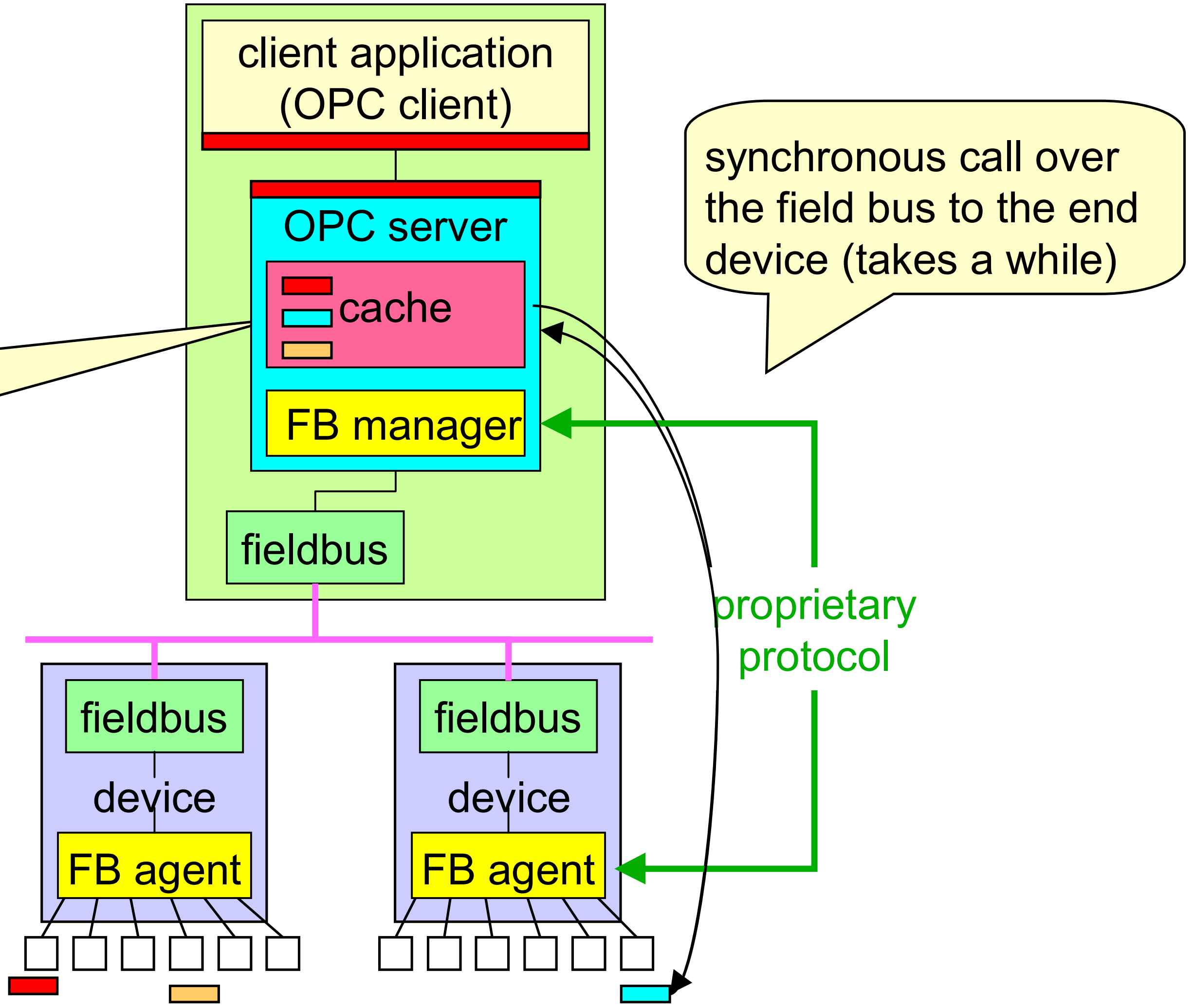
- in a particular group (myGroup_DataChange) or
- in any of the groups (myGroups_GlobalDataChange)

In the second case, only the group in which the item changed will be sent.

OPC DA: "Cache" or "Device" ?

"SynchRead" reads the data either from
cache (local to the PC) or
reads synchronous from the device.

"Write" is always to device
(DA 3.0 allows write to cache)



synchronous call over the field bus to the end device (takes a while)

server samples items
(at the RequestedUpdateRate)
and puts them into cache

no need for “device access” when fieldbus operates cyclically…

OPC DA: When are subscribed data transmitted ?

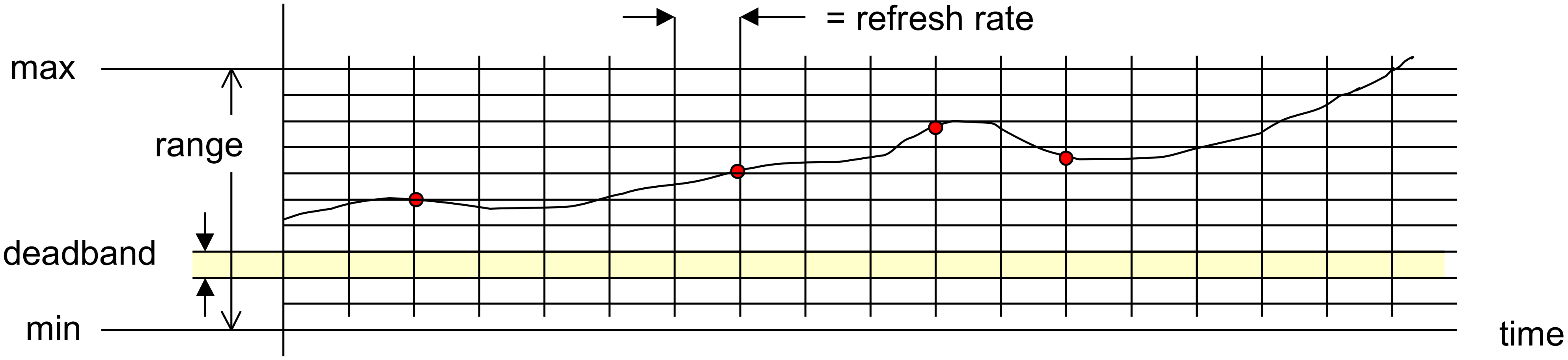
A group has two properties to control when a data change is to be transmitted:

myGroup.Refreshrate:

the rate at which the server samples the values, expressed in seconds ! (1/rate)
earliest interval between changes of value (throttles changes, but may miss some)

myGroup.Deadband

applied only to analog values: deadband = % the range (in Engineering Units).
value is transmitted if the difference since last transmission exceeds deadband.
Problem: applies to whole group without considering individual items, seldom implemented.



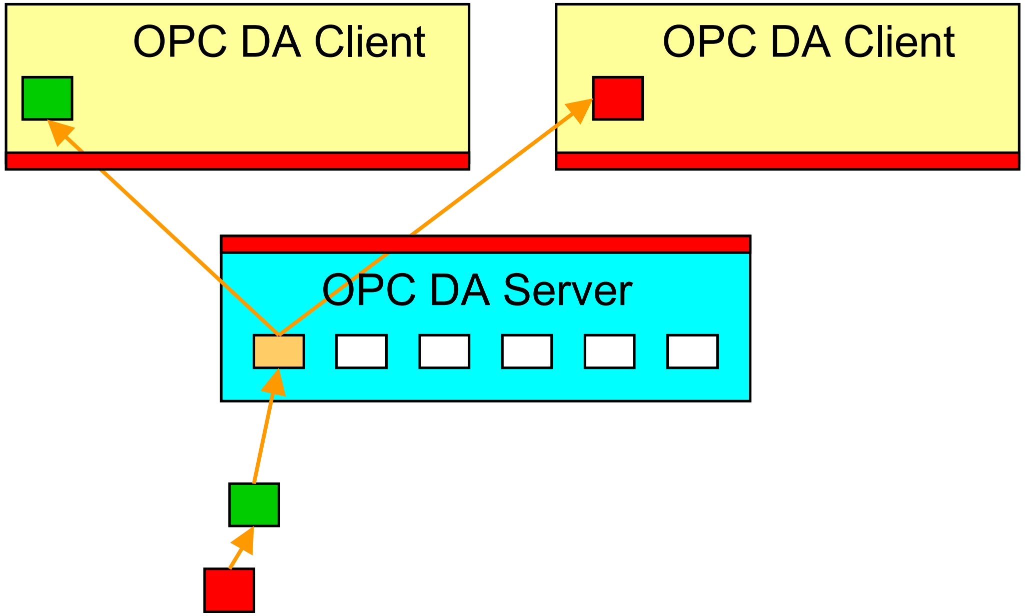
OPC DA: communication paradigm

OPC DA works according to the “shared memory” paradigm.

This means that a newer value overwrites the older one, no queues or history are kept.

The server does not guarantee that different clients see the same snapshot of the plant.

The server does not guarantee that all changes to variables are registered, changes may be missed if the polling period is too low.



OPC DA: Programming Example

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- Objects, types and properties
- Communication model
- **Simple Programming Example**
- Standard and components

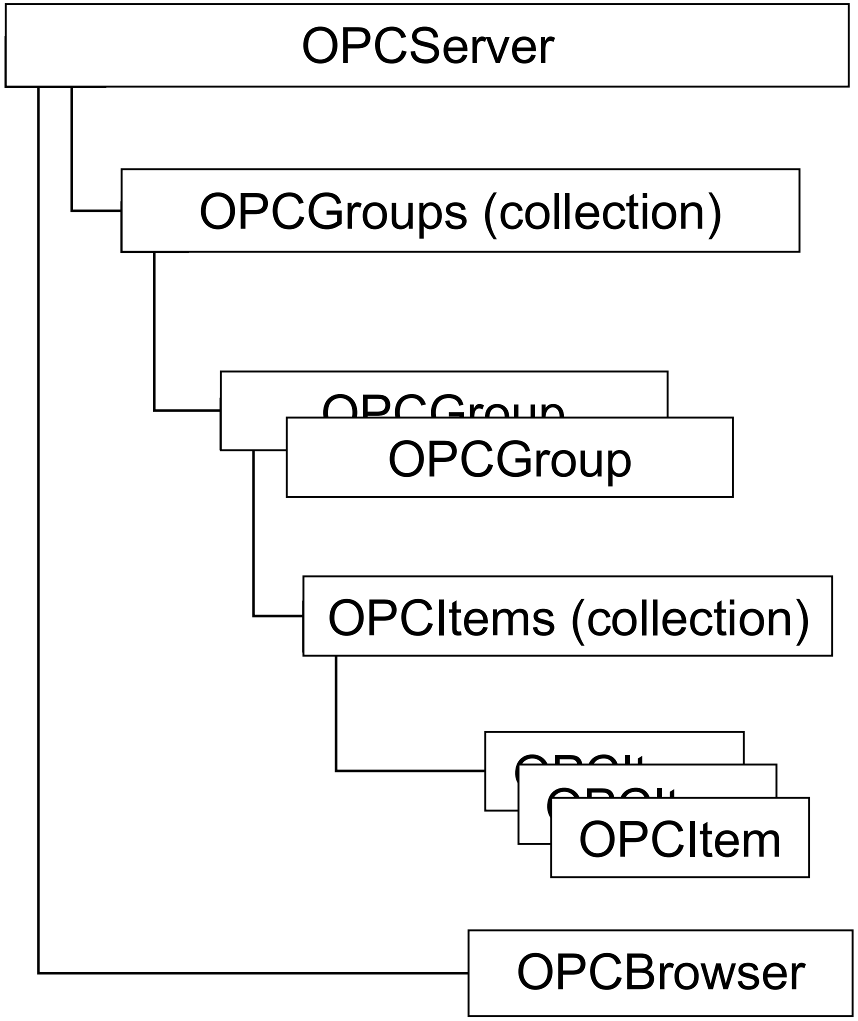
OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

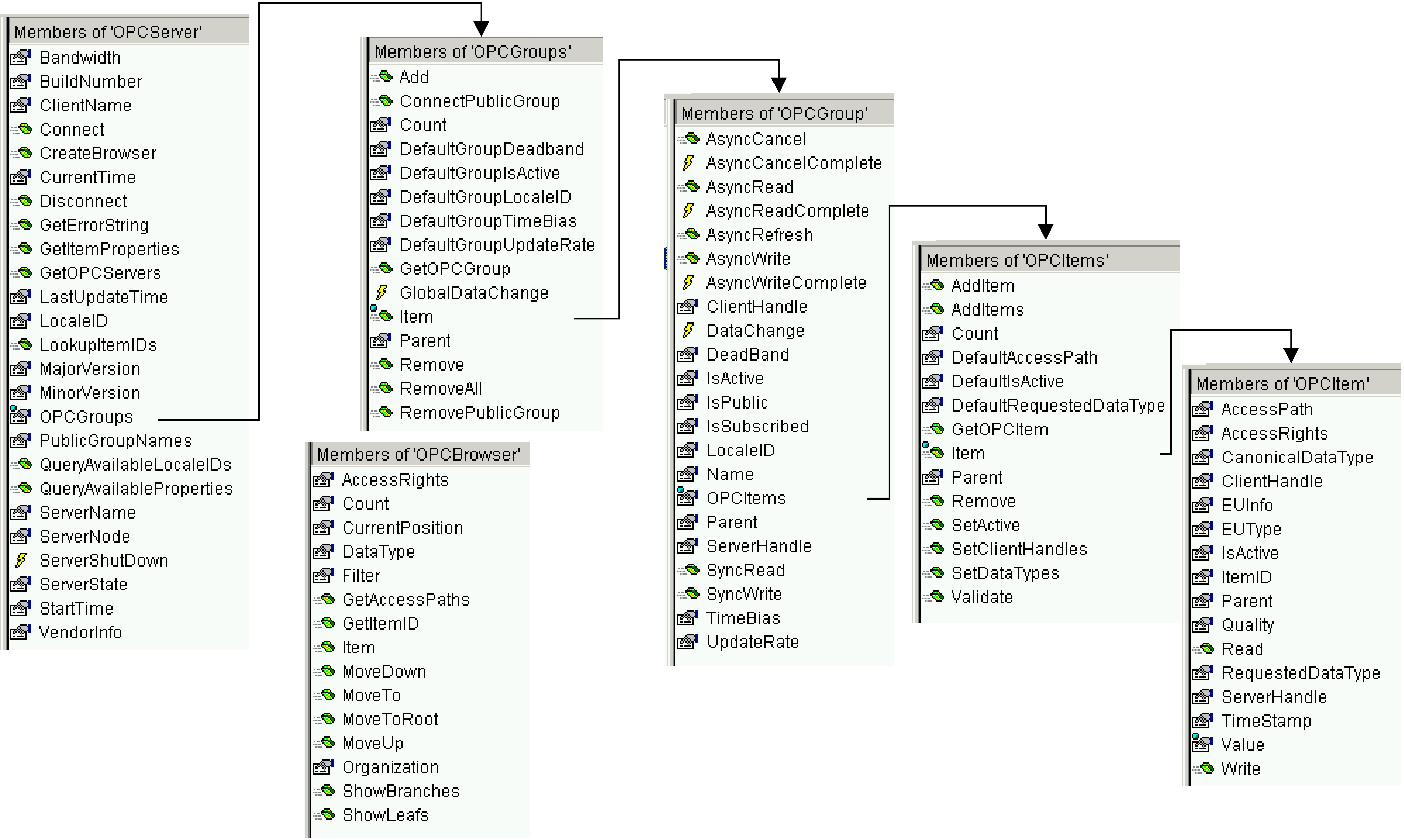
- Overview

OPC DA: Object hierarchy at the client



Description
An instance of an OPC Server. You must create an OPCServer object before you can get references to other objects. It contains the OPCGroups Collection and creates OPCBrowser objects.
A collection containing all of the OPCGroup objects this client has created within the scope of the OPCServer that the Automation Application has connected to via OPCServer.Connect()
An instance of an OPCGroup object. this object maintains state information and provides the mechanism to access data for the OPCItems Collection object that the OPCGroup object references.
A collection containing all of the OPCItem objects this client has created within the scope of the OPCServer, and corresponding OPCGroup object that the Automation Application has created.
An automation object that maintains the item's definition, current value, status information, last update time. Note the Custom Interface does not provide a separate Item Object.
An object that browses item names in the server's configuration. There exists only one instance of an OPCBrowser object per instance of an OPC Server object.

OPC DA: Automation interface summary



Members of 'OPCServer'

- Bandwidth
- BuildNumber
- ClientName
- Connect
- CreateBrowser
- CurrentTime
- Disconnect
- GetErrorString
- GetItemProperties
- GetOPCServers
- LastUpdateTime
- LocaleID
- LookupItemIDs
- MajorVersion
- MinorVersion
- OPCGroups
- PublicGroupNames
- QueryAvailableLocaleIDs
- QueryAvailableProperties
- ServerName
- ServerNode
- ServerShutDown
- ServerState
- StartTime
- VendorInfo

Members of 'OPCGroups'

- Add
- ConnectPublicGroup
- Count
- DefaultGroupDeadband
- DefaultGroupIsActive
- DefaultGroupLocaleID
- DefaultGroupTimeBias
- DefaultGroupUpdateRate
- GetOPCGroup
- GlobalDataChange
- Item
- Parent
- Remove
- RemoveAll
- RemovePublicGroup

Members of 'OPCBrowser'

- AccessRights
- Count
- CurrentPosition
- DataType
- Filter
- GetAccessPaths
- GetItemID
- Item
- MoveDown
- MoveTo
- MoveToRoot
- MoveUp
- Organization
- ShowBranches
- ShowLeafs

Members of 'OPCGroup'

- AsyncCancel
- AsyncCancelComplete
- AsyncRead
- AsyncReadComplete
- AsyncRefresh
- AsyncWrite
- AsyncWriteComplete
- ClientHandle
- DataChange
- DeadBand
- IsActive
- IsPublic
- IsSubscribed
- LocaleID
- Name
- OPCItems
- Parent
- ServerHandle
- SyncRead
- SyncWrite
- TimeBias
- UpdateRate

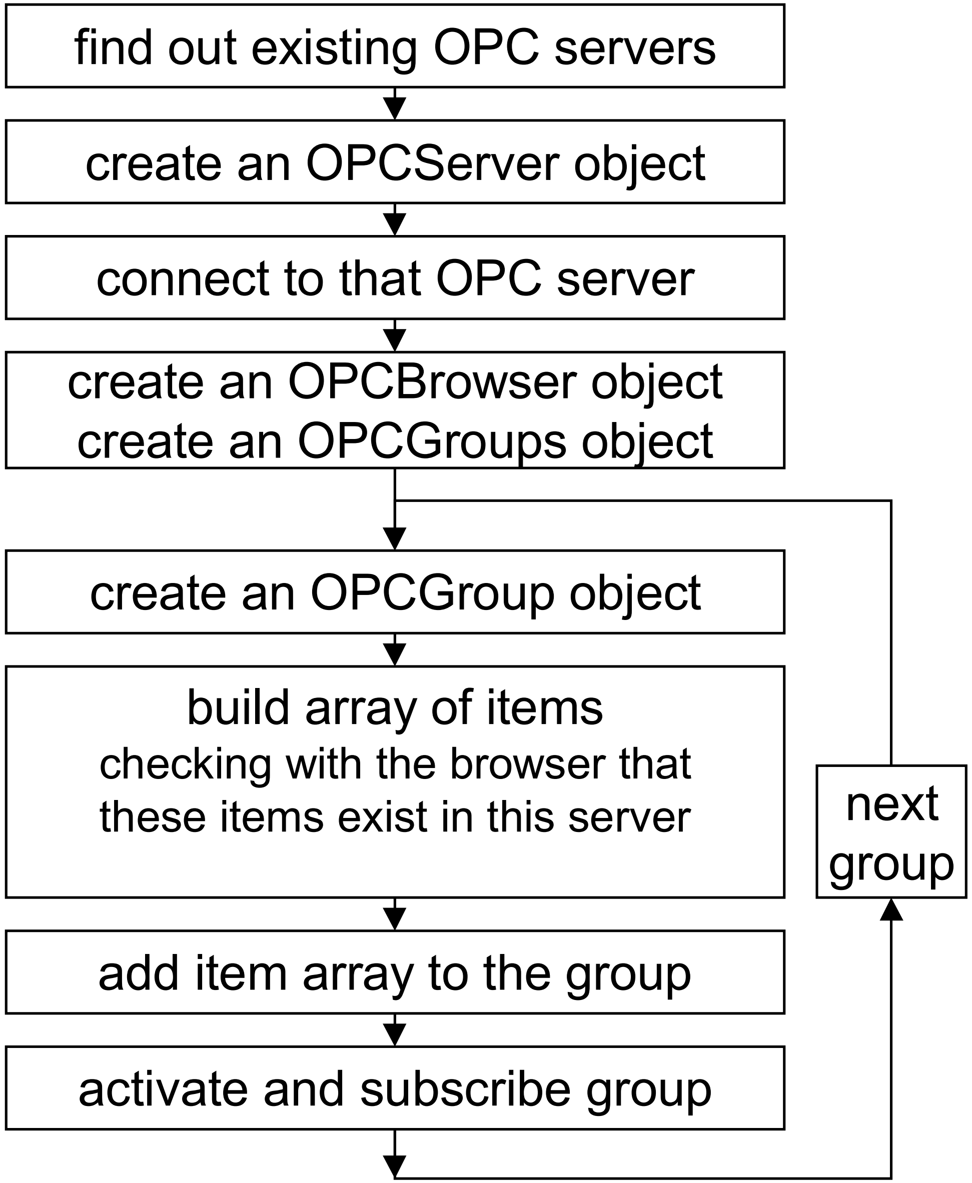
Members of 'OPCItems'

- AddItem
- AddItems
- Count
- DefaultAccessPath
- DefaultIsActive
- DefaultRequestedDataType
- GetOPCItem
- Item
- Parent
- Remove
- SetActive
- SetClientHandles
- SetDataTypes
- Validate

Members of 'OPCItem'

- AccessPath
- AccessRights
- CanonicalDataType
- ClientHandle
- EUInfo
- EUType
- IsActive
- ItemID
- Parent
- Quality
- Read
- RequestedDataType
- ServerHandle
- TimeStamp
- Value
- Write

OPC DA: Program - initialising a connection


find out existing OPC servers
create an OPCServer object
connect to that OPC server
create an OPCBrowser object
create an OPCGroups object
create an OPCGroup object
build array of items
checking with the browser that
these items exist in this server
add item array to the group
activate and subscribe group
next
group

```
myDummyServer.GetOPCServers

myServer = new OPCServer

myServer.Connect

Set myBrowser = myServer.Browser
Set myGroups = myServer.Groups

Set myGroup1 = myGroups.Add
Set MyItems = MyGroup1.OPCItems

FQItems1[1] = "Device1.Temp1"
ClientHandle1[1] = 101
ReDim ServerHandle1(nrItems)
ReDim ServerErrors1(nrItems)
ReDim Value1(nrItems)

myGroup1.AddItems

myGroup1.IsActive
myGroup1.IsSubscribed
```

OPC DA: Program - Declarations

```
Option Base 1                          'OPC arrays indices start with 1
Dim WithEvents MyServer As OPCServer   'OPC Server Object (Events optional)
Dim WithEvents MyGroups As OPCGroups   'OPC Group Collection (Events opt.)
Dim WithEvents MyGroup  As OPCGroup    'OPC Group Object
' items
Dim nrItems          As Integer
Dim MyItems             As OPCItems    'OPC Item Collection Object
Dim MyItem              As OPCItem     'OPC Item Object
Dim ItemsID(2)          As String      'fully qualified items (see later)
Dim ClientHandles(2)    As Long
Dim ServerHandles()     As Long        ' must be a dynamic array
Dim ServerErrors()      As Long        ' must be a dynamic array
```

Reference: "OPC Automation 2.0" must be included into Visual Basic or C#

(if missing: copy `opcdaauto.dll` to `C:\WINNT\System32\opddaauto)`
and register it: `C:\>regsvr32 C:\WINNT\System32\opddaauto.`

A simple way to do it: install Software Toolbox's TopServer (freeware)

OPC DA: Program - Finding the OPC servers

The GetOPCServers function applied to a dummy Server object allow to list the existing servers on this node or on another node (over DCOM - security must be set correctly).
The information about which OPC servers exist is taken from the registry, where it has been
by each server at its installation time

```
Private Sub ShowServers(netNodeName As String)
    Dim dummyServer As OPCServer
    Dim Servers As Variant                                            ' this is an array of strings
    Dim cntServers As Integer

    Set dummyServer = New OPCServer                                   ' create a dummy server object
    Servers = dummyServer.GetOPCServers(netNodeName)                  ' returns all available servers

    For cntServers = LBound(Servers) To UBound(Servers)               ' display the names
        MsgBox Servers(cntServers)
    Next cntServers

    Set Getserver = Nothing                                        ' delete this object (was created by New)
    Exit Sub
```

OPC DA: Program - Connecting to the OPC server

```
Set MyServer = New OPCServer                 ' create server object
MyServer.Connect ("Matrikon.OPC.Simulation") ' connect to Matrikon server
```

Before connecting, it is safe to check the name of the server from the server's list.
Also, it is preferable to include the connection in a separate routine since it can fail:

```
Function ServerGetCare(Name As String, ServerNode As String) As OPCServer
    On Error GoTo ServerGetCareErr
    Dim MyOPCServer As New OPCServer

    MyOPCServer.Connect ServerName, ServerNode ' connect risky
    Set ServerGetCare = MyOPCServer
    Exit Function

ServerGetCare_Err:                          ' error handler if connect fails
    Err.Clear
    MsgBox "Could not connect"
    Set MyServer = Nothing
    Exit Function
```

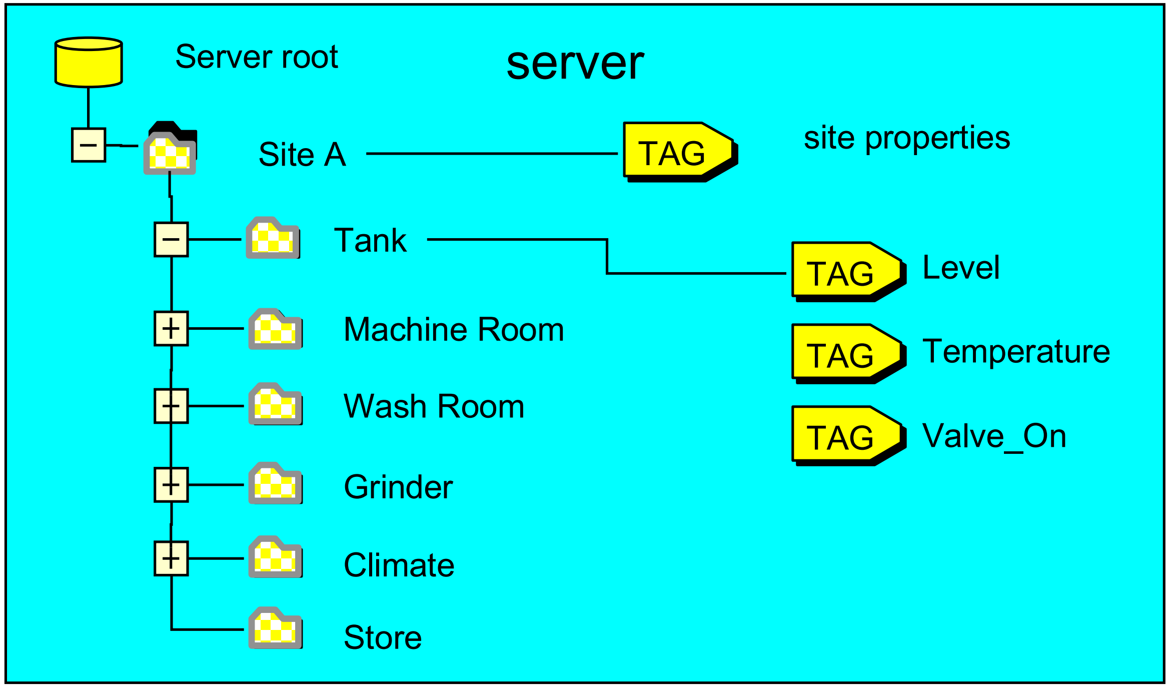
OPC DA: Program - Browsing the server

The object OPCBrowser (of type "collection") acts as a pointer to the server's tree:

```
Dim MyServer As OPCServer
Dim MyBrowser As OPCBrowser
Dim vName As Variant

MyServer.Connect "Matrikon.OPC.Simulation", "Orion"    'server and node name (DCOM)

Set MyBrowser = MyServer.CreateBrowser                 ' create an OPC browser

MyBrowser.ShowBranches                                  ' show the branches
For Each vName In MyBrowser
   MsgBox "Branch: " & vName                            ' display the branch name
Next vName

MyBrowser.ShowLeafs                                     ' explore the leaves
For Each vName In MyBrowser
   MsgBox "Leaf: " & vName                              ' display the leaves's name
Next vName
```

OPC DA: Navigating

```
MyBrowser.MoveDown (strBranch)          ' go down the selected branch tree

MyBrowser.MoveUp                        ' go up the selected branch tree
```



There may be leaves at every branch, since a branch may have properties

OPC DA: Program - get the Fully Qualified ItemID

To get the "fully qualified itemID", one positions the browser at the place where the leaf is attached to the branch and calls GetItemID

```
myOPCBrowser.MoveDown("TankArea")

myOPCBrowser.MoveDown("Tank1")

FQI = myOPCBrowser.GetItemId ("Level")
```

e.g. FQI could be "Controller1;Tanks!Level"

Of course, one can write an Item ID directly when defining a group, but it is safer to browse the server and get the FQI from there, since the delimiter depends on the server.

OPC DA: Program - Creating OPCGroups and OPCItems

```
Set MyGroups = MyServer.OPCGroups            ' create groups collection
Set MyGroup1 = MyGroups.Add("GRP1")          ' add group, name private
Set MyItems = MyGroup1.OPCItems              ' define the OPCItems of group
```

```
FQItemIDs(1) = "Area2.Tank1.Level"           ' fully qualified itemID
ClientHandles(1) = 5                         ' arbitrary
FQItemIDs(2) = "Area2.Tank1.Temperature"     ' fully qualified itemID
ClientHandles(2) = 6                         ' arbitrary (but different)
nrItems = 2
```

```
  MyItems.AddItems _                         ' adds the items to collection
→   nrItems, _                               ' input parameter
→   FQItemIDs, _                             ' input fully qualified ID
→   ClientHandles, _                         ' input ClientHandles
←   ServerHandles, _                         ' return parameter ServerHandles
←   ServerErrors                             ' return parameter ServerErrors
```

```
MyGroup1.ClientHandle = 1                    ' handle of the group (no s) !
MyGroup1.IsActive = True                     ' now ready to send and receive
MyGroup1.IsSubscribed = True                 ' and to generate events
```

The role of the ServerHandles and ClientHandles will be explained later…

OPC DA: Data structures at the client

The client prepares data structures for its items and gives the server the corresponding pointers so the server can update them.

Items to be written and read can be mixed in the same group.

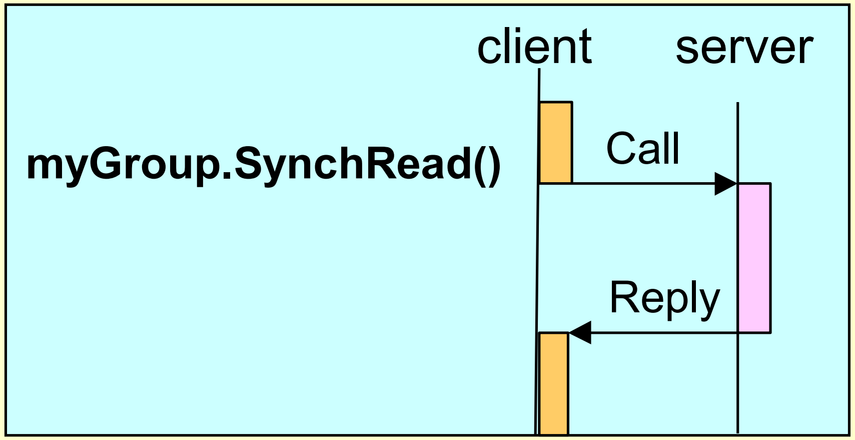
The type of the item (Boolean, Float,…) is implicit, but known at the server

communicated to server by registering group		returned by server when registering		dynamic changes (refreshed on change)		
FullyQualifiedItemID	**ClientHandle**	**ServerHandle**	**ServerError**	**Value**	**Quality**	**TimeStamp**
"Channel1.Device1.Temp1"	100	34543	0	123.4	OK	12:09.234
"Channel1.Device1.Speed1"	102	22532	0	999.8	OK	12:02.214
"Channel1.PLC2.Door"	203	534676	0	0	OK	12:03.002
"Channel1.PLC2.Valve3"	204	787234	0	1	OK	12:02.345
"Channel1.PLC2.CloseDoor"	205	58432	0	0	BAD	12:02.345
..		..	..	..	..	..

Note: OPC indices start with 1 !

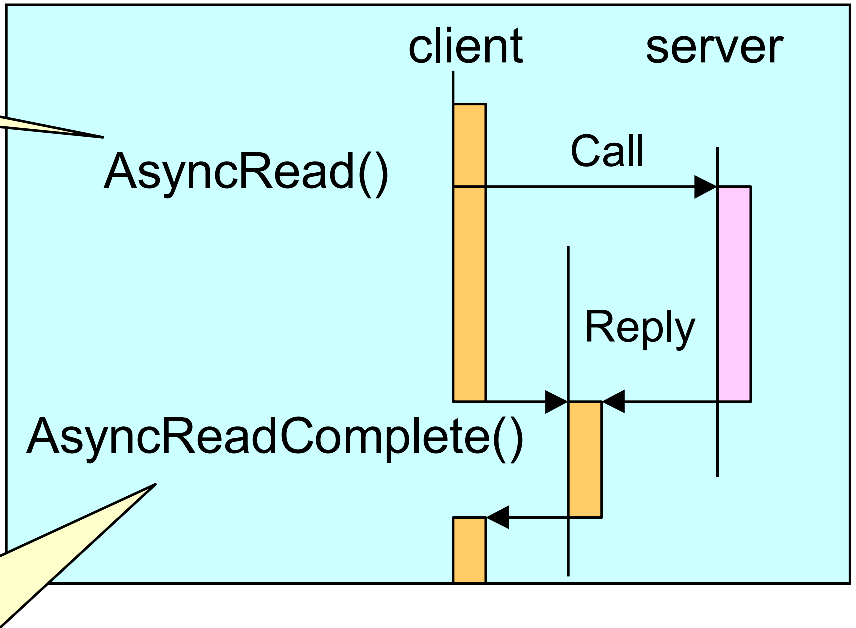
OPC DA: Synchronous Read of a group

```
Dim thisGroup As OPCGroup
Dim cntItems As Integer
Dim source As Integer
Dim serverHandles(2) As Long
Dim values() As Variant
Dim errors() As Long

serverHandles(1) =  ServerHandle(11) '
serverHandles(2) =  ServerHandle(14)

source = OPCcache                   ' could also be OPCDevice
thisGroup.SyncRead
→ source,
→ nrItems,
→ serverHandles,                    ' identifies the items to be read !
← values,                           ' returns be a dynamic array
← errors                            ' returns a dynamic array

For cntItems = LBound(serverHandles) To UBound(serverHandles) ' 1..n
    MsgBox CStr(cntItems) & " : " & values(cntItems)
Next cntItems
```



OPC DA: Asynchronous read of single Items

```
    Dim WithEvents MyGroup
    ...
    MyGroup.AsyncRead
 →     nrItems,
 →     ServerHandles,
 ←     ServerErrors,
 →     TransactionID,
 →     CancelID
```



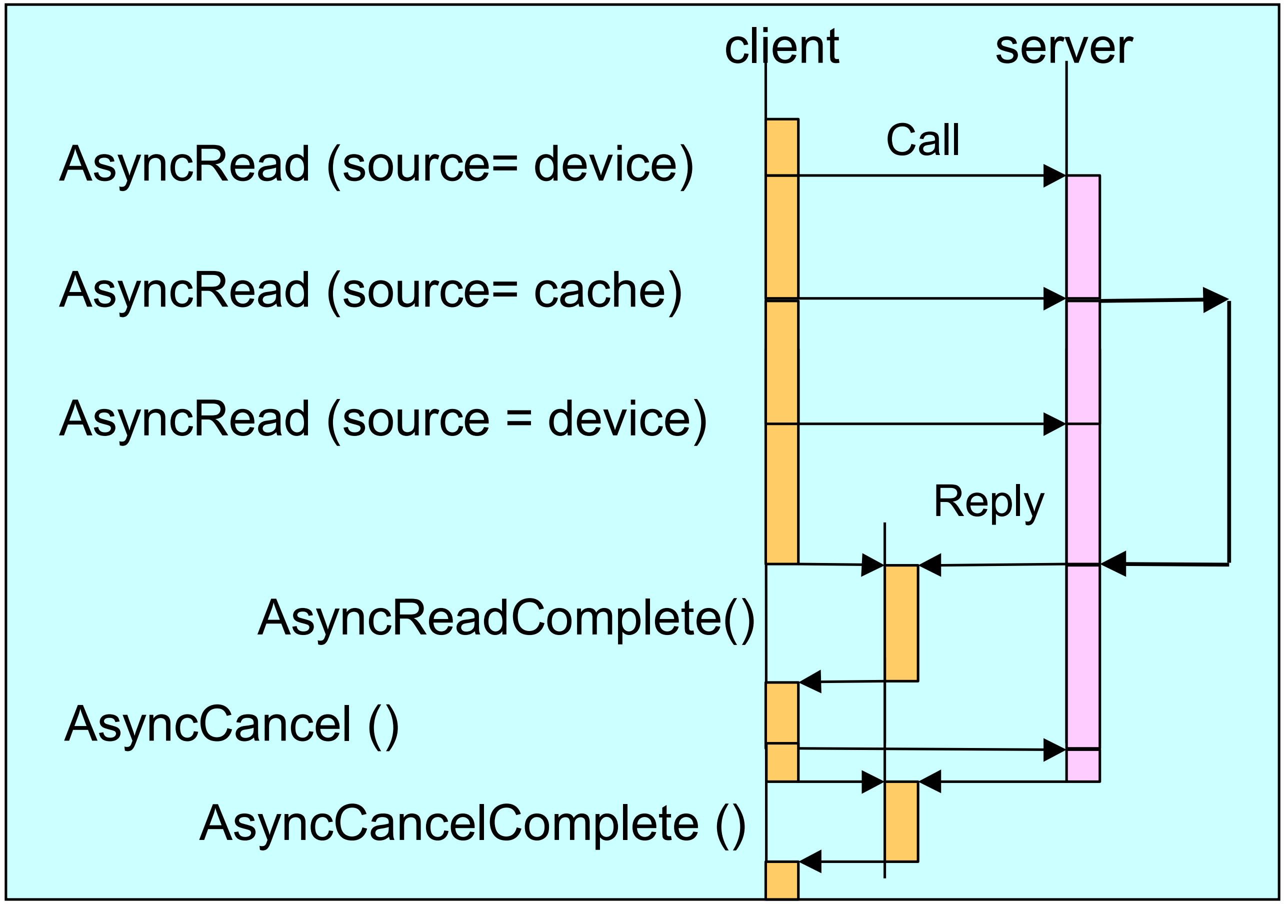
```
  Private Sub Mygroup_AsyncReadComplete (
←   ByVal TransactionID As Long,
←   ByVal NumItems As Long,
←   ClientHandles() As Long,
←   ItemValues() As Variant,
←   Qualities() As Long,
←   TimeStamps() As Date,
←   Errors() As Long)

    MsgBox ("Async Read Complete")
End Sub
```

Asynchronous read separates Call and Reply.

Call supplies the ServerHandles

Reply returns the corresponding ClientHandles

OPC DA: Transaction ID


client
server
Call
AsyncRead (source= device)
AsyncRead (source= cache)
AsyncRead (source = device)
Reply
AsyncReadComplete()
AsyncCancel ()
AsyncCancelComplete ()

Although the AsynchReadComplete carries the ClientHandle of each item,
it does not tell which AsynchRead caused the AsynchReadComplete event to fire.

Call and Reply are linked by the TransactionID: this ID is returned in AsynchReadComplete
It can also be used to cancel the operation

OPC DA: Reading (by events) the OPC group

```
Dim WithEvents MyGroup
...
Private Sub MyGroup_DataChange( _
    ByVal TransactionID As Long, _
    ByVal NrItems As Long, _
    ClientHandles() As Long, _            ' returned by the server to the client
    ItemValues() As Variant, _
    Qualities() As Long, _
    TimeStamps() As Date)

    Dim cntItems As Integer
    For cntItems = LBound(ClientHandles) To UBound(ClientHandles)     ' index 1..n
        TextValue(cntItems - 1).Text     = ItemValues(cntItems)       ' display
        TextTimeStamp(cntItems - 1).Text = DateAdd("h", 9, TimeStamps(cntItems))
        TextQuality(cntItems - 1).Text   = Qualities(cntItems)
    Next cntItems
End Sub
```

This function is called each time an item in the group changes
The ClientHandles (here: 5 and 6) identifies the variables, not the “fully qualified itemID”
The values are displayed in the TextValue, TextTimeStamp and TextQuality fields.
The refresh rate is given in the group definition.

OPC DA: Groups Events

Although transmission by groups is more efficient than AsyncRead, it can be improved by using Groups Events (Global Data Change)

This event is fired whenever a variable of a group changes.

If the group is subscribed also to a Group Event (DataChange), I.e. if the group is declared WithEvents, then both Events will be fired.

The application must sort out the groups and the items.

OPC DA: GlobalDataChange

```
Dim WithEvents MyGroups As OPCGroups
...
Private Sub MyGroups_GlobalDataChange(
    ByVal TransactionID As Long,     ' =0 if called by Refresh
    ByVal GroupHandle As Long,
    ByVal NumItems As Long,
    ClientHandles() As Long,        ' identifies the items
    ItemValues() As Variant,        ' value of the items
    Qualities() As Long,            ' value of the items
    TimeStamps() As Date)           ' timestamps of the items

Select Case GroupHandle             ' depending on the group ...
    Case 1
       ' treat group 1
    Case 2
       ' treat group 2
```

The GlobalDataChange event is fired when any item in a group changed.
(if Groups is also with events, the corresponding Group_DataChange will also be called)

OPC DA: Server Events

```
Dim WithEvents MyServer As OPCServer            ' define the event
.. ..
Private Sub MyServer_ServerShutDown(ByVal Reason As String)
       MsgBox "my OPC Server " & MyServer.ServerName & " quit"
End Sub
```

This event signals to the client that the server shut down.

The client must declare its server „WithEvents“ and provide the corresponding event Subroutine

This should stop all actions, otherwise exceptions will occur.

OPC DA: Do not forget cleanup !

To speed up connection/disconnection, an OPC server remembers its groups and clients when a client disconnects.

To do this, an OPC server initialises its structures with a client counter of 2, instead of 1. Therefore, it is imperative to shut down explicitly the server, otherwise links will subside (and you will have kill the server to clear them).

```
Private Sub ServerShutdown
    Dim dummyServer As OPCServer
    Dim Servers As Variant                              ' this is an array of strings
    Dim cntServers As Integer

    Set myGroup1 = Nothing                              ' create a dummy server object
    Set myGroups = Nothing                              ' returns all available servers
    MyServer.Remove
    MyServer.RemoveAllGroups
    MyServer.Disconnect                             ' delete this object (was created by New)
    Set MyServer = Nothing
```

OPC DA: Standard and components

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- Objects, types and properties
- Communication model
- Simple programming example
- **Standard and components**

OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

OPC DA: Libraries

The OPC DA specification is not formal, conformance can hardly be checked against this document.

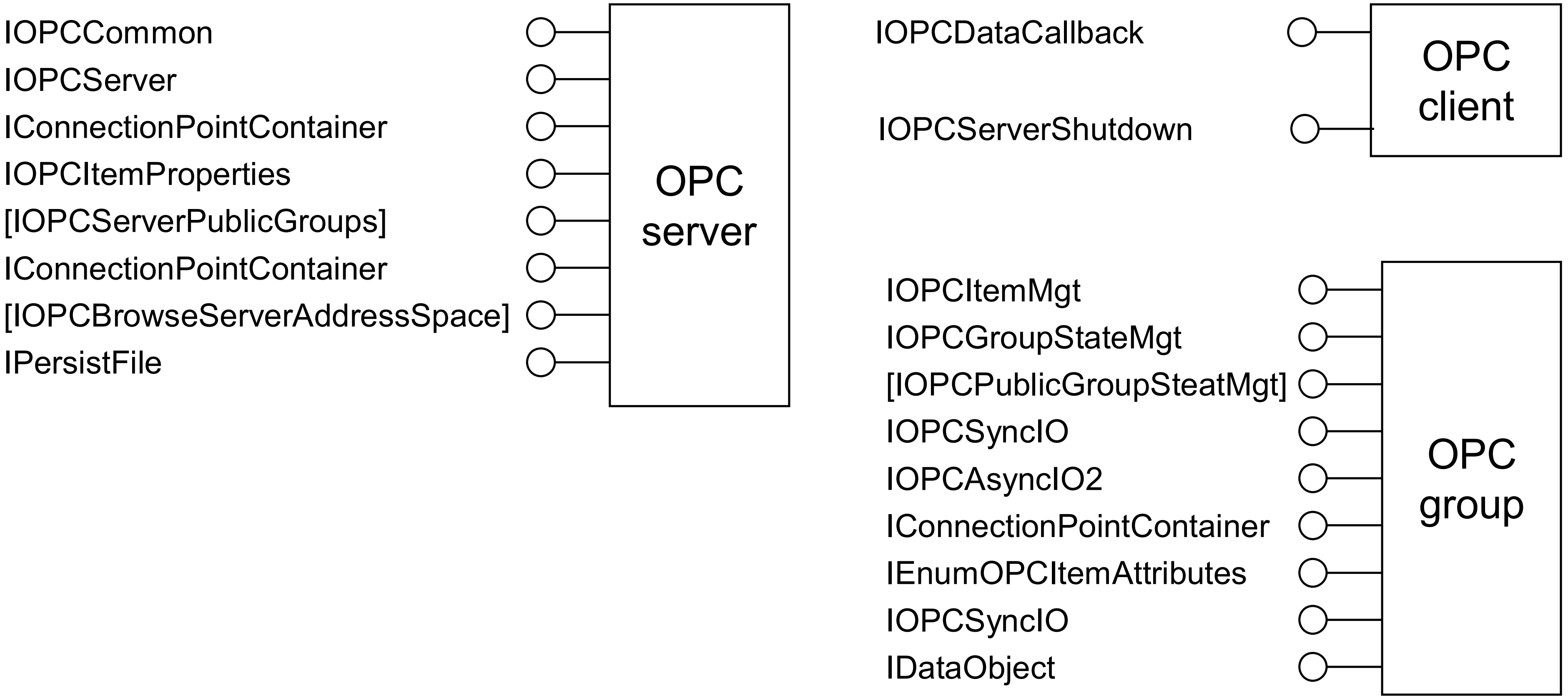
To ensure that the standard is observed, the OPC foundation distributes on its website the DLLs (opcdaauto.dll, opccomn_ps,…) that contain the type libraries to access the OPC server.

The vendors are not compelled to implement all features. For instance, the description of the variables is seldom used. Calling unimplemented functions causes exceptions that must be caught in Visual Basic with "On Error …" statements.

There exist three versions of DA, 1.0, 2.0 and 3.0, that behave differently, however, older servers do not have a property indicating which version they support.

OPC DA: Custom Interface

While the Automation Interface is easy to use and quite powerful, some OPC functions are missing and special operations can only be done in Visual C++ using the custom COM interface.
This is only recommended for experienced programmers.



OPC DAOPCGroup Custom Interface: comparison (1)

This checklist for experienced programmers (custom interface) shows the differences between the DA versions

Required Interfaces	**DA 1.0**	**DA 2.0**	**DA 3.0**
OPCGroup			
IUnknown	Required	Required	Required
IOPCItemMgt	Required	Required	Required
IOPCGroupStateMgt	Required	Required	Required
IOPCGroupStateMgt2	N/A	N/A	Required
IOPCPublicGroupStateMgt	Optional	Optional	N/A
IOPCSyncIO	Required	Required	Required
IOPCSyncIO2	N/A	N/A	Required
IOPCAsyncIO2	N/A Required	Required	
IOPCAsyncIO3	N/A	N/A	Required
IOPCItemDeadbandMgt	N/A	N/A	Required
IOPCItemSamplingMgt	N/A	N/A	Optional
IConnectionPointContainer	N/A	Required	Required
IOPCAsyncIO	Required	Optional	N/A
IDataObject	Required	Optional	N/A

OPC DA OPCServer 1.0, 2.0 & 3.0 comparison (2)

This checklist for experienced programmers (custom interface) shows the differences between the DA versions

Required Interfaces	**1.0**	**2.0**	**3.0**
OPCServer			
IOPCServer	Required	Required	Required
IOPCCommon	N/A Required	Required	
IConnectionPointContainer	N/A	Required	Required
IOPCItemProperties	N/A Required	N/A	
IOPCBrowse	N/A	N/A	Required
IOPCServerPublicGroups	Optional	Optional	N/A
IOPCBrowseServerAddressSpace	Optional	Optional	N/A
IOPCItemIO	N/A	N/A	Required

The differences do not yet appear in the automation interface

OPC DA: Assessment

What is OPC ?

Which are the read and write operations ?

Is communication done by items, by groups or by collection of groups ?

What is the difference between cache and device reading ?

Can a change of an OPC variable be notified as an event, or shall the client poll ?

How is browsing done ?

Why is browsing necessary, even when one knows the variable's location in the server ?

To probe further….

OPC Foundation:
  Specifications http://www.opcfoundation.org

SoftwareToolbox
  Examples in Visual Basic
  http://www.softwaretoolbox.com/Tech_Support/TechExpertiseCenter/OPC/opc.html

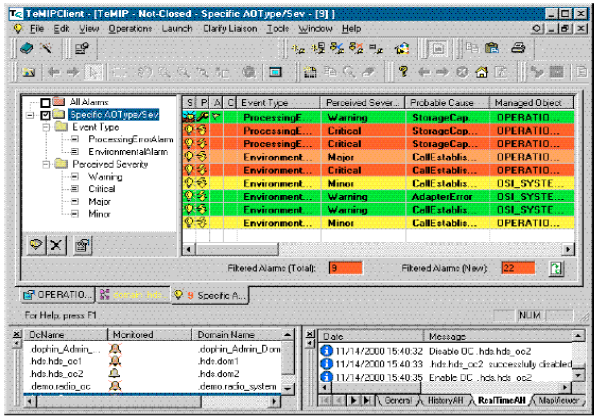
The Code Project
  OPC and .NET
  http://www.codeproject.com/useritems/opcdotnet.asp

Matrikon
  Free client and server:
  http://www.matrikon.com

WinTech
  Toolkit for an OPC server
  http://www.win-tech.com/html/opcstk.htm

NewAge Automation
  Toolkit for an OPC server
  http://www.newageautomation.com

Industrial Automation
Automation Industrielle
Industrielle Automation


TeMIPClient - [TeMIP - Not-Closed - Specific AOType/Sev - [9]]
File Edit View Operations Launch Clarify Liaison Tools Window Help

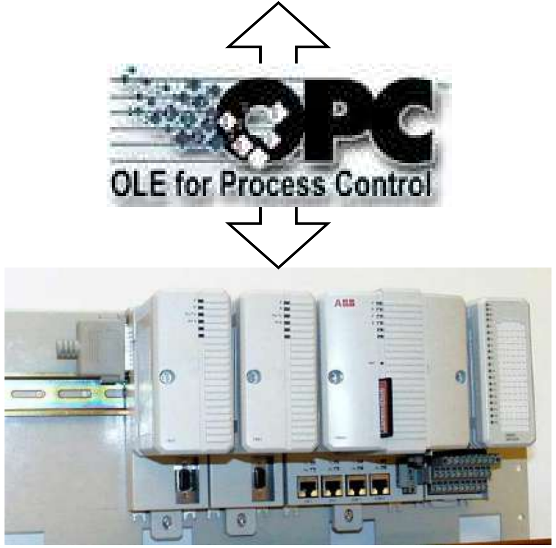
- □ All Alarms
- ☑ Specific AOType/Sev
  - Event Type
    - ProcessingErrorAlarm
    - EnvironmentalAlarm
  - Perceived Severity
    - Warning
    - Critical
    - Major
    - Minor

S	P	A	C	Event Type	Perceived Sever...	Probable Cause	Managed Object
				ProcessingE...	Warning	StorageCap...	OPERATIO...
				ProcessingE...	Critical	StorageCap...	OPERATIO...
				ProcessingE...	Critical	StorageCap...	OPERATIO...
				Environment...	Major	CallEstablis...	OPERATIO...
				Environment...	Critical	CallEstablis...	OPERATIO...
				Environment...	Minor	CallEstablis...	OSI_SYSTE...
				Environment...	Warning	AdapterError	OSI_SYSTE...
				Environment...	Warning	CallEstablis...	OSI_SYSTE...
				Environment...	Minor	CallEstablis...	OPERATIO...

Filtered Alarms (Total): 9 Filtered Alarms (New): 22

OcName	Monitored	Domain Name
.dophin_Admin_...		.dophin_Admin_Dom
.hds.hds_oc1		.hds.dom1
.hds.hds_oc2		.hds.dom2
.demo.radio_oc		.demo.radio_system

Date	Message
11/14/2000 15:40:32	Disable OC .hds.hds_oc2
11/14/2000 15:40:33	.hds.hds_oc2 successfuly disabled
11/14/2000 15:40:35	Enable OC .hds.hds_oc2



4 Access to devices

4.3 OLE for Process Control (OPC)

4.3.3 Alarms & Events

Prof. Dr. H. Kirrmann
ABB Research Centre, Baden, Switzerland

AE: Overview

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- Objects, types and properties
- Communication model
- Simple Programming Example
- Standard and components

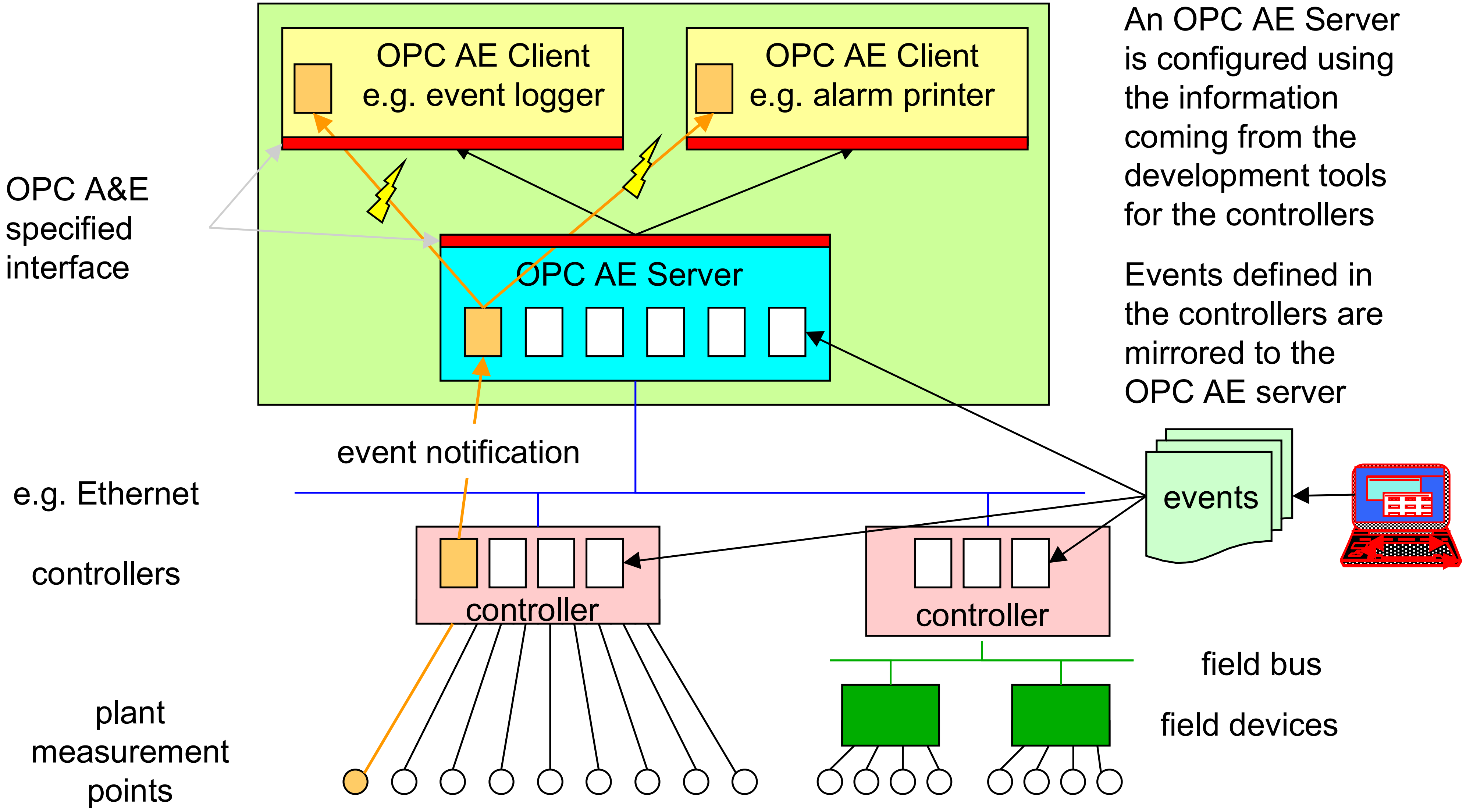
OPC Alarms and Events Specification

- **Overview: definitions and objects**
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

AE: Configuration



An OPC AE Server is configured using the information coming from the development tools for the controllers

Events defined in the controllers are mirrored to the OPC AE server

AE: Purpose

The controllers (PLC) generate events in response to changes in the plant variables. together with their precise time of occurrence, type, severity and associated message for the human operator.

An OPC Event server registers these events and makes them available to several clients A particular class of events are the alarms, which are detailed events that may require acknowledgement.

The OPC Alarms & Events Interface gives access to the OPC Event server, allowing to:

- browse the OPC A&E Server for predefined events.
- enable or disable alarms and events
- subscribe to alarms and events of interest
- receive the event and alarm notifications with the associated attributes
- acknowledge alarms

AE: Definitions

An event is a general change of state that is relevant to the OPC server.
An event signal a change:
1) in the field device ("production started")
2) in the OPC server ("alarm acknowledged")
3) in the application ("operator action")

An alarm is a state of the process that requires attention and is relevant to the OPC server.
An alarm is represented by an alarm condition, (or short: condition), a state machine indicating if the alarm has been enabled, triggered or acknowledged.

An event or an alarm does not transmit analogue process values,
but they transmit information about their origin, the time of their occurrence and a message intended for a human operator.

Alarms and events may not get lost
(contrarily to OPC DA, which does not guarantee completeness)

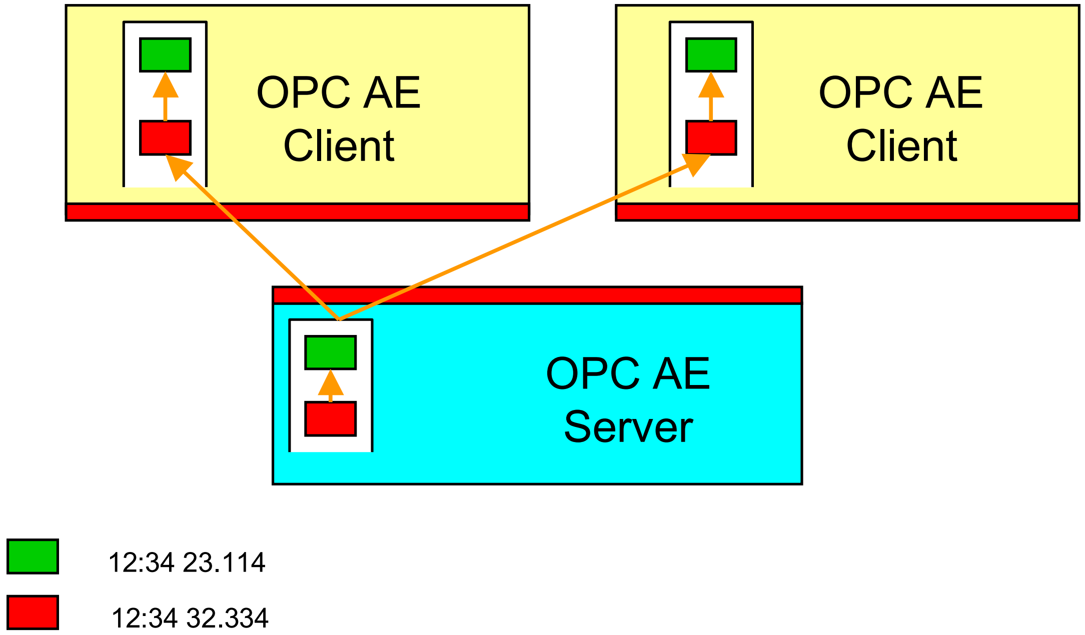
Alarms and event are precisely time-stamped by their source,
(contrarily to process variables, which are time-stamped by the receiving OPC server).

AE: communication paradigm

OPC AE works according to the “message passing” paradigm, contrarily to OPC DA, that works according to the "shared memory" paradigm.

This means that an event is kept in a queue until all clients have read it (or timed out).

The server guarantees that different clients will see all events in the same sequence.

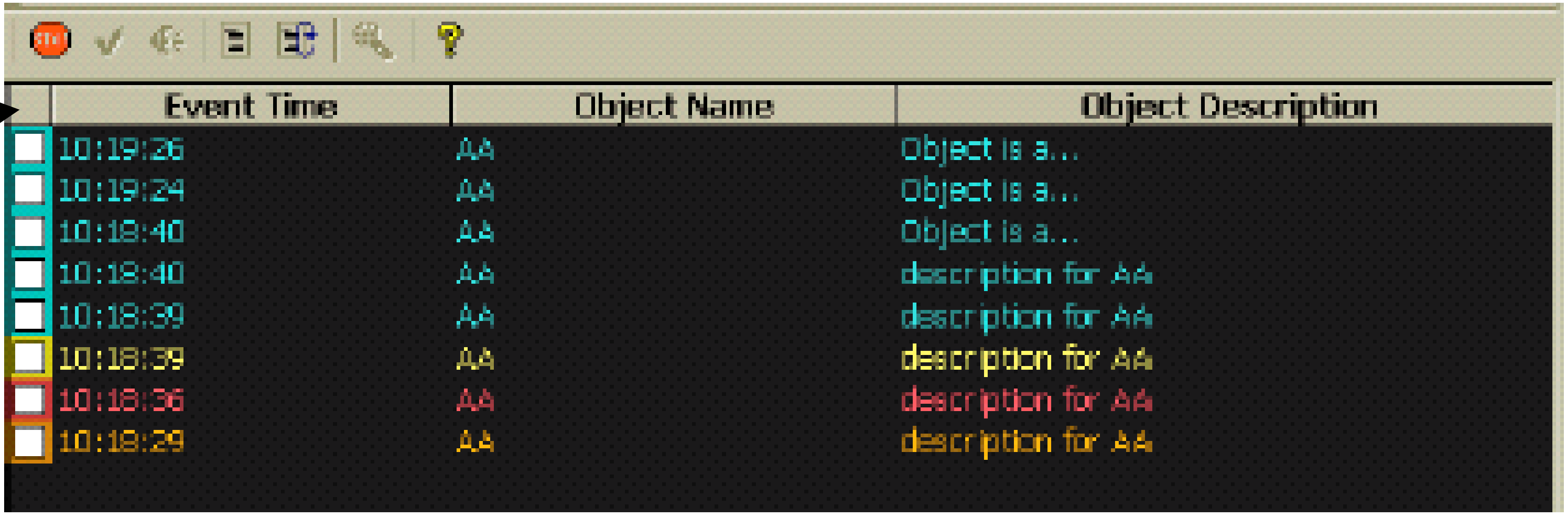


AE: Displaying Alarms and Events

Alarms and events are usually displayed differently on an operator screen.

- Events are displayed in an event list that can become quite long (typically 1000 entries),
  entries are not cleared when the source of the event returns to normal

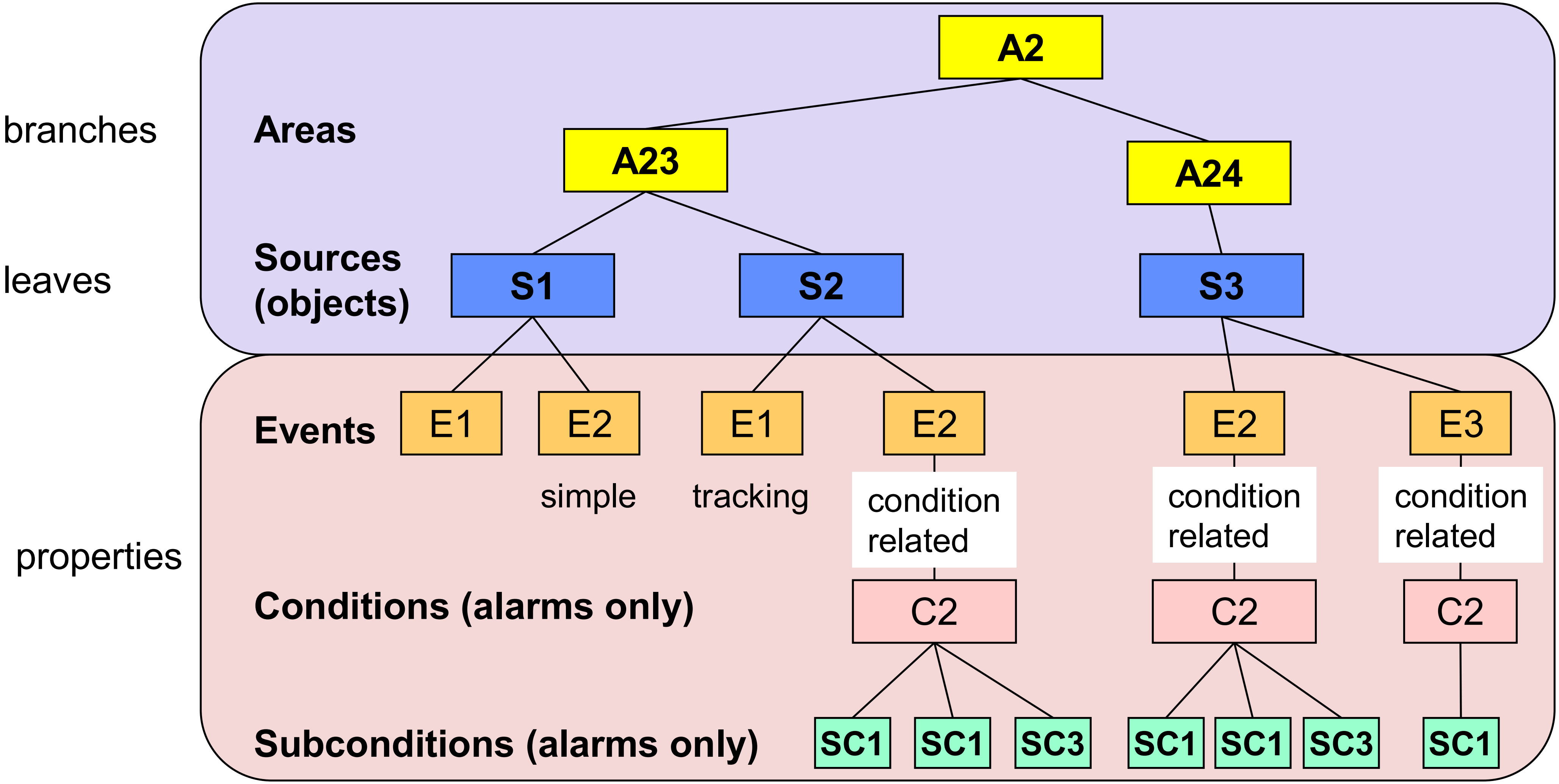
- Alarms are displayed in a short list (typically 50 alarms)
  appearance changes when the alarm is acknowledged,
  an alarm line is cleared when the alarm signal is cleared.



Ack checkbox

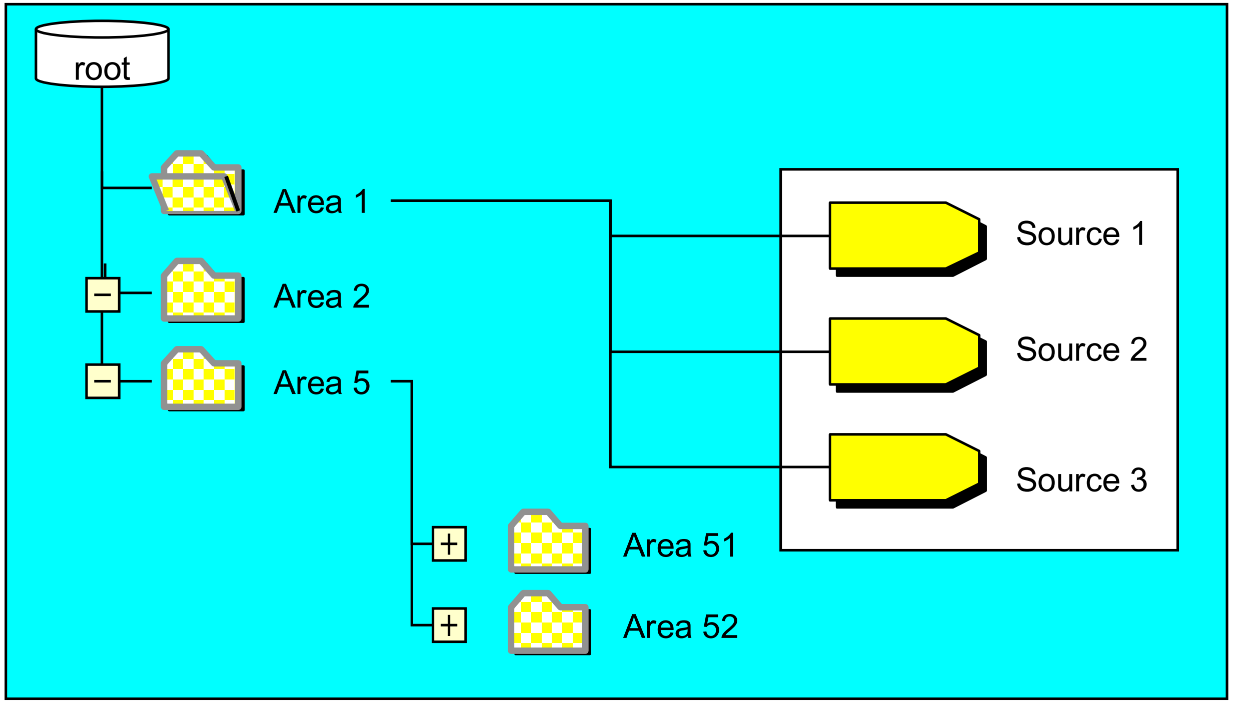
	Event Time	Object Name	Object Description
□	10:19:26	AA	Object is a...
□	10:19:24	AA	Object is a...
□	10:18:40	AA	Object is a...
□	10:18:40	AA	description for AA
□	10:18:39	AA	description for AA
□	10:18:39	AA	description for AA
□	10:18:36	AA	description for AA
□	10:18:29	AA	description for AA

AE: Server Organization



An event is identified by a source (owner object in the controller) and an event name
this combination must be unique in the AE Server.

AE: Browsing the AE Server



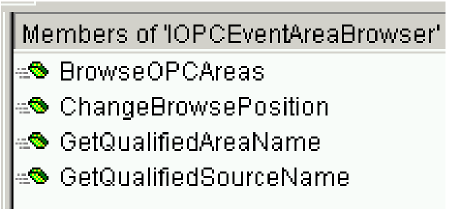
Alarms and Event are organized by area, which themselves may contain other areas.

Contrarily to branches in OPC DA, area and sources have properties that allow to disable or enable events or alarms by area or by source, corresponding to parts of the plants, rooms or specific equipment of the plant.

AE: Browsing methods

Like all other OPC Servers, an OPC A&E presents an interface that allows the client to browse the server to explore its structure, with the methods:

```
BrowseOPCArea
ChangeBrowsePosition (up, down, root)

GetQualifiedAreaName
GetQualifiedSourceName
```



Members of 'IOPCEventAreaBrowser'
- BrowseOPCAreas
- ChangeBrowsePosition
- GetQualifiedAreaName
- GetQualifiedSourceName

There is no "GetQualifiedItemID, since the condition name is known from the source.

AE: Events

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- Objects, types and properties
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification

- Overview: definitions and objects
- **Events**
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification

- Overview

AE: Events kinds

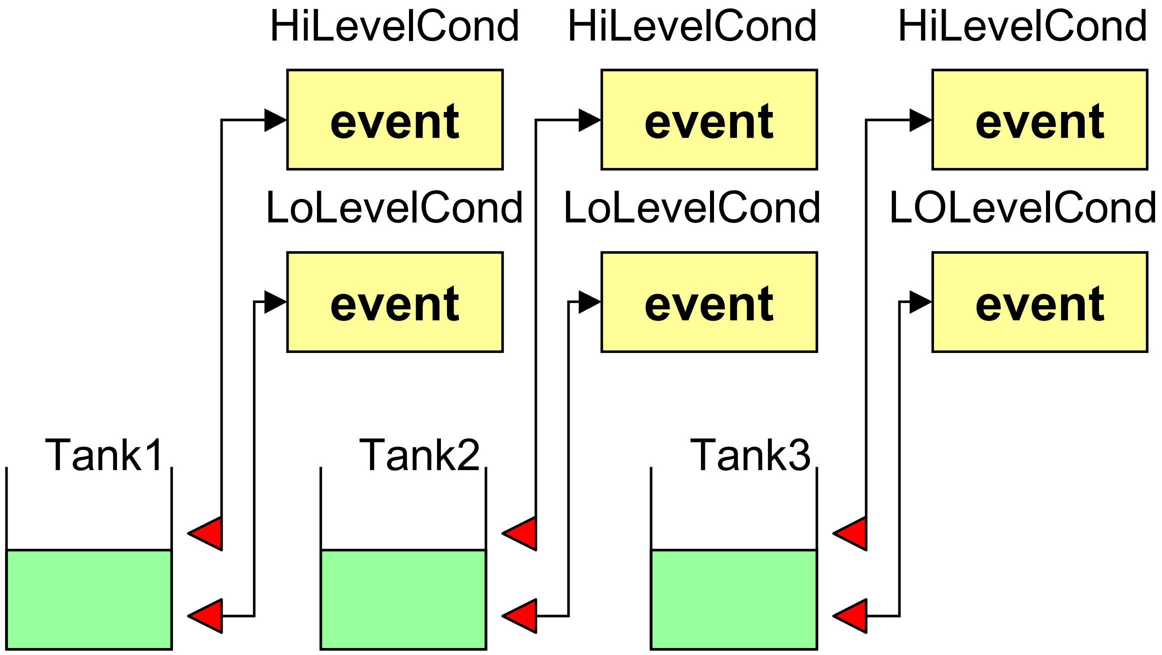
OPC AE defines three kinds of events:

- simple: process control system related events (change of a boolean variable)
- condition-related: notifies a change of an alarm condition (CLEARED, ACKNOWLEDGED), (see later)
- tracking-related: origin outside of the process (e.g. operator intervention)

AE: Event- identification

An event is identified by

- its source (the object that generates the event. e.g. Tank1) and
- the event name (which can be the same as in another object, e.g. HiLevelCond)



AE: Event PLC Function block

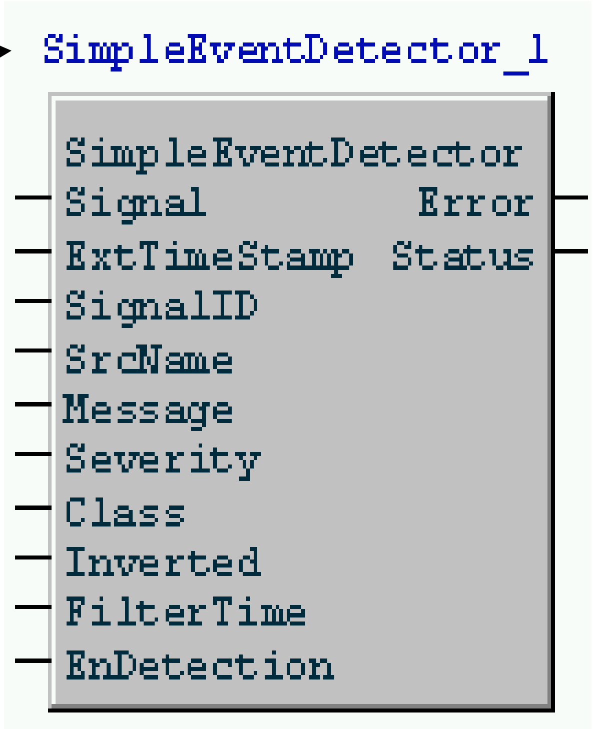
Simple Event function blocks in a controller are used to signal a simple event.

The event is identified by the concatenation of the name of the containing object (SrcName) and the event handling function block name (here: SimpleEventDetector_1).

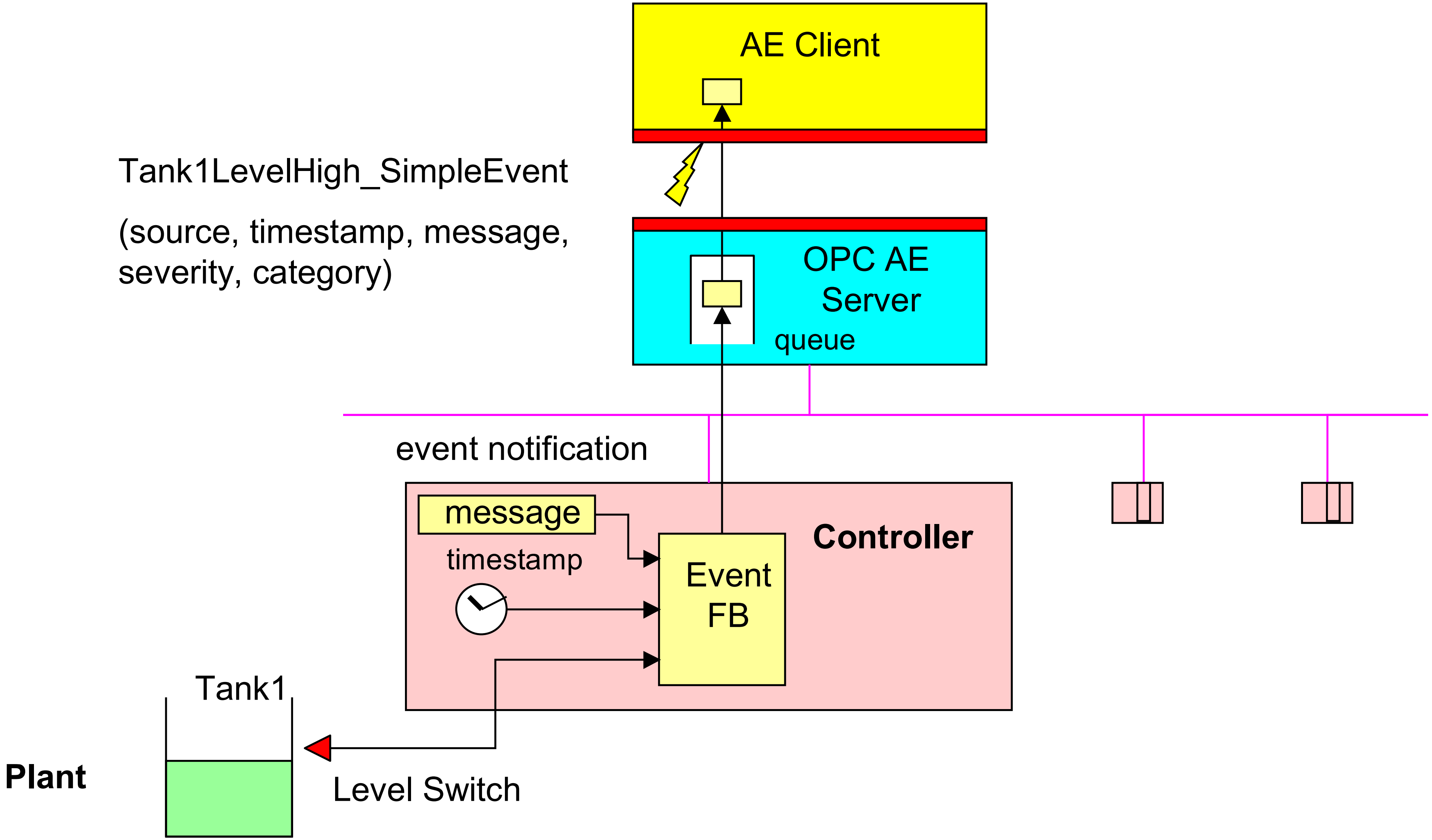
The source name can be that of the containing code module (owner object), assuming that a plant object is represented by a code module.

name of the **event**

event signal (boolean expression)
is an external signal to be used ?
signal name for external signal (20 characters)
name of the **source** (30 characters)
message (60 characters)



AE: Events - Notification



AE: Events - Time Stamp

There are three places where events can be time-stamped:

- at the device that originally produced the data (external event - low-level event)
  allowing Sequence-Of-Events with a high resolution, down to microseconds

- at the controller, (internal event) using the controller's clock to time-stamp messages
  giving precision not greater than the period of the tasks, about 1 ms.

- at the OPC Server, when an event message arrives (tracking events)
  not more precise than DA, about 10 ms)

The OPC server can be configured to register the time stamp at the instant of the event transition (positive or negative) and the instant of the acknowledgement.

AE: Properties of an Event-object

	Property	Meaning
all events	Source	source object (area + source)
	Time	time of occurrence
	Message	associated message for the operator
	EventCategory	user-defined
	Severity	priority (1..1000)
	OPCEventAttribute	
condition -related events	ConditionName	name of the condition within the source
	SubCondition	name of the active subcondition (subconditions are exclusive)
	ChangeActiveState	
	ChangeAckState	
	ChangeEnableState	
	ChangeQuality	
	ChangeSeverity	
	ChangeSubCondition	
	ChangeMessage	
	ChangeAttribute	
	ConditionAction	
	ConditionAcknowleddged	
	Quality	
	AckRequired	
	ActiveTime	
	Cookie	server handle used for acknowledgement of alarms
	ActorID	identified who acknowledged the alarm (for client-side acknowledgement)

AE: Alarm conditions

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- Objects, types and properties
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- **Alarm Conditions**
- Automation Interface

OPC Historical Data Specification

- Overview

AE: Alarms - Condition Definition

An (alarm) condition is a named state machine that describes the state of an alarm

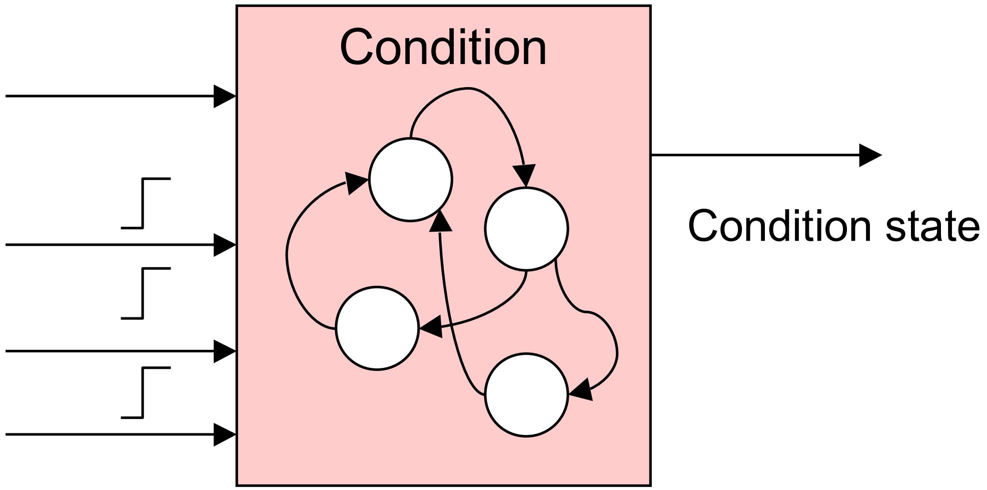
The condition state is defined by three variables:

- Enabled: the condition is allowed to send event notifications
- Active: the alarm signal is true
- Acknowledged: the alarm has been acknowledged

Alarm signal
(e.g. FIC101.PV > 100 AND FIC101.PV < 150)

Acknowledgement signal
(a positive transition of a boolean variable)

Enable (positive transition)
Disable (positive transition)



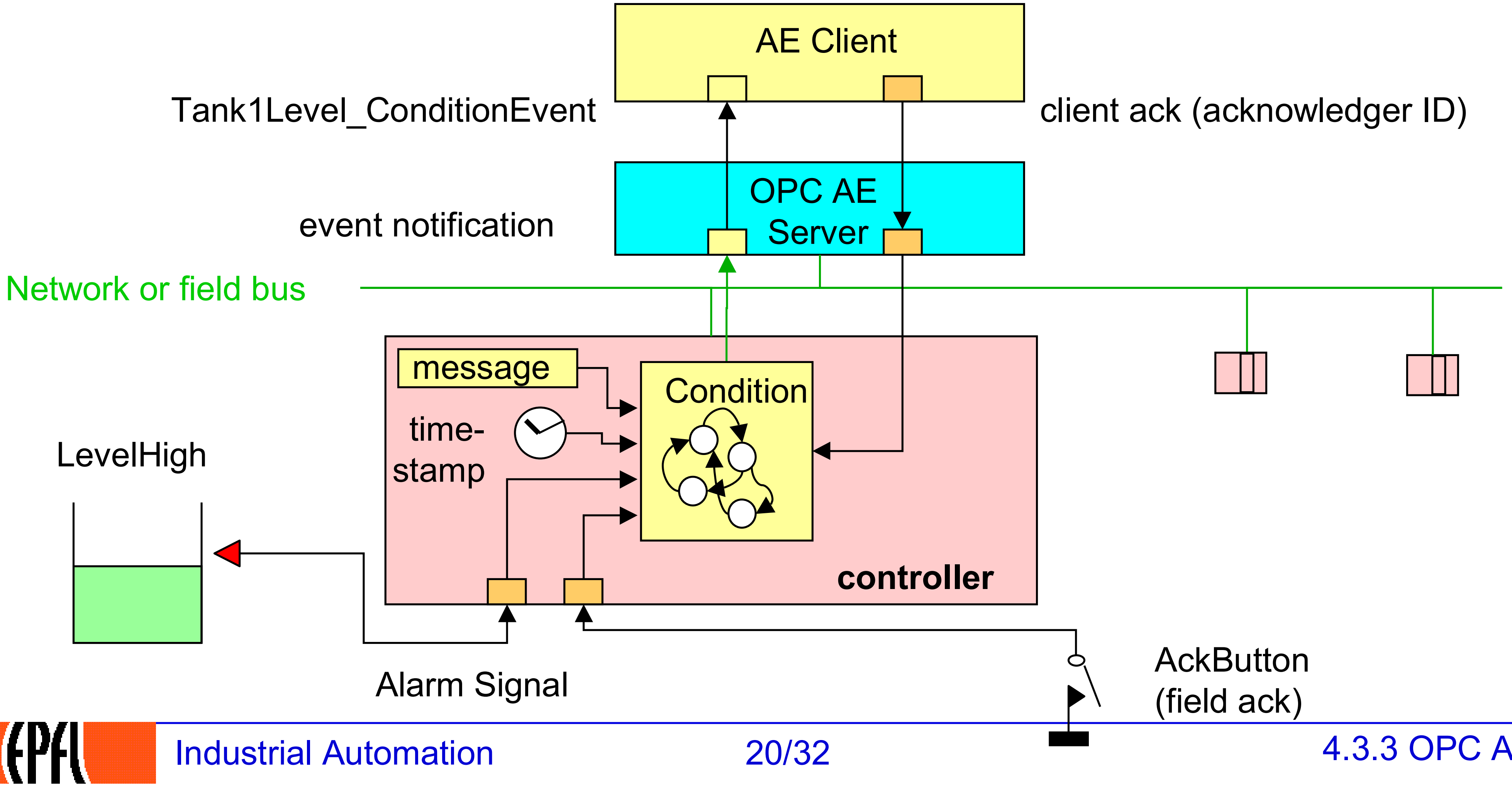
Condition

Condition state

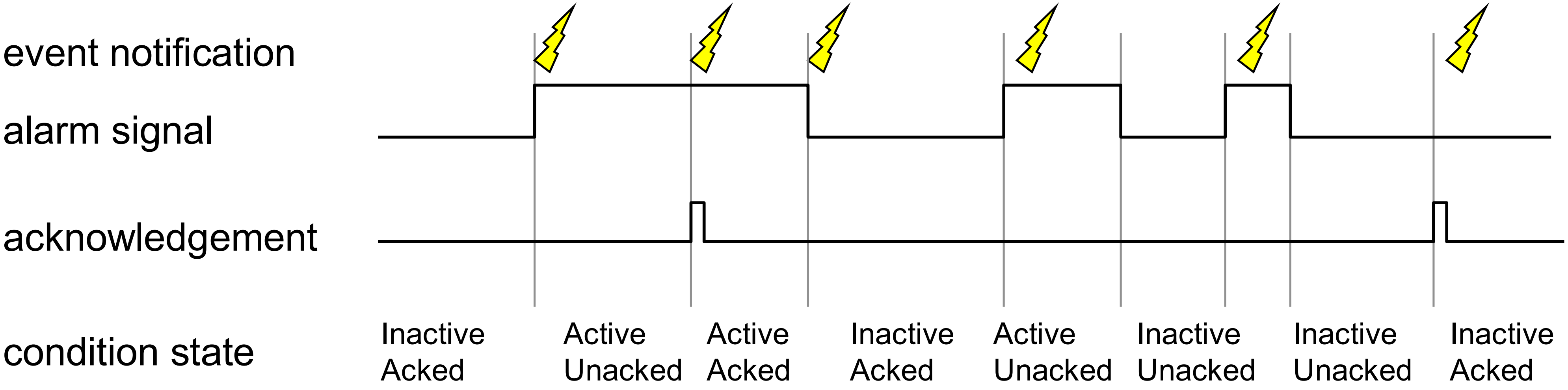
AE: Alarms - Acknowledgement

An alarm condition becomes active when the PLC produces an alarm signal describing an abnormal state (e.g. the level of the tank is too high).

The operator is expected to acknowledge this condition (client ack)
Alternatively, a local operator may press a button that the PLC reads (field ack)

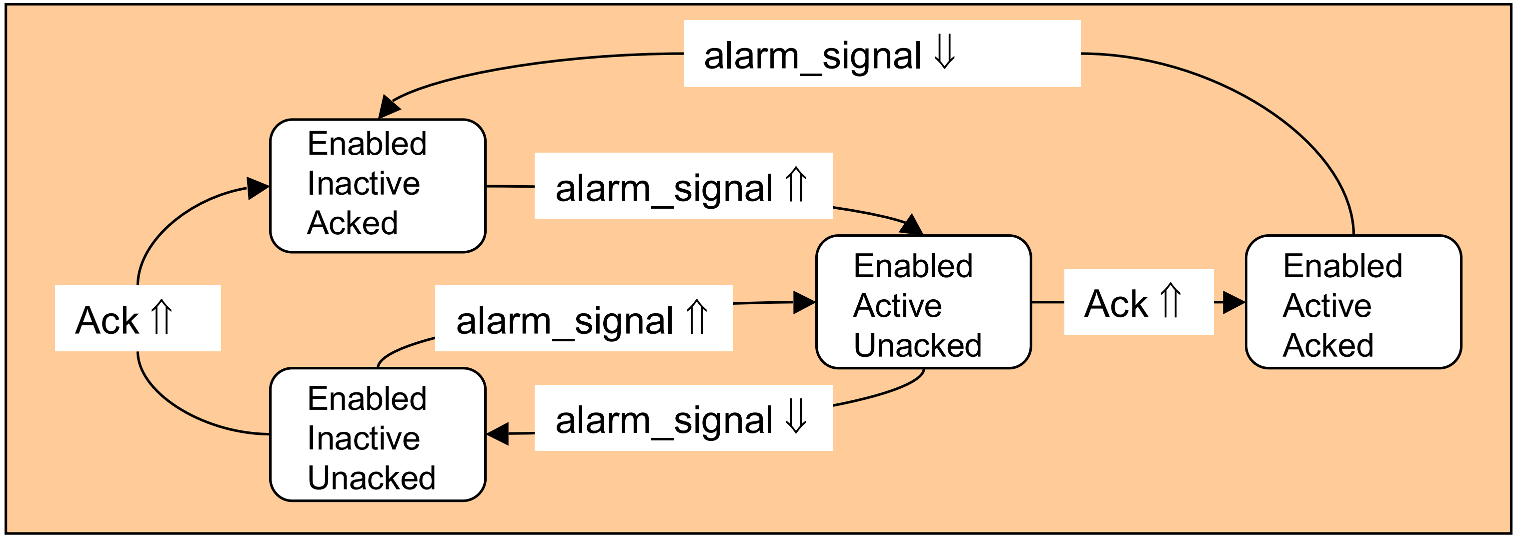


AE: Alarms - Condition states and acknowledgement



condition state transitions

(here: always enabled)



An event is generated each time the alarm signal changes state, or is acknowledged

AE: Alarms - Condition properties

Name	Name, unique within Server, assigned to the condition
Active	alarm expression is in the state represented by the condition
ActiveSubCondition	If condition active, name of SubCondition (see later)
Quality	quality of data upon which condition is computed
Enabled	condition may become active
Acked	alarm has been acknowledged
LastAckTime	last time that alarm was acknowledged
SubCondLastActive	last time that subcondition became active (see later)
CondLastActive	last time that condition became active
LastInactive	last time that condition became inactive
AcknowledgerID	who acknowledged the alarm
Comment	

AE: Alarms - Subconditions

A condition may be subdivided into mutually exclusive subconditions

This allows to signal an alarm identified by the object name and give details in the subcondition.

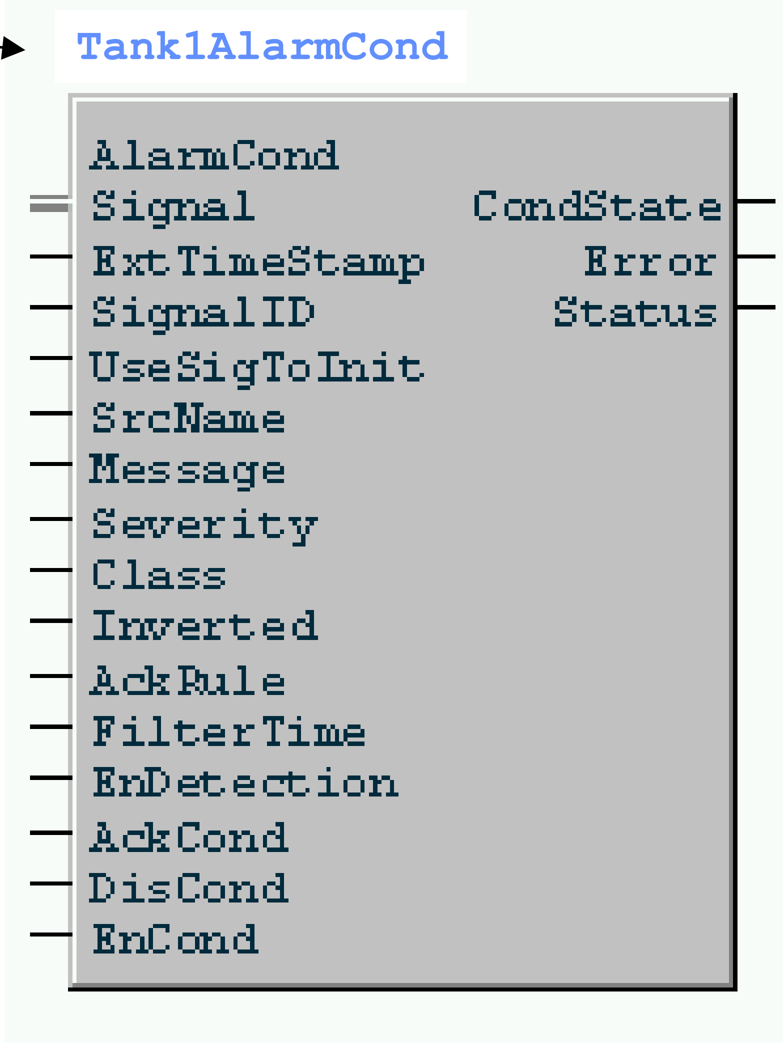
(for instance: “level high”, “level very high”, “overflow”)

Name	Name, unique within the condition, assigned to the sub-condition
Definition	An expression that defines the sub-state
Severity	priority (different subconditions may have different severity levels)
Description	Text string to be included in the event notification

An alarm condition has at least one subcondition, that defines the severity.

AE: Alarms : Example of Function Block (AC800)

name of the **condition**



alarm signal (boolean expression)

is an external signal to be used ?

external signal name (20 characters)

name of the **source** (30 characters)

message (60 characters)

(= Priority, 1..1000)

User defined (1..9999)

invert signal

acknowledgement method

shortest condition considered (0..3600)

enable detection (state)

field **acknowledgement** (positive edge)

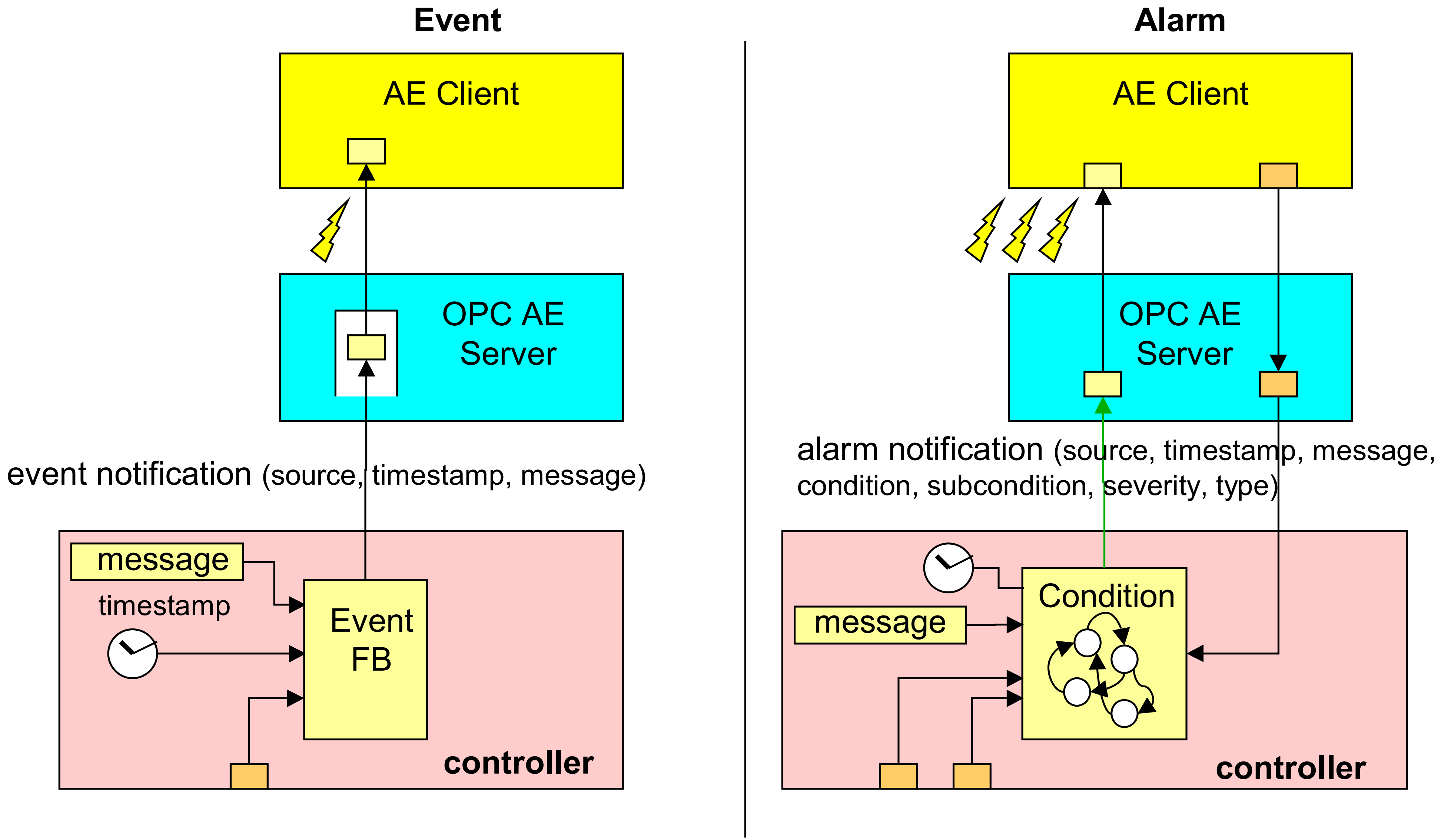
disable condition (positive edge)

enable condition (positive edge)

active, acked,..

This function block has only one subcondition

AE: Summary alarms and events



event notification (source, timestamp, message)

alarm notification (source, timestamp, message, condition, subcondition, severity, type)

AE: Automation Interface

OPC Common

- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access

- Overview: browsing the server
- Objects, types and properties
- Communication model
- Simple Programming Example
- Standard and components

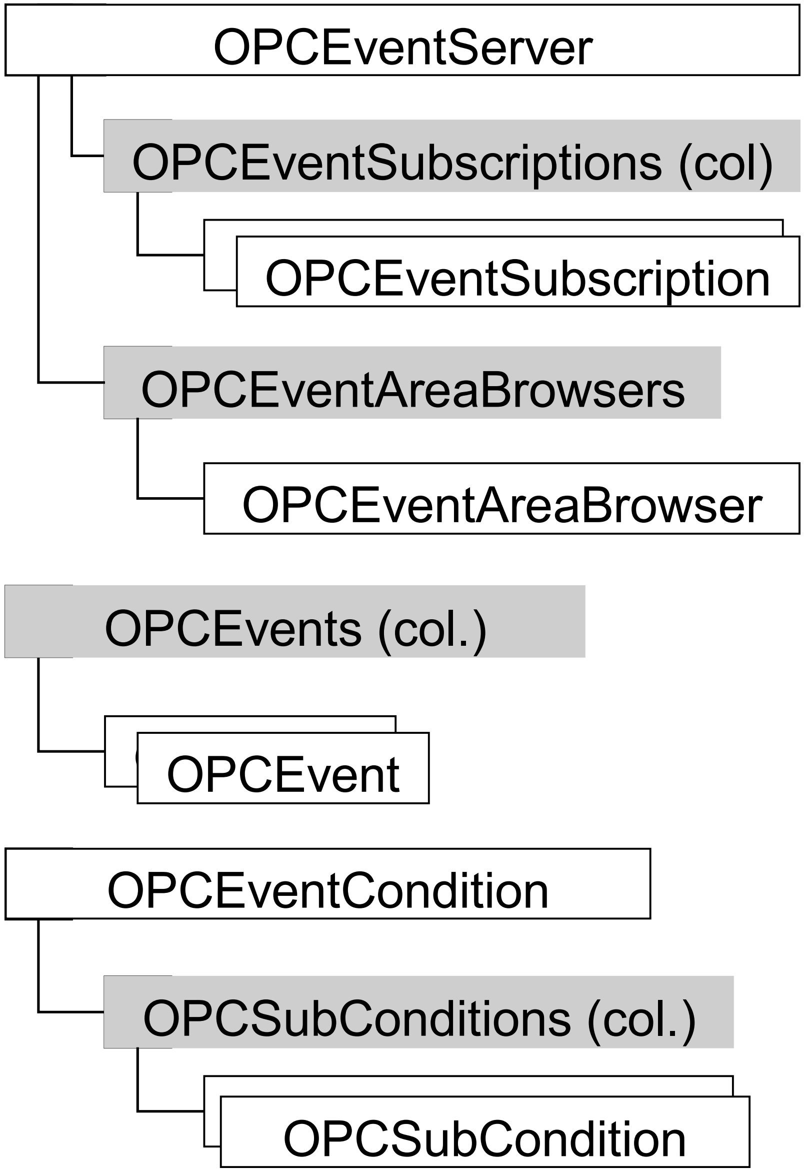
OPC Alarms and Events Specification

- Overview: definitions and objects
- Events
- Alarm Conditions
- **Automation Interface**

OPC Historical Data Specification

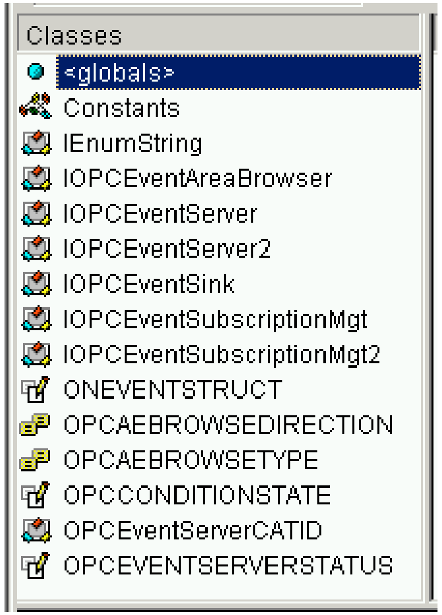
- Overview

AE: Object hierarchy



OPCEventServer	An instance of an OPC AE Server.
OPCEventSubscriptions (col)	A collection containing all OPCEventSubscription objects this client has created
OPCEventSubscription	An object that maintains state information and provides the mechanisms for events and alarms notification
OPCEventAreaBrowsers	A collection of browsers for the server (only one instance of an OPCBrowser object per instance of an OPCServer object.)
OPCEventAreaBrowser	An object that browses items in the server’s configuration. It accesses the arrays of OPCAreas and OPCAreaSources
OPCEvents (col.)	A collection that holds the OPCEvents objects. When the Automation Wrapper receives a callback from the AE Server, it forwards the response as an OPCEvents collection object.
OPCEvent	An object that represents one specific event of a subscription
OPCEventCondition	An object that holds the current state of a condition instance, identified by its Source and Condition Name
OPCSubConditions (col.)	A collection that holds the subconditions associated with the event condition
OPCSubCondition	represents one subcondition associated with the event condition

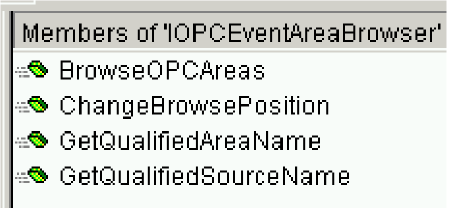
AE: Automation Interface (Summary 1/2)



Classes

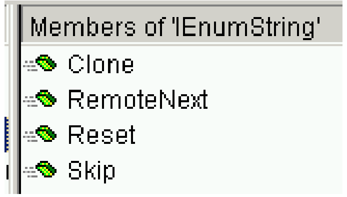
- <globals>
- Constants
- IEnumString
- IOPCEventAreaBrowser
- IOPCEventServer
- IOPCEventServer2
- IOPCEventSink
- IOPCEventSubscriptionMgt
- IOPCEventSubscriptionMgt2
- ONEVENTSTRUCT
- OPCAEBROWSEDIRECTION
- OPCAEBROWSETYPE
- OPCCONDITIONSTATE
- OPCEventServerCATID
- OPCEVENTSERVERSTATUS

Methods



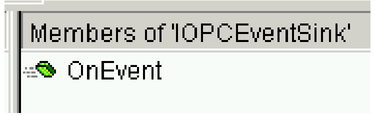
Members of 'IOPCEventAreaBrowser'

- BrowseOPCAreas
- ChangeBrowsePosition
- GetQualifiedAreaName
- GetQualifiedSourceName



Members of 'IEnumString'

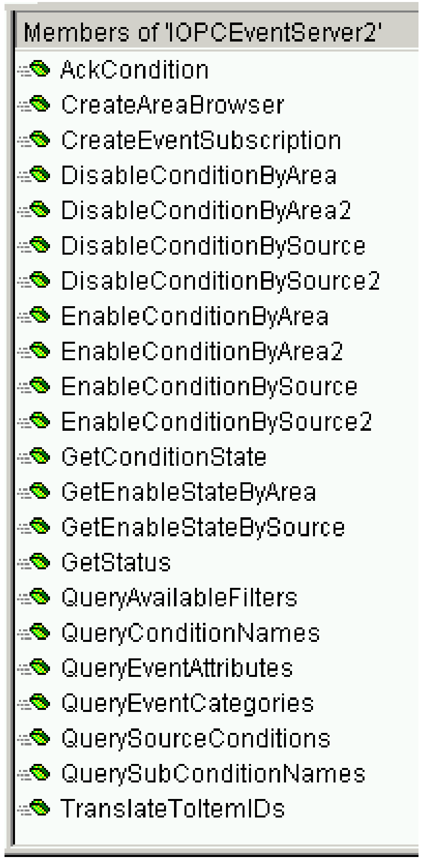
- Clone
- RemoteNext
- Reset
- Skip



Members of 'IOPCEventSink'

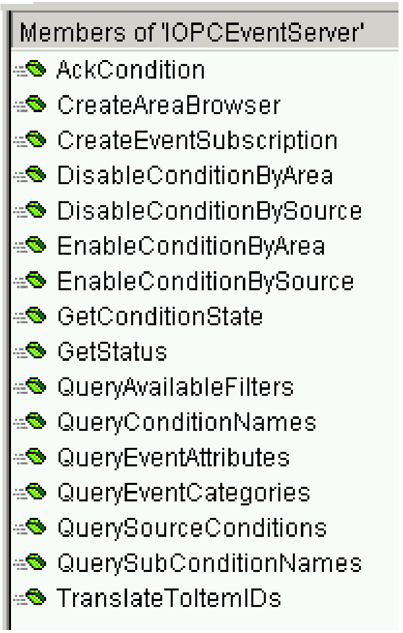
- OnEvent

Event Server



Members of 'IOPCEventServer2'

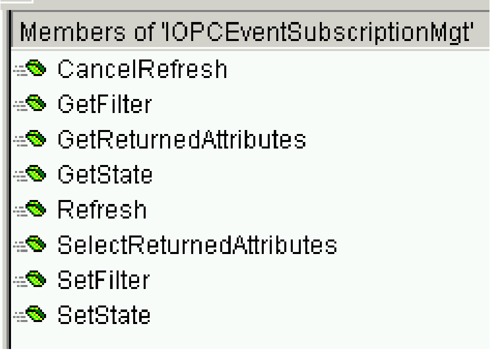
- AckCondition
- CreateAreaBrowser
- CreateEventSubscription
- DisableConditionByArea
- DisableConditionByArea2
- DisableConditionBySource
- DisableConditionBySource2
- EnableConditionByArea
- EnableConditionByArea2
- EnableConditionBySource
- EnableConditionBySource2
- GetConditionState
- GetEnableStateByArea
- GetEnableStateBySource
- GetStatus
- QueryAvailableFilters
- QueryConditionNames
- QueryEventAttributes
- QueryEventCategories
- QuerySourceConditions
- QuerySubConditionNames
- TranslateToItemIDs



Members of 'IOPCEventServer'

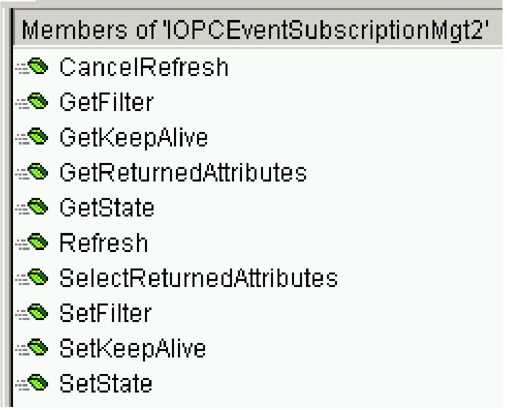
- AckCondition
- CreateAreaBrowser
- CreateEventSubscription
- DisableConditionByArea
- DisableConditionBySource
- EnableConditionByArea
- EnableConditionBySource
- GetConditionState
- GetStatus
- QueryAvailableFilters
- QueryConditionNames
- QueryEventAttributes
- QueryEventCategories
- QuerySourceConditions
- QuerySubConditionNames
- TranslateToItemIDs

Event Subscription Mgt



Members of 'IOPCEventSubscriptionMgt'

- CancelRefresh
- GetFilter
- GetReturnedAttributes
- GetState
- Refresh
- SelectReturnedAttributes
- SetFilter
- SetState

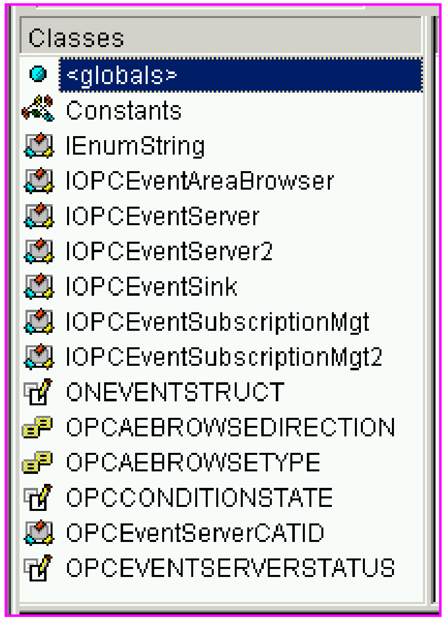


Members of 'IOPCEventSubscriptionMgt2'

- CancelRefresh
- GetFilter
- GetKeepAlive
- GetReturnedAttributes
- GetState
- Refresh
- SelectReturnedAttributes
- SetFilter
- SetKeepAlive
- SetState

AE: Automation Interface (Summary 2/2)

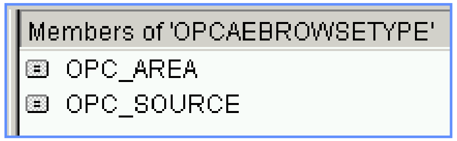
Classes



Classes

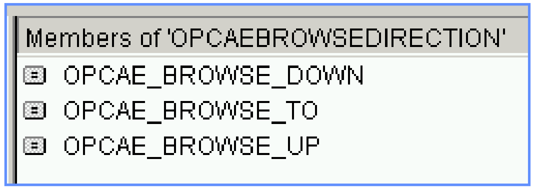
- <globals>
- Constants
- IEnumString
- IOPCEventAreaBrowser
- IOPCEventServer
- IOPCEventServer2
- IOPCEventSink
- IOPCEventSubscriptionMgt
- IOPCEventSubscriptionMgt2
- ONEVENTSTRUCT
- OPCAEBROWSEDIRECTION
- OPCAEBROWSETYPE
- OPCCONDITIONSTATE
- OPCEventServerCATID
- OPCEVENTSERVERSTATUS

Enums



Members of 'OPCAEBROWSETYPE'

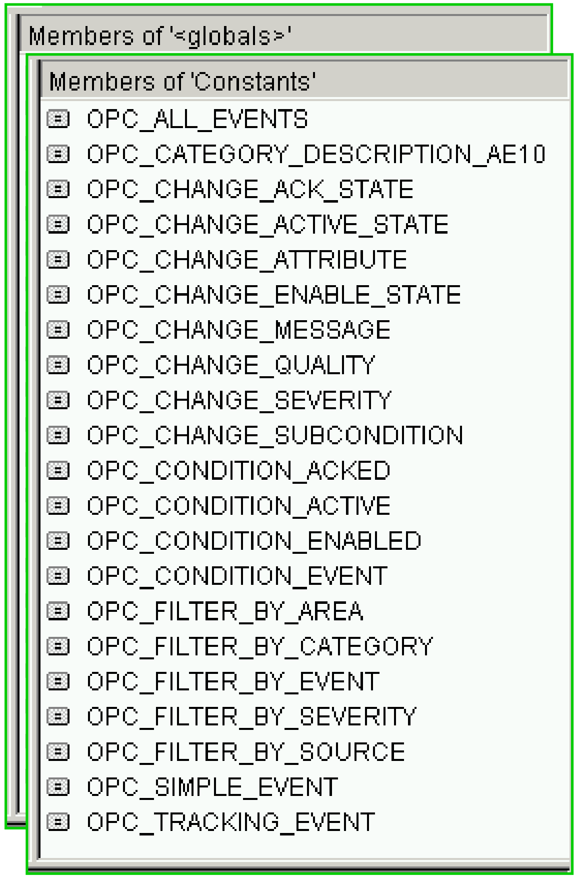
- OPC_AREA
- OPC_SOURCE



Members of 'OPCAEBROWSEDIRECTION'

- OPCAE_BROWSE_DOWN
- OPCAE_BROWSE_TO
- OPCAE_BROWSE_UP

Constants

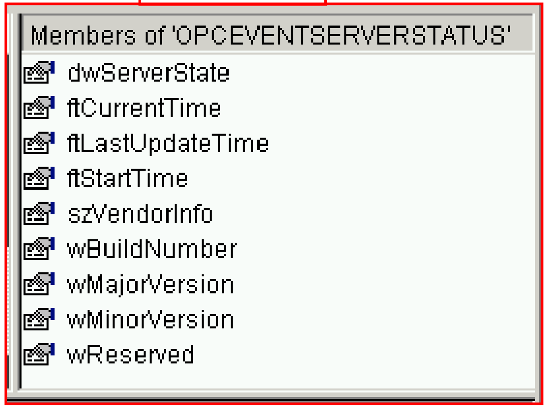


Members of '<globals>'

Members of 'Constants'

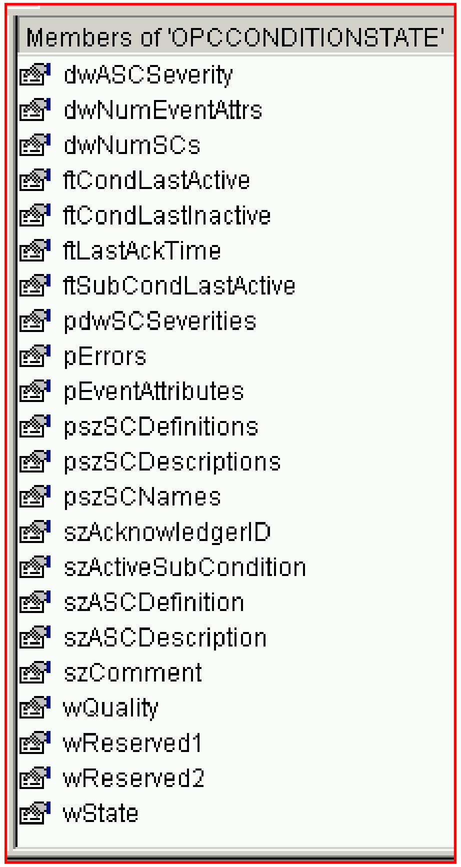
- OPC_ALL_EVENTS
- OPC_CATEGORY_DESCRIPTION_AE10
- OPC_CHANGE_ACK_STATE
- OPC_CHANGE_ACTIVE_STATE
- OPC_CHANGE_ATTRIBUTE
- OPC_CHANGE_ENABLE_STATE
- OPC_CHANGE_MESSAGE
- OPC_CHANGE_QUALITY
- OPC_CHANGE_SEVERITY
- OPC_CHANGE_SUBCONDITION
- OPC_CONDITION_ACKED
- OPC_CONDITION_ACTIVE
- OPC_CONDITION_ENABLED
- OPC_CONDITION_EVENT
- OPC_FILTER_BY_AREA
- OPC_FILTER_BY_CATEGORY
- OPC_FILTER_BY_EVENT
- OPC_FILTER_BY_SEVERITY
- OPC_FILTER_BY_SOURCE
- OPC_SIMPLE_EVENT
- OPC_TRACKING_EVENT

Types



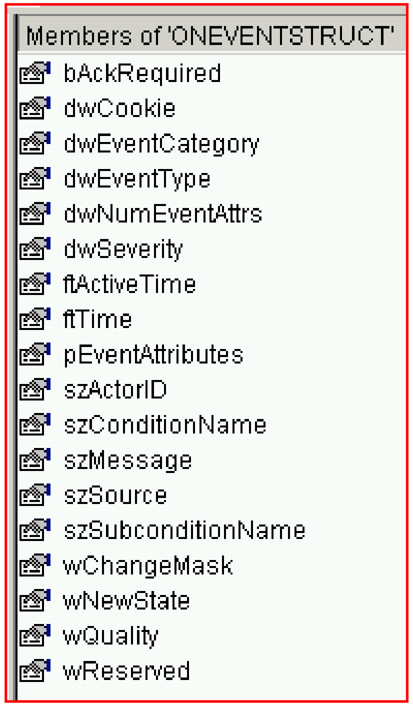
Members of 'OPCEVENTSERVERSTATUS'

- dwServerState
- ftCurrentTime
- ftLastUpdateTime
- ftStartTime
- szVendorInfo
- wBuildNumber
- wMajorVersion
- wMinorVersion
- wReserved



Members of 'OPCCONDITIONSTATE'

- dwASCSeverity
- dwNumEventAttrs
- dwNumSCs
- ftCondLastActive
- ftCondLastInactive
- ftLastAckTime
- ftSubCondLastActive
- pdwSCSeverities
- pErrors
- pEventAttributes
- pszSCDefinitions
- pszSCDescriptions
- pszSCNames
- szAcknowledgerID
- szActiveSubCondition
- szASCDefinition
- szASCDescription
- szComment
- wQuality
- wReserved1
- wReserved2
- wState



Members of 'ONEVENTSTRUCT'

- bAckRequired
- dwCookie
- dwEventCategory
- dwEventType
- dwNumEventAttrs
- dwSeverity
- ftActiveTime
- ftTime
- pEventAttributes
- szActorID
- szConditionName
- szMessage
- szSource
- szSubconditionName
- wChangeMask
- wNewState
- wQuality
- wReserved

To probe further….

OPC Foundation:
  Specifications http://www.opcfoundation.org

SoftwareToolbox
  Examples in Visual Basic
  http://www.softwaretoolbox.com/Tech_Support/TechExpertiseCenter/OPC/opc.html

The Code Project
  OPC and .NET
  http://www.codeproject.com/useritems/opcdotnet.asp

Matrikon
  Free client and server:
  http://www.matrikon.com

WinTech
  Toolkit for an OPC server
  http://www.win-tech.com/html/opcstk.htm

NewAge Automation
  Toolkit for an OPC server
  http://www.newageautomation.com