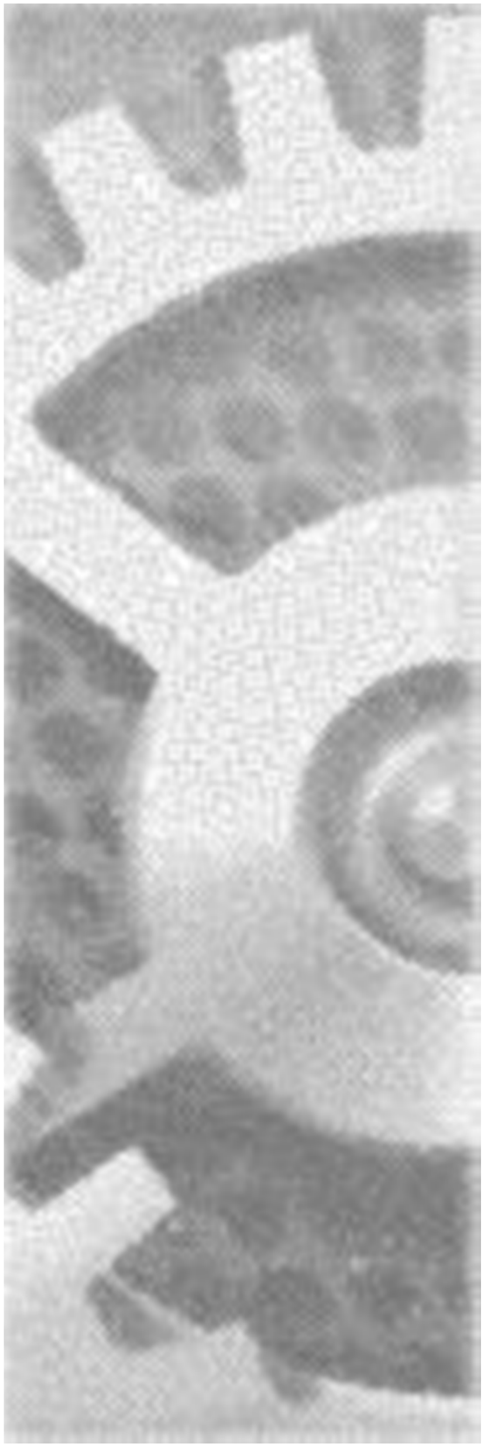
OPC UA usage within ACTL

DESY, CTA
David Melkumyan
SLOW-Telescopes Meeting

April
2016



Agenda

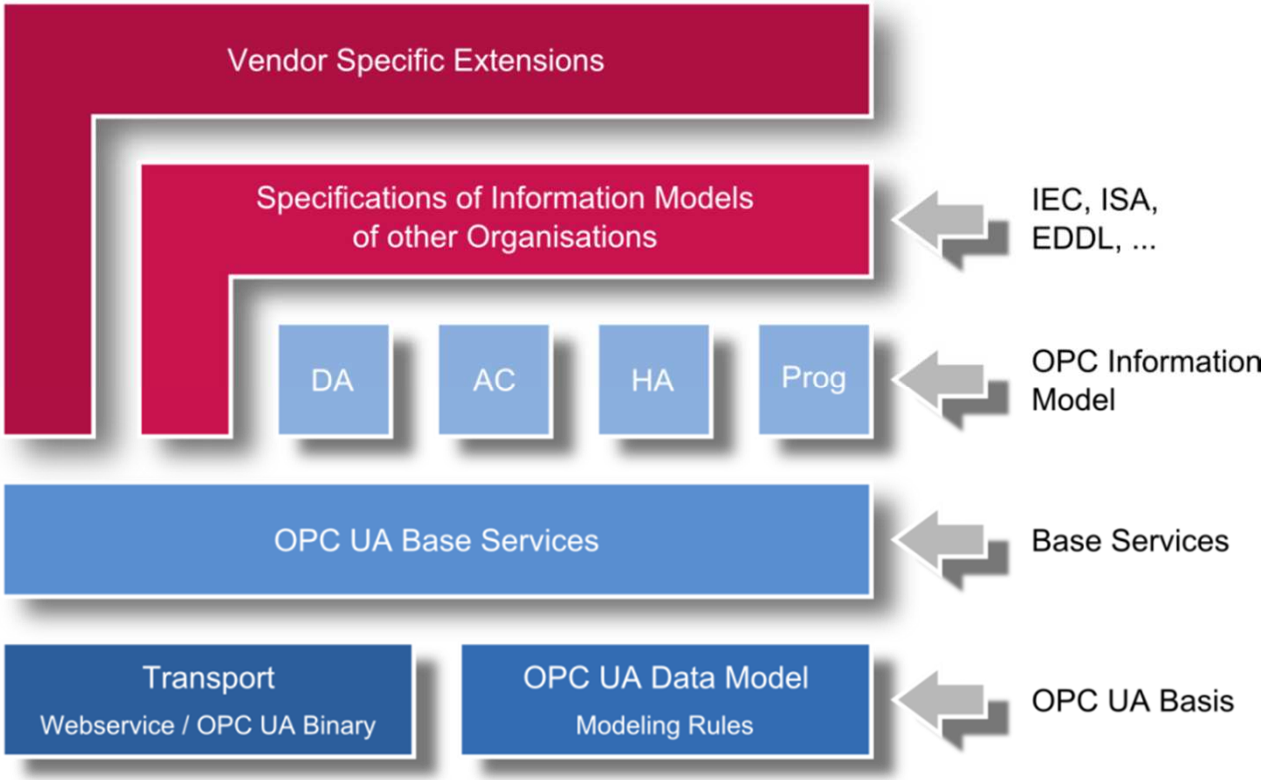
- Introduction to OPC UA
- OPC UA SDKs
- OPC UA Generic clients
- OPC UA Data Access for ACS
- OPC UA for device integration into ACS
- Examples for OPC UA Server and ACS bridge
- Summary

Introduction to OPC UA

- OPC (Openness, Productivity and Collaboration; formerly "OLE for Process Control") Unified Architecture (UA) is industrial communication protocol for interoperability developed by the OPC Foundation.
- OPC UA was accepted as International Electrotechnical Commission (IEC) standard 62541 in 2011.

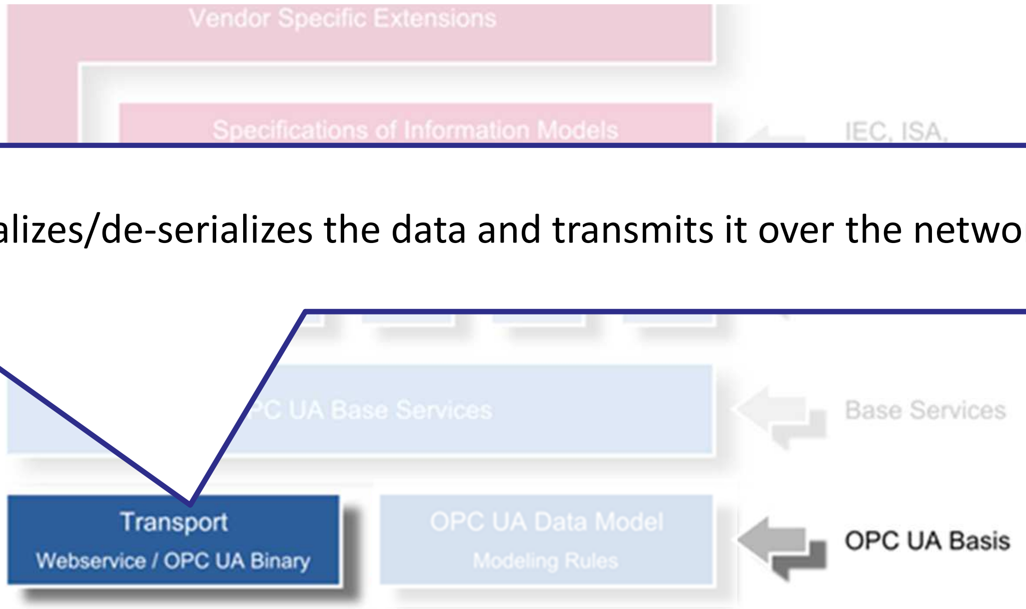
Introduction to OPC UA

- The OPC UA is a Service Orientated Architecture (SOA) and is based on different logical levels.



Introduction to OPC UA

- OPC (Openness, Productivity and Collaboration; formerly "OLE for Process Control") Unified Architecture (UA) is industrial communication protocol for interoperability developed by the OPC Foundation.
- OPC UA was accepted as International Electrotechnical Commission (IEC) standard 62541 in 2011.
- The OPC UA is a Service Orientated Architecture (SOA) and is based on different logical levels.

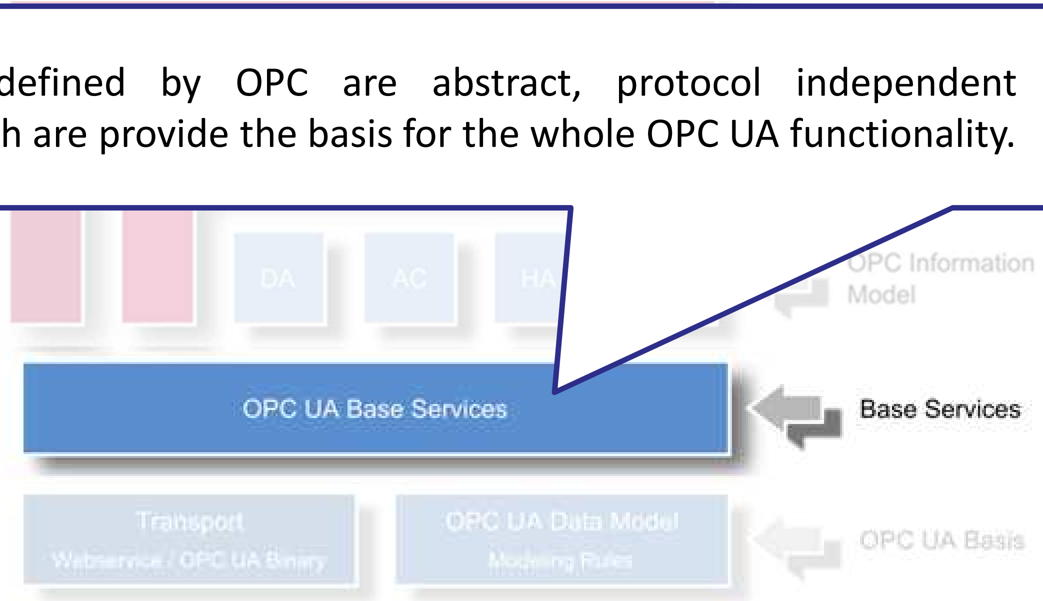


The transport serializes/de-serializes the data and transmits it over the network.

Introduction to OPC UA

- OPC (Openness, Productivity and Collaboration; formerly "OLE for Process Control") Unified Architecture (UA) is industrial communication protocol for interoperability developed by the OPC Foundation.
- OPC UA was accepted as International Electrotechnical Commission (IEC) standard 62541 in 2011.
- The OPC UA is a Service Orientated Architecture (SOA) and is based on different logical levels.

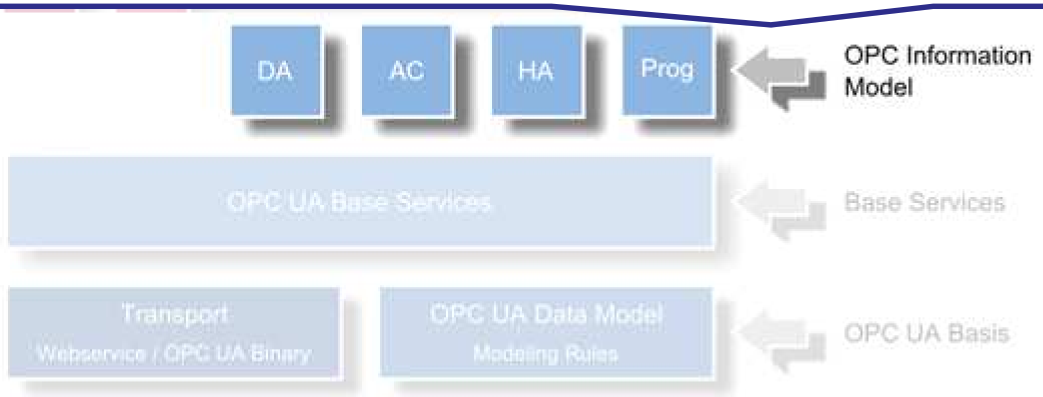
Base Services defined by OPC are abstract, protocol independent method descriptions which are provide the basis for the whole OPC UA functionality.



Introduction to OPC UA

- OPC (Openness, Productivity and Collaboration; formerly "OLE for Process Control") Unified Architecture (UA) is industrial communication protocol for interoperability developed by the OPC Foundation.
- OPC UA was accepted as International Electrotechnical Commission (IEC) standard 62541 in 2011.
- The OPC UA is a Service Orientated Architecture (SOA) and is based on different logical levels.

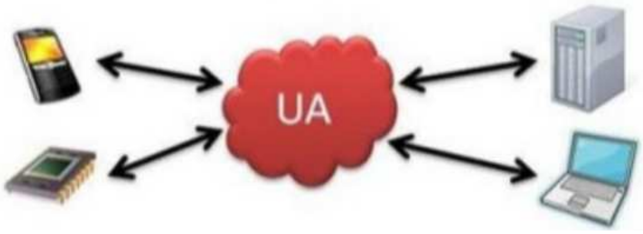
OPC IM is a full-mesh network (FMN) based on Nodes (metadata, diagnostics, objects, attributes, methods, alarms, events and types)



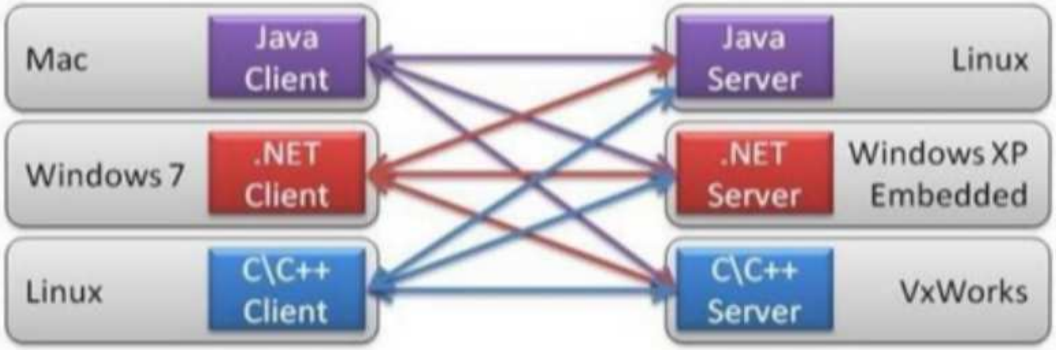
OPC UA key features and benefits

Platform independence and Scalability:

- PC hardware, cloud-based servers, PLCs, micro-controllers (ARM etc.)

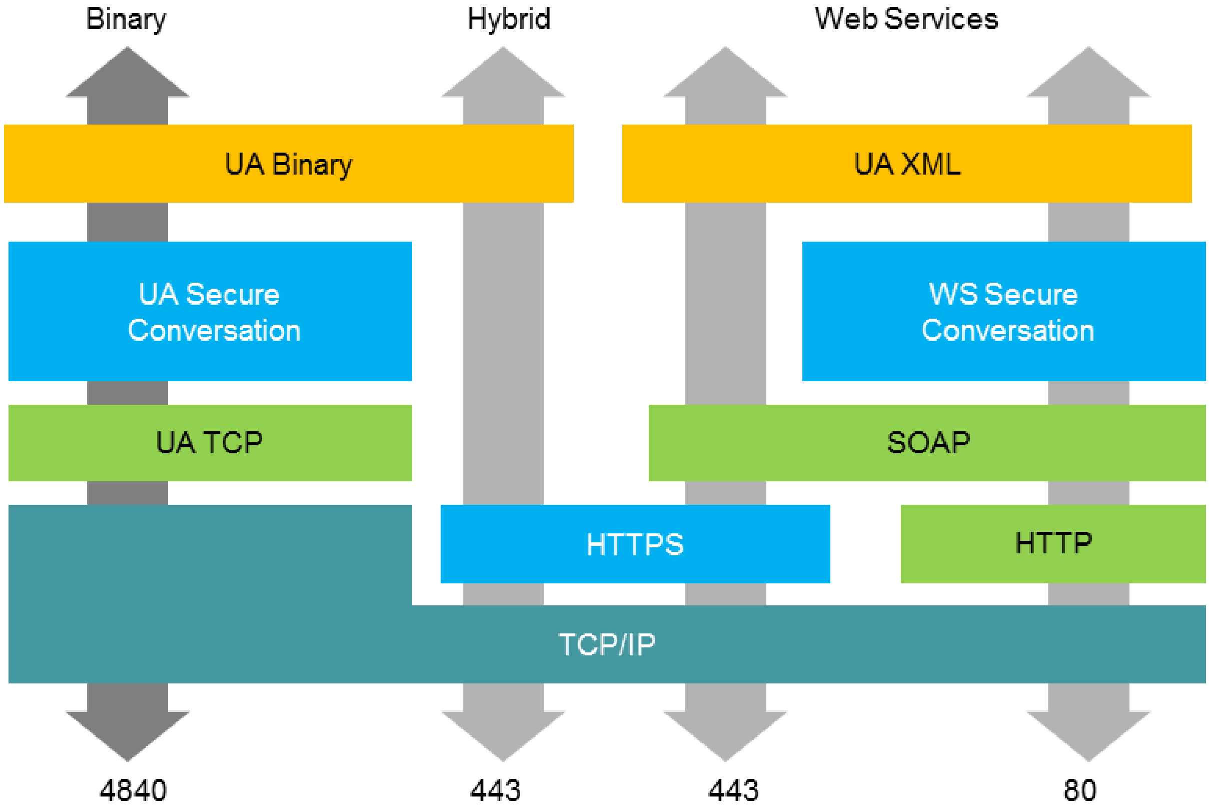


- Microsoft Windows, Apple OSX, Android, Linux, etc.
- Multi-platform implementation, including portable ANSI C, Java, .NET



OPC UA key features and benefits

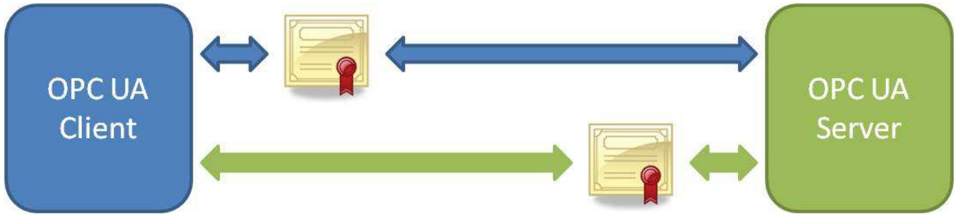
Secure: numerous transport protocols: e.g. ultra-fast OPC-binary transport or the more universally compatible SOAP-based protocols (e.g. HTTP, HTTPS)



OPC UA key features and benefits

Secure: encryption, authentication, authorization and auditing

- **Session encryption, message signing and authentication**: each client/server can be is identified through OpenSSL certificates providing control over which applications and systems are permitted to connect with each other



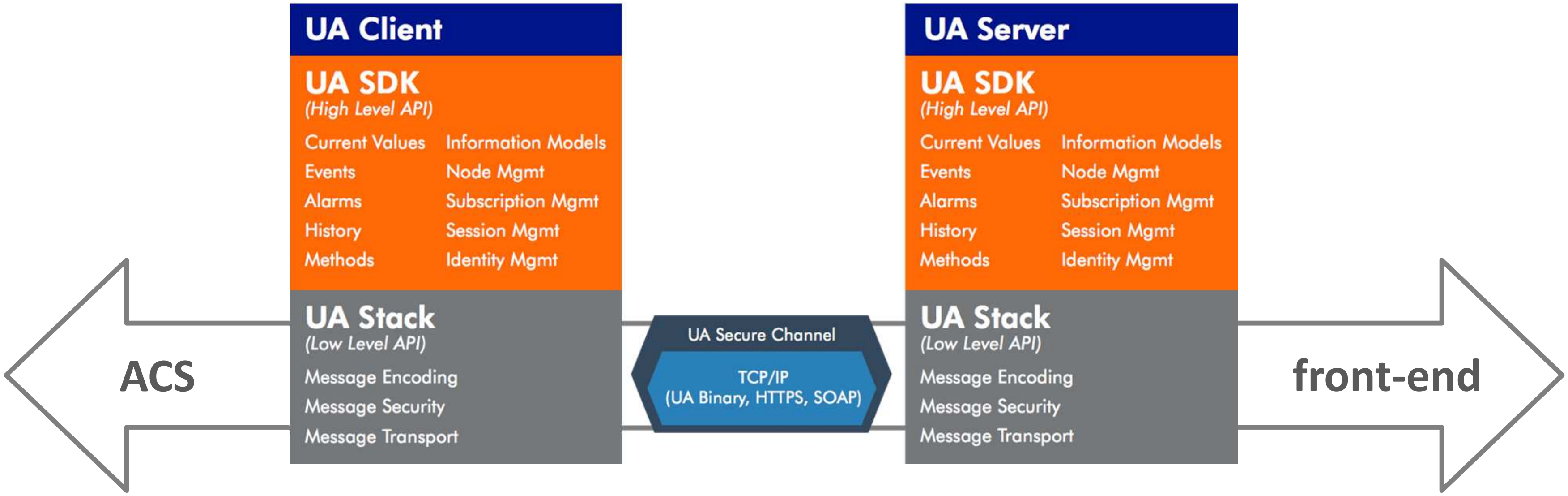
- **Sequenced Packets**: exposure to message replay attacks is eliminated with sequencing
- **User Control**: applications can require users to authenticate and can further restrict/enhance their capabilities with access rights
- **Auditing**: activities by user and/or system are logged providing an access audit trail

OPC UA key features and benefits

- **Extensible:** ability to add new features (e.g. transport protocols, security algorithms, encoding standards, or application-services) without affecting existing applications
- **Discovery:** find the availability of OPC Servers on local PCs and/or networks
- **Address space:** all data is represented hierarchically allowing for complex structures to be discovered by OPC Clients
- **On-demand:** read and write data/information based on access-permissions
- **Subscriptions:** monitor data/information and report-by-exception when values change based on a client's criteria
- **Events:** notify important information based on client's criteria
- **Methods:** clients can execute methods defined on the server
- **Heartbeat** for connection monitoring in both directions (the server as well as the client recognize failures)

OPC UA SDKs

- The OPC UA Stack implements only the TCP/IP communication protocol defined and provided by the **OPC Foundation.**
- The **Software Development Kit** (SDK) implements the base services.



- For developing OPC UA clients/servers inside CTA commercially available SDKs are used. It was decided to purchase SDKs from **Prosys** (Java) and from **Unified Automation** (C++) .

C++ OPC UA SDK from UA

The C++ SDK from Unified Automation (UA) is officially supported for a limited number of operating systems:

- binary (pre-compiled version) and evaluation (for evaluation/testing purposes) editions are available for the MS Win and OpenSuse Linux platforms only.
- No binaries are provided by UA for Scientific Linux (or Cent OS) - binaries must be compiled from source.
- On Desktop Linux (x86, x86-64) compilation the SDKs from source normally does not require any changes or porting.
- Cross-compiling for other architectures (e.g. ARM) normally works without problems with **GCC ≥ 4.1.2** (supports atomic compiler operations).

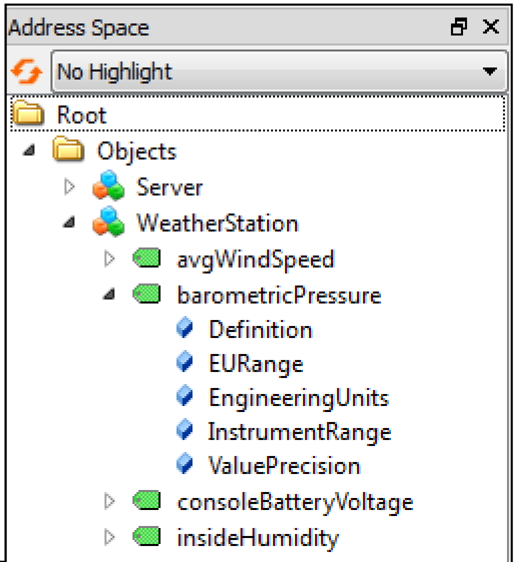
C++ OPC UA SDK from UA

- DESY has purchased a Source Code Developer License from UA.
- In a special agreement with UA this license can be used by the DESY named developer(s) to compile SDKs for CTA member institutions which have purchased a binary license for a certain platform from UA.
- Currently DESY is providing binaries for SL6 and for several ARM-based platforms only.
- Compiling binaries for more platforms depends on the complexity of the system and also on resources available at DESY.

Software: UaExpert Client (from UnifiedAutomation)

UA Expert is an C++ based OPC UA Client used to demonstrate the functionality and performance of OPC UA.

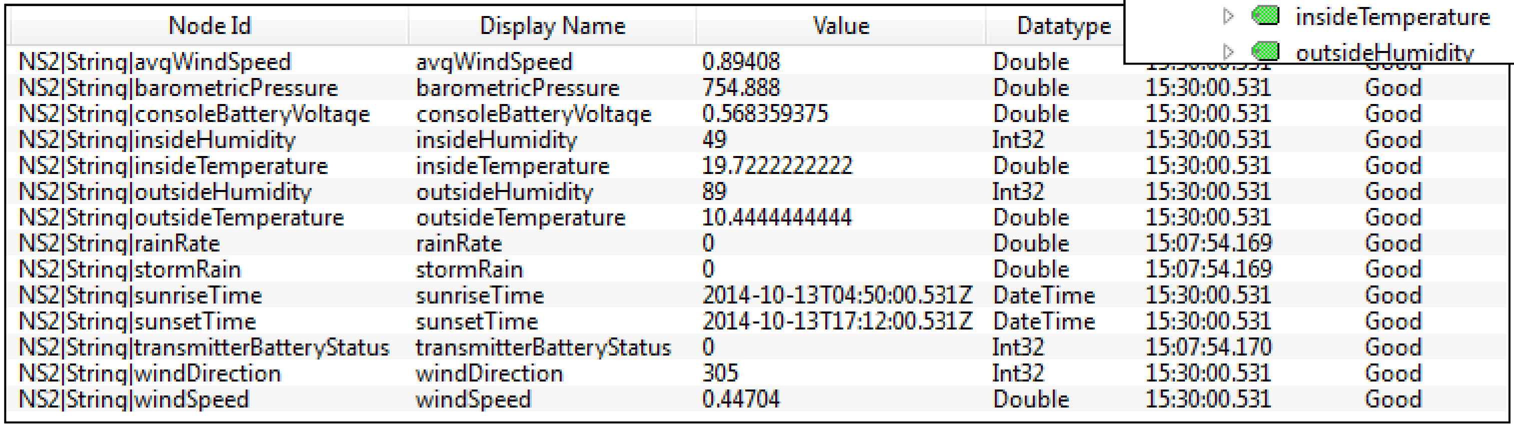
- browsing OPC UA address Space
- reading and writing of variable values/attributes
- monitoring of data changes
- monitoring of events
- calling methods
- reading/updating of history data
- adding/removing nodes and references
- available for Windows/Linux (32-based)



Address Space

No Highlight

- Root
  - Objects
    - Server
    - WeatherStation
      - avgWindSpeed
      - barometricPressure
        - Definition
        - EURange
        - EngineeringUnits
        - InstrumentRange
        - ValuePrecision
      - consoleBatteryVoltage
      - insideHumidity
      - insideTemperature
      - outsideHumidity



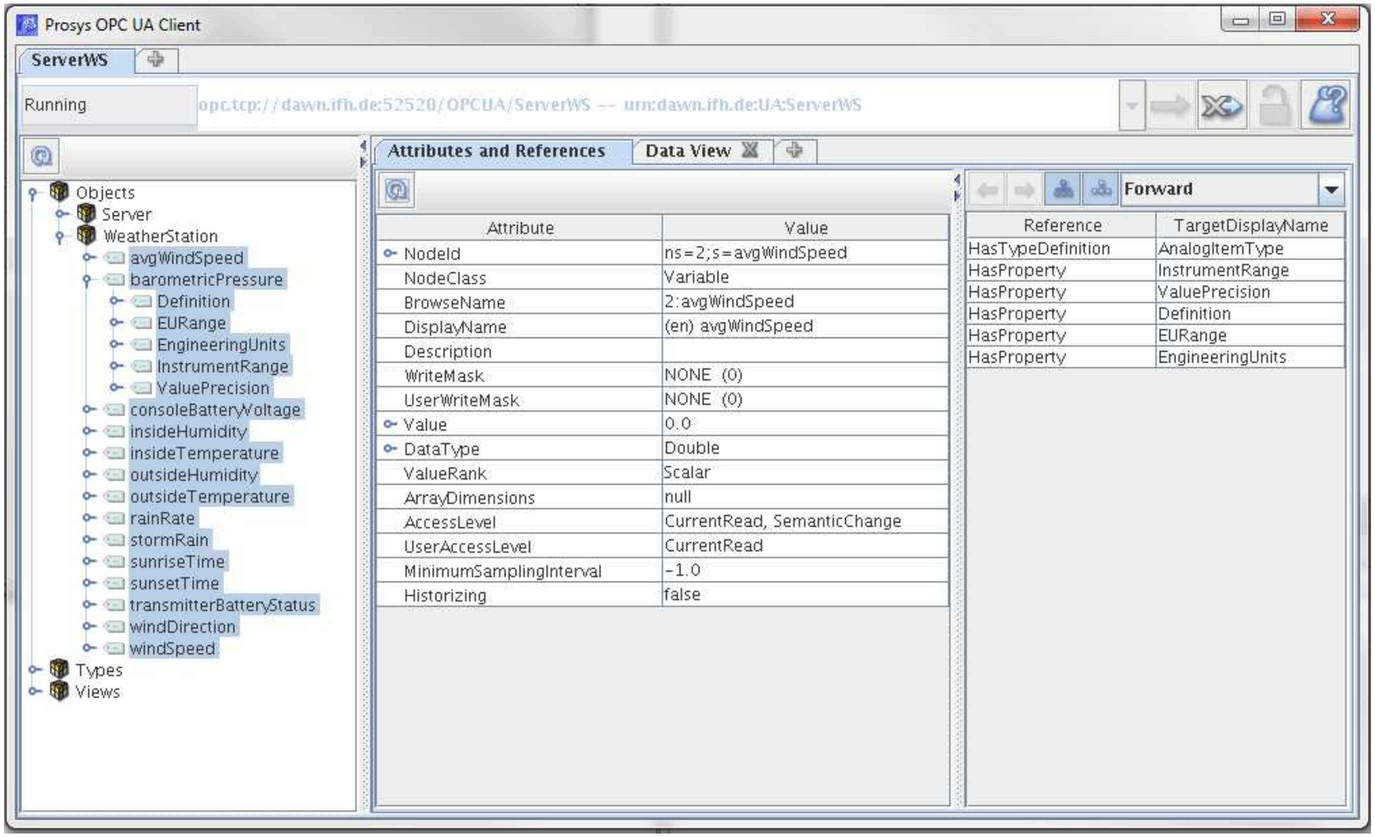
Node Id	Display Name	Value	Datatype		
NS2|String|avgWindSpeed	avgWindSpeed	0.89408	Double	15:30:00.531	Good
NS2|String|barometricPressure	barometricPressure	754.888	Double	15:30:00.531	Good
NS2|String|consoleBatteryVoltage	consoleBatteryVoltage	0.568359375	Double	15:30:00.531	Good
NS2|String|insideHumidity	insideHumidity	49	Int32	15:30:00.531	Good
NS2|String|insideTemperature	insideTemperature	19.7222222222	Double	15:30:00.531	Good
NS2|String|outsideHumidity	outsideHumidity	89	Int32	15:30:00.531	Good
NS2|String|outsideTemperature	outsideTemperature	10.4444444444	Double	15:30:00.531	Good
NS2|String|rainRate	rainRate	0	Double	15:07:54.169	Good
NS2|String|stormRain	stormRain	0	Double	15:07:54.169	Good
NS2|String|sunriseTime	sunriseTime	2014-10-13T04:50:00.531Z	DateTime	15:30:00.531	Good
NS2|String|sunsetTime	sunsetTime	2014-10-13T17:12:00.531Z	DateTime	15:30:00.531	Good
NS2|String|transmitterBatteryStatus	transmitterBatteryStatus	0	Int32	15:07:54.170	Good
NS2|String|windDirection	windDirection	305	Int32	15:30:00.531	Good
NS2|String|windSpeed	windSpeed	0.44704	Double	15:30:00.531	Good

data monitoring view

Software: Java OPC UA Client (from Prosys)

Prosys **OPC UA Client** is a generic OPC UA client developed with Prosys OPC UA Java SDK. OPC UA Client implements the three information models:

- Data Access
- Historical Access
- Alarms & Conditions



Running opc.tcp://dawn.ifh.de:52520/OPCUA/ServerWS -- urn:dawn.ifh.de:UA:ServerWS

- Objects
  - Server
  - WeatherStation
    - avgWindSpeed
    - barometricPressure
      - Definition
      - EURange
      - EngineeringUnits
      - InstrumentRange
      - ValuePrecision
    - consoleBatteryVoltage
    - insideHumidity
    - insideTemperature
    - outsideHumidity
    - outsideTemperature
    - rainRate
    - stormRain
    - sunriseTime
    - sunsetTime
    - transmitterBatteryStatus
    - windDirection
    - windSpeed
- Types
- Views

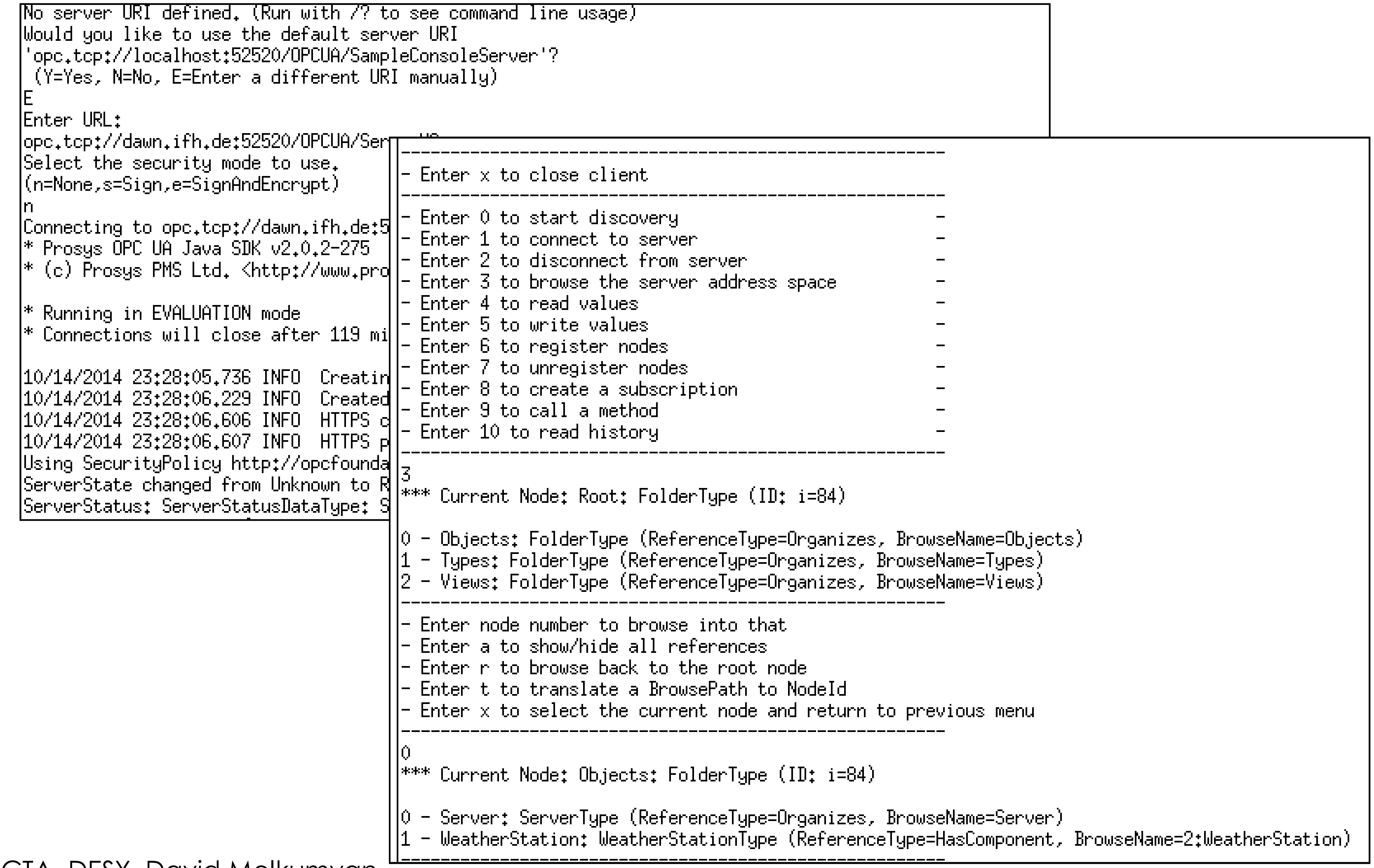
Attribute	Value
NodeId	ns=2;s=avgWindSpeed
NodeClass	Variable
BrowseName	2:avgWindSpeed
DisplayName	(en) avgWindSpeed
Description	
WriteMask	NONE (0)
UserWriteMask	NONE (0)
Value	0.0
DataType	Double
ValueRank	Scalar
ArrayDimensions	null
AccessLevel	CurrentRead, SemanticChange
UserAccessLevel	CurrentRead
MinimumSamplingInterval	-1.0
Historizing	false

Reference	TargetDisplayName
HasTypeDefinition	AnalogItemType
HasProperty	InstrumentRange
HasProperty	ValuePrecision
HasProperty	Definition
HasProperty	EURange
HasProperty	EngineeringUnits

Software: Java Console Client (from Prosys)

Prosys **SampleConsoleClient** is a generic command line based OPC UA client.

```
> ./SampleConsoleClient
```

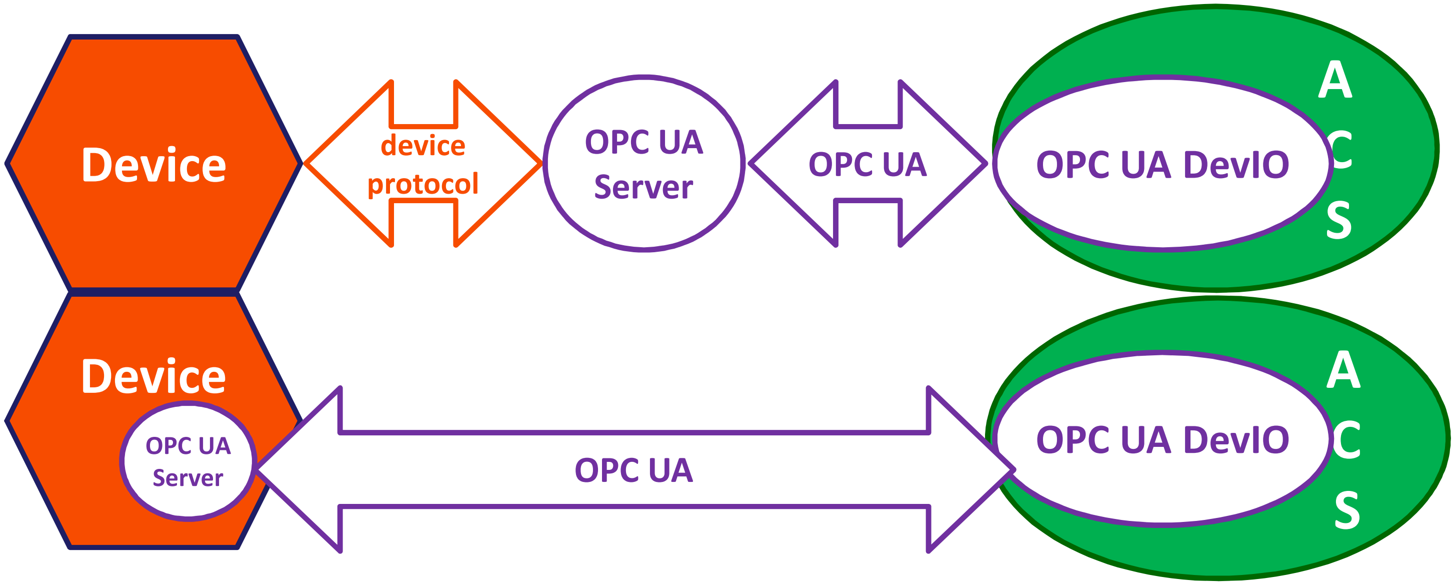


```
No server URI defined. (Run with /? to see command line usage)
Would you like to use the default server URI
'opc.tcp://localhost:52520/OPCUA/SampleConsoleServer'?
 (Y=Yes, N=No, E=Enter a different URI manually)
E
Enter URL:
opc.tcp://dawn.ifh.de:52520/OPCUA/Ser
Select the security mode to use.
(n=None,s=Sign,e=SignAndEncrypt)
n
Connecting to opc.tcp://dawn.ifh.de:5
* Prosys OPC UA Java SDK v2.0.2-275
* (c) Prosys PMS Ltd. <http://www.pro

* Running in EVALUATION mode
* Connections will close after 119 mi

10/14/2014 23:28:05.736 INFO  Creatin
10/14/2014 23:28:06.229 INFO  Created
10/14/2014 23:28:06.606 INFO  HTTPS c
10/14/2014 23:28:06.607 INFO  HTTPS p
Using SecurityPolicy http://opcfounda
ServerState changed from Unknown to R
ServerStatus: ServerStatusDataType: S
```

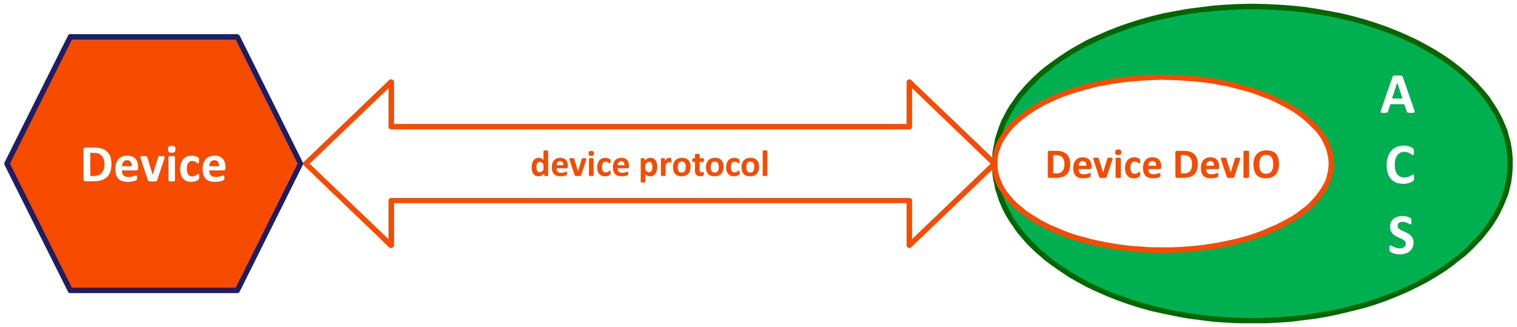
```
-------------------------------------------------------
- Enter x to close client
-------------------------------------------------------
- Enter 0 to start discovery                          -
- Enter 1 to connect to server                        -
- Enter 2 to disconnect from server                   -
- Enter 3 to browse the server address space          -
- Enter 4 to read values                              -
- Enter 5 to write values                             -
- Enter 6 to register nodes                           -
- Enter 7 to unregister nodes                         -
- Enter 8 to create a subscription                    -
- Enter 9 to call a method                            -
- Enter 10 to read history                            -
-------------------------------------------------------
3
*** Current Node: Root: FolderType (ID: i=84)

0 - Objects: FolderType (ReferenceType=Organizes, BrowseName=Objects)
1 - Types: FolderType (ReferenceType=Organizes, BrowseName=Types)
2 - Views: FolderType (ReferenceType=Organizes, BrowseName=Views)
-------------------------------------------------------
- Enter node number to browse into that
- Enter a to show/hide all references
- Enter r to browse back to the root node
- Enter t to translate a BrowsePath to NodeId
- Enter x to select the current node and return to previous menu
-------------------------------------------------------
0
*** Current Node: Objects: FolderType (ID: i=84)

0 - Server: ServerType (ReferenceType=Organizes, BrowseName=Server)
1 - WeatherStation: WeatherStationType (ReferenceType=HasComponent, BrowseName=2:WeatherStation)
-------------------------------------------------------
```

Device integration into ACS: two scenarios …

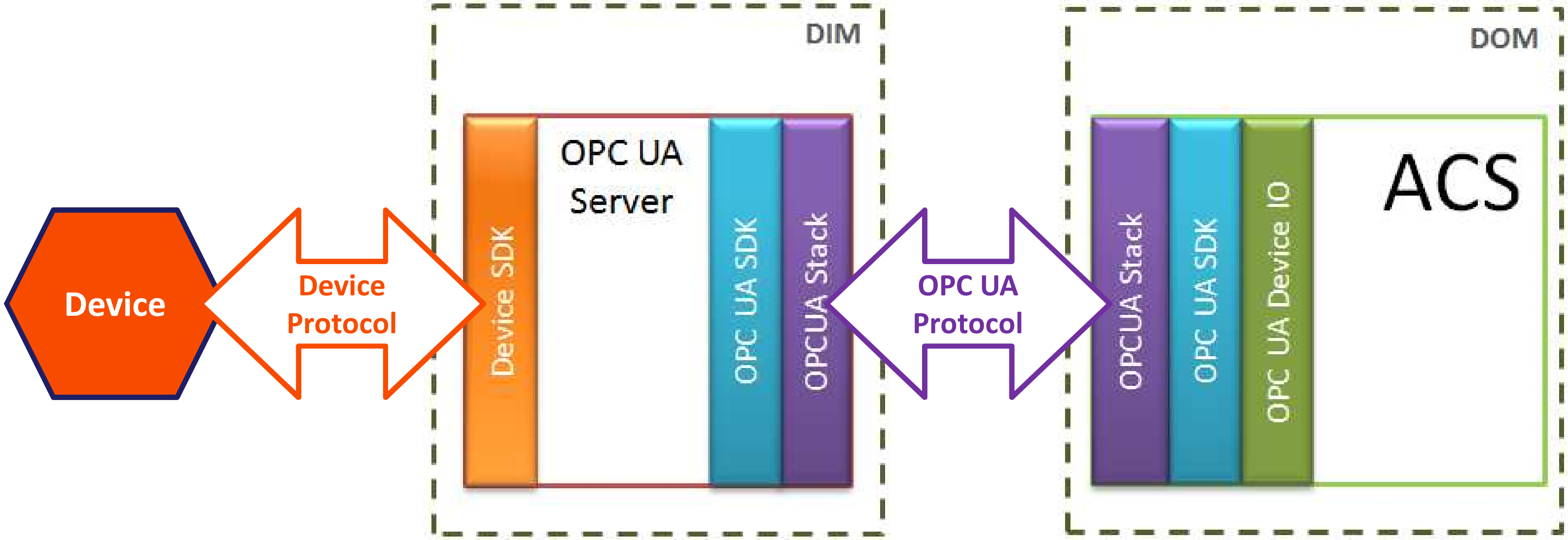
- via OPC UA (server) and OPC UA DevIO (client)



- directly via device specific DevIO



Device integration into ACS via OPC UA

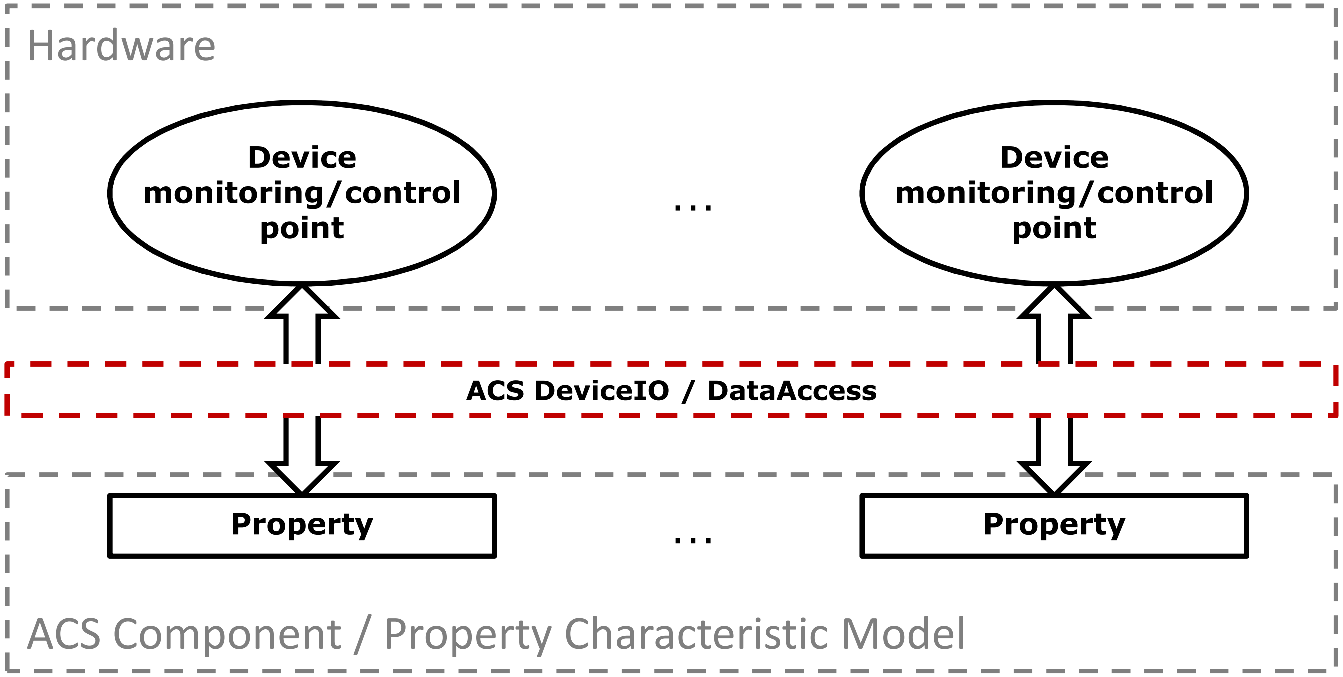


- OPC UA was approved as the interface for controlling devices and their integration into ACS within ACTL software;
- control devices are communicating either by using **native OPC UA server firmware** or by implementing **OPC UA server** software on the device access level;
- OPC UA client functionality is directly integrated into ACS by means of the **ACS Data Access Support** (DevIO) abstraction layer;

Introduction

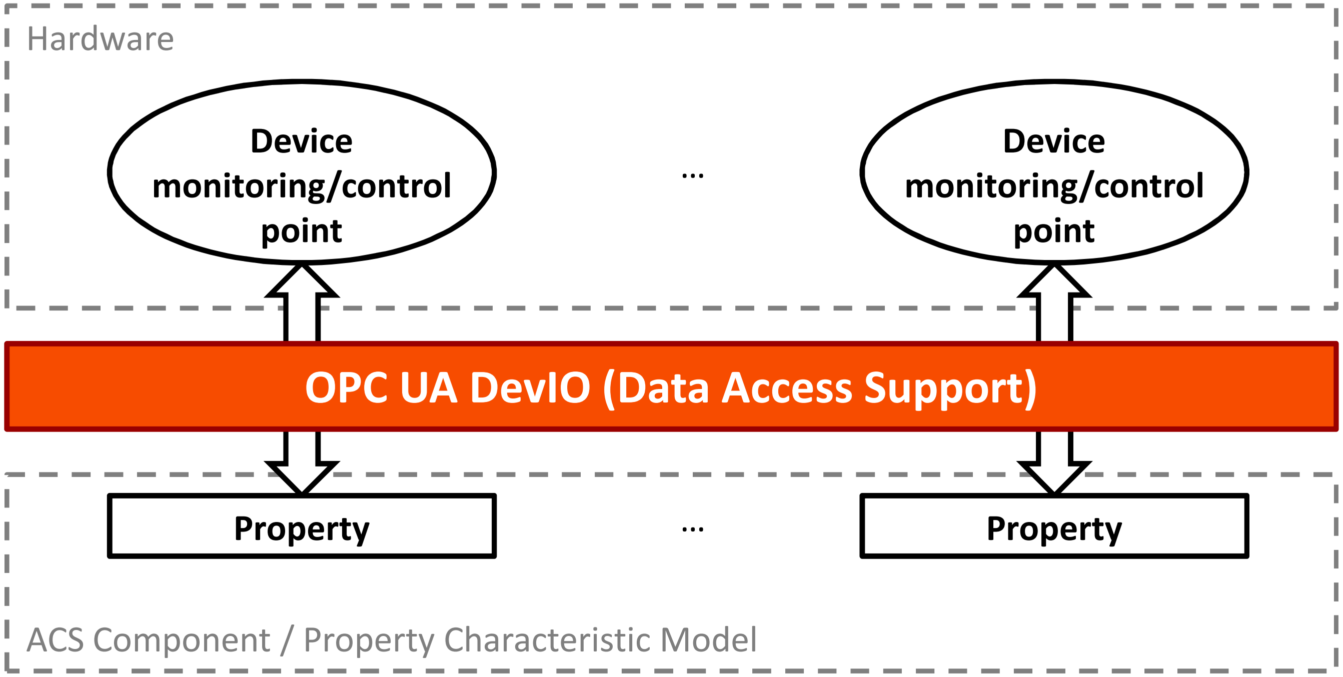
- ACS provides a generic abstraction between the Property and the hardware monitor/control points in the Component / Property Characteristic model.

- ACS Device IO decouples the ACS property from the access to the hardware.


Hardware
Device monitoring/control point
…
Device monitoring/control point
ACS DeviceIO / DataAccess
Property
…
Property
ACS Component / Property Characteristic Model

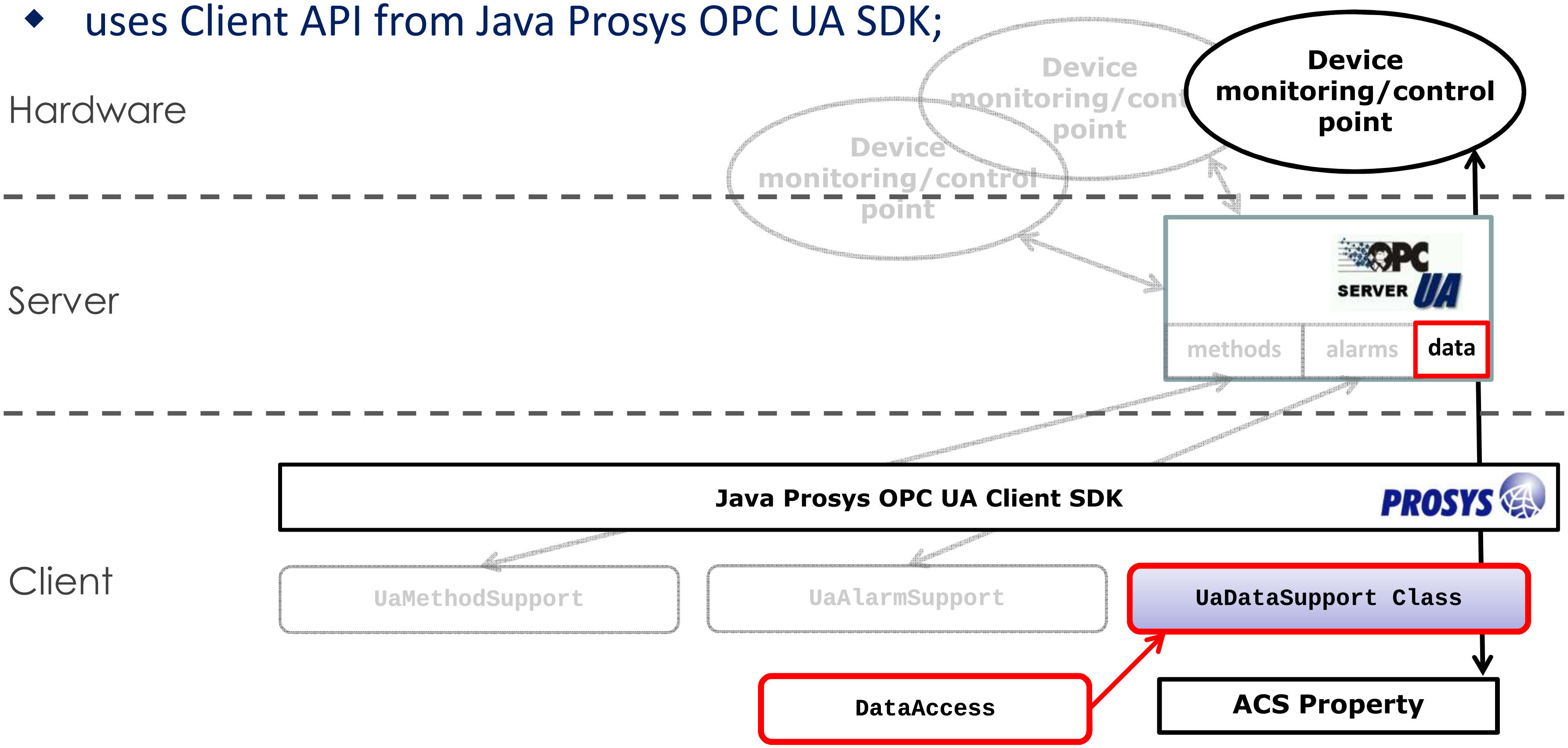
OPC UA Device I/O for ACS

- OPC UA DevIO is a one of the implementations of the ACS DevIO layer for monitoring/control point access via OPC UA


Hardware
Device monitoring/control point
...
Device monitoring/control point
OPC UA DevIO (Data Access Support)
Property
...
Property
ACS Component / Property Characteristic Model

OPC UA Data Access Support for ACS

- implements functionality of the DataAccess (DevIO) class for monitoring/control point access via OPC UA Server :
  - read/write and subscriptions
- uses Client API from Java Prosys OPC UA SDK;

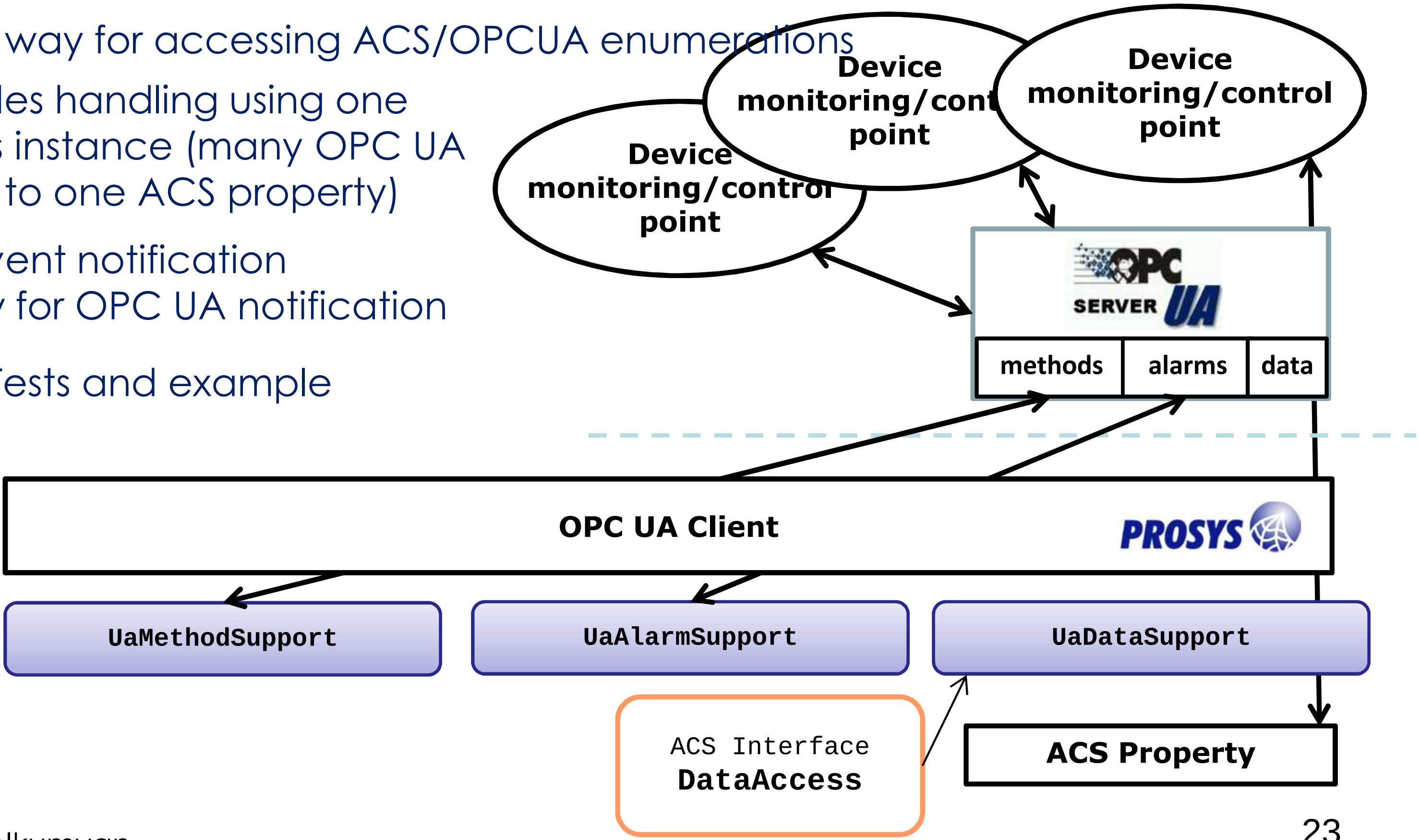


OPC UA Data Access Support for ACS

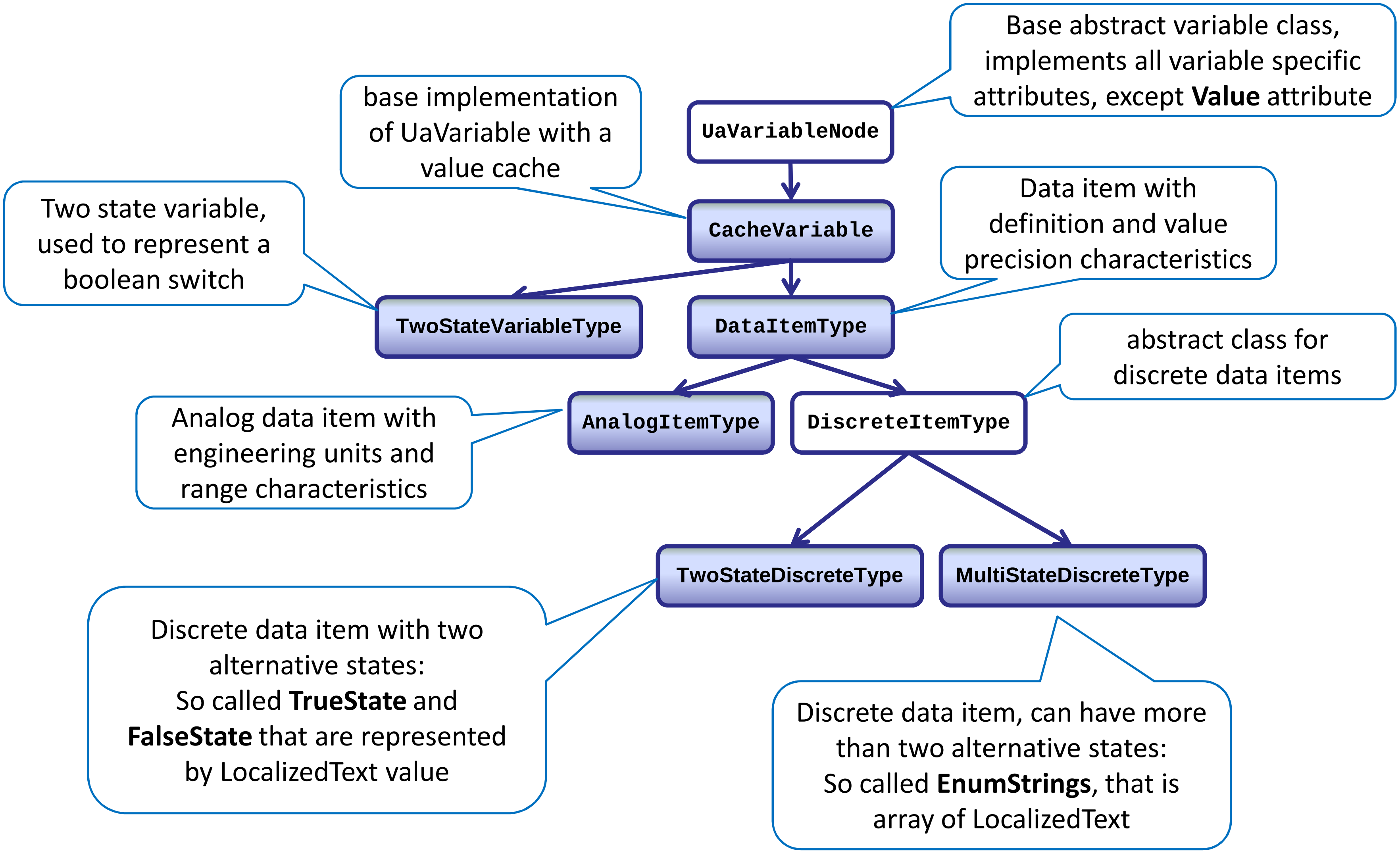
https://forge.in2p3.fr/projects/ctaactl/wiki/DevIO

Additionally, provides

- methods for synchronous/asynchronous calls
- alarms
- a “generic” way for accessing ACS/OPCUA enumerations
- multiply nodes handling using one DataAccess instance (many OPC UA data nodes to one ACS property)
- simplified event notification functionality for OPC UA notification
- set of JUnit Tests and example



Prosys OPC UA server and address space



Base abstract variable class, implements all variable specific attributes, except **Value** attribute

base implementation of UaVariable with a value cache

Two state variable, used to represent a boolean switch

Data item with definition and value precision characteristics

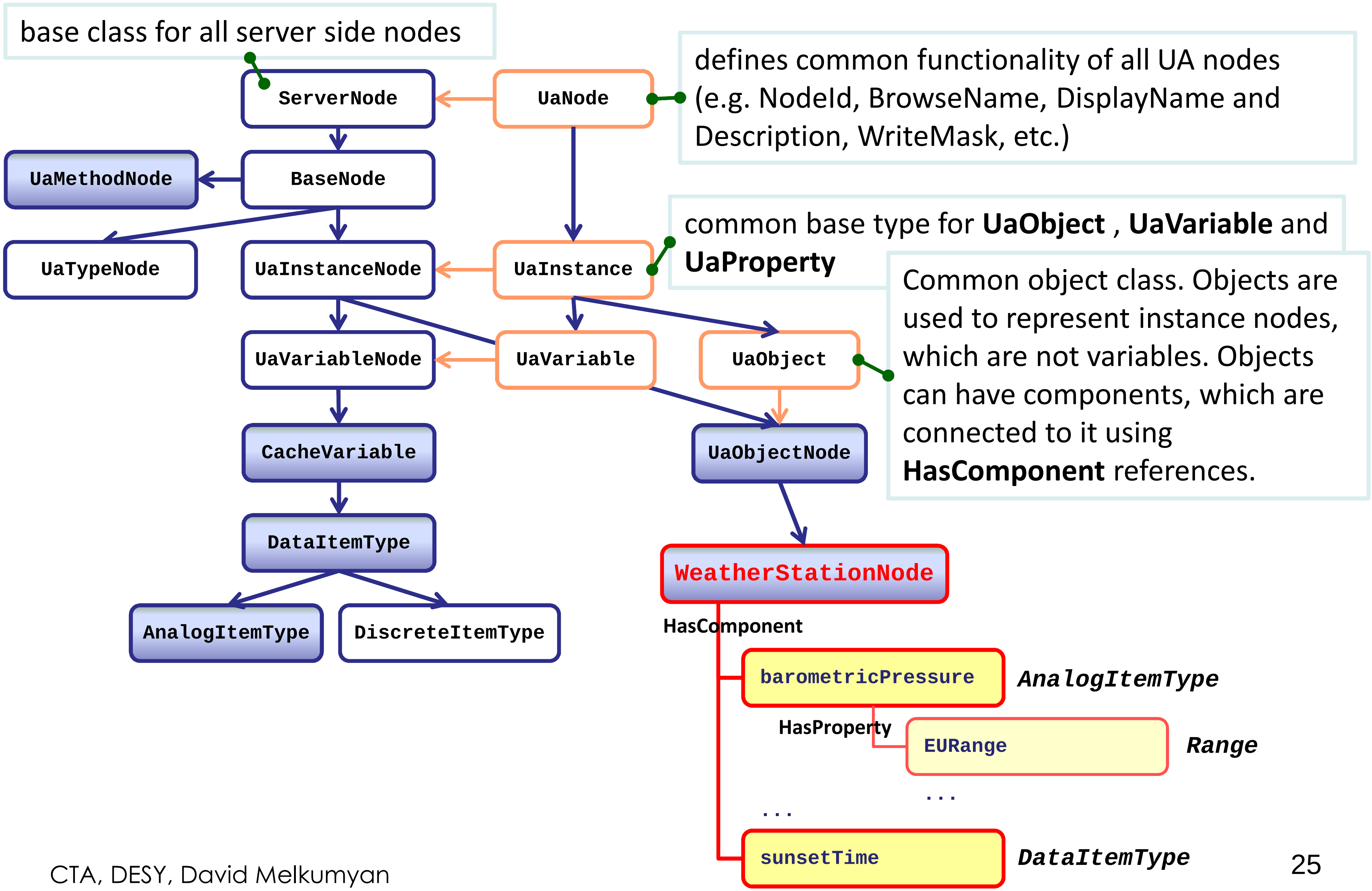
abstract class for discrete data items

Analog data item with engineering units and range characteristics

Discrete data item with two alternative states:
So called **TrueState** and **FalseState** that are represented by LocalizedText value

Discrete data item, can have more than two alternative states:
So called **EnumStrings**, that is array of LocalizedText

Prosys OPC UA server and address space (WS server)



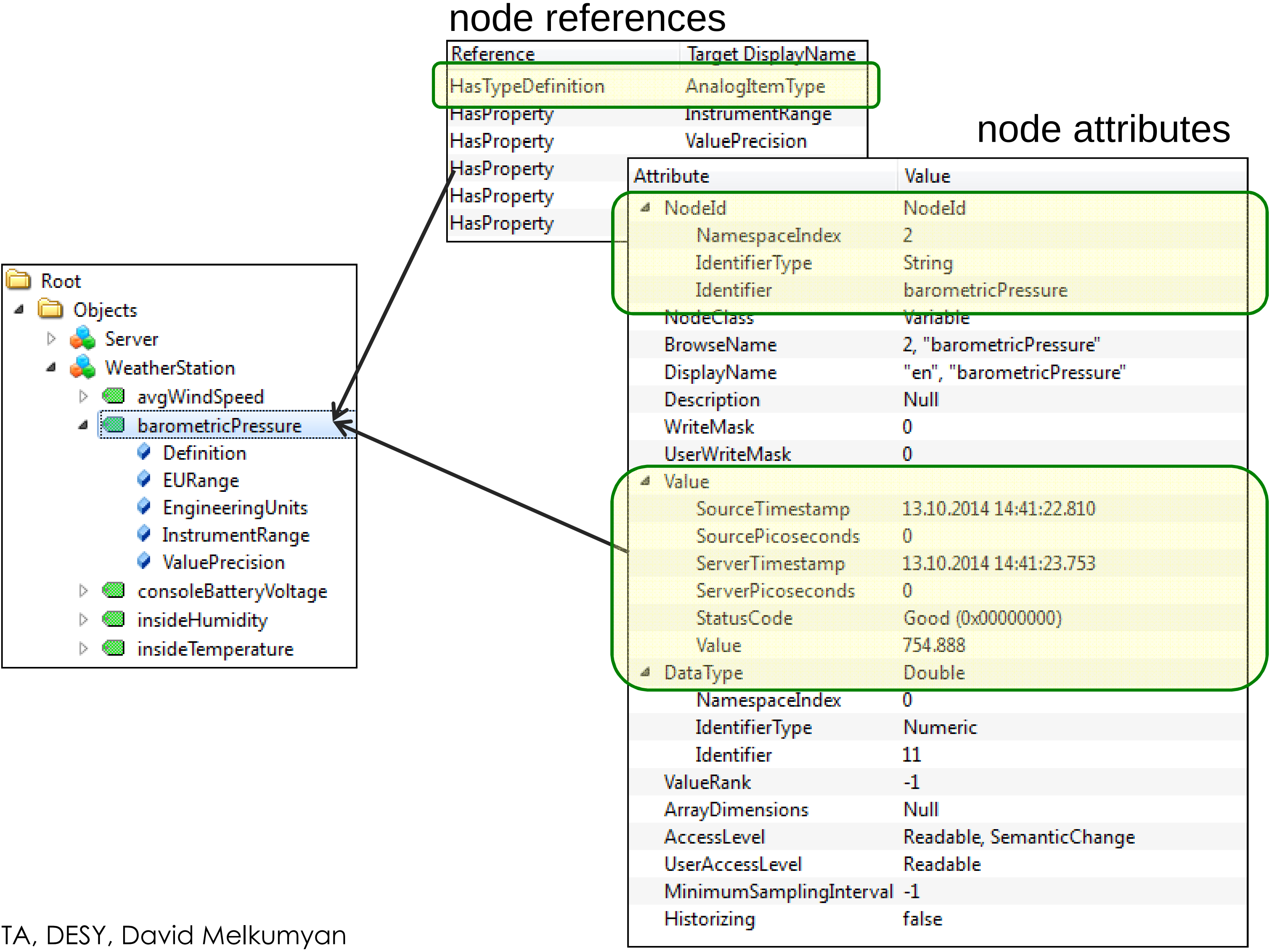
base class for all server side nodes

defines common functionality of all UA nodes (e.g. NodeId, BrowseName, DisplayName and Description, WriteMask, etc.)

common base type for **UaObject** , **UaVariable** and **UaProperty**

Common object class. Objects are used to represent instance nodes, which are not variables. Objects can have components, which are connected to it using **HasComponent** references.

OPC UA server and address space (WS server)



node references

Reference	Target DisplayName
HasTypeDefinition	AnalogItemType
HasProperty	InstrumentRange
HasProperty	ValuePrecision
HasProperty	
HasProperty	
HasProperty	

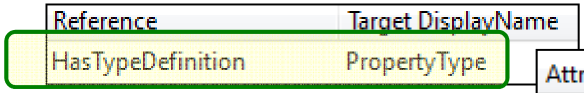
- Root
  - Objects
    - Server
    - WeatherStation
      - avgWindSpeed
      - barometricPressure
        - Definition
        - EURange
        - EngineeringUnits
        - InstrumentRange
        - ValuePrecision
      - consoleBatteryVoltage
      - insideHumidity
      - insideTemperature

node attributes

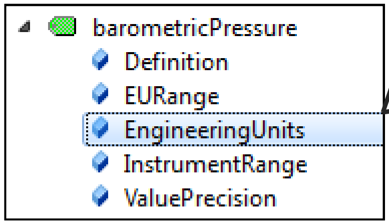
Attribute	Value
NodeId	NodeId
NamespaceIndex	2
IdentifierType	String
Identifier	barometricPressure
NodeClass	Variable
BrowseName	2, "barometricPressure"
DisplayName	"en", "barometricPressure"
Description	Null
WriteMask	0
UserWriteMask	0
Value	
SourceTimestamp	13.10.2014 14:41:22.810
SourcePicoseconds	0
ServerTimestamp	13.10.2014 14:41:23.753
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	754.888
DataType	Double
NamespaceIndex	0
IdentifierType	Numeric
Identifier	11
ValueRank	-1
ArrayDimensions	Null
AccessLevel	Readable, SemanticChange
UserAccessLevel	Readable
MinimumSamplingInterval	-1
Historizing	false

OPC UA server and address space (WS server)

node references

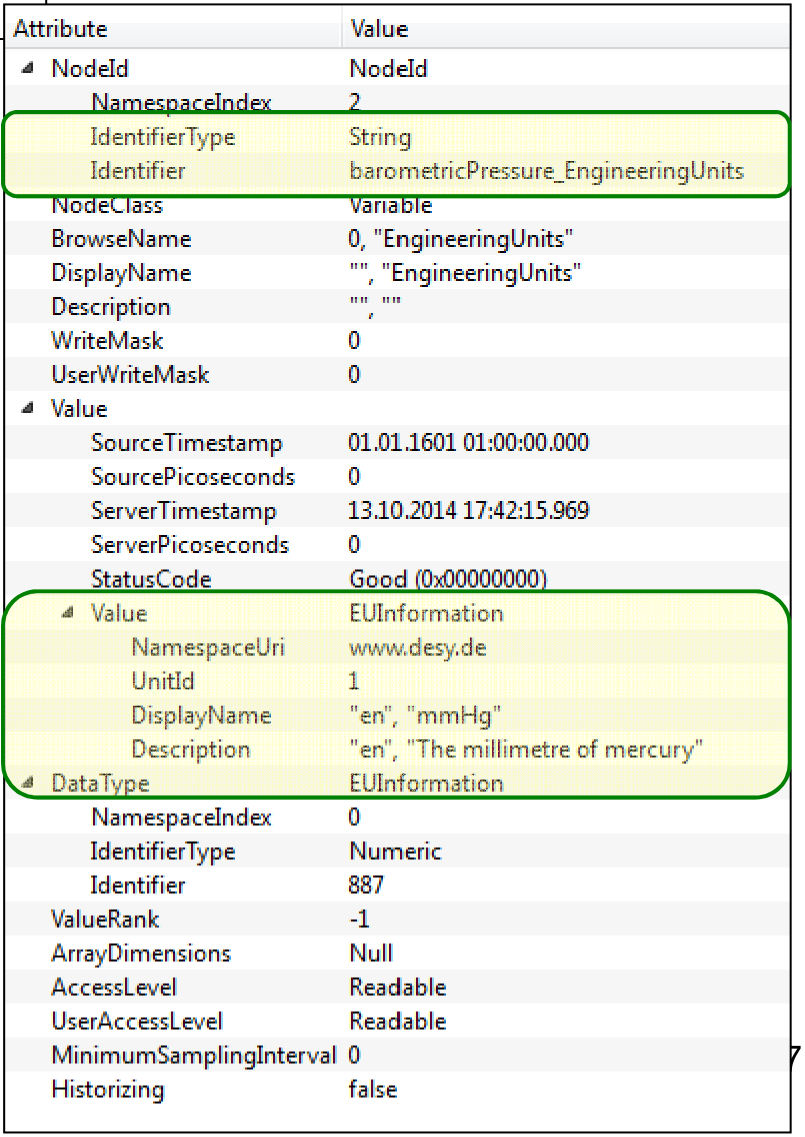


Reference	Target DisplayName
HasTypeDefinition	PropertyType



- barometricPressure
  - Definition
  - EURange
  - EngineeringUnits
  - InstrumentRange
  - ValuePrecision

node attributes



Attribute	Value
NodeId	NodeId
NamespaceIndex	2
IdentifierType	String
Identifier	barometricPressure_EngineeringUnits
NodeClass	Variable
BrowseName	0, "EngineeringUnits"
DisplayName	"", "EngineeringUnits"
Description	"", ""
WriteMask	0
UserWriteMask	0
Value	
SourceTimestamp	01.01.1601 01:00:00.000
SourcePicoseconds	0
ServerTimestamp	13.10.2014 17:42:15.969
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	EUInformation
NamespaceUri	www.desy.de
UnitId	1
DisplayName	"en", "mmHg"
Description	"en", "The millimetre of mercury"
DataType	EUInformation
NamespaceIndex	0
IdentifierType	Numeric
Identifier	887
ValueRank	-1
ArrayDimensions	Null
AccessLevel	Readable
UserAccessLevel	Readable
MinimumSamplingInterval	0
Historizing	false

Source code available at CTA SVN Repository (`svn.in2p3.fr`)

OPC UA Server for Weather Station location:
`“ACTL/projects/OpcUaServerWeatherStation”`

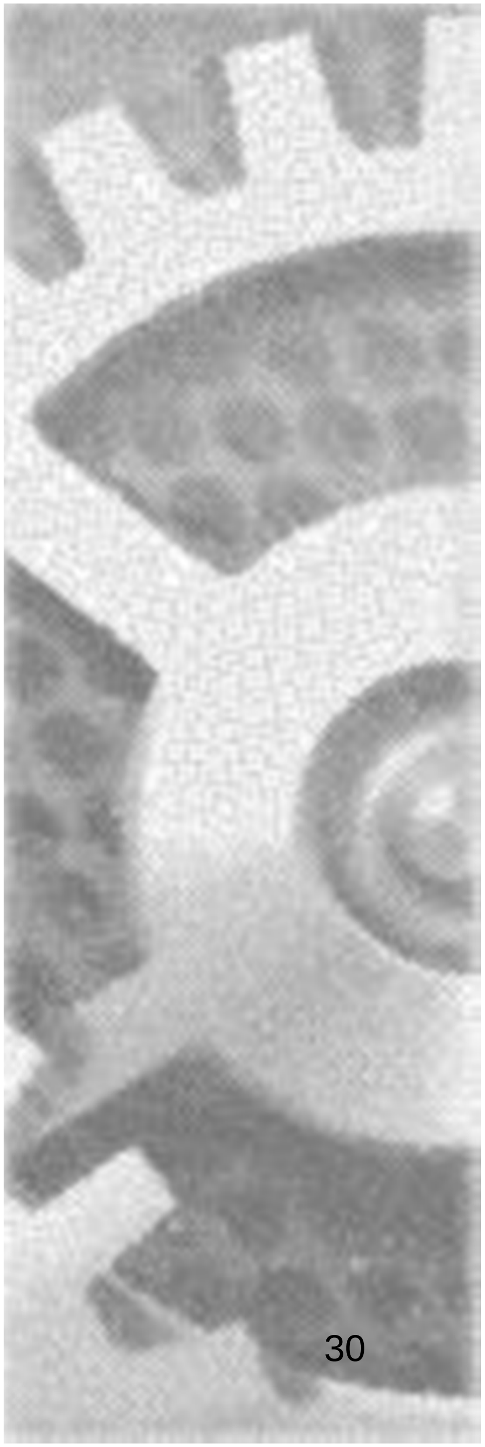
ACS component for the drive system (bridge):
`“ACTL/projects/DriveSystemMST”`

Summary

- OPC UA DevIO for ACS is available
- C++ OPC UA Server/Client SDK binaries are provided to several groups
- OPC UA Server CGT are available (MOS, UaModel, etc.)
- Bootcamp for OPC UA server (was organized in Oct 2014)
- Guidelines for device integration into ACS via OPC UA (WIP)
- Several projects (OPC UA servers, ACS bridges) are available on SVN

Scheduled:

- Publishing/sharing documents on OPC UA (e.g. Guidelines, SDKs, Tools, CGF)
- Guidelines for creating OPC UA servers (TODO)



thank you