ELFORSK

Energimyndigheten

Wind power communication – Design and implementation of test environment for IEC61850/UCA2

Elforsk rapport 02:16

Anders Johnsson, Jörgen Svensson

April 2002

Wind power communication – Design and implementation of test environment for IEC61850/UCA2

Elforsk rapport 02:16

Anders Johnsson, Jörgen Svensson April 2002

Wind power communication – Design and implementation of test environment for IEC61850/UCA2

Elforsk rapport 02:16

Authors:

Anders Johnsson	Vattenfall Utveckling AB
Jörgen Svensson	SYCON Energikonsult

Förord

Föreliggande rapport utgör en komplettering av tidigare utgivna specifikation av funktionella krav för etablering och drift av ett system för datakommunikation mellan kontrollsystemet i ett vindkraftverk och datorer för fjärrövervakning (SCADA), Elforsk rapport 01:25.

Arbetet har, liksom det tidigare projektet, utförts av en dansk-svensk arbetsgrupp med representanter från följande företag: Vattenfall Utveckling AB och Sycon Energikonsult som representanter för Elforsk AB, Sydkraft Vind AB, Tech-wise A/S som representanter för Elsam A/S, SEAS Distribution A.m.b.A som representanter för Energi E2 A/S.

Projektet har genomförts inom ramen för Elforsks vindkraftprogram (projekt 2172). Programmet finansieras av Vattenfall, Sydkraft, Birka Energi, Göteborg Energi, Umeå Energi, Falkenberg Energi, Helsingborg Energi, Varberg Energi, Graninge AB och Energimyndigheten.

Stockholm april 2002

Ulf Arvidsson

Elforsk AB
Programområde El- och värmeproduktion

Sammanfattning

En version av den preliminära standarden IEC 61850 baserad på kommunikations-protokollet MMS (ISO 9506) har testats i ett vindkraftverk i Sverige och ett motsvarande projekt med ett OPC-gränssnitt har genomförts i Danmark. Resultat och erfarenheter från de tester som gjorts i Sverige och i Danmark redovisas i separat Elforsk rapport 02:14 ”Wind power communications – Verification report and recommendation”.

Denna rapport innehåller en detaljerad beskrivning av den implementering av IEC 61850/MMS som gjorts i ett vindkraftverk på Gotland. Syftet är med denna rapport är beskriva de tekniska lösningar i form av modeller, system och programvaror som använts men ett viktigt syfte är också att detta material skall kunna fungera som underlag för liknande tester eller pilotinstallationer på andra platser.

Summary

A version of the draft standard IEC 61850 based on the communication protocol MMS (ISO 9506) has been implemented and tested at a wind power plant in Sweden and a similar project for an OPC-interface has been conducted in Denmark. Results and conclusions from the tests conducted in Sweden and in Denmark are documented in a separate Elforsk report, 02:14 "Wind power communications – Verification report and recommendation".

This report contains detailed descriptions on the implementation of IEC 61850/MMS in a wind power plant at Gotland, Sweden. The purpose of the report is to present the technical solutions in the way of models, systems, software programs, and tools used. An important purpose is also that this material should be possible to use as basis for similar tests or pilot installations at other locations.

Contents

Appendices

1 Introduction

Elforsk has sponsored a join Swedish-Danish work to find, and to make recommendations on, a common solution for communication with wind power plants. The first stage of the work resulted in the requirement specification Functional requirements on Communication System for Wind Turbine Applications (Elforsk report 01:25) [1]. During the work a number of possible communication solutions were identified. The two most promising solutions have been tested in order to verify to what extent they fulfil the requirements in the specification. A version of the IEC 61850 standard [2] based on the communication protocol MMS [3], [4] has been tested at a wind power plant at Näsudden on Gotland in Sweden and an OPC-interface has been tested at Ny Nøjsomhedsodde at west Lolland in Denmark. This document includes a description of the design choices made for the test implementation of MMS-based communication at one of Vattenfalls wind power plants at Gotland, as well as a detailed description of the implementation of IEC61850/UCA2 software including information models and information exchange services.

2 Verification environment

2.1 The test object

Vattenfall has located the national control centre for wind power at Näsudden on the island Gotland. From there 39 wind power plants at different locations in Sweden are supervised. 8 of these are located at Näsudden.

The test object is a wind turbine from Nordic Wind Power, NWP. Sigvards 2, which is the name of the turbine, was taken into operation during year 2000. The wind turbine is located approx. 600 m from the control centre. A local radio based Ethernet-LAN connects the wind turbine and the control centre. The wind turbine is also equipped with telephone modem.

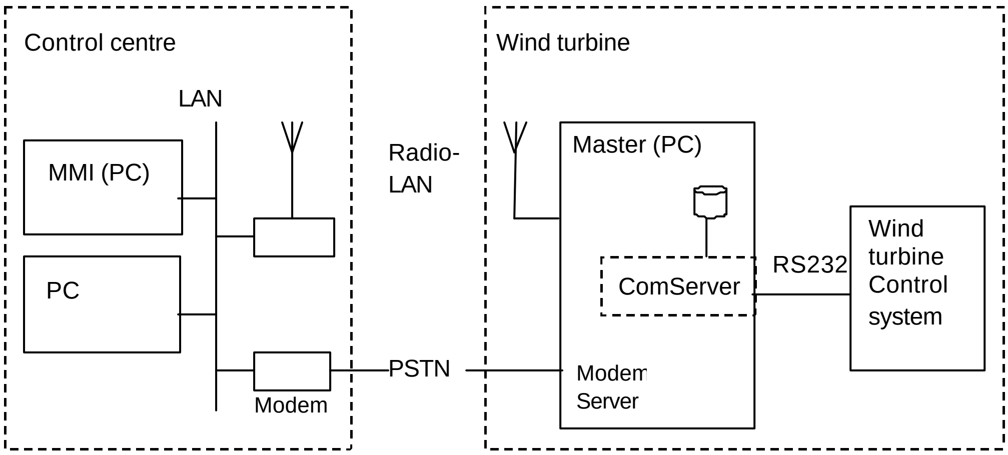

Figure 1: Overview control centre, wind power plants and communication networks.

As all the other wind turbines at Näsudden Sigvards 2 is equipped with a PC, called Master-PC, that handles and stores data from the wind turbine. The control centre operator can connect to the Master-PC and retrieve both off-line and on-line data, as well as operate the wind turbine.

2.2 Data communication flow

The wind turbine control system is from Mita-Teknik A/S. It is connected with the Master-PC through an RS232 link. The control system uses Mita-Teknik's own protocol to periodically send data to the Master-PC. The protocol is described in WP3000 Manual, Communication 7.21 [16]. Four different kinds of data packets are sent once every 100 ms, 1 s or 10 s.

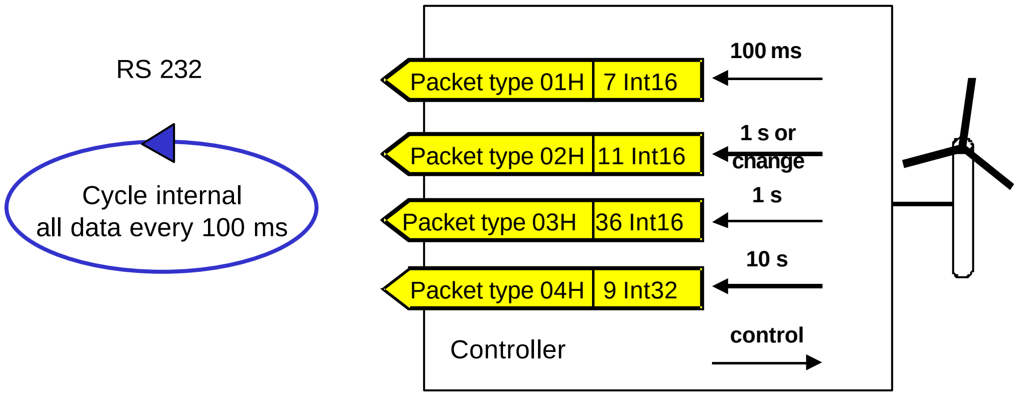

Figure 2: Control system communication interface

Packet type 01H includes the following analogue values that are sent every 100 ms:
- Generator speed
- Generator slip
- Power
- Current L1
- Acceleration X
- Acceleration Y
- Yaw pressure

Packet type 02H includes the following binary state values that are sent every 1 s:
- Word 0 (Error, Warning, Free to yaw, Free to operate, Fee run, etc) (16 bits)
- Word 1 (Phase compensation (order), Disk brake 1 activated (order), etc) (16 bits)
- Word 2 (Free / not used)
- Word 3 (Status word to Weier)
- Word 4 (Status word from Weier)
- Word 5 (Reset level)
- Word 6 (Status code)
- Word 7 (Active fault code)
- Word 8 (Active fault code 2)
- Word 9 (Active fault code 3)
- Word 10 (Active fault code 4)

Packet type 03H includes totally 36 measurement values (signed integers) that are sent every 1 s. The first ten are:
- Cos.phi
- Grid voltage L1
- Grid voltage L2
- Grid voltage L3
- Grid current L1
- Grid current L2
- Grid current L3

- Reactive power
- Grid frequency
- Generator current (Weier)

Packet typ 04H includes the following measurement values (signed long) that are sent every 10 s:

- Energy G1
- Energy G2
- Energy consumption
- Time G1
- Time G2
- Time with fault status
- Time with grid OK
- Time with sufficient wind for electricity production
- Total time

3 Design choices

3.1 Server interface issues

There are several ways to implement wind power plant interfaces, see Figure 3. Taking into account today's non-standardised information and information exchange methods there are many topologies possible how and where a standard interface is implemented. Three possible topologies are discussed in the following. The objective of these examples is to depict the principle ways to apply the IEC61850 standard.

NOTE A smooth migration from today's solutions to a standardised solution is likely to be implemented making use of today's solutions and applying the advanced standard specification where appropriate.

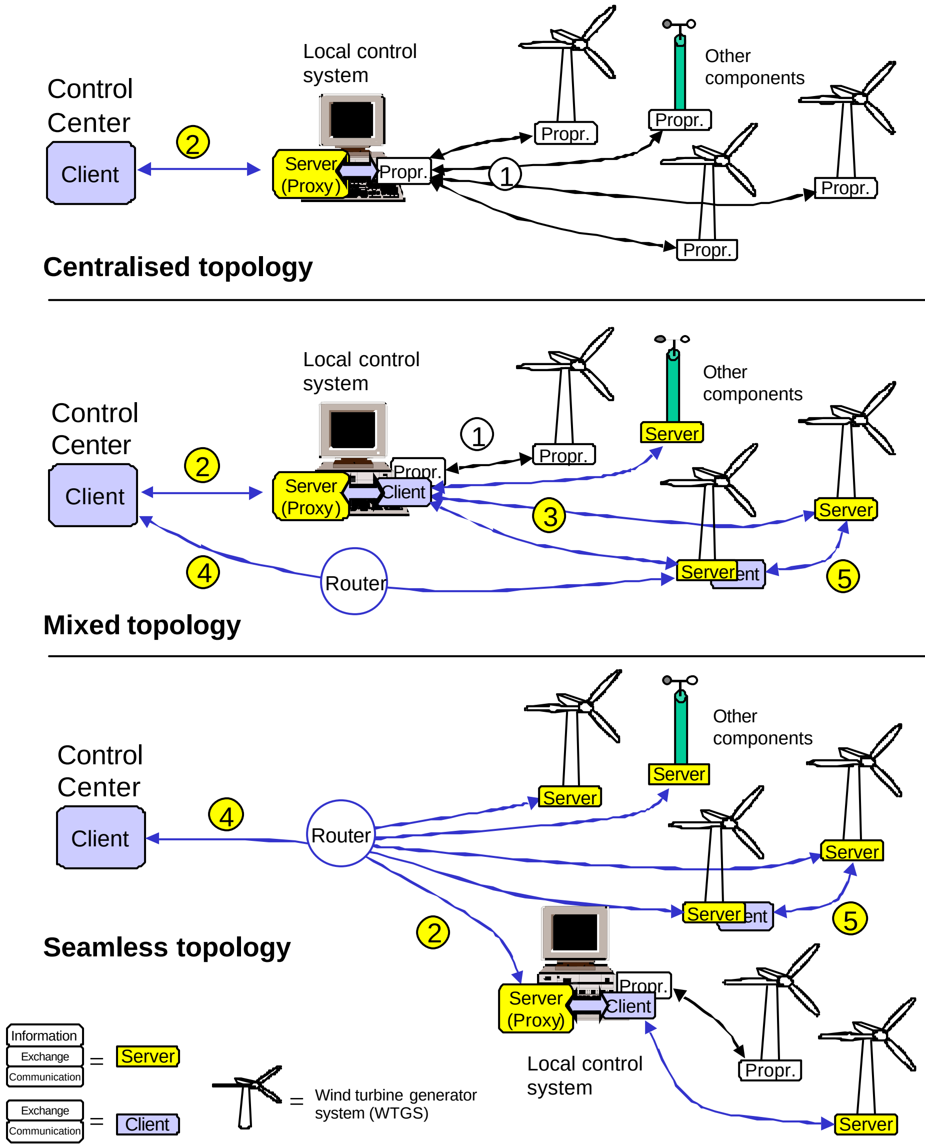

Figure 3 – Communication topology for wind power plant applications

3.1.1 Centralised topology

As depicted in Figure 3 the application of the use of the software is as shown on the top. The centralised topology provides a centralised view to all information of all parts of the wind power plant. The communication between the centralised server and the various parts of the wind power plant — interface (1) — is unchanged proprietary. The standardised information and information exchange methods are implemented in the centralised server located in the local control system. The client in the control centre can access the server — interface (2) — using standardised information exchange methods.

In this topology the various parts of the wind power plant use non-standardised solutions. The mapping of the process values to the information models is realised in the centralised local control system. The benefit of this topology is:

- The wind power plant can be connected by standardised means to the control centre.
- The wind power plants, their information and communication systems can be used as is.

This topology has been chosen for the project.

3.1.2 Mixed topology

In this advanced application three additional implementation approaches are introduced. In case — interface (3) — the information and information exchange methods are directly realised in the various parts of the wind power plant.

The mapping of the process values to the information models is realised in the various parts. The centralised server allows to filter and process the information of the wind power plant before a small amount of information is exchanged with the control centre. The interface (4) adds the possibility to replace the centralised system. The access is directly through a router with all security measures. The interface (5) introduces the possibility to apply a standard also for communication between any parts of the wind power plant.

The advantages are:

- The number of communication solutions to be implemented and maintained inside the wind power plant is reduced to one — interface (3).
- Direct (but secure) seamless access can be provided — interface (4).

3.1.3 Seamless topology

A flat topology provides a single means for the information and information exchange methods regardless of the location. In addition a “sub wind power plant” may be connected.

NOTE Depending on the requirements all five possibilities (in any combination) are likely to be used in one or the other way (may be at the same time). This standard can be applied to any topology. The standard may also be applied inside a single wind turbine generator system to provide communication between the intelligent subsystems (e.g., generator controller, generator controller, ...).

3.2 Location of information and services

One major issue to be decided is:
Where shall the information models and the service models be located (close to the communication software or in the application)?
With the current approach provided by NettedAutomation and Tamarack the information models and the service models are integrated into the communication software of the server. Two extreme and one mixed possibilities are depicted in Figure 4. **The architecture shown at the top is applied in the project.** The interface between the services (service models) and the information model is hidden. The information model is integrated in the provided server software. The interface to the application is realised as a "value interface" implemented as a DLL interface (the DLL provides all software on top of the Winsocket interface of Windows..

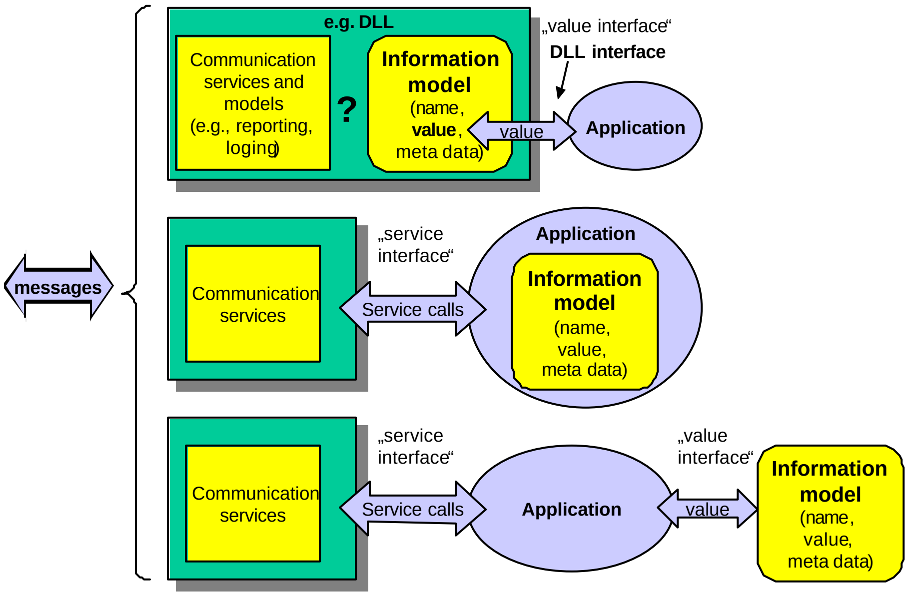

Figure 4 – Interface between communication and application (server)

The inter-process-communication (IPC) between the communication software and the application depends mainly on the requirements and operating system used. There is no way to find "the best approach that fits every size".

NOTE – The approach how to interface with the communication software is independent of the protocols used to communicate between clients and server.

The approach of Dynamic Link Libraries (DLLs) has been chosen for the project.

The server devices are usually less powerful than computers in control centers. Therefore the software architecture and the software running on servers are crucial with regard to real-time behaviour and performance.

3.3 Client interface

The focus for the tests is on the server side and there are a couple of important issues on the client side that is not dealt with in this project. For example, how can automatic data storage of collected data values be implemented, and how can reports be automatically created. Furthermore, how shall information be handled for display to the operator and how can commands from the operator be executed using MMS.

3.4 Modelling approach:

In the existing SCADA data from all wind power plants, including the data from Sigvards 2, is presented in a similar way to the control centre operator. The HMI interface shows grouped data values from the wind power plant. The logical grouping is based on the physical origin of the data, i.e. from which part of the wind turbine generator system the data comes from. See figure 5.

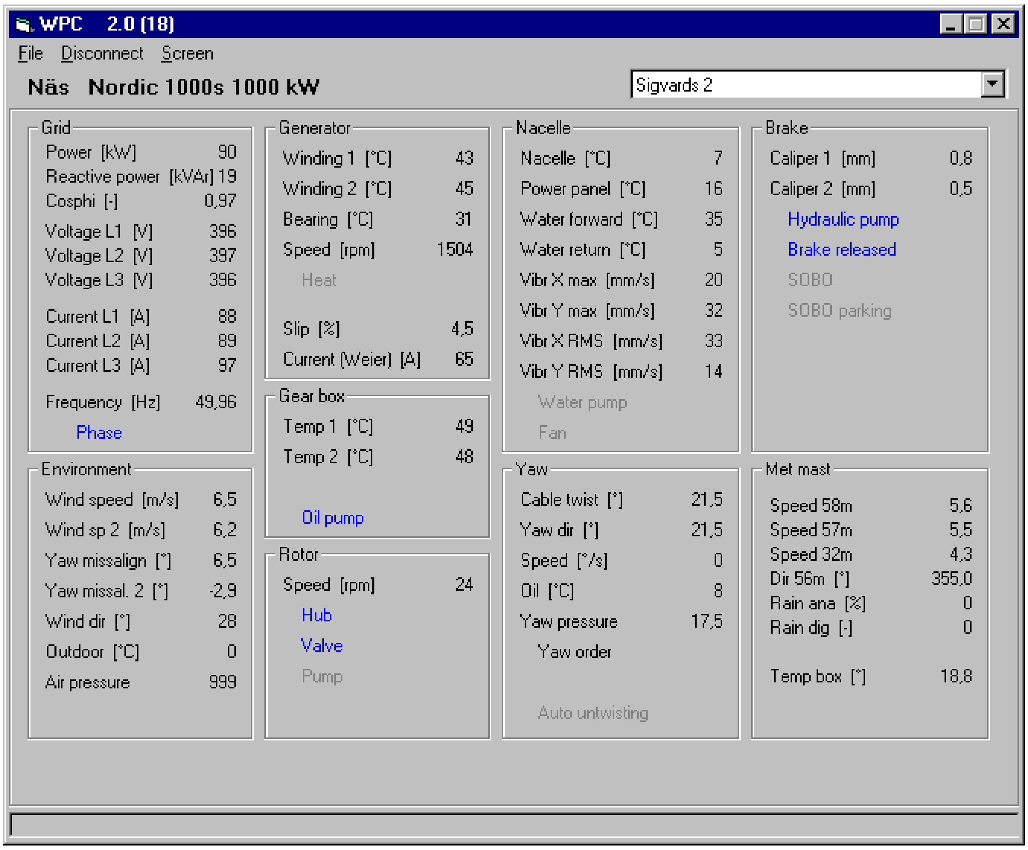

Figure 5 – HMI presentation of data from Sigvards 2 (Nordic1000).

The modelling approach of IEC 61850 is based on the concept of Logical Nodes as described IEC61850-5 and IEC 61850-7. These logical nodes interact to perform monitoring and control functions. In order to use IEC 61850 for wind power plant communication one or several logical nodes needed to be defined for Sigvards 2.

The grouping after physical components (in line with the existing ComServer client) is also applied as the modelling approach in this project. Each data is allocated to one of nine logical nodes (defined in IEC61850-x).

To produce a more detailed model with a hierarchical structure of a large number of logical nodes was not feasible within this project but will be needed for a future standard.

3.5 Implementation approach

The implementation of the information model and the server that hosts the model is shown in Figure 6. The process data packets (provided as arrays of variables from a Visual basic program) on the right hand side are moved to the server DLL.

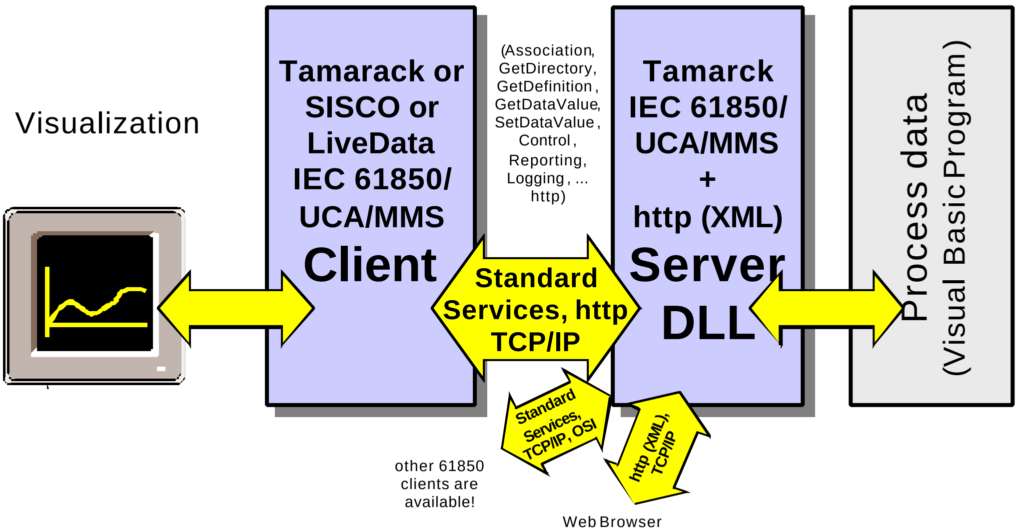

Figure 6 – Mapping approach (overview)

The DLL approach has been chosen to allow an easy integration of the server into the existing software environment. The server "serves" one or more clients for real-time information exchange implemented in the IEC 61850/UCA/MMS server.

The server DLL includes a Web server for the HTTP access (exchanging HTML and XML coded pages) providing values in a non-real-time manner.

More powerful ways to integrate the IEC 61850 compliant software are possible with other IPCs (inter process communication) and other platforms. See Figure 7 for an example of other approaches.

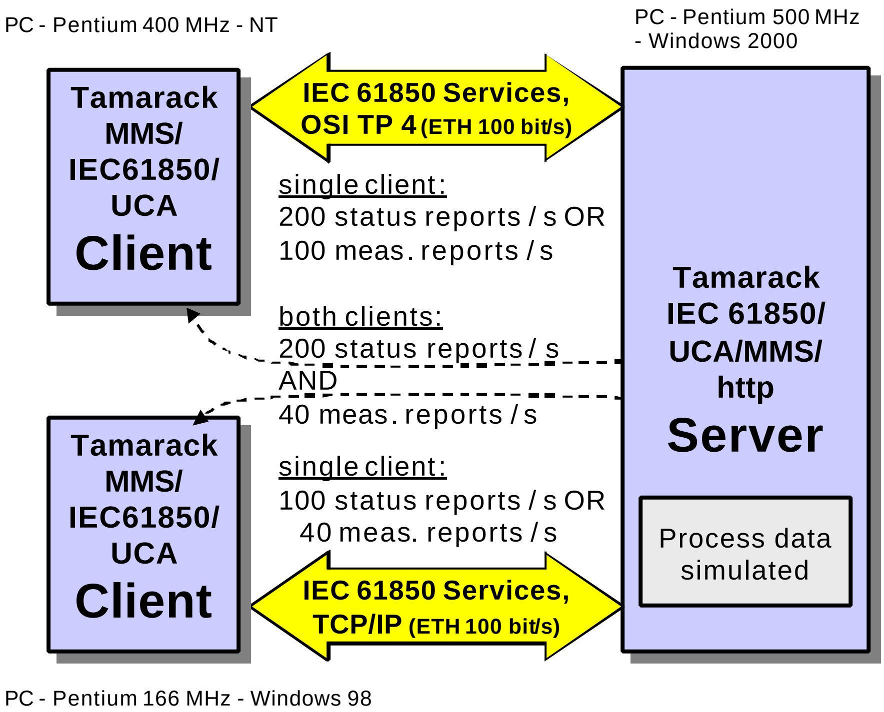

Figure 7 – Performance measurements for other (non-DLL) integration approach

Other implementations have shown that up to 1000 messages (e.g., reports) per second can be exchanged with one server.

4 Description of the test system implementation

4.1 Test server

The data model is implemented within the server software and connected to the real time data source under Windows NT. The data source Visual Basic Mita-MMS ComServer Adapter (VB-MMCA) interfaces with the Tamarack server software via a build-in IPC (inter process communication). This IPC and the server software is implemented as a Microsoft Windows DLL. The VB-MMCA software writes/updates the real time values to the DLL; the rest is handled by the DLL (server).

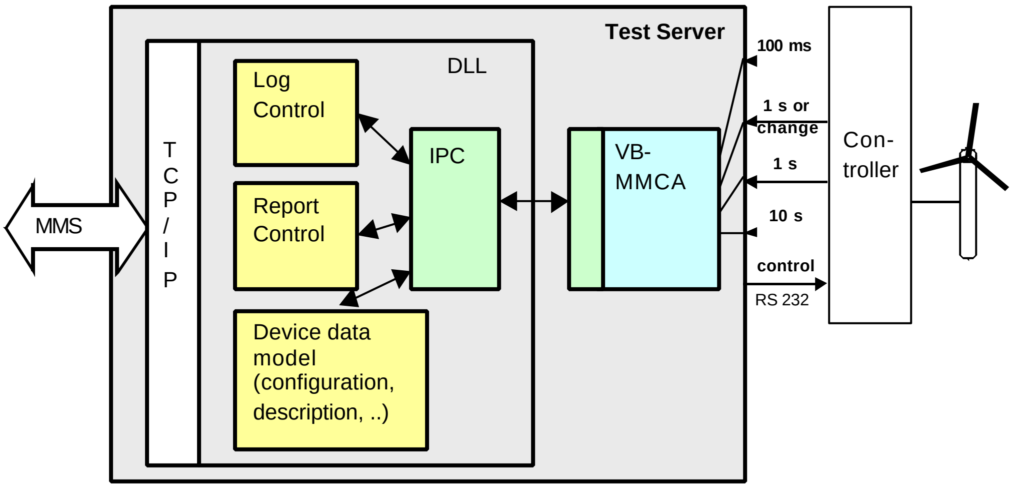

Figure 8: Test server PC.

4.1.1 MMS Server DLL

Figure 9 depicts the interface between the Visual Basic application and the server DLL. The VB-MMCA application controls the DLL. The first action is to open the DLL (openDLL). OpenDLL is called at the start of the application program for internal initiations in the MMS server DLL and a closeDLL call is initiated when the program is closed in order to close the program in a correct way. The process data values are stored in the server applying the “storePacketx” calls. When all data values are stored, the application calls the “serviceDLL”. This call starts the processing of the server DLL. See also Appendix A.

The DLL runs only when it is called by the application. As a consequence of this DLL-typical behavior, the server DLL processes incoming and outgoing messages as often as the application calls the server DLL.

When the serviceDLL is called all internal processing in the DLL, such as responding to Get Variable requests, sending reports, etc is conducted. Thus, the response times for client requests depend on how often this function is called. The server does not respond on a particular request until the serviceDLL has been called.

For the test, the "ServeDLL" is called every 75 millisecond. Thus the DLL runs just 12 times per second. This is fast enough for the test.
Note - The Server DLL applied in the test has to be called by the application. In a revised version of the Server DLL (not used in the project) the incoming service requests will also call the DLL. The Server DLL will process the request immediately. This will improve the performance.

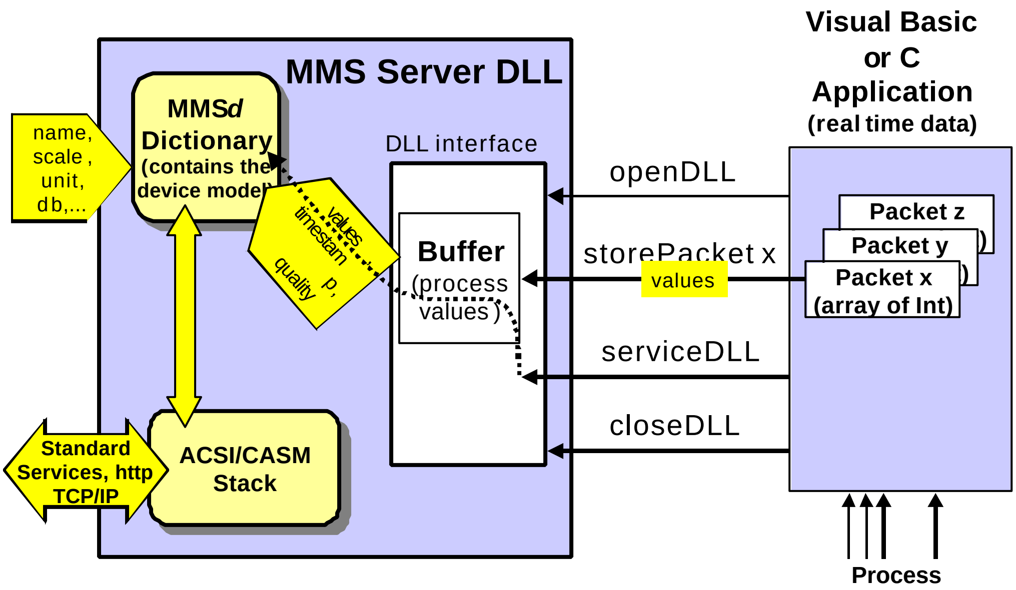

Figure 9 – Server implementation (overview)

More details of the DLL interface are shown in Figure 10. The packets (1, ...4) are first copied to the VB Test Server program that maps the data values received in the packets to the model-specific packets (e.g., WTur). For each logical node two "storeDLLpkxy" calls are defined, one for measurand and one for status information. The time stamp, scale and unit are located in the Server DLL. The value of the timestamp is added to the process data as soon as they are stored in the DLL.

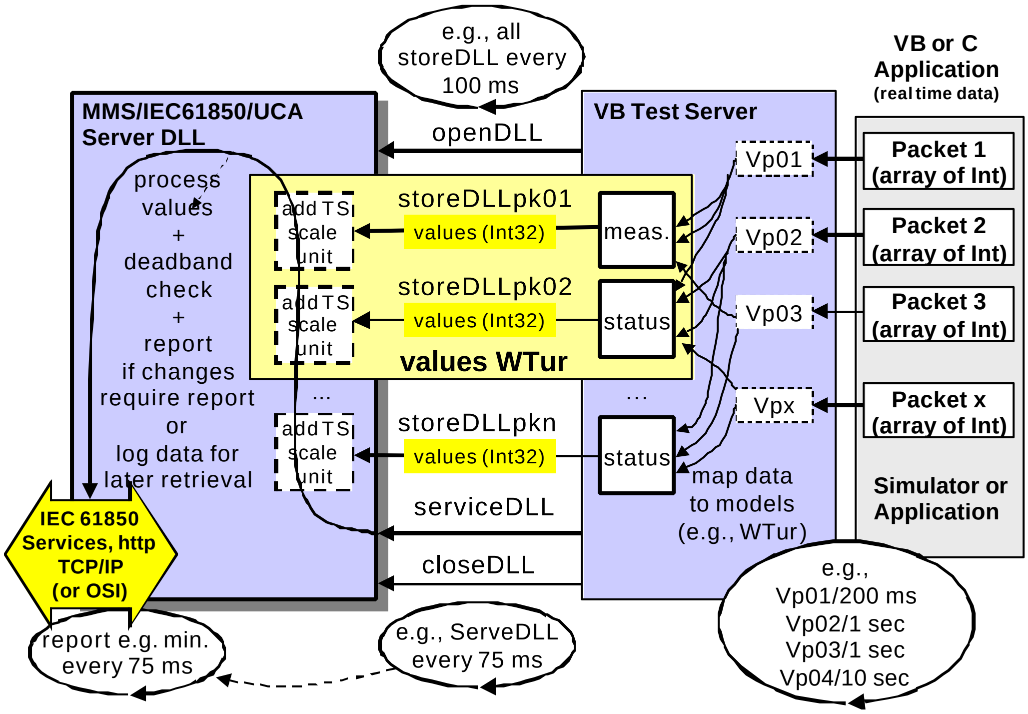

Figure 10 – Server details

In simulated mode (real application is replaced by a simulation program) the four packets will be filled up with increasing values and then be moved to the DLL packets.

4.1.2 Source of real-time data

The mapping of the real-time data values that are communicated with the WPP device model is depicted in Figure 11. The communication with the MMS DLL server is handled by the "Store service". The DLL provides for each independent set of data (logical node) two services: one for measurands and one for status.

As basis for the Visual Basic program, functions and subroutines of the existing ComServer program, used in the present supervision system developed by Vattenfall, are reused. They handle data received over serial link connections (RS232) from the control systems of the wind power plants and stores the data in ordered lists for future processing. In order to deliver the data to the DLL the data values in the list for the test object is reorganised so that the data for each defined logical node can be delivered in separate service calls (storeDLL).

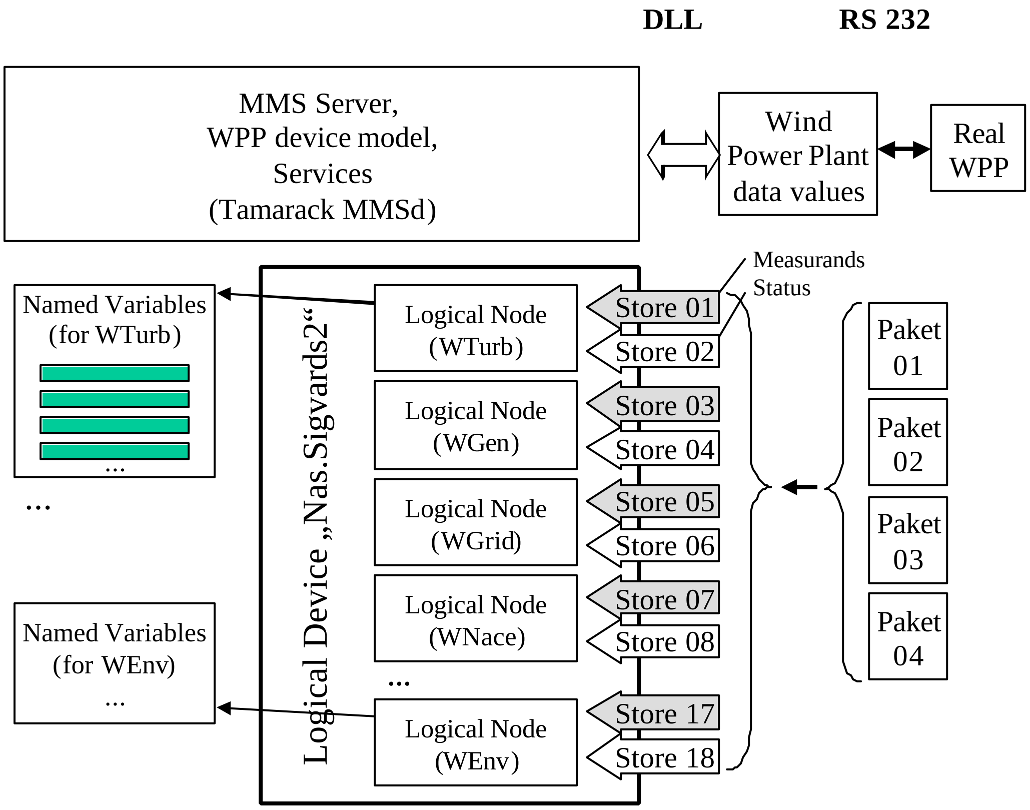

Figure 11 – Source of data and mapping

After storing the real data values to the MMS Named Variables, the communication with the MMS client is as defined in the IEC 61850 and UCA 2.0 standard (providing, e.g., services like Get, Set, Reporting, Logging, ...).

4.1.3 Information Exchange Services

The test server provide the following services through the IEC61850/UCA interface based on the MMS protocol:

- Online services (polling - Get/Read -, reporting, and setting)
- Logging (off-line)
- Controlling the device (stop, resume, start)
- Identification and self-description of data objects

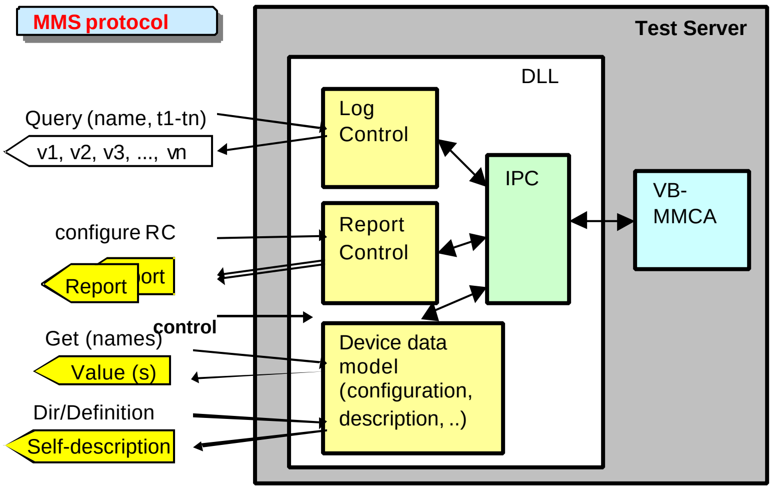

Figure 12: MMS Services

4.1.4 Server Software

The server includes the following software for the MMS interface:

Wpp_srv.dll (344 kB)
Server DLL that implements the whole information model as well as the IEC 61850 service models and services. The server DLL communicates via TCP/IP (Win-socket interface in Windows) with the client. The Server DLL also contains the Web server providing the HTTP.

WppTest (.exe, .vbp, .frm, .vbw)
Visual Basic simulation program that allows to run the server without connecting to the real application program. This simulation provides the same data values as the real application. The simulation allows to automatically increment the simulated values or to enter values manually.

In order to provide a web interface the following software is included:

root.htm, trailer.htm, header.htm
To act as a web server the DLL needs these Javascript files in the same directory in which the DLL is stored, for the transmission of the values via html pages.

lib1.js, lib2.js
To act as a web server the DLL needs these files in the same directory in which the DLL is stored, for the visualization of the received html coded values.

Logo.gif, automation_klein_2001_01_ss.gif, anima_unten.jpg, anim_mitte_neu.gif, anim_oben.jpg
When the server is acting as a web server these files are some common elements like the NettedAutomation logo placed on the web browser in the client.

The web interface is further described in Appendix E.

4.2 Test clients (MMI)

The clients used to demonstrate the capabilities of the server are:

- Tamarack's general test client
- Tamarack's logging test client
- SISCO's MMS explorer
- LifeCycle's MMS client with the access from Windows based applications (e.g., Excel, Word) via DDE channels.

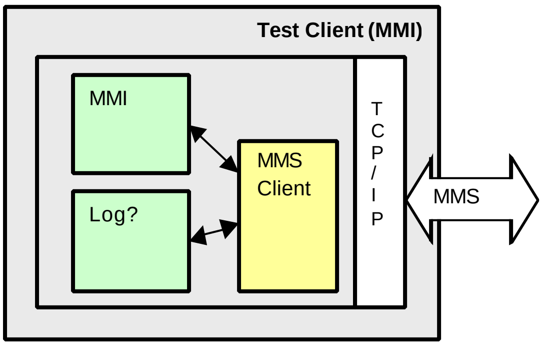

Figure 13: Test client PC.

4.3 Presentation of data (MMI)

Figure 14 and 15 show the same data in two different ways. The SISCO MMS Object Explorer shows the data using a hierarchical structure whereas the Tamarack client shows the chosen data in a more simple way.

IEC61850 does not define how data shall be used and presented by a client software. Nor does it define any application program interface (API). Thus, different MMI client software have different ways of showing the data.

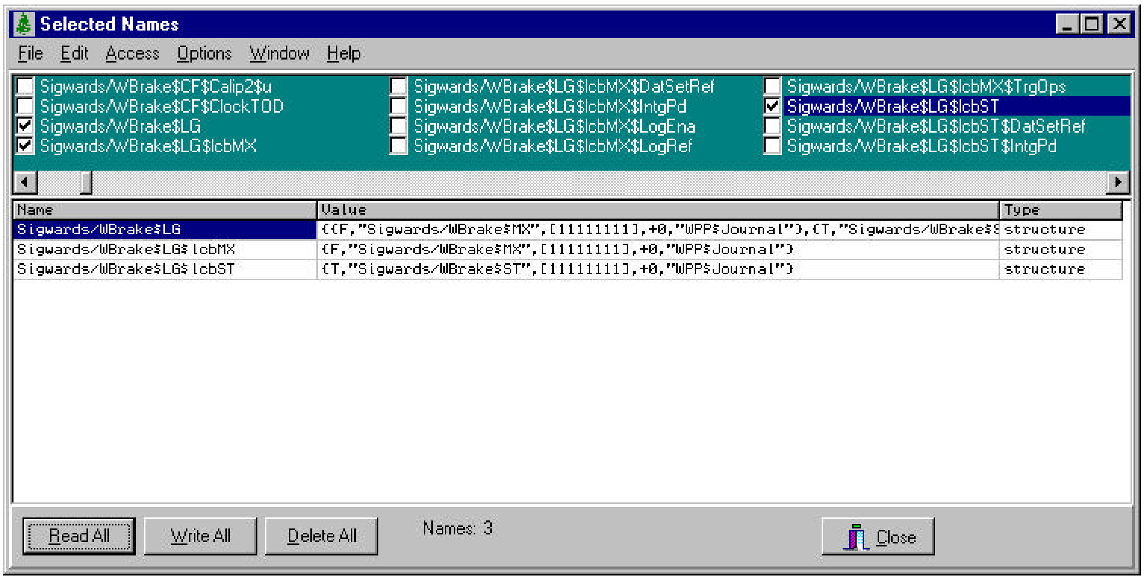

Figure 14: Tamarack client.

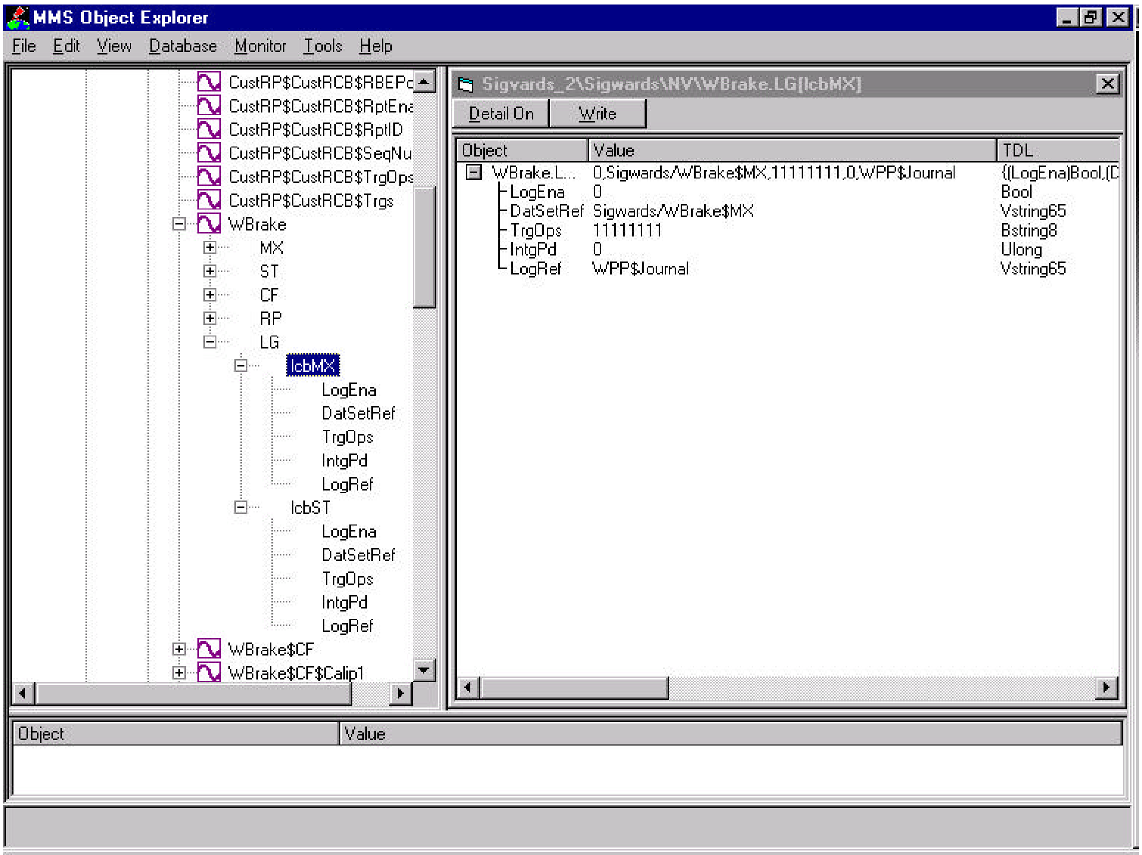

Figure 15: SISCO MMS Object Explorer client.

5 Software packages

This clause includes short descriptions of the software packages used. More information can be found in the product descriptions for the Tamarack [13], SISCO [14], and LiveData software [15] as well as in the Mita-Teknik WP3000 manual [16].

5.1 Tamarack

Tamarack provides a server (MMS*d*) and a general test client. During the course of the project they have also developed and provided a client test tool for testing of the logging services.
NettedAutomation has provided the WPP information model for the MMS*d* package and acted as the system integrator.

5.2 LiveData

The LiveData client can be connected to several servers at the same time. The LiveData server supports all CASM (and most of IEC61850) services. The client supports basic services only. LiveData's focus is on servers!
The LiveData client provide the possibility to include links in an Excel spreadsheet or to make the connection from a VB object using the reference.

5.3 SISCO

The SISCO server supports all CASM (and most of IEC61850) services including log Model. The log model is mapped to the MMS journaling model.
The MMS Object Explorer client supports all CASM (and most of IEC61850) services including log model.

6 References

[1] Bjerge C, et al; ”Functional requirements on Communication System for Wind Turbine Applications”, Report 01:25, Elforsk, Stockholm, June 2001.

[2] IEC TC57, Draft standard “IEC 61850 - Communication networks and systems in substations” – Part 1 to Part 10. For present status see www.iec.ch.

  [2.1] IEC, IEC 61850-5 “Part 5: Communication requirements for functions and device models”

  [2.2] IEC, IEC 61850-7-2 “Part 7-2: Basic communication structure for substations and feeder equipment - Abstract communication service interface (ACSI)”

  [2.3] IEC, IEC 61850-8-1 “Part 8-1: Specific communication service mapping (SCSM) - Mapping to MMS (ISO/IEC 9506 Part 1 and Part 2)”

[3] ISO/IEC 9506-1:1990 Industrial automation systems -- Manufacturing Message Specification -- Part 1: Service definition 1

[4] ISO/IEC 9506-2:1990 Industrial automation systems -- Manufacturing Message Specification -- Part 2: Protocol specification 2

[5] Tamarack Software Overview, <<ARCH.DOC>>

[6] Tamarack Development Environment <<Tools.doc>>

[7] User's Guide to MMSdPREP <<Prep20.doc>>

[8] MMSd Protocol Implementation Conformance Statement <<MMSPICS.doc>>

[9] IEEE “IEEE-SA TR 1550 - IEEE-SA Technical Report on Utility Communications Architecture (UCATM)”, November 1999.

[10] Internet RFC 793 et al., TCP – Transmisson Control Protocol

[11] Internet RFC 791 et al., IP – Internet Protocol

[12] W3C- World Wide Web Consortium, “Extensible Markup Language (XML) 1.0”, Feb, 1998, (Second edition Oct 2000), www.w3.org.

[13] Tamarack software product description. See also www.tamarack.com.

[14] SISCO software product description. See also www.sisconet.com.

[15] LiveData software product description. See also www.livedata.com.

[16] Mita-Teknik WP3000 Manual, Communication, 7.21.

Appendices

A Implementation of the MMS-server DLL in the Visual Basic MITA/MMS ComServer adapter

A.1 MITA/ComServer adapter for WPP Sigvards

Functions developed for the Visual Basic implementation used for the existing SCADA system was used as basis for the Visual basic program that interact with the MMS server dll. The VB object receives data from the wind power controller over an RS232 link and stores this data in lists for further processing. The VB object then regroups the data so that data for each logical node is forward using a separate function call.

The following function calls are used to deliver different kind of data to the different logical nodes:

```
> Private Declare Function storeDLLpk01 Lib "wpp_srv.dll" Alias "_storeUCA01@4" (ByRef pk01 As Long) As Long
> Private Declare Function storeDLLpk02 Lib "wpp_srv.dll" Alias "_storeUCA02@4" (ByRef pk02 As Long) As Long
> Private Declare Function storeDLLpk03 Lib "wpp_srv.dll" Alias "_storeUCA03@4" (ByRef pk03 As Long) As Long
> Private Declare Function storeDLLpk04 Lib "wpp_srv.dll" Alias "_storeUCA04@4" (ByRef pk04 As Long) As Long
> Private Declare Function storeDLLpk05 Lib "wpp_srv.dll" Alias "_storeUCA05@4" (ByRef pk05 As Long) As Long
> Private Declare Function storeDLLpk06 Lib "wpp_srv.dll" Alias "_storeUCA06@4" (ByRef pk06 As Long) As Long
> Private Declare Function storeDLLpk07 Lib "wpp_srv.dll" Alias "_storeUCA07@4" (ByRef pk07 As Long) As Long
> Private Declare Function storeDLLpk08 Lib "wpp_srv.dll" Alias "_storeUCA08@4" (ByRef pk08 As Long) As Long
> Private Declare Function storeDLLpk09 Lib "wpp_srv.dll" Alias "_storeUCA09@4" (ByRef pk09 As Long) As Long
> Private Declare Function storeDLLpk10 Lib "wpp_srv.dll" Alias "_storeUCA10@4" (ByRef pk10 As Long) As Long
> Private Declare Function storeDLLpk11 Lib "wpp_srv.dll" Alias "_storeUCA11@4" (ByRef pk11 As Long) As Long
> Private Declare Function storeDLLpk12 Lib "wpp_srv.dll" Alias "_storeUCA12@4" (ByRef pk12 As Long) As Long
> Private Declare Function storeDLLpk13 Lib "wpp_srv.dll" Alias "_storeUCA13@4" (ByRef pk13 As Long) As Long
> Private Declare Function storeDLLpk14 Lib "wpp_srv.dll" Alias "_storeUCA14@4" (ByRef pk14 As Long) As Long
> Private Declare Function storeDLLpk15 Lib "wpp_srv.dll" Alias "_storeUCA15@4" (ByRef pk15 As Long) As Long
> Private Declare Function storeDLLpk16 Lib "wpp_srv.dll" Alias "_storeUCA16@4" (ByRef pk16 As Long) As Long
> Private Declare Function storeDLLpk17 Lib "wpp_srv.dll" Alias "_storeUCA17@4" (ByRef pk17 As Long) As Long
> Private Declare Function storeDLLpk18 Lib "wpp_srv.dll" Alias "_storeUCA18@4" (ByRef pk18 As Long) As Long
```

Each node is represented by a number. The following numbers and definitions are used for the data lists:

```
> Const MAX_PK01 = 8     ' Length of MX packet Turbine
> Const MAX_PK02 = 7     ' Length of ST packet Turbine
> Const MAX_PK03 = 6     ' Length of MX packet Generator
> Const MAX_PK04 = 2     ' Length of ST packet Generator
> Const MAX_PK05 = 10    ' Length of MX packet Grid
> Const MAX_PK06 = 0     ' Length of ST packet Grid
> Const MAX_PK07 = 9     ' Length of MX packet Nacelle
> Const MAX_PK08 = 0     ' Length of ST packet Nacelle
> Const MAX_PK09 = 1     ' Length of MX packet Gear
> Const MAX_PK10 = 0     ' Length of ST packet Gear
> Const MAX_PK11 = 1     ' Length of MX packet Brake
> Const MAX_PK12 = 0     ' Length of ST packet Brake
> Const MAX_PK13 = 1     ' Length of MX packet Rotor
> Const MAX_PK14 = 0     ' Length of ST packet Rotor
> Const MAX_PK15 = 5     ' Length of MX packet Yaw
> Const MAX_PK16 = 0     ' Length of ST packet Yaw
```

> Const MAX_PK17 = 3 ' Length of MX packet Environment
> Const MAX_PK18 = 0 ' Length of ST packet Environment

> Dim DLLpacket01(MAX_PK01) As Long ' MX packet Turbine
> Dim DLLpacket02(MAX_PK02) As Long ' ST packet Turbine
> Dim DLLpacket03(MAX_PK03) As Long ' MX packet Generator
> Dim DLLpacket04(MAX_PK04) As Long ' ST packet Generator
> Dim DLLpacket05(MAX_PK05) As Long ' MX packet Grid
> Dim DLLpacket06(MAX_PK06) As Long ' ST packet Grid
> Dim DLLpacket07(MAX_PK07) As Long ' MX packet Nacelle
> Dim DLLpacket08(MAX_PK08) As Long ' ST packet Nacelle
> Dim DLLpacket09(MAX_PK09) As Long ' MX packet Gear
> Dim DLLpacket10(MAX_PK10) As Long ' ST packet Gear
> Dim DLLpacket11(MAX_PK11) As Long ' MX packet Brake
> Dim DLLpacket12(MAX_PK12) As Long ' ST packet Brake
> Dim DLLpacket13(MAX_PK13) As Long ' MX packet Rotor
> Dim DLLpacket14(MAX_PK14) As Long ' ST packet Rotor
> Dim DLLpacket15(MAX_PK15) As Long ' MX packet Yaw
> Dim DLLpacket16(MAX_PK16) As Long ' ST packet Yaw
> Dim DLLpacket17(MAX_PK17) As Long ' MX packet Environment
> Dim DLLpacket18(MAX_PK18) As Long ' ST packet Environment

Additional function calls to the DLL:

> Public Declare Function openDLL Lib "wpp_srv.dll" Alias "_openUCA@0" () As Long
> Public Declare Function closeDLL Lib "wpp_srv.dll" Alias "_closeUCA@0" () As Long
> Public Declare Function serviceDLL Lib "wpp_srv.dll" Alias "_serviceUCA@0" () As Long
> Public Declare Function enrollCtlUCA Lib "wpp_srv.dll" Alias "_enrollCtlUCA@4" (ByVal fcn As Long) As Long

OpenDLL is called at program start for internal initiations in the MMS server and closeDLL is called to end the program in a proper way. The function serviceDLL initiates all internal proccesing in the DLL. Thus the time intervall between calling serviceDLL set the speed for the whole MMS server.

A.2 Adapter for Meteorological station

In addition to the DLL for the wind power turbine a DLL for a meteorological station was implemented. The implementation for this DLL is similar to the one for the wind turbine but the met. station is not devided into several logical nodes, just one.

The following function call is used to deliver the data to the met.station logical node:

> Private Declare Function storeDLLpk01 Lib "wpp_srv.dll" Alias "_storeUCA01@4" (ByRef pk01 As Long) As Long 'Integer) As Long

Each node is represented by a number. The following numbers and definitions are used for the data lists:

> Const MAX_VPK05 = 26 ' Length of Vattenfall packet 05
> Global Vpacket05(MAX_VPK05) As Integer ' Vattenfall packet 05

B The Wind power plant information model

Foreword to appendix B and C

The contents of this appendix is based on the report ’Specification of the wind power plant information model based on IEC 61850 (UCA™2.0) and The implementation with Tamarack's MMSd” by Karlheinz Schwarz, NettedAutomation GmbH, Germany. The full report is part of the overall documentation of the project and can be distributed on request.

NettedAutomation GmbH has acted as sub-contractor during this project. Karlheinz Schwarz has made the development and implementation of the IEC 61850/MMS server, including modelling, implementation and Thorsten Greeb has made the neccesary programming work. During the implementation of the DLL server approach they have also had support from George Schimmel (Tamarack) who provided the basic server software. I would especially like to thank Karlheinz Schwarz for his contribution to this project. In addition to delivering a high quality implementation Karlheinz Schwarz has provided the project team with a huge amount of information, documents and papers which has been very valuable to the project.

Anders Johnsson
Vattenfall Utveckling

B.1 Introduction

This document provides the **information model of devices** for wind power plants (WPP) according to the architecture provided by Vattenfall Utveckling AB (Stockholm, Sweden). This model is used for the **test implementation** applying the MMS*d* server software provided by Tamarack (Ann Arbor, USA). The WPP **information model** and the mapping of the information model to the MMS data dictionary have been implemented by NettedAutomation (Karlsruhe, Germany).

The WPP information model has been mapped to MMS objects according to the latest drafts of IEC 61850-7-2, 7-3, 7-4, and IEC 61850-8-1. The data dictionary makes use of UCA definitions as well.

The WPP information model is based on the real-time data values carried by the message packets defined in the document WP3000 MANUAL (P99106/RE_NY by Mita-Teknik); updated on 10 April 2001 (Excel sheets, MITA_MMS.xls).

To keep the report small several companion documents have been written that provide more details on several aspects. These documents are listed at the beginning, and they are referenced in this report.

The project has been sponsored by Elforsk AB, Sweden.

It is my wish that this report and the companion documents referenced in this report help the interested people in the wind power and other distributed energy application domains to understand the approach provided with the Standard IEC 61850 of IEC TC 57 WG 10-12.

Dipl.-Ing. Karlheinz Schwarz
NettedAutomation GmbH
Im Eichbaeumle 108
D-76139 Karlsruhe
Germany
Phone +49-721-684844
Fax +49-721-679387
karlheinz.schwarz@nettedautomation.com
www.nettedautomation.com

B.2 References of input documents for the project

WP3000	*MANUAL (P99106/RE_NY by Mita-Teknik) <<E07_21.DOC>>*
Update	*Updated messages as per 2001-04-18 <<MITA_MMS.xls>>*
IEC 61850-7-1:	*Communication networks and systems in substations – Part 7-1: Basic communication structure for substation and feeder equipment – Principles and models - (March 2001)*
IEC 61850-7-2:	*Communication networks and systems in substations – Part 7-2: Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI) - (March 2001)*
IEC 61850-7-3:	*Communication networks and systems in substations – Part 7-3: Basic communication structure for substation and feeder equipment – Common data classes - (March 2001)*
IEC 61850-7-4:	*Communication networks and systems in substations – Part 7-4: Basic communication structure for substation and feeder equipment – Compatible logical node classes and data classes - (March 2001)*
IEC 61850-8-1:	*Communication networks and systems in substations – Part 8-1: Specific communication service mapping (SCSM) – Mapping to MMS(ISO/IEC 9506 Part 1 and Part 2) - (March 2001)*
IEEE TR 1550	*Utility Communications Architecture (UCA™)*
IEC 61400-25:	*Working Draft – Communications for monitoring and control of wind power plants*

NOTE The knowledge of the basic concepts and architecture of IEC 61850 and UCA 2.0 is required for the interpretation of the WPP information model and information exchange services.

The software applied in this project has been provided by:

Tamarack Consulting, Inc.
George Schimmel
1900 W. Stadium Blvd, Suite D
Ann Arbor, MI 48103
USA
Tel +1 (734) 761-8369
Fax +1 (734) 761-8383
gs@tamarack.com
www.tamarack.com

I would like to thank Anders Johnsson and Anders Anderson (both Vattenfall) for their contributions to this project, and George Schimmel (Tamarack) for his support during the implementation of the DLL server approach. A special thank goes to Thorsten Greeb (NettedAutomation) for his excellent programming work.

B.3 Companion documents related to this report

This report is accompanied by a set of other documents that provide details on some aspects only briefly discussed or just mentioned in this report. These companion documents are provided to give a more comprehensive view what the standard really provides and how it impacts the development or life cycle of software, devices, systems, and plants.

These documents are copyrighted by NettedAutomation GmbH (Karlsruhe, Germany) or SCC (Karlsruhe Germany).

These documents are intended for those readers that are interested to get more details and background information.

> For people that want to get a brief overview about the information that will be communicated and the methods how the information is exchanged are recommended to read clauses 3 - 7, and clause 13 of this document.

The other clauses provide more details with regard to the implementation and application.

Related documents:

[1] *Karlheinz Schwarz;* **Comprehensive overview**: IEC 60870-6-TASE.2 - The Inter-control Center Communication Protocol (ICCP) and IEC 61850 - Communication networks and systems in substations (UCA™); Seminar November 2001; <<Schwarz-Karlheinz_TASE2_IEC61850_2001-11-19.pdf>>

[2] *Karlheinz Schwarz;* Use of the standard **IEC 61850 outside the areas of Electrical Utilities**; November 2001; <<UCA-IEC61850_in-non-Utility-areas_2001-08-27.pdf>>

[3] *Karlheinz Schwarz;* **The life cycles of the standard IEC 61850**, standard-compliant devices, and plants applying these devices; November 2001; <<IEC61850_Life-Cycle-Point-of-view_2001-11-24.pdf>>

[4] *Karlheinz Schwarz;* Specification of the wind power plant information model based on IEC 61850 (UCA™2.0) and Discussion of the **application programmers interface versus protocol interface**; November 2001; <<Interfaces_2001-11-24.pdfc>>

[5] *Karlheinz Schwarz;* Experience with IEC 61850 compliant communication for Wind Power Plants – **IEC 61850 server and client implementations based on DLLs**; November 2001; <<IEC61850-DLL-Demo.pdf>>

[6] *Karlheinz Schwarz;* Specification of the wind power plant information model based on IEC 61850 (UCA™2.0) and **The use of Web technologies (HTTP, HTML, XML, and Javascript) in Tamarack's MMSd server**; November 2001; <<Data_Model_XML_2001-11-24.pdf>>

[7] *Karlheinz Schwarz;* **Comparison of IEC 60870-5-101/-103/-104 and IEC 60870-6-TASE.2 with IEC 61850**; November 2001; <<Comparison_101-61850_2001-08-28.pdf>>

[8] *Karlheinz Schwarz;* **IEC 61850 and UCA™ 2.0 or UCA™ 2.0 harmonized with IEC 61850 ?**; November 2001; <<Comparison-UCA-61850_2001-11-10.pdf>>

[9] *Karlheinz Schwarz;* **Seamless communication with IEC 61850 (UCA™) for distributed power generation**; November 2001; <<Seamless-Comm_with-61850_for-Windpower_2001-10-31.pdf>>

[10] *Tamarack Consulting*; **Information Package**; June 2001

[11] *NettedAutomation*; File with the complete high level Information model: "WPP.mdl" and "MetMast.mdl".

B.4 Context of the project

IEC TC 57 is working on a set of standards (IEC 61850 part 1 to 10) that define substation-specific and common information, information description methods, and information ex-change methods for monitoring and control systems for substations for power transmission and distribution. The first parts of the standard IEC 61850 have been published as International Standards. The other parts are in the process of standardization.

The project of Vattenfall and NettedAutomation uses the draft standards of IEC 61850 and the basic software and software tools developed in the context of these phases.

The main objectives of the project are:

— to **define the information model** of a wind power plant (for a Vattenfall wind power plant located on Gotland, Sweden).

— to **define the mapping of real-time data values to the information model** (WPP – wind power plant).

— to **implement and describe the mapping of the WPP information model to a MMS server** using the approach provided by Tamarack (MMS*d*).

— to **demonstrate the services** provided by the MMS server

— to **analyze the modeling approach** of IEC 61850, and analyze the use of IEC-61850-conformant software.

The **information** defined in the project comprises mainly wind power plant specific information like status, counters, measurands, and control information of various parts of a wind power plant, e.g., turbine, generator, gear, rotor, and grid.

NOTE The IEC 61850 standard focuses on the common, non-vendor-specific information. Those information items which tend to vary greatly between vendor-specific implementations can for example be specified in bilateral agreements, in user groups, or in amendments to the standard.

The object oriented **information description methods** allow precise and complete specification of the information.

The **information exchange methods** defines a client-server relation that provides:

- real-time data access and retrieval (polling),
- controlling devices,
- event/alarm reporting and logging (publisher/subscriber),
- self-description of devices (device data directory),
- data typing and discovery of data types, and
- file transfer

The **communication profile** comprises TCP/IP (Winsocket) relying on a proprietary link between a control computer and the wind power plant.

The main focus of this project is the information models, services and how to connect the applications with the IEC-61850-compliant software.

The report [3] describes the overall process from the first steps in the standardization process to the development of IEC-61850-compliant products and the use of these products.

B.5 Summary of the results of the project

General

The following sub-clauses list the results of the project viewed from the modeling and implementation point of view.

Information modeling

All process information exchanged between a WPP and the WPP control center has been modeled with the methods of IEC 61850-7-x. Description information (Metadata) has been added to the simple process values (e.g., SI units, scale, deadband).

The **specification** of the information model is independent of the information exchange services. Additional models can be specified at any time. After the specification and realization of the core WPP model there was a requirement to add a “meteorological information model”. This model has been integrated into the existing model without any change of the existing model.

For details see clauses B.7 to B.10.

Information exchange services

The information exchange methods used provide:

- real-time data access and retrieval,
- controlling devices,
- event/alarm reporting and logging,
- self-description of devices,
- data typing and discovery of data types, and
- file transfer

For details see clause B.14.

Tools to support the engineering and configuration of the server software

The Tamarack preprocessor tool provides a helpful tool (MMS*d* PREP) to engineer and configure the WPP information model. The MMS*d* PREP is a tool that processes ASCII based information model input files and produces the required program code source.

Interface between application and IEC-61850-compliant software

The selection of the interface between the application and the IEC-61850-compliant software is one of the most crucial issues to be decided when applying communication software.

The information model, the service models, and the services are provided as a single DLL. The DLL receives process data from the application for updating the state of the information model implemented in the server.

For details and discussion of this approach see [5].

B.6 Application scope of the approach analyzed in the project

The current communication solution between the WPP and the control center uses a proprietary communication protocol developed by Vattenfall (see Figure 1). The raw process values received via the RS 232 point-to-point link are processed by the proprietary program in the personal computer. The process values can be accessed by a proprietary protocol that communicates over TCP/IP. On the client side the proprietary software receives the process values as requested. The type of the process values, their identification and meaning (semantic) is hidden in the programs on both ends of the communication. Prior to the exchange of the values there needs to be an “exchange” of the semantic for the common understanding of the values exchanged.

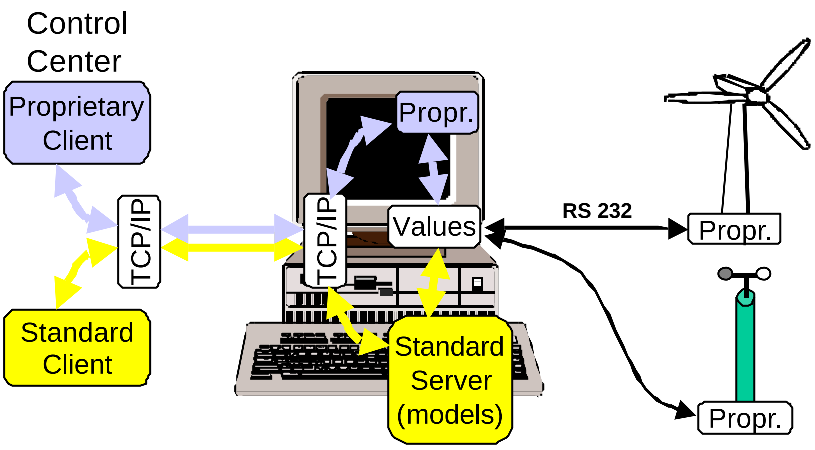

Figure 1 – Interface between communication and application (server)

For the project this **proprietary communication** has been replaced by the **standardized approach** of IEC 61850.

The major difference between the two approaches is that in the current solution (1) the **semantic is hidden** in the software, (2) the **services** and **messages** used to exchange information are **proprietary**.

The major benefit of the approach of IEC 61850 is the explicit definition of semantic and the standardized services and messages.

NOTE – The current proprietary solution meets today’s requirements. Retrofitting the proprietary solution would provide just a replacement of one solution by another. The standard IEC 61850 provides crucial benefits in the case of a **heterogeneous environment** where many **different systems** with a lot of non-standardized protocols have to co-operate. After all the standard reduces the number of communication solutions of one vendor, too.

B.7 Introduction and overview of the IEC 61850 modeling approach

Basic modeling approach

The basic principle of describing the information model is depicted in Figure 2. The server provides a **window** into existing information of a wind turbine generator system (WTGS). This window **represents** the real wind turbine generator system.

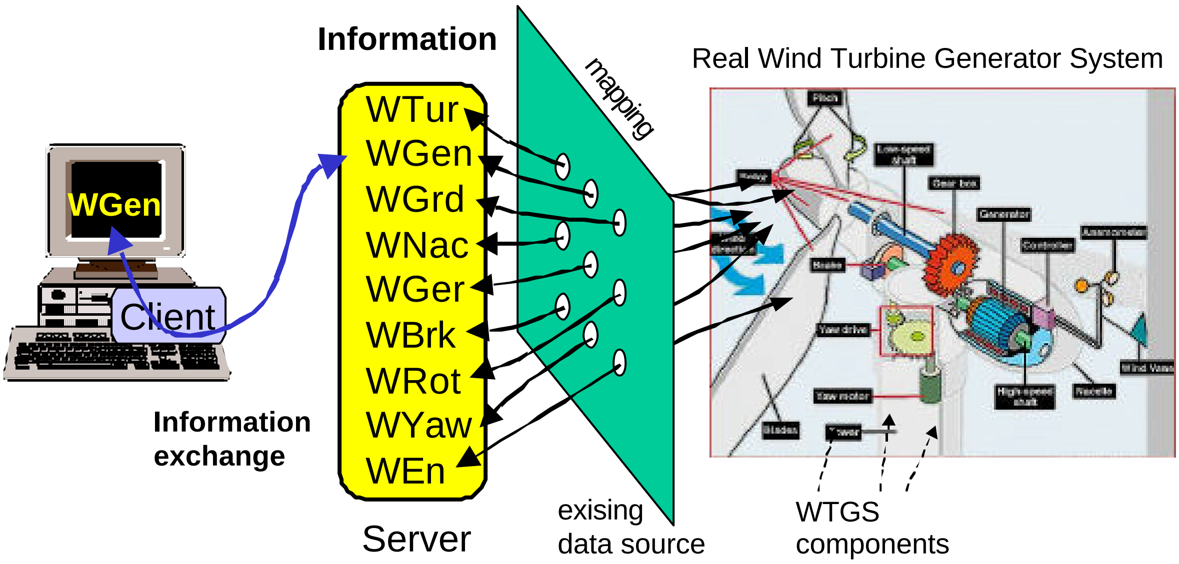

Figure 2 – Communication of existing information

The **existing information** is mapped to the **information model** in the server. How this mapping is realized is outside the scope of the standard IEC 61850; this mapping is described this report.

The information of the various components of the **WTGS** visible to the client is grouped as follows:

Group (name)	Short description
WTurb	general wind turbine information
WGen	wind generator information
WGrid	wind grid information
WNace	wind nacelle information
WGear	wind gear information
WBrake	wind brake information
WRotor	wind rotor information
WYaw	wind yaw information
WEnv	wind turbine information

These groups provide the information (e.g., generator speed, duty factor sent to generator, slip, generator current) that can be accessed with the information exchange methods (e.g., get, set, log, report).

The communication models and services and the information to be exchanged are being modeled applying an object oriented approach. Gear and generator, for instance, are be modeled as separate objects. Each may include measurements, calculated information (e.g., reactive power, power factor), and control services. The communication system provides a hierarchical naming method for the objects (measurements, etc).

EXAMPLE – The "generator speed" may be named "Vattenfall_Turbine5/WGen.MX.GenSpeed" or "Vattenfall_Turbine5/WGen.MX.DFacSToGen" may be the name for the duty factor sent to generator.

Before details of the information model is presented, the different levels of the standard IEC 61850 that are applied in the project are briefly discussed.

The levels of modeling in IEC 61850

The hierarchy of the models used in the standard IEC 61850 is depicted in Figure 3. The **information** (e.g. status and measurements for the wind turbine) to be exchanged for monitoring and control systems for wind power plants is shown at the top. The information comprises the most crucial definitions (**THE content**) of the whole project. In the middle the services (independent of the models and communication) are shown. The communication system on the bottom is independent of the services.

The three layers are independent of each other! This means different communication networks, e.g., RS 232 as well as 100 MBit/s Ethernet can be applied.

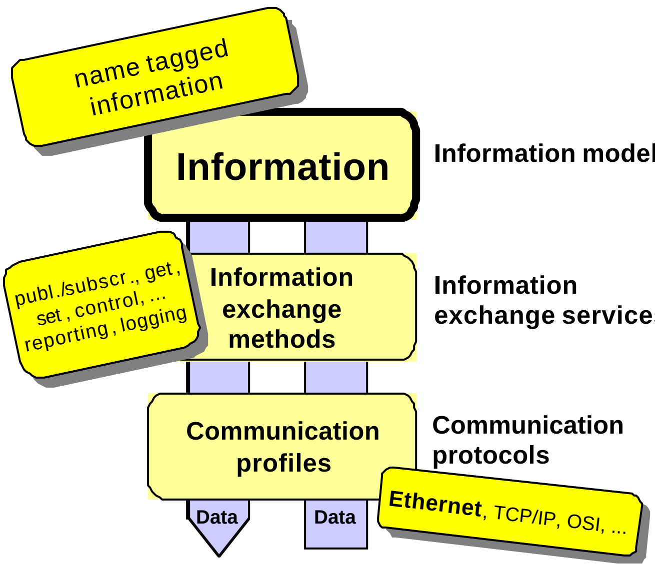

Figure 3 – Layers

The information exchange methods and the communication profiles – that are independent of the information models – are very important for real systems, because they carry the pay load of the information exchange. These aspects are discussed later in the report.

The next clauses focus mainly on the highest level, the information models.

Each of the above WTGS components comprises an list of **process information** (e.g., Generator speed, slip, ...) that needs to be defined precisely by a **name,** a **type,** and a **description** for the semantic. This allows sender and receiver of the exchanged data to interpret the data in a consistent way. The type and the description provide all details about the values required to for the exchange and interpretation.

Figure 4 depicts the "wrapping" or "information hiding" of the process value to be communicated in the center circle. The kernel and the second shell specify a data object. The data object has several attributes and additional characteristics that provide details for the value (or give more detail behind the process value: Metadata = data behind data):

- Name,
- Decription,
- Type (e.g., analog value)
- Functional constraint (e.g., value can be read, reported or controlled)
- Configuration information (e.g., SI unit, deadband for reporting)

The most outer shell is the information exchange method, e.g., set, get, report.

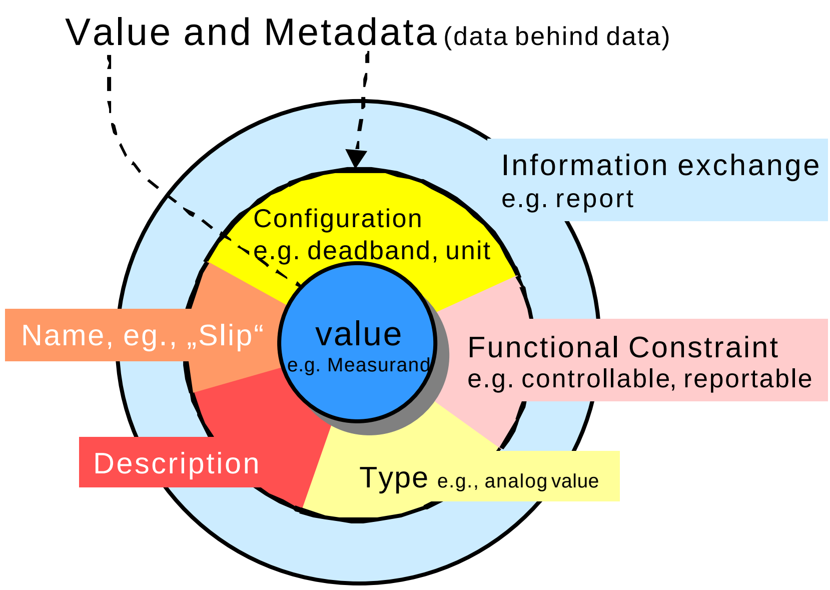

Figure 4 – Data object definition and Information hiding

A more detailed definition of the various attributes and the different levels of definition are depicted in Figure 5.

For various applications (status, analog, or controllable values) different sets of attributes for data objects (second shell) are defined. These sets are called Common Data Classes (CDC) in IEC 61850-7-3. Three CDCs are shown on the left hand side of the figure. Each CDC has a set of attributes (e.g, measured value, quality, timestamp, ... limits).

The highest level of the model are the lists of data objects that are grouped in so called logical nodes (WTGS components, e.g., wind generator). The logical nodes are lists of named data objects. Data objects are just instances of the CDCs. As shown in the figure, a data object "Pos" is derived from the CDC "controllable point". The logical node has a name, too. In the example of the figure it is named "XCBR".

Since a device may have several logical nodes derived from "XCBR" the implemented logical nodes may be assigned an instance-specific index: XCBR → XCBR17. The position of the circuit breaker (XCBR) with the number 17 will be designated as: XCBR.ST.Pos. ST indicates that the data is of the functional constraint "Status".

Below this modeling level, which comprises all information that describes the – external visible – information of the application, there is the level of exchange methods (services) which is defined in IEC 61850-7-2.

The services are still abstract. The services need to be mapped to a concrete application layer (e.g., MMS). The network needed for the exchange of the messages between devices is not shown at all. The main advantage of this modeling approach is the **abstraction** used.

During the definition of the information model we can – absolutely – ab-stract from any service and communication means.

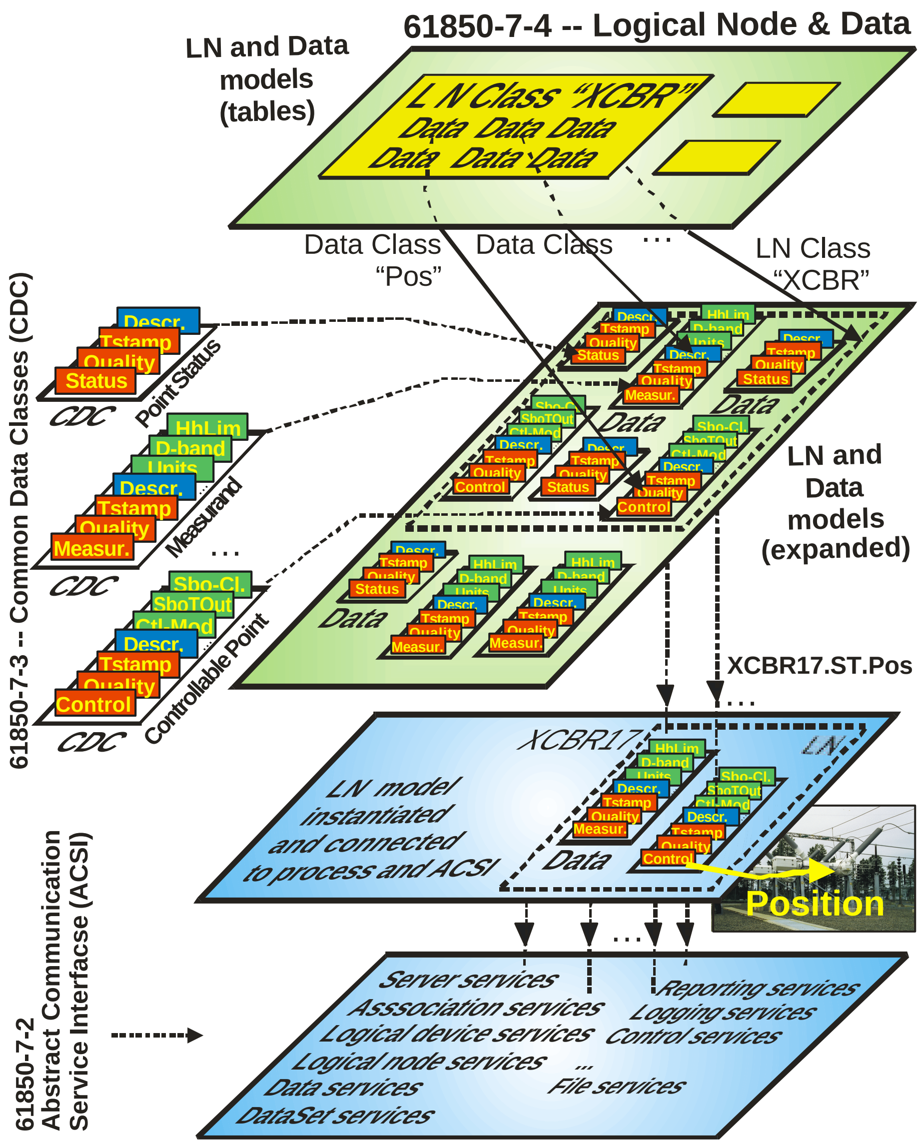

Figure 5 – Re-usability of common definitions

Now, after we went down close to the bits on the wire, we will return to the top level and stay there for a while.

The container for logical nodes and data objects

The information model for the WPP is located in a **logical device** "Sigvards2" as depicted in Figure 6. The logical device comprises all logical nodes of the model. The **logical nodes** contain data, data sets, and the reporting and logging objects. The logical device is contained in a server.

There are one server, one logical device, and nine logical nodes defined for the project.

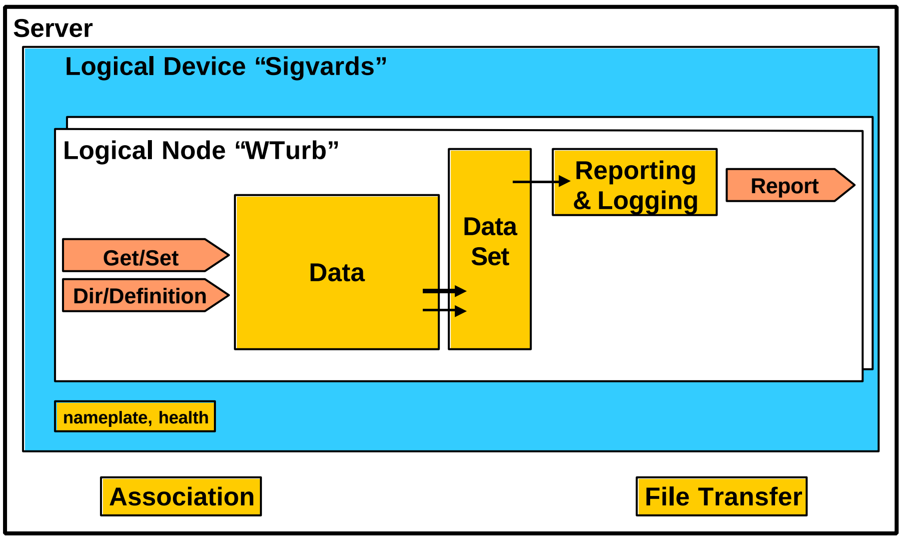

Figure 6 – Server building blocks

The logical device "Sigvards" is the representation of the wind power plant. The logical device hosts the following logical nodes:

Logical Node	**Description**
WTurb	logical node representing general wind turbine information
WGen	logical node representing wind generator information
WGrid	logical node representing wind grid information
WNace	logical node representing wind nacelle information
WGear	logical node representing wind gear information
WBrake	logical node representing wind brake information
WRotor	logical node representing wind rotor information
WYaw	logical node representing wind yaw information
WEnv	logical node representing wind turbine information

Additional logical nodes to describe system information are:

Logical Node	**Description**
LLN0	logical node zero – general info about the logical device
LPHD	Physical device information – general info about the real WPP

Each logical node contains a list of data objects.

Any other number of logical nodes may be defined as appropriate to the application or as defined in a standard (e.g., in the coming standard IEC 61400-25 of IEC TC 88 PT 25).

Logical nodes and process data

The following tables comprise all process data objects (measurands and status) modeled and implemented in the project. The data objects for the nine logical nodes are listed. The purpose of these lists is just to give an overview.

The mapping of the **real data** (according to WP3000 MANUAL) to the logical nodes can be found in chapter B.13.

LN Wind turbine (LN WTurb)

The LN WTurb comprises the following list of measurands and status information:

Measurands	Status information
Energy G1	Free to yaw
Energy G2	Free to operate
Energy consumption	Free run
Total time	Safety chain
Time G1	Error
Time G2	Warning
Time with fault status	Remote control from info possible
Time with grid ok	Reset level
Time with wind for prod	Status Code
	Active fault code
	Active fault code2
	Active fault code3
	Active fault code4

LN Wind generator (LN WGen)

The LN WGen comprises the following list of measurands and status information:

Measurands	Status information
Generator speed	Thyristor opening
Duty factor sent to generator	Generator connected
Slip	Heat generator (order)
Gen current (Weier)	Status word from Weier
Gen bearing temp	
Generator temp	
Generator 2 temp	

LN Wind grid (LN WGrid)

The LN WGrid comprises the following list of measurands and status information:

Measurands	Status information
Power	Phase compensation (order)
Cosphi	
Voltage L1	
Voltage L2	
Voltage L3	
Current L1	
Current L2	
Current L3	
Reactive power	
Frequency	

LN Wind nacelle (LN WNace)

The LN WNace comprises the following list of measurands and status information:

Measurands	Status information
Acceleration X	Water pump (order)
Acceleration Y	Ventilator nacelle (order)
Vibration X max	
Vibration Y max	
Vibration X RMS	
Vibration Y RMS	
Nacelle temp	
Power panel temp	
Water to cooler temp	
Water from cooler temp	

LN Wind gear (LN WGear)

The LN WGear comprises the following list of measurands and status information:

Measurands	Status information
Gear oil temp Gear oil 2 temp	Oil pump (order)

LN Wind brake (LN WBrake)

The LN WBrake comprises the following list of measurands and status information:

Measurands	Status information
Caliper 1 Caliper 2	Brake proc. 50 Brake proc. 75 Brake proc. 199 Brake proc. 200 Disk brake 1 activated (order) Disk brake 2 activated (order) Soft brake (order) Hydraulic pump brake (order)

LN Wind rotor (LN WGRotor)

The LN WRotor comprises the following list of measurands and status information:

Measurands	Status information
Rotor speed Rotor position	Hydraulic pump hub (order) Hub hydraulic, pump (current feedback) Hub hydraulic, solenoid (current feedback)

LN Wind yaw (LN WYaw)

The LN WYaw comprises the following list of measurands and status information:

Measurands	Status information
Yaw pressure Yaw missalignment Yaw missalignment 2 Cable twist Yaw speed Yaw oil temp	Yaw CCW Yaw CW Auto rewinding cables Hydraulic pump yaw (order)

LN Wind environment (LN WEnv)

The LN WEnv comprises the following list of measurands and status information:

Measurands	Status information
Wind speed Wind speed 2 Outdoor temp Air pressure	Heat wind gauges (order)

Logical node details

General

The table notation of IEC 61850-7-4 is used as a basis for the following sub-clauses. The tables provide two additional columns “Unit” and “Scale”.

The tables specify the semantic (Data Description), the name (Data Class Name), the type (CDC – common data class), the indication if the data object is optional or mandatory, and Unit and Scale for measured values.

These tables comprise all details needed for the complete specification of the information models. These data objects expand when replacing the common data class (CDC) with the specification of the CDCs.

Logical node "WTurb" representing general wind turbine information

This logical node contains all data classes that represent the general wind turbine information.

Note The notation of tables "Table 2 – Basic Logical Node information ", "Table 4 – Measurands", and "Table 8 – Status information" refer to IEC 6850-7-4. Details of the data class structures of the data classes used can be found in clause B.10.

LN: Wind turbine **Name: WTurb**

Data Description	**Data Class Name**	**CDC**	**M/O**	**Unit**	**Scale** Note 1)
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Energy G1	WhG1	MV	M	kWh	1
Energy G2	WhG2	MV	M	kWh	1
Energy consumption	WhConspt	MV	M	kWh	1
Total time	TimeTotal	MV	M	h	0,1
Time G1	TimeG1	MV	M	h	0,1
Time G2	TimeG2	MV	M	h	0,1
Time with fault status	TimeFltSt	MV	M	h	0,1
Time with grid ok	TimeGridOk	MV	M	h	0,1
Time with wind for prod	TimeWndProd	MV	M	h	0,1
Table 8 – Status information					
Free to yaw	FreeToYaw	SPS	M		
Free to operate	FreeToOp	SPS	M		
Free run	FreeRun	SPS	M		
Safety chain	SafeChn	SPS	M		
Error	Error	SPS	M		
Warning	Warn	SPS	M		
Remote control from info possible	RemCtlInf	SPS	M		
Reset level	RstLvl	ISI	M		
Status Code	StCod	ISI	M		
Active fault code	ActvFltCod	ISI	M		
Active fault code2	ActvFltCod2	ISI	M		
Active fault code3	ActvFltCod3	ISI	M		
Active fault code4	ActvFltCod4	ISI	M		

Note 1 – The values for scale are configuration specific.

Details of the data type (CDC) can be found in clause B.9.

Data instances will be addressed as follows:

Sigvards/Wturb.MX.WhConspt

Note 1 – MMS named variables have "$" instead ".".

Note 2 – The MX (functional constraint in IEC 61850 = functional component in UCA™) is added in front of the data class name according to IEC 61850-8-1.

Logical node "WGen" representing wind generator information

This logical node contains all data classes that represent the wind generator information.

LN: Wind generator **Name: WGen**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Generator speed	GenSpeed	MV	M	rpm	1
Duty factor sent to generator	DFacSToGen	MV	M		
Slip	Slip	MV	M	%	0,1
Gen current (Weier)	GenA	MV	M	A	0,1
Gen bearing temp	GenBeTemp	MV	M	°C	1
Generator temp	GenTemp	MV	M	°C	1
Generator 2 temp	Gen2Temp	MV	M	°C	1
Table 8 – Status information					
Thyristor opening	Tyropen	SPS	M		
Generator connected	GenCon	SPS	M		
Heat generator (order)	HeatGen	SPS	M		
Status word from Weier	SWW	ISI	M		

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node "WGrid" representing wind grid information

This logical node contains all data classes that represent the wind grid information.

LN: Wind grid **Name: WGrid**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Power	Power	MV	M	kW	1
Current L1	APhsA	MV	M	A	1
Cosphi	CosPhi	MV	M	-	0,01
Voltage L1	VPhsA	MV	M	V	1
Voltage L2	VPhsB	MV	M	V	1
Voltage L3	VPhsC	MV	M	V	1
Current L1	APhsA	MV	M	A	1
Current L2	APhsB	MV	M	A	1
Current L3	APhsC	MV	M	A	1
Reactive power	VAr	MV	M	kVAr	1
Frequency	Hz	MV	M	Hz	0,01
Table 8 – Status information					
Phase compensation (order)	PhCom	SPS	M		

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node "WNace" representing wind nacelle information

This logical node contains all data classes that represent the wind nacelle information.

LN: Wind nacelle **Name: WNace**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 4 – Measurands					
Acceleration X	AccX	MV	M	m/s2	0,01
Acceleration Y	AccY	MV	M	m/s2	0,01
Vibration X max	VibXMax	MV	M	mm/s	1
Vibration Y max	VibYMax	MV	M	mm/s	1
Vibration X RMS	VibXRMS	MV	M	mm/s	1
Vibration Y RMS	VibYRMS	MV	M	mm/s	1
Nacelle temp	NaclTemp	MV	M	°C	1
Power panel temp	PwrPnlTemp	MV	M	°C	1
Water to cooler temp	WtrToClrTemp	MV	M	°C	1
Water from cooler temp	WtrFrmClrTemp	MV	M	°C	1
Table 8 – Status information					
Water pump (order)	WtrPump	SPS	M		
Ventilator nacelle (order)	VentNac	SPS			

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node "WGear" representing wind gear information

This logical node contains all data classes that represent the wind gear information.

LN: Wind gear **Name: WGear**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Gear oil temp	GeaOilTemp	MV	M	°C	1
Gear oil 2 temp	GeaOil2Temp	MV	M	°C	1
Table 8 – Status information					
Oil pump (order)	OilPump	SPS	M		

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node "WBrake" representing wind brake information

This logical node contains all data classes that represent the wind brake information.

LN: Wind brake **Name: WBrake**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Caliper 1	Calip1	MV	M	mm	0,1
Caliper 2	Calip2	MV	M	mm	0,1
Table 8 – Status information					
Brake proc. 50	Brake50	SPS	M		
Brake proc. 75	Brake75	SPS	M		
Brake proc. 199	Brake199	SPS	M		
Brake proc. 200	Brake200	SPS	M		
Disk brake 1 activated (order)	DiskBrk1	SPS	M		
Disk brake 2 activated (order)	DiskBrk2	SPS	M		
Soft brake (order)	SoftBrk	SPS	M		
Hydraulic pump brake (order)	HydPmpBrk	SPS	M		

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node "WRotor" representing wind rotor information

This logical node contains all data classes that represent the wind rotor information.

LN: Wind rotor **Name: WRotor**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Rotor speed	RotSpd	MV	M	rpm	0,1
Rotor position	RotPos	MV	M		
Table 8 – Status information					
Hydraulic pump hub (order)	HydPmpHub	SPS	M		
Hub hydraulic, pump (current feedback)	HubHydrPump	SPS	M		
Hub hydraulic, solenoid (current feedback)	HubHydrSole	SPS	M		

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node "WYaw" representing wind yaw information

This logical node contains all data classes that represent the wind yaw information.

LN: Wind yaw **Name: WYaw**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Yaw pressure	YawP	MV	M	MPa	0,01
Yaw missalignment	YawMalgmt	MV	M	°	0,1
Yaw missalignment 2	YawMalgmt2	MV	M	°	0,1
Cable twist	CablTwst	MV	M	°	0,1
Yaw speed	Yawspd	MV	M	°/s	0,1
Yaw oil temp	YawOilTemp	MV	M	°C	1
Table 8 – Status information					
Yaw CCW	YawCCW	SPS	M		
Yaw CW	YawCW	SPS	M		
Auto rewinding cables	AuRewCab	SPS	M		
Hydraulic pump yaw (order)	HydPmpYaw	SPS	M		

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node "WEnv" representing wind environment information

This logical node contains all data classes that represent the wind environment information.

LN: Wind environment **Name: WEnv**

Data Description	**Data Class Name**	**CDC**	**M/O**	**Unit**	**Scale**
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Wind speed	WindSpd	MV	M	m/s	0,1
Wind speed 2	WindSpd2	MV	M	m/s	0,1
Outdoor temp	OutdrTemp	MV	M	°C	1
Air pressure	AirPres	MV	M	hPa	0,1
Table 8 – Status information					
Heat wind gauges (order)	HeatWndGau	SPS	M		

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node zero (LLN0)

This logical node contains all data classes that represent the common information of the **logical device** and (may be) the **external equipment** (i.e. the power plant controller).

LN: Logical node node zero **Name: LLN0**

Data Description	Data Class Name	CDC	M/O
Table 2 – Basic Logical Node information			
Mode	Mode	ISC	M
Behaviour	Beh	ISI	M
Health	Health	ISI	M
Name plate	Name	PLATE	M
Operation hours	Operh	ISI	O
Run Diagnostics	Diag	SPC	O
LED reset	LEDRs	SPC	O
External equipment health (i.e. the power plant controller)	EEHealth	ISI	O
External equipment name plate (i.e. the power plant controller)	EEName	PLATE	O

Details of the data class structure can be found in clause B.10 and following clauses.

Logical node physical device information (LPHD)

This logical node contains all data classes that represent the common information of the **physical device** that hosts the logical device.

LN: Logical node node zero **Name: LLN0**

Data Description	Data Class Name	CDC	M/O
Table 2 – Basic Logical Node information			
Mode	Mode	ISC	M
Behaviour	Beh	ISI	M
Health	Health	ISI	M
Name plate	Name	PLATE	M

Details of the data class structure can be found in clause B.10 and following clauses.

B.8 Meteorological data information model

The meteorological data information model provides meteorological information. The information model contains just one logical node because usually this information is provided by a separate computer.

The logical node may also be implemented in any other information model.

This model has been implemented long after the other models had been completed. For this model no basic software had to be modified. The model has been added as another application running on the same software.

Any other model can be defined using the same services.

This logical node contains all data classes that represent the meteorological information.

LN: meteorological information **Name: WMet**

Data Description	Data Class Name	CDC	M/O	Unit	Scale
Table 2 – Basic Logical Node information					
Mode	Mode	ISC	M		
Behaviour	Beh	ISI	M		
Health	Health	ISI	M		
Name plate	Name	PLATE	M		
Resetable operation counter	OperCntRs	ISC	O		
Table 4 – Measurands					
Wind speed 10m	WindSpd10	MV	O	mm/s	100
Wind speed 38m	WindSpd38	MV	O	mm/s	100
Wind speed 54m	WindSpd54	MV	O	mm/s	100
Wind speed 75m	WindSpd75	MV	O	mm/s	100
Wind speed 96m	WindSpd96	MV	O	mm/s	100
Wind speed 120m	WindSpd120	MV	O	mm/s	100
Wind speed 145m	WindSpd145	MV	O	mm/s	100
Wind speed 40m	WindSpd40	MV	O	mm/s	100
Wind speed 56m	WindSpd56	MV	O	mm/s	100
Wind speed 77m	WindSpd77	MV	O	mm/s	100
Wind speed 98m	WindSpd98	MV	O	mm/s	100
Wind speed 122m	WindSpd122	MV	O	mm/s	100
Wind direction 40m	WindDir40	MV	O	°	0,1
Wind direction 56m	WindDir56	MV	O	°	0,1
Wind direction 77m	WindDir77	MV	O	°	0,1
Wind direction 98m	WindDir98	MV	O	°	0,1
Wind direction 122m	WindDir122	MV	O	°	0,1
Temperature 1,5 m	Temp1	MV	O	°C	0,1
Temperature 10 m	Temp10	MV	O	°C	0,1
Temperature 38 m	Temp38	MV	O	°C	0,1
Temperature 54 m	Temp54	MV	O	°C	0,1
Temperature 75 m	Temp75	MV	O	°C	0,1
Temperature 96 m	Temp96	MV	O	°C	0,1
Temperature 120 m	Temp120	MV	O	°C	0,1
Temperature 145 m	Temp145	MV	O	°C	0,1
Air pressure	AirPres	MV	O	hPa	0,1
Rain	Rain	MV	O	-	1

Details of the data class structure can be found in clause B.10 and following clauses.

B.9 Common data classes expanded

Introduction

The following common data classes (CDC) of IEC 61850-7-3 are applied for the WPP information model:

ISC – Controllable Integer Status (IEC 61850-7-3 clause 6.6.4).
ISI – Integer Status (IEC 61850-7-3 clause 6.4.4).
PLATE – Name Plate (IEC 61850-7-3 clause 6.4.7).
MV – Measured Value (IEC 61850-7-3 clause 6.5.2).
SPS – Single Point Status (IEC 61850-7-3 clause 6.4.2).

Examples of data objects are completely expanded in the following clause.

Note - The following tables show an excerpt only. Some optional information has been skipt.

Mode data object example

The following examples of the logical node "WGen" show how the common data classes are expanded (data classes and their attributes are shown in detail). The "Mode" data object (CDC = ISC) for example is composed of information belonging to

- "CO" (Control) → WGen.CO.Mode → with one component WGen.CO.Mode.ctlVal
- "ST" (Status) → WGen.ST.Mode → with four components stVal, q, q.validity, and t
- "CF" (Configuration) → WGen.CF.Mode → with one comp. WGen.CF.Mode.ctlModel

Table 2 – Basic Logical Node information

Description	**Data Name, Functional Constraint, and Attribute Name**	**Value**	**Note**
Mode (ISC – Controllable Integer Status)	WGen.CO.Mode WGen.CO.Mode.ctlVal WGen.ST.Mode WGen.ST.Mode.stVal WGen.ST.Mode.q WGen.ST.Mode.q.validity WGen.ST.Mode.t WGen.CF.Mode WGen.CF.Mode.ctlModel	*current* *current* *current* *current* *current* *current* *current* *current* *current*	Note 1
Behaviour (ISI – Integer Status)	WGen.ST.Beh WGen.ST.Beh.stVal WGen.ST.Beh.q WGen.ST.Beh.q.validity	*current* *current* *current* *current*	Note 1

Description	**Data Name, Functional Constraint, and Attribute Name**	**Value**	**Note**
	WGen.ST.Beh.t	*current*	
Health (ISI – Integer Status)	WGen.ST.Health WGen.ST.Health.stVal WGen.ST.Health.q WGen.ST.Health.q.validity WGen.ST.Health.t	*current* *current* *current* *current* *current*	Note 2
Name plate (PLATE – Name Plate)	WGen.ST.Name WGen.ST.Name.vendor WGen.ST.Name.serNum	*current* *current* *current*	

Note 1

Mode / Behavior definition (Mode can be controlled by client, Behavior is current value reported to client)

On *enabled*

Blocked *function active, but no outputs generated, no reporting, controls are rejected, functional and configuration data visible*

Test *function active, outputs generated, controls are accepted, reporting is flagged as test, functionality and configuration data are visible.*

Test/Blocked *function active, outputs generated, controls are accepted, reporting is flagged as test, functionality and configuration data visible.*

Off *disabled. No operation/function, objects are visible, configuration visible/accessible, functional data NOT visible, control commands are rejected (negative response, no reporting, no logging)*

Note 2

Health (this is an indication)

Ok *"green"*

Warning *"yellow"*

Alarm *"red"*

Measurand data object example

Each data object, e.g., "Generator speed "of common data class (CDC) "MV" (measured value) is composed of

- "MX" (Measurands) → WGen.MX.GenSpeed → with five components dbVal, dbVal.i, q, q.validity, and t
- "CF" (Configuration) → WGen.CF.GenSpeed → with five components WGen.CF.GenSpeed, ...units, ...sVC, ...sVC.iScaleFactor, ...db

Table 4 – Measurands

Description	Data Name, Functional Constraint, and Attribute Name	Value
Generator speed	WGen.MX.GenSpeed	*current*
(MV – Measured Value)	WGen.MX.GenSpeed.dbVal	*current*
	WGen.MX.GenSpeed.dbVal.i	*current*
	WGen.MX.GenSpeed.q	*current*
	WGen.MX.GenSpeed.q.validity	*current*
	WGen.MX.GenSpeed.t	*current*
	WGen.CF.GenSpeed	-
	WGen.CF.GenSpeed.units	Rpm
	WGen.CF.GenSpeed.sVC	-
	WGen.CF.GenSpeed.sVC.iScaleFact	1
	or	*current*
	WGen.CF.GenSpeed.db	

The complete Measured value Common Data Class (MV)

The following table shows all 17 possible attribute from which we have chosen 10 for the project.

MV Attribute Definition					
Name	**Type**	**FC**	**TrgOp**	**Value / Value Range**	**M/O**
mVal	AnalogueValue	sv	dchg		O
dbVal	AnalogueValue	mx	fchg		O
range	Range	mx	fchg		O
q	Quality	mx	qchg		M
t	TimeStamp	mx			M
subID	VISIBLE STRING	sv			O
configuration and description					
d	Description	dc		Text	O
units	SIUnits	cf			O
sVC	ScaledValueConfig	cf			O
db	INTEGER	cf		> 0	O
hhLim	REAL	cf			O
hLim	REAL	cf			O
lLim	REAL	cf			O
llLim	REAL	cf			O
min	REAL	cf			O

MV Attribute Definition					
Name	**Type**	**FC**	**TrgOp**	**Value / Value Range**	**M/O**
max	REAL	cf			O
smpRate	SampleRate	cf			O

Status data object example

Each data object, e.g., “Generator connected ”of common data class (CDC) “SPS” (single point status) is composed of

- “ST” (Status) → WGen.ST.GenCon → with four components stVal, q, q.validity, and t

Each data object, e.g., “Status word from Weier ”of common data class (CDC) “ISI” (integer status) is composed of

- “ST” (Status) → WGen.ST.SWW → with four components stVal, q, q.validity, and t

Table 8 – Status information

Description	Data Name, Functional Constraint, and Attribute Name	Value
Generator connected (SPS – Single Point Status)	WGen.ST.GenCon.stVal WGen.ST.GenCon.q WGen.ST.GenCon.q.validity WGen.ST.GenCon.t	*current* *current* *current* *current*
Status word from Weier (ISI – Integer Status)	WGen.ST.SWW.stVal WGen.ST.SWW.q WGen.ST.SWW.q.validity WGen.ST.SWW.t	*current* *current* *current* *current*

B.10 Logical node expanded (example)

The logical node "**WTurb**" is expanded (i.e., the CDC is applied, all CDC “names” are replaced by the detailed specification of the CDC) to show an example of all data that belong to the logical node and that can be accessed using the standardized services.

Only one logical node is expanded. These lists can be retrieved by the client through MMS services (GetName...).

The complete list of all 984 MMS NamedVariables of all logical nodes of the server can be found in B.15. The 984 MMS NamedVariables comprise all Data objects of all logical nodes as well as the NamedVariables that represent the attributes of the report and log control objects.

Example – NamedVariable: “Buffer Time” (Data Set = data to be reported) attribute of the measurement “MX” basic report control block “brcb”, of functional characteristic “RP”, of the logical node “WGrid”, of the logical device “Siegvards”:

455	Sigvards/WGridRPbrcbMX$BufTim
456	Sigvards/WGridRPbrcbMX$DatSet

To get an overview about the data objects, please check the B.15 for all detail. B.15 does provide the reference of the data objects only. The types (common data class) is listed in the following table in the first column in brackets, e.g., (ISC – Controllable Integer Status).

The other logical nodes could be expanded in the same manner.

Logical node "WTurb:

Description	**Data Name, Functional Constraint, and Attribute Name**	**Value**
Basic Logical Node information		
Mode	WTurb.CO.Mode	*current*
(ISC – Controllable Integer Status)	WTurb.CO.Mode.ctlVal	*current*
	WTurb.ST.Mode	*current*
	WTurb.ST.Mode.stVal	*current*
	WTurb.ST.Mode.q	*current*
	WTurb.ST.Mode.q.validity	*current*
	WTurb.ST.Mode.t	*current*
	WTurb.CF.Mode	*current*
	WTurb.CF.Mode.ctlModel	*current*

Description	Data Name, Functional Constraint, and Attribute Name	Value
Behaviour (ISI – Integer Status)	WTurb.ST.Beh WTurb.ST.Beh.stVal WTurb.ST.Beh.q WTurb.ST.Beh.q.validity WTurb.ST.Beh.t	*current* *current* *current* *current* *current*
Health (ISI – Integer Status)	WTurb.ST.Health WTurb.ST.Health.stVal WTurb.ST.Health.q WTurb.ST.Health.q.validity WTurb.ST.Health.t	*current* *current* *current* *current* *current*
Name plate (PLATE – Name Plate)	WTurb.ST.Name WTurb.ST.Name.vendor WTurb.ST.Name.serNum	*current* *current* *current*
Measurands		
Energy G1 (MV – Measured Value)	WTurb.MX.WhG1 WTurb.MX.WhG1.dbVal WTurb.MX.WhG1.dbVal.i WTurb.MX.WhG1.q WTurb.MX.WhG1.q.validity WTurb.MX.WhG1.t WTurb.CF.WhG1 WTurb.CF.WhG1.units WTurb.CF.WhG1.sVC WTurb.CF.WhG1.sVC.iScaleFactor WTurb.CF.WhG1.db	*current* *current* *current* *current* *current* *current* - kWh - 1 *current*
Energy G2 (MV – Measured Value)	WTurb.MX.WhG2 WTurb.MX.WhG2.dbVal WTurb.MX.WhG2.dbVal.i WTurb.MX.WhG2.q WTurb.MX.WhG2.q.validity WTurb.MX.WhG2.t WTurb.CF.WhG2 WTurb.CF.WhG2.units WTurb.CF.WhG2.sVC WTurb.CF.WhG2.sVC.iScaleFactor WTurb.CF.WhG2.db	*current* *current* *current* *current* *current* *current* - kWh - 1 *current*
Energy consumption (MV – Measured Value)	WTurb.MX.WhConspt WTurb.MX.WhConspt.dbVal WTurb.MX.WhConspt.dbVal.i WTurb.MX.WhConspt.q WTurb.MX.WhConspt.q.validity	*current* *current* *current* *current* *current*

Description	Data Name, Functional Constraint, and Attribute Name	Value
	WTurb.MX.WhConspt.t WTurb.CF.WhConspt WTurb.CF.WhConspt.units WTurb.CF.WhConspt.sVC WTurb.CF.WhConspt.sVC.iScaleFactor WTurb.CF.WhConspt.db	*current* - kWh - 1 *current*
Total time (MV – Measured Value)	WTurb.MX.TimeTotal WTurb.MX.TimeTotal.dbVal WTurb.MX.TimeTotal.dbVal.i WTurb.MX.TimeTotal.q WTurb.MX.TimeTotal.q.validity WTurb.MX.TimeTotal.t WTurb.CF.TimeTotal WTurb.CF.TimeTotal.units WTurb.CF.TimeTotal.sVC WTurb.CF.TimeTotal.sVC.iScaleFactor WTurb.CF.TimeTotal.db	*current* *current* *current* *current* *current* *current* - h - 0.1 *current*
Time G1 (MV – Measured Value)	WTurb.MX.TimeG1 WTurb.MX.TimeG1.dbVal WTurb.MX.TimeG1.dbVal.i WTurb.MX.TimeG1.q WTurb.MX.TimeG1.q.validity WTurb.MX.TimeG1.t WTurb.CF.TimeG1 WTurb.CF.TimeG1.units WTurb.CF.TimeG1.sVC WTurb.CF.TimeG1.sVC.iScaleFactor WTurb.CF.TimeG1.db	*current* *current* *current* *current* *current* *current* - h - 0.1 *current*
Time G2 (MV – Measured Value)	WTurb.MX.TimeG2 WTurb.MX.TimeG2.dbVal WTurb.MX.TimeG2.dbVal.i WTurb.MX.TimeG2.q WTurb.MX.TimeG2.q.validity WTurb.MX.TimeG2.t WTurb.CF.TimeG2 WTurb.CF.TimeG2.units WTurb.CF.TimeG2.sVC WTurb.CF.TimeG2.sVC.iScaleFactor WTurb.CF.TimeG2.db	*current* *current* *current* *current* *current* *current* - h - 0.1 *current*
Time with fault status (MV – Measured Value)	WTurb.MX.TimeFltSt WTurb.MX.TimeFltSt.dbVal WTurb.MX.TimeFltSt.dbVal.i	*current* *current* *current*

Description	Data Name, Functional Constraint, and Attribute Name	Value
	WTurb.MX.TimeFltSt.q WTurb.MX.TimeFltSt.q.validity WTurb.MX.TimeFltSt.t WTurb.CF.TimeFltSt WTurb.CF.TimeFltSt.units WTurb.CF.TimeFltSt.sVC WTurb.CF.TimeFltSt.sVC.iScaleFactor WTurb.CF.TimeFltSt.db	*current* *current* *current* - h - 0.1 *current*
Time with grid ok (MV – Measured Value)	WTurb.MX.TimeGridOk WTurb.MX.TimeGridOk.dbVal WTurb.MX.TimeGridOk.dbVal.i WTurb.MX.TimeGridOk.q WTurb.MX.TimeGridOk.q.validity WTurb.MX.TimeGridOk.t WTurb.CF.TimeGridOk WTurb.CF.TimeGridOk.units WTurb.CF.TimeGridOk.sVC WTurb.CF.TimeGridOk.sVC.iScaleFactor WTurb.CF.TimeGridOk.db	*current* *current* *current* *current* *current* *current* - h - 0.1 *current*
Time with wind for prod (MV – Measured Value)	WTurb.MX.TimeWndProd WTurb.MX.TimeWndProd.dbVal WTurb.MX.TimeWndProd.dbVal.i WTurb.MX.TimeWndProd.q WTurb.MX.TimeWndProd.q.validity WTurb.MX.TimeWndProd.t WTurb.CF.TimeWndProd WTurb.CF.TimeWndProd.units WTurb.CF.TimeWndProd.sVC WTurb.CF.TimeWndProd.sVC.iScaleFactor WTurb.CF.TimeWndProd.db	*current* *current* *current* *current* *current* *current* - h - 0.1 *current*
Status information		
Free to yaw (SPS – Single Point Status)	WTurb.ST.FreeToYaw.stVal WTurb.ST.FreeToYaw.q WTurb.ST.FreeToYaw.q.validity WTurb.ST.FreeToYaw.t	*current* *current* *current* *current*
Free to operate (SPS – Single Point Status)	WTurb.ST.FreeToOp.stVal WTurb.ST.FreeToOp.q WTurb.ST.FreeToOp.q.validity WTurb.ST.FreeToOp.t	*current* *current* *current* *current*
Free run (SPS – Single Point Status)	WTurb.ST.FreeRun.stVal WTurb.ST.FreeRun.q	*current* *current*

Description	Data Name, Functional Constraint, and Attribute Name	Value
	WTurb.ST.FreeRun.q.validity WTurb.ST.FreeRun.t	*current* *current*
Safety chain (SPS – Single Point Status)	WTurb.ST.SafeChn.stVal WTurb.ST.SafeChn.q WTurb.ST.SafeChn.q.validity WTurb.ST.SafeChn.t	*current* *current* *current* *current*
Error (SPS – Single Point Status)	WTurb.ST.Error.stVal WTurb.ST.Error.q WTurb.ST.Error.q.validity WTurb.ST.Error.t	*current* *current* *current* *current*
Warning (SPS – Single Point Status)	WTurb.ST.Warn.stVal WTurb.ST.Warn.q WTurb.ST.Warn.q.validity WTurb.ST.Warn.t	*current* *current* *current* *current*
Remote control from info possible (SPS – Single Point Status)	WTurb.ST.RemCtlInf.stVal WTurb.ST.RemCtlInf.q WTurb.ST.RemCtlInf.q.validity WTurb.ST.RemCtlInf.t	*current* *current* *current* *current*
Reset level (ISI – Integer Status)	WTurb.ST.RstLvl.stVal WTurb.ST.RstLvl.q WTurb.ST.RstLvl.q.validity WTurb.ST.RstLvl.t	*current* *current* *current* *current*
Status Code (ISI – Integer Status)	WTurb.ST.StCod.stVal WTurb.ST.StCod.q WTurb.ST.StCod.q.validity WTurb.ST.StCod.t	*current* *current* *current* *current*
Active fault code (ISI – Integer Status)	WTurb.ST.ActvFltCod.stVal WTurb.ST.ActvFltCod.q WTurb.ST.ActvFltCod.q.validity WTurb.ST.ActvFltCod.t	*current* *current* *current* *current*
Active fault code2 (ISI – Integer Status)	WTurb.ST.ActvFltCod2.stVal WTurb.ST.ActvFltCod2.q WTurb.ST.ActvFltCod2.q.validity WTurb.ST.ActvFltCod2.t	*current* *current* *current* *current*
Active fault code3 (ISI – Integer Status)	WTurb.ST.ActvFltCod3.stVal WTurb.ST.ActvFltCod3.q WTurb.ST.ActvFltCod3.q.validity WTurb.ST.ActvFltCod3.t	*current* *current* *current* *current*
Active fault code4 (ISI – Integer Status)	WTurb.ST.ActvFltCod3.stVal WTurb.ST.ActvFltCod3.q WTurb.ST.ActvFltCod3.q.validity	*current* *current* *current*

Description	Data Name, Functional Constraint, and Attribute Name	Value
	WTurb.ST.ActvFltCod3.t	*current*

B.11 Services provided by the WPP information model for the test

The following table lists all service models and services implement in the server implementation.

CO	Control	MS	Monitoring and supervision
D-CR	Data collection and retrieval	PR	Performance and reporting
AE-M	Alarm and event management	CF	Configuration

Service model	Description	Services	CO	MS	D-CR	PR	AE - M	CF
Server	Represents the external visible behaviour of a device. All other ACSI models are part of the server.	ServerDirectory						**x**
Application association	Provision of how two or more devices can be connected. Provides different views to a device: restricted access to the server's information and functions.	Associate Abort Release	**x** **x** **x**					**x** **x** **x**
Logical device	Represents a group of functions; each function is defined as a logical node.	LogicalDeviceDirectory						**x**
Logical node	Represents a specific function of the substation system, e.g., wind turbine.	LogicalNodeDirectory						**x**
Data	Provides a means to specify typed information, e.g., position of a switch with quality information, and timestamp.	GetDataValues SetDataValues GetDataDefinition GetDataDirectory		**x** **x**	**x** **x**	**x** **x**		 **x** **x**
Data set	Allow to group various data together.	GetDataSetValue SetDataSetValue GetDataSetDirectory		**x** **x**	**x** **x**	**x** **x**		 **x** **x**
Reporting and logging	Describes the conditions for generating reports and logs based on parameters set by the client. Reports may be triggered by changes of process data values (e.g.,	Report GetReportControl- AttrValue SetReportControlAttrVa lue GetLogControlValue SetLogControlValue		**x** **x** **x** **x** **x**	**x** **x** **x** **x** **x**	**x** **x** **x** **x** **x**	**x** **x** **x** **x** **x**	 **x** **x**

Service model	Description	Services	CO	MS	D-CR	PR	AE - M	CF
	state change or deadband) or by quality changes. Logs can be queried for later retrieval. Reports may be send immediately or deferred (buffered). Reports provide change-of-state and sequence-of-events information exchange.	QueryLogByTime QueryLogByEntry GetLogStatusValue		**x**	**x**	**x**	**x**	
Control	Describes the services to control, e.g., devices or parameter setting groups.	Operate	**x**					

B.12 Representation of device name plates

General

The device nameplate specified in this report is consistent with IEC 61850-7-3 and -7-4.

NOTE – The implementation used in the project provides the UCA2.0 compliant information which is slightly different.

Application of common data classes

The common data class ISC (Controllable Integer Status) applied to the data class Mode is as depicted in the following example:

Data Name (semantic)	**Attributes**	**Value**
Mode.CO.	ctlVal	*current*
Mode.ST.	stVal	*current*
Mode.ST.	q.validity	*current*
Mode.ST.	t	*current*
Mode.CF.	ctlModel	*current*

The common data class ISI (Integer Status) applied to data classes Behaviour and Health is as depicted in the following example:

Data Name (semantic)	**Attributes**	**Value**
Beh.ST.	stVal	*current*
Beh.ST.	q.validity	*current*
Beh.ST.	t	*current*

Named Variables for Logical node zero (LLN0)

Description	**Data Name, Functional Constraint, and Attribute Name**	**Value**
Mode	Mode Mode.CO Mode.CO.ctlVal Mode.ST Mode.ST.stVal	*current* *current* *current* *current* *current*

Description	**Data Name, Functional Constraint, and Attribute Name**	**Value**
	Mode.ST.q Mode.ST.q.validity Mode.ST.t Mode.CF Mode.CF.ctlModel	*current* *current* *current* *current* *current*
Behaviour	Beh Beh.ST Beh.ST.stVal Beh.ST.q Beh.ST.q.validity Beh.ST.t	*current* *current* *current* *current* *current* *current*
Health	Health Health.ST Health.ST.stVal Health.ST.q Health.ST.q.validity Health.ST.t	*current* *current* *current* *current* *current* *current*
Name plate	Name Name.ST Name.ST.vendor Name.ST.serNum	*current* *current* *current* *current*

Named Variables for Physical device information (LPHD)

Description	**Data Name, Functional Constraint, and Attribute Name**	**Value**
Physical device name plate	PhName PhName.ST PhName.ST.vendor PhName.ST.serNum	*current* *current* *current* *current*
Physical device health	PhHealth PhHealth.ST PhHealth.ST.stVal PhHealth.ST.q PhHealth.ST.q.validity PhHealth.ST.t	*current* *current* *current* *current* *current* *current*

B.13 Mapping of the real process data to logical nodes and data objects

The mapping approach

The implementation of the **information model** and the server that hosts the model is shown in Figure 7. The process data packets (provided as arrays of variables from a Visual basic program) on the right hand side are moved to the server DLL. More details on the use of the DLL approach can be found in [5].

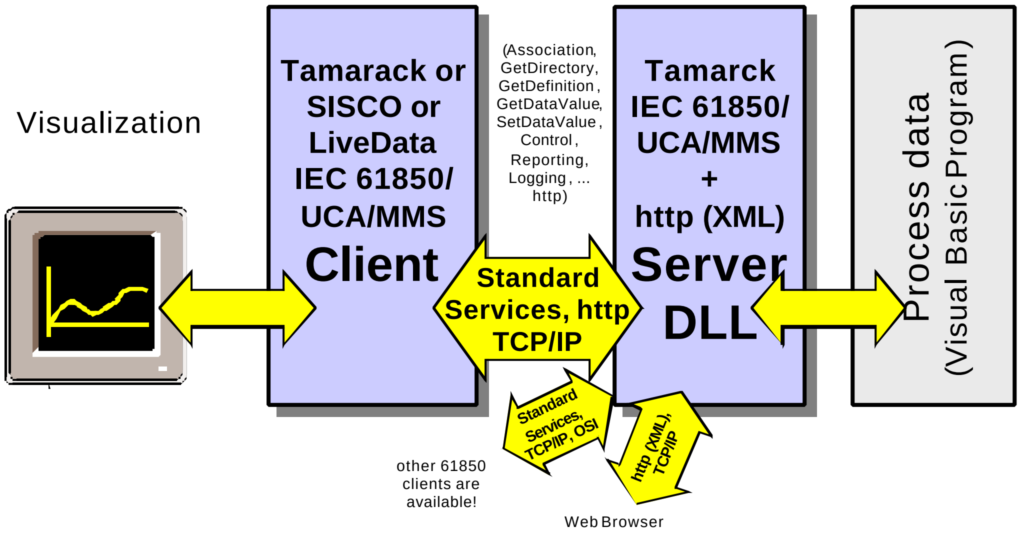

Figure 7 – Mapping approach (overview)

The DLL approach has been chosen to allow an easy integration of the server into the existing software environment. The server "serves" one or more clients for real-time information exchange implemented in the IEC 61850/UCA/MMS server.

The server provides an additional software for the HTTP access (exchanging HTML and XML coded pages) providing values in a non-real-time manner.

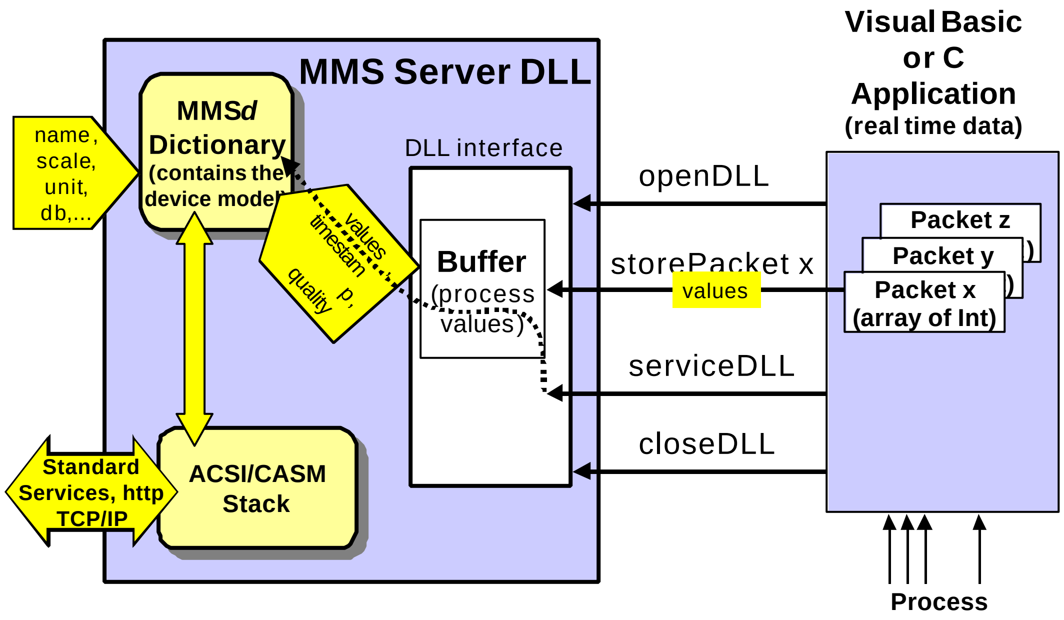

Figure 8 – Server implementation (overview)

Figure 8 depicts the interface between the Visual Basic application and the server DLL. The application controls the DLL. The first action is to open the DLL (openDLL). The process data values are stored in the server applying the "storePacketx" calls. When all data values are store, the application calls the "serviceDLL" call. This call starts the processing of the server DLL.

The DLL runs only when it is called by the application. As a consequence of this DLL-typical behavior, the server DLL processes incoming and outgoing messages as often as the application calls the server DLL.

Specification of the packets

The **real process data** are the data that are offered by the Visual Basic application in a PC in the wind power plant. These data are organised into several packets.

The **real process data** is sent every 100 ms. The Packet type field specifies what data the Data field includes

The following packet types may contain more data than are defined in the WP3000 MANUAL. These additional data have been added to the logical node models on request of Vattenfall during the modeling process (MITA_MMS.xls – 2001-04-18).

The **unit** and **scale** values are used for configuration of the server.

Packet type 01H

Packet type 01H is updated every 100 ms.

Description	Data Class Name	Logical node	Unit	Scale
Generator speed	GenSpeed	WGen	rpm	1
Slip	Slip	WGen	%	0,1
Power	Power	WGrid	kW	1
Current L1	APhsA	WGrid	A	0,1
Acceleration X	AccX	WNace	m/s2	0,01
Acceleration Y	AccY	WNace	m/s2	0,01
Yaw pressure	YawP	WYaw	MPa	0,01

Packet type 02H

Packet type 02H is updated when entries changes or every 1 s.

Description	Data Class Name	Logical node
Digital signals word 0:		
Error	Error	WTurb
Warning	Warn	WTurb
Free to yaw	FreeToYaw	WTurb
Free to operate	FreeToOp	WTurb
Free run	FreeRun	WTurb
Thyristor opening	Tyropen	WGen
Generator connected	GenCon	WGen
Brake proc. 50	Brake50	WBrake
Brake proc. 75	Brake75	WBrake
Brake proc. 199	Brake199	WBrake
Brake proc. 200	Brake200	WBrake
Safety chain	SafeChn	WTurb
Yaw CCW	YawCCW	WYaw
Yaw CW	YawCW	WYaw
Auto rewinding cables	AuRewCab	WYaw
Remote control from info possible	RemCtlInf	WTurb
Digital signals word 1:		
Phase compensation (order)	PhCom	WGrid
Disk brake 1 activated (order)	DiskBrk1	WBrake
Disk brake 2 activated (order)	DiskBrk2	WBrake

Description	Data Class Name	Logical node
Soft brake (order)	SoftBrk	WBrake
Hydraulic pump brake (order)	HydPmpBrk	WBrake
Hydraulic pump yaw (order)	HydPmpYaw	WYaw
Hydraulic pump hub (order)	HydPmpHub	WRotor
Water pump (order)	WtrPump	WNace
Hub hydraulic, pump (current feedback)	HubHydrPump	WRotor
Hub hydraulic, solenoid (current feedback)	HubHydrSole	WRotor
Ventilator nacelle (order)	VentNac	WNace
Oil pump (order)	OilPump	WGear
-	-	
-	-	
Heat generator (order)	HeatGen	WGen
Heat wind gauges (order)	HeatWndGau	WEnv
Digital signals word 2:		
Free	DSW2	Not used
Status word from Weier:		
Status word from Weier	SWW	WGen
Reset level:		
Reset level	RstLvl	WTurb
Status code:		
Status Code	StCod	WTurb
Active fault code:		
Active fault code	ActvFltCod	WTurb

Packet type 03H

Packet type 03H is updated every 1 s.

Description	Data Class Name	Logical node	Unit	Scale
Cosphi	CosPhi	WGrid	-	0,01
Voltage L1	VPhsA	WGrid	V	1
Voltage L2	VPhsB	WGrid	V	1
Voltage L3	VPhsC	WGrid	V	1
Current L1	APhsA	WGrid	A	1
Current L2	APhsB	WGrid	A	1
Current L3	APhsC	WGrid	A	1

Description	**Data Class Name**	**Logical node**	**Unit**	**Scale**
Reactive power	VAr	WGrid	kVAr	1
Frequency	Hz	WGrid	Hz	0,01
Gen current (Weier)	GenA	WGen	A	0,1
Wind speed	WindSpd	WEnv	m/s	0,1
Wind speed 2	WindSpd2	WEnv	m/s	0,1
Yaw missalignment	YawMalgmt	WYaw	°	0,1
Yaw missalignment 2	YawMalgmt2	WYaw	°	0,1
Cable twist	CablTwst	WYaw	°	0,1
Yaw speed	Yawspd	WYaw	°/s	0,1
Rotor speed	RotSpd	WRotor	rpm	0,1
Vibration X max	VibXMax	WNace	mm/s	1
Vibration Y max	VibYMax	WNace	mm/s	1
Vibration X RMS	VibXRMS	WNace	mm/s	1
Vibration Y RMS	VibYRMS	WNace	mm/s	1
Caliper 1	Calip1	WBrake	mm	0,1
Caliper 2	Calip2	WBrake	mm	0,1
Gen bearing temp	GenBeTemp	WGen	°C	1
Yaw oil temp	YawOilTemp	WYaw	°C	1
Generator temp	GenTemp	WGen	°C	1
Generator 2 temp	Gen2Temp	WGen	°C	1
Gear oil temp	GeaOilTemp	WGear	°C	1
Gear oil 2 temp	GeaOil2Temp	WGear	°C	1
Outdoor temp	OutdrTemp	WEnv	°C	1
Nacelle temp	NaclTemp	WNace	°C	1
Power panel temp	PwrPnlTemp	WNace	°C	1
Water to cooler temp	WtrToClrTemp	WNace	°C	1
Water from cooler temp	WtrFrmClrTemp	WNace	°C	1
Air pressure	AirPres	WEnv	hPa	0,1

Packet type 04H

Packet type 04H is updated every 10 s.

Description	**Data Class Name**	**Logical node**	**Unit**	**Scale**
Energy G1	WhG1	WTurb	kWh	1
Energy G2	WhG2	WTurb	kWh	1
Energy consumption	WhConspt	WTurb	kWh	1
Time G1	TimeG1	WTurb	h	0,1
Time G2	TimeG2	WTurb	h	0,1
Time with fault status	TimeFltSt	WTurb	h	0,1
Time with grid ok	TimeGridOk	WTurb	h	0,1
Time with wind for prod	TimeWndProd	WTurb	h	0,1
Total time	TimeTotal	WTurb	h	0,1

B.14 Introduction to the reporting and logging model

General

The service models and services are listed in clause B.11. Most of these services are trivial. The reporting and logging is explained in more details in this clause to give a brief overview and introduction.

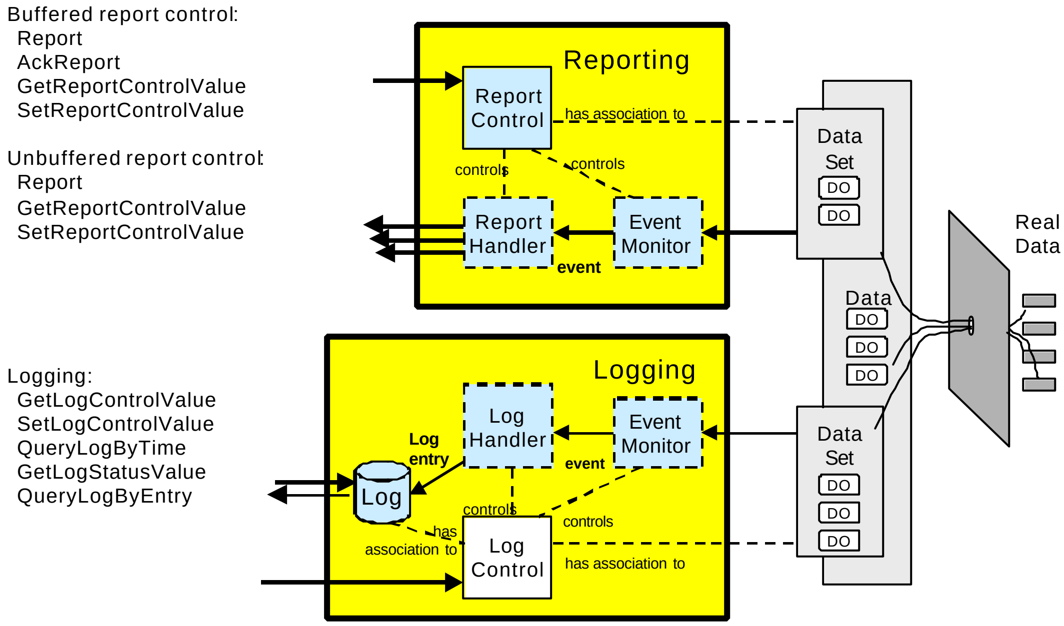

Figure 9 – Basic building blocks for reporting and logging

Overview

An overview of the reporting and logging model of IEC 61850 is shown in Figure 9.

The data to be reported are represented by a **data set** (a data set references data objects). The values of a data set can be used for a report to clients or to be stored in a log (for later retrieval).

The details of the attributes of the reporting and logging control object can be found in IEC 61850-7-2. In the following explanation just the basic behavior is described and explained.

Each logical node provides one basic reporting control block (brcbMX) for measurands (MX) and one (brcbST) for status information (ST). Additionally each logical node provides one log control block (lcbMX) for measurands and one for status information (lcbST).

An example of the control attribute values for these control blocks can be found in the variable table in B.15:

- Log control: Lines 11 to 23 for WBrake

- Report control: Lines 33 to 55 for WBrake

Figure 10 lists the basic features of the reporting and logging model.

- timely reports serve as an early indication to clients,
- logging of events for later retrieval (sequence-of-event),
- the impact on network bandwidth is minimised,
- sending reports only when required (controlled by several attributes),
- low frequency integrity scan and client initiated general interrogation.
- *polling data values at any time (Get)*

Trigger options:

data-change
filtered-data-change
quality-change
cyclic-integrity
general-interrogation
data-set-directory

Figure 10 – Reporting and logging features

Figure 11 shows characteristics of the two possible report methods. The software in the project applies the buffered report control.

Buffered report control

Events (trigger options data-change and/or quality-change) are buffered (to some practical limit) for transmission, such that

data is not lost

due to transport flow control constraints or loss of connection.

Unbuffered report control

Events are transmitted on a

'best efforts' basis.

If no association exists, or if the transport data flow is not fast enough to support it, events may be lost.

Figure 11 – Buffered and unbuffered reporting

Figure 12 depicts the behavior of deadbanding. The deadband of a measurement value constrains the number of reports sent due to changes of the measured value. The application of a deadband allows that the value of the data object will be reported only if the value has changed more than the deadband value since the last report.

A small value causes more reports than a bigger value.

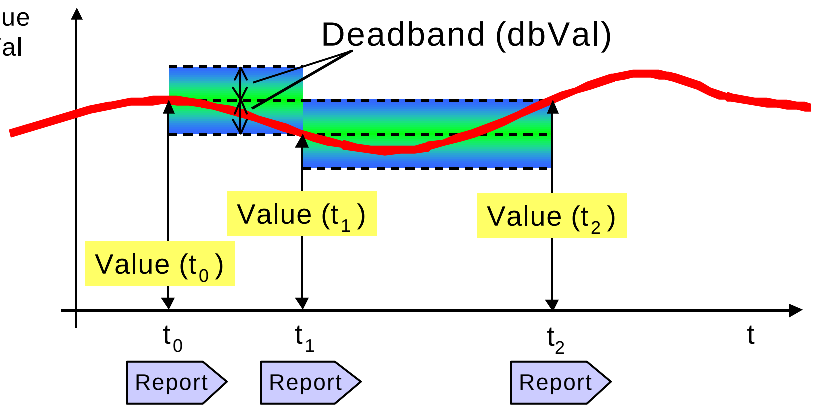

Figure 12 – Deadbanding in the report control model

Figure 13 shows a measured value that is defined in a context of additional information and services. First of all, the measured value may be accessed by a Set and a Get service. The Get service may be used at any time to retrieve the current value of the data object.

The measured value provides several attributes like Unit, Scale, etc. that describe details of the value required to interpret the value when exchanged and received by a client. This metadata can be accessed by appropriate services.

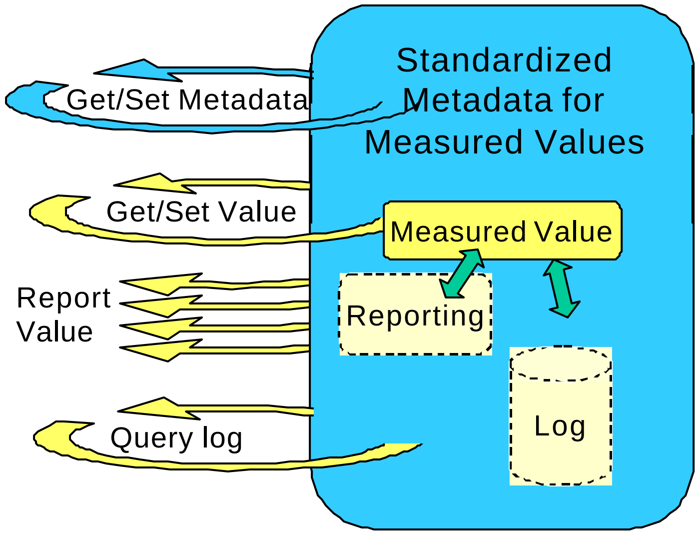

Figure 13 – Metadata and reporting

The value may also be logged for later retrieval.

The basics of the logging model

The Journal (Logs in 61850) allows to read (query in 61850) any set of sequentially stored values of named MMS variables (Data in 61850). Figure 14 depicts three queries.

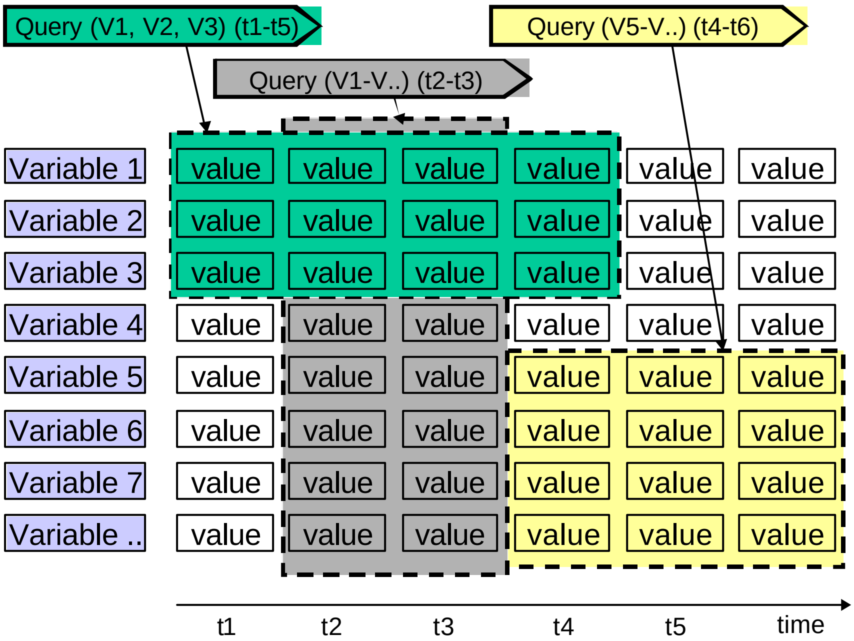

Figure 14 – Logging example

The main objective of the log model is to store and query a sequence of (event) data (**SOE – Sequence of Events**).

The Journal entry (values of all variables at a specific time) may also be written from another device using the service "Write Journal".

All Journal Entries can additionally be sent as immediate Reports, too.

The main characteristics of reporting and logging are:

- timely reports serve as an early indication to clients,
- logging of events for later retrieval (sequence-of-event),
- the impact on network bandwidth is minimized,
- sending reports only when required (controlled by several attributes),
- low frequency integrity scan and client initiated general interrogation.

Reporting provides mechanisms to report packed data instance values immediately or after some buffer time. The logging model provides mechanisms to store events in the log in sequence. A client may query a range of log entries at any time.

Reporting and logging as well as the basic services of the data model provide flexible data retrieval schemas, e.g.:

- **change-of-state notification** of clients: immediate reports,
- **sequence-of-events**: storing and querying log entries,

— **polling data at any time**: GetDataValues and GetDataSetValues

Logging of Wind Power Plant data

The following list data is used in this document to demonstrate the logging functionality of IEC 61850-7-2. The log model of IEC 61850-7-2 is mapped to the MMS Journal Model defines to 61850-8-1.

Data	**Data name**	**MMS Named Variable**
Energy G1	WhG1	WTurbMXWhG1
Energy G2	WhG2	WTurbMXWhG2
Energy consumption	WhConspt	WTurbMXWhConspt
Total time	TimeTotal	WTurbMXTimeTotal
Time G1	TimeG1	WTurbMXTimeG1
Time G2	TimeG2	WTurbMXTimeG2
Time with fault status	TimeFltSt	WTurbMXTimeFltSt
Time with grid ok	TimeGridOk	WTurbMXTimeGridOk
Time with wind for prod	TimeWndProd	WTurbMXTimeWndProd

The logging building blocks are depicted in Figure 15.

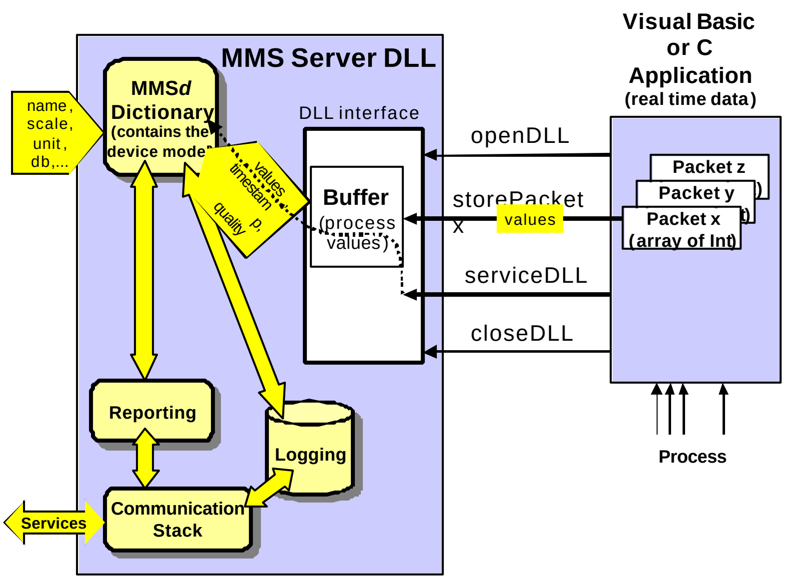

Figure 15 – The Logging blocks

These Data are referenced by the MMS Named Variable List (Data Set in 61850) "**WTurb$MX**". The MMS Journal Name (=Log Name in 61850) is "**WPP$Journal**".

The values of the Data Set "**WTurb$MX"** a stored in the log as Journal entries (log entries in 61850). These log entries can be queried by a client later on.

B.15 List of MMS object names of NamedVariables of the server

The following table lists all **984** NamedVariables of the server visible to the network.

How to read the tables:

- The scope is domain-specific (Domain Name is “Sigvards”).
- Between the “/” and the first “$” the logical node is designated (e.g., WBrake).
- After the first “$” the Functional Constraint (FC) is included – coulered (e.g., CF for configuration, LG for logging, MX for Measurands, RP for reporting, and ST for Status.
- After the second “$” the Data Object is designated (e.g., Calip1)
- After the last “$” the Data Attribute is designated (e.g., db for deadband)

The names are hierarchical. The variable ‘Sigvards/WBrake$CF“ contains all variables one level below and so on. Reading “Sigvards/WBrake$CF“ retrieves all subordinate variable of this „tree“ starting with “Sigvards/WBrake$CF“.

The list has been copied from the original C-code produced bei the MMS*d* PREP. These variables are part of the DLL <<wpp_srv.dll>>. The very same list could be produced by the service *LogicalNodeDirectory* and *GetDataDirectory*. Then the server returns all these names.

1	Sigvards/WBrake$CF
2	Sigvards/WBrakeCFCalip1
3	Sigvards/WBrakeCFCalip1$db
4	Sigvards/WBrakeCFCalip1$s
5	Sigvards/WBrakeCFCalip1$u
6	Sigvards/WBrakeCFCalip2
7	Sigvards/WBrakeCFCalip2$db
8	Sigvards/WBrakeCFCalip2$s
9	Sigvards/WBrakeCFCalip2$u
10	Sigvards/WBrakeCFClockTOD
11	Sigvards/WBrake$LG
12	Sigvards/WBrakeLGlcbMX
13	Sigvards/WBrakeLGlcbMX$DatSetRef
14	Sigvards/WBrakeLGlcbMX$IntgPd
15	Sigvards/WBrakeLGlcbMX$LogEna
16	Sigvards/WBrakeLGlcbMX$LogRef
17	Sigvards/WBrakeLGlcbMX$TrgOps
18	Sigvards/WBrakeLGlcbST
19	Sigvards/WBrakeLGlcbST$DatSetRef
20	Sigvards/WBrakeLGlcbST$IntgPd
21	Sigvards/WBrakeLGlcbST$LogEna
22	Sigvards/WBrakeLGlcbST$LogRef
23	Sigvards/WBrakeLGlcbST$TrgOps
24	Sigvards/WBrake$MX
25	Sigvards/WBrakeMXCalip1
26	Sigvards/WBrakeMXCalip1$mVali
27	Sigvards/WBrakeMXCalip1$q
28	Sigvards/WBrakeMXCalip1$t
29	Sigvards/WBrakeMXCalip2
30	Sigvards/WBrakeMXCalip2$mVali
31	Sigvards/WBrakeMXCalip2$q
32	Sigvards/WBrakeMXCalip2$t
33	Sigvards/WBrake$RP
34	Sigvards/WBrakeRPbrcbMX
35	Sigvards/WBrakeRPbrcbMX$BufTim
36	Sigvards/WBrakeRPbrcbMX$DatSet
37	Sigvards/WBrakeRPbrcbMX$IntgPd
38	Sigvards/WBrakeRPbrcbMX$OptFlds
39	Sigvards/WBrakeRPbrcbMX$RBEPd
40	Sigvards/WBrakeRPbrcbMX$RptEna
41	Sigvards/WBrakeRPbrcbMX$RptID
42	Sigvards/WBrakeRPbrcbMX$SeqNum
43	Sigvards/WBrakeRPbrcbMX$TrgOps
44	Sigvards/WBrakeRPbrcbMX$Trgs
45	Sigvards/WBrakeRPbrcbST
46	Sigvards/WBrakeRPbrcbST$BufTim
47	Sigvards/WBrakeRPbrcbST$DatSet
48	Sigvards/WBrakeRPbrcbST$IntgPd
49	Sigvards/WBrakeRPbrcbST$OptFlds
50	Sigvards/WBrakeRPbrcbST$RBEPd
51	Sigvards/WBrakeRPbrcbST$RptEna
52	Sigvards/WBrakeRPbrcbST$RptID

53	Sigvards/WBrakeRPbrcbST$SeqNum
54	Sigvards/WBrakeRPbrcbST$TrgOps
55	Sigvards/WBrakeRPbrcbST$Trgs
56	Sigvards/WBrake$ST
57	Sigvards/WBrakeSTBrake199
58	Sigvards/WBrakeSTBrake199$q
59	Sigvards/WBrakeSTBrake199$stVal
60	Sigvards/WBrakeSTBrake199$t
61	Sigvards/WBrakeSTBrake200
62	Sigvards/WBrakeSTBrake200$q
63	Sigvards/WBrakeSTBrake200$stVal
64	Sigvards/WBrakeSTBrake200$t
65	Sigvards/WBrakeSTBrake50
66	Sigvards/WBrakeSTBrake50$q
67	Sigvards/WBrakeSTBrake50$stVal
68	Sigvards/WBrakeSTBrake50$t
69	Sigvards/WBrakeSTBrake75
70	Sigvards/WBrakeSTBrake75$q
71	Sigvards/WBrakeSTBrake75$stVal
72	Sigvards/WBrakeSTBrake75$t
73	Sigvards/WBrakeSTDiskBrk1
74	Sigvards/WBrakeSTDiskBrk1$q
75	Sigvards/WBrakeSTDiskBrk1$stVal
76	Sigvards/WBrakeSTDiskBrk1$t
77	Sigvards/WBrakeSTDiskBrk2
78	Sigvards/WBrakeSTDiskBrk2$q
79	Sigvards/WBrakeSTDiskBrk2$stVal
80	Sigvards/WBrakeSTDiskBrk2$t
81	Sigvards/WBrakeSTHydPmpBrk
82	Sigvards/WBrakeSTHydPmpBrk$q
83	Sigvards/WBrakeSTHydPmpBrk$stVal
84	Sigvards/WBrakeSTHydPmpBrk$t
85	Sigvards/WBrakeSTSoftBrk
86	Sigvards/WBrakeSTSoftBrk$q
87	Sigvards/WBrakeSTSoftBrk$stVal
88	Sigvards/WBrakeSTSoftBrk$t
89	Sigvards/WEnv$CF
90	Sigvards/WEnvCFAirPres
91	Sigvards/WEnvCFAirPres$db
92	Sigvards/WEnvCFAirPres$s
93	Sigvards/WEnvCFAirPres$u
94	Sigvards/WEnvCFClockTOD
95	Sigvards/WEnvCFOutdrTemp
96	Sigvards/WEnvCFOutdrTemp$db
97	Sigvards/WEnvCFOutdrTemp$s
98	Sigvards/WEnvCFOutdrTemp$u
99	Sigvards/WEnvCFWindSpd
100	Sigvards/WEnvCFWindSpd$db
101	Sigvards/WEnvCFWindSpd$s
102	Sigvards/WEnvCFWindSpd$u
103	Sigvards/WEnvCFWindSpd2
104	Sigvards/WEnvCFWindSpd2$db

105	Sigvards/WEnvCFWindSpd2$s
106	Sigvards/WEnvCFWindSpd2$u
107	Sigvards/WEnv$LG
108	Sigvards/WEnvLGlcbMX
109	Sigvards/WEnvLGlcbMX$DatSetRef
110	Sigvards/WEnvLGlcbMX$IntgPd
111	Sigvards/WEnvLGlcbMX$LogEna
112	Sigvards/WEnvLGlcbMX$LogRef
113	Sigvards/WEnvLGlcbMX$TrgOps
114	Sigvards/WEnvLGlcbST
115	Sigvards/WEnvLGlcbST$DatSetRef
116	Sigvards/WEnvLGlcbST$IntgPd
117	Sigvards/WEnvLGlcbST$LogEna
118	Sigvards/WEnvLGlcbST$LogRef
119	Sigvards/WEnvLGlcbST$TrgOps
120	Sigvards/WEnv$MX
121	Sigvards/WEnvMXAirPres
122	Sigvards/WEnvMXAirPres$mVali
123	Sigvards/WEnvMXAirPres$q
124	Sigvards/WEnvMXAirPres$t
125	Sigvards/WEnvMXOutdrTemp
126	Sigvards/WEnvMXOutdrTemp$mVali
127	Sigvards/WEnvMXOutdrTemp$q
128	Sigvards/WEnvMXOutdrTemp$t
129	Sigvards/WEnvMXWindSpd
130	Sigvards/WEnvMXWindSpd$mVali
131	Sigvards/WEnvMXWindSpd$q
132	Sigvards/WEnvMXWindSpd$t
133	Sigvards/WEnvMXWindSpd2
134	Sigvards/WEnvMXWindSpd2$mVali
135	Sigvards/WEnvMXWindSpd2$q
136	Sigvards/WEnvMXWindSpd2$t
137	Sigvards/WEnv$RP
138	Sigvards/WEnvRPbrcbMX
139	Sigvards/WEnvRPbrcbMX$BufTim
140	Sigvards/WEnvRPbrcbMX$DatSet
141	Sigvards/WEnvRPbrcbMX$IntgPd
142	Sigvards/WEnvRPbrcbMX$OptFlds
143	Sigvards/WEnvRPbrcbMX$RBEPd
144	Sigvards/WEnvRPbrcbMX$RptEna
145	Sigvards/WEnvRPbrcbMX$RptID
146	Sigvards/WEnvRPbrcbMX$SeqNum
147	Sigvards/WEnvRPbrcbMX$TrgOps
148	Sigvards/WEnvRPbrcbMX$Trgs
149	Sigvards/WEnvRPbrcbST
150	Sigvards/WEnvRPbrcbST$BufTim
151	Sigvards/WEnvRPbrcbST$DatSet
152	Sigvards/WEnvRPbrcbST$IntgPd
153	Sigvards/WEnvRPbrcbST$OptFlds
154	Sigvards/WEnvRPbrcbST$RBEPd
155	Sigvards/WEnvRPbrcbST$RptEna
156	Sigvards/WEnvRPbrcbST$RptID

157	Sigvards/WEnvRPbrcbST$SeqNum
158	Sigvards/WEnvRPbrcbST$TrgOps
159	Sigvards/WEnvRPbrcbST$Trgs
160	Sigvards/WEnv$ST
161	Sigvards/WEnvSTHeatWndGau
162	Sigvards/WEnvSTHeatWndGau$q
163	Sigvards/WEnvSTHeatWndGau$stVal
164	Sigvards/WEnvSTHeatWndGau$t
165	Sigvards/WGear$CF
166	Sigvards/WGearCFClockTOD
167	Sigvards/WGearCFGeaOil2Temp
168	Sigvards/WGearCFGeaOil2Temp$db
169	Sigvards/WGearCFGeaOil2Temp$s
170	Sigvards/WGearCFGeaOil2Temp$u
171	Sigvards/WGearCFGeaOilTemp
172	Sigvards/WGearCFGeaOilTemp$db
173	Sigvards/WGearCFGeaOilTemp$s
174	Sigvards/WGearCFGeaOilTemp$u
175	Sigvards/WGear$LG
176	Sigvards/WGearLGlcbMX
177	Sigvards/WGearLGlcbMX$DatSetRef
178	Sigvards/WGearLGlcbMX$IntgPd
179	Sigvards/WGearLGlcbMX$LogEna
180	Sigvards/WGearLGlcbMX$LogRef
181	Sigvards/WGearLGlcbMX$TrgOps
182	Sigvards/WGearLGlcbST
183	Sigvards/WGearLGlcbST$DatSetRef
184	Sigvards/WGearLGlcbST$IntgPd
185	Sigvards/WGearLGlcbST$LogEna
186	Sigvards/WGearLGlcbST$LogRef
187	Sigvards/WGearLGlcbST$TrgOps
188	Sigvards/WGear$MX
189	Sigvards/WGearMXGeaOil2Temp
190	Sigvards/WGearMXGeaOil2Temp$mVali
191	Sigvards/WGearMXGeaOil2Temp$q
192	Sigvards/WGearMXGeaOil2Temp$t
193	Sigvards/WGearMXGeaOilTemp
194	Sigvards/WGearMXGeaOilTemp$mVali
195	Sigvards/WGearMXGeaOilTemp$q
196	Sigvards/WGearMXGeaOilTemp$t
197	Sigvards/WGear$RP
198	Sigvards/WGearRPbrcbMX
199	Sigvards/WGearRPbrcbMX$BufTim
200	Sigvards/WGearRPbrcbMX$DatSet
201	Sigvards/WGearRPbrcbMX$IntgPd
202	Sigvards/WGearRPbrcbMX$OptFlds
203	Sigvards/WGearRPbrcbMX$RBEPd
204	Sigvards/WGearRPbrcbMX$RptEna
205	Sigvards/WGearRPbrcbMX$RptID
206	Sigvards/WGearRPbrcbMX$SeqNum
207	Sigvards/WGearRPbrcbMX$TrgOps
208	Sigvards/WGearRPbrcbMX$Trgs

209	Sigvards/WGearRPbrcbST
210	Sigvards/WGearRPbrcbST$BufTim
211	Sigvards/WGearRPbrcbST$DatSet
212	Sigvards/WGearRPbrcbST$IntgPd
213	Sigvards/WGearRPbrcbST$OptFlds
214	Sigvards/WGearRPbrcbST$RBEPd
215	Sigvards/WGearRPbrcbST$RptEna
216	Sigvards/WGearRPbrcbST$RptID
217	Sigvards/WGearRPbrcbST$SeqNum
218	Sigvards/WGearRPbrcbST$TrgOps
219	Sigvards/WGearRPbrcbST$Trgs
220	Sigvards/WGear$ST
221	Sigvards/WGearSTOilPump
222	Sigvards/WGearSTOilPump$q
223	Sigvards/WGearSTOilPump$stVal
224	Sigvards/WGearSTOilPump$t
225	Sigvards/WGen$CF
226	Sigvards/WGenCFClockTOD
227	Sigvards/WGenCFDFacSToGen
228	Sigvards/WGenCFDFacSToGen$db
229	Sigvards/WGenCFDFacSToGen$s
230	Sigvards/WGenCFDFacSToGen$u
231	Sigvards/WGenCFGen2Temp
232	Sigvards/WGenCFGen2Temp$db
233	Sigvards/WGenCFGen2Temp$s
234	Sigvards/WGenCFGen2Temp$u
235	Sigvards/WGenCFGenA
236	Sigvards/WGenCFGenA$db
237	Sigvards/WGenCFGenA$s
238	Sigvards/WGenCFGenA$u
239	Sigvards/WGenCFGenBeTemp
240	Sigvards/WGenCFGenBeTemp$db
241	Sigvards/WGenCFGenBeTemp$s
242	Sigvards/WGenCFGenBeTemp$u
243	Sigvards/WGenCFGenSpeed
244	Sigvards/WGenCFGenSpeed$db
245	Sigvards/WGenCFGenSpeed$s
246	Sigvards/WGenCFGenSpeed$u
247	Sigvards/WGenCFGenTemp
248	Sigvards/WGenCFGenTemp$db
249	Sigvards/WGenCFGenTemp$s
250	Sigvards/WGenCFGenTemp$u
251	Sigvards/WGenCFMode
252	Sigvards/WGenCFMode$ctlModel
253	Sigvards/WGenCFSlip
254	Sigvards/WGenCFSlip$db
255	Sigvards/WGenCFSlip$s
256	Sigvards/WGenCFSlip$u
257	Sigvards/WGenCFSWW
258	Sigvards/WGenCFSWW$db
259	Sigvards/WGen$CO
260	Sigvards/WGenCOMode

261	Sigvards/WGenCOMode$ctlVal
262	Sigvards/WGen$LG
263	Sigvards/WGenLGlcbMX
264	Sigvards/WGenLGlcbMX$DatSetRef
265	Sigvards/WGenLGlcbMX$IntgPd
266	Sigvards/WGenLGlcbMX$LogEna
267	Sigvards/WGenLGlcbMX$LogRef
268	Sigvards/WGenLGlcbMX$TrgOps
269	Sigvards/WGenLGlcbST
270	Sigvards/WGenLGlcbST$DatSetRef
271	Sigvards/WGenLGlcbST$IntgPd
272	Sigvards/WGenLGlcbST$LogEna
273	Sigvards/WGenLGlcbST$LogRef
274	Sigvards/WGenLGlcbST$TrgOps
275	Sigvards/WGen$MX
276	Sigvards/WGenMXDFacSToGen
277	Sigvards/WGenMXDFacSToGen$mVali
278	Sigvards/WGenMXDFacSToGen$q
279	Sigvards/WGenMXDFacSToGen$t
280	Sigvards/WGenMXGen2Temp
281	Sigvards/WGenMXGen2Temp$mVali
282	Sigvards/WGenMXGen2Temp$q
283	Sigvards/WGenMXGen2Temp$t
284	Sigvards/WGenMXGenA
285	Sigvards/WGenMXGenA$mVali
286	Sigvards/WGenMXGenA$q
287	Sigvards/WGenMXGenA$t
288	Sigvards/WGenMXGenBeTemp
289	Sigvards/WGenMXGenBeTemp$mVali
290	Sigvards/WGenMXGenBeTemp$q
291	Sigvards/WGenMXGenBeTemp$t
292	Sigvards/WGenMXGenSpeed
293	Sigvards/WGenMXGenSpeed$mVali
294	Sigvards/WGenMXGenSpeed$q
295	Sigvards/WGenMXGenSpeed$t
296	Sigvards/WGenMXGenTemp
297	Sigvards/WGenMXGenTemp$mVali
298	Sigvards/WGenMXGenTemp$q
299	Sigvards/WGenMXGenTemp$t
300	Sigvards/WGenMXSlip
301	Sigvards/WGenMXSlip$mVali
302	Sigvards/WGenMXSlip$q
303	Sigvards/WGenMXSlip$t
304	Sigvards/WGen$RP
305	Sigvards/WGenRPbrcbMX
306	Sigvards/WGenRPbrcbMX$BufTim
307	Sigvards/WGenRPbrcbMX$DatSet
308	Sigvards/WGenRPbrcbMX$IntgPd
309	Sigvards/WGenRPbrcbMX$OptFlds
310	Sigvards/WGenRPbrcbMX$RBEPd
311	Sigvards/WGenRPbrcbMX$RptEna
312	Sigvards/WGenRPbrcbMX$RptID

313	Sigvards/WGenRPbrcbMX$SeqNum
314	Sigvards/WGenRPbrcbMX$TrgOps
315	Sigvards/WGenRPbrcbMX$Trgs
316	Sigvards/WGenRPbrcbST
317	Sigvards/WGenRPbrcbST$BufTim
318	Sigvards/WGenRPbrcbST$DatSet
319	Sigvards/WGenRPbrcbST$IntgPd
320	Sigvards/WGenRPbrcbST$OptFlds
321	Sigvards/WGenRPbrcbST$RBEPd
322	Sigvards/WGenRPbrcbST$RptEna
323	Sigvards/WGenRPbrcbST$RptID
324	Sigvards/WGenRPbrcbST$SeqNum
325	Sigvards/WGenRPbrcbST$TrgOps
326	Sigvards/WGenRPbrcbST$Trgs
327	Sigvards/WGen$ST
328	Sigvards/WGenSTGenCon
329	Sigvards/WGenSTGenCon$q
330	Sigvards/WGenSTGenCon$stVal
331	Sigvards/WGenSTGenCon$t
332	Sigvards/WGenSTHeatGen
333	Sigvards/WGenSTHeatGen$q
334	Sigvards/WGenSTHeatGen$stVal
335	Sigvards/WGenSTHeatGen$t
336	Sigvards/WGenSTMode
337	Sigvards/WGenSTMode$q
338	Sigvards/WGenSTMode$stVal
339	Sigvards/WGenSTMode$t
340	Sigvards/WGenSTSWW
341	Sigvards/WGenSTSWW$q
342	Sigvards/WGenSTSWW$stVal
343	Sigvards/WGenSTSWW$t
344	Sigvards/WGenSTTyrOpen
345	Sigvards/WGenSTTyrOpen$q
346	Sigvards/WGenSTTyrOpen$stVal
347	Sigvards/WGenSTTyrOpen$t
348	Sigvards/WGrid$CF
349	Sigvards/WGridCFAPhsA
350	Sigvards/WGridCFAPhsA$db
351	Sigvards/WGridCFAPhsA$s
352	Sigvards/WGridCFAPhsA$u
353	Sigvards/WGridCFAPhsA0
354	Sigvards/WGridCFAPhsA0$db
355	Sigvards/WGridCFAPhsA0$s
356	Sigvards/WGridCFAPhsA0$u
357	Sigvards/WGridCFAPhsB
358	Sigvards/WGridCFAPhsB$db
359	Sigvards/WGridCFAPhsB$s
360	Sigvards/WGridCFAPhsB$u
361	Sigvards/WGridCFAPhsC
362	Sigvards/WGridCFAPhsC$db
363	Sigvards/WGridCFAPhsC$s
364	Sigvards/WGridCFAPhsC$u

365	Sigvards/WGridCFClockTOD
366	Sigvards/WGridCFCosPhi
367	Sigvards/WGridCFCosPhi$db
368	Sigvards/WGridCFCosPhi$s
369	Sigvards/WGridCFCosPhi$u
370	Sigvards/WGridCFHz
371	Sigvards/WGridCFHz$db
372	Sigvards/WGridCFHz$s
373	Sigvards/WGridCFHz$u
374	Sigvards/WGridCFPower
375	Sigvards/WGridCFPower$db
376	Sigvards/WGridCFPower$s
377	Sigvards/WGridCFPower$u
378	Sigvards/WGridCFVAr
379	Sigvards/WGridCFVAr$db
380	Sigvards/WGridCFVAr$s
381	Sigvards/WGridCFVAr$u
382	Sigvards/WGridCFVPhsA
383	Sigvards/WGridCFVPhsA$db
384	Sigvards/WGridCFVPhsA$s
385	Sigvards/WGridCFVPhsA$u
386	Sigvards/WGridCFVPhsB
387	Sigvards/WGridCFVPhsB$db
388	Sigvards/WGridCFVPhsB$s
389	Sigvards/WGridCFVPhsB$u
390	Sigvards/WGridCFVPhsC
391	Sigvards/WGridCFVPhsC$db
392	Sigvards/WGridCFVPhsC$s
393	Sigvards/WGridCFVPhsC$u
394	Sigvards/WGrid$LG
395	Sigvards/WGridLGlcbMX
396	Sigvards/WGridLGlcbMX$DatSetRef
397	Sigvards/WGridLGlcbMX$IntgPd
398	Sigvards/WGridLGlcbMX$LogEna
399	Sigvards/WGridLGlcbMX$LogRef
400	Sigvards/WGridLGlcbMX$TrgOps
401	Sigvards/WGridLGlcbST
402	Sigvards/WGridLGlcbST$DatSetRef
403	Sigvards/WGridLGlcbST$IntgPd
404	Sigvards/WGridLGlcbST$LogEna
405	Sigvards/WGridLGlcbST$LogRef
406	Sigvards/WGridLGlcbST$TrgOps
407	Sigvards/WGrid$MX
408	Sigvards/WGridMXAPhsA
409	Sigvards/WGridMXAPhsA$mVali
410	Sigvards/WGridMXAPhsA$q
411	Sigvards/WGridMXAPhsA$t
412	Sigvards/WGridMXAPhsA0
413	Sigvards/WGridMXAPhsA0$mVali
414	Sigvards/WGridMXAPhsA0$q
415	Sigvards/WGridMXAPhsA0$t
416	Sigvards/WGridMXAPhsB

417	Sigvards/WGridMXAPhsB$mVali
418	Sigvards/WGridMXAPhsB$q
419	Sigvards/WGridMXAPhsB$t
420	Sigvards/WGridMXAPhsC
421	Sigvards/WGridMXAPhsC$mVali
422	Sigvards/WGridMXAPhsC$q
423	Sigvards/WGridMXAPhsC$t
424	Sigvards/WGridMXCosPhi
425	Sigvards/WGridMXCosPhi$mVali
426	Sigvards/WGridMXCosPhi$q
427	Sigvards/WGridMXCosPhi$t
428	Sigvards/WGridMXHz
429	Sigvards/WGridMXHz$mVali
430	Sigvards/WGridMXHz$q
431	Sigvards/WGridMXHz$t
432	Sigvards/WGridMXPower
433	Sigvards/WGridMXPower$mVali
434	Sigvards/WGridMXPower$q
435	Sigvards/WGridMXPower$t
436	Sigvards/WGridMXVAr
437	Sigvards/WGridMXVAr$mVali
438	Sigvards/WGridMXVAr$q
439	Sigvards/WGridMXVAr$t
440	Sigvards/WGridMXVPhsA
441	Sigvards/WGridMXVPhsA$mVali
442	Sigvards/WGridMXVPhsA$q
443	Sigvards/WGridMXVPhsA$t
444	Sigvards/WGridMXVPhsB
445	Sigvards/WGridMXVPhsB$mVali
446	Sigvards/WGridMXVPhsB$q
447	Sigvards/WGridMXVPhsB$t
448	Sigvards/WGridMXVPhsC
449	Sigvards/WGridMXVPhsC$mVali
450	Sigvards/WGridMXVPhsC$q
451	Sigvards/WGridMXVPhsC$t
452	Sigvards/WGrid$RP
453	Sigvards/WGridRPbrcbMX
454	Sigvards/WGridRPbrcbMX$BufTim
455	Sigvards/WGridRPbrcbMX$DatSet
456	Sigvards/WGridRPbrcbMX$IntgPd
457	Sigvards/WGridRPbrcbMX$OptFlds
458	Sigvards/WGridRPbrcbMX$RBEPd
459	Sigvards/WGridRPbrcbMX$RptEna
460	Sigvards/WGridRPbrcbMX$RptID
461	Sigvards/WGridRPbrcbMX$SeqNum
462	Sigvards/WGridRPbrcbMX$TrgOps
463	Sigvards/WGridRPbrcbMX$Trgs
464	Sigvards/WGridRPbrcbST
465	Sigvards/WGridRPbrcbST$BufTim
466	Sigvards/WGridRPbrcbST$DatSet
467	Sigvards/WGridRPbrcbST$IntgPd
468	Sigvards/WGridRPbrcbST$OptFlds

469	Sigvards/WGridRPbrcbST$RBEPd
470	Sigvards/WGridRPbrcbST$RptEna
471	Sigvards/WGridRPbrcbST$RptID
472	Sigvards/WGridRPbrcbST$SeqNum
473	Sigvards/WGridRPbrcbST$TrgOps
474	Sigvards/WGridRPbrcbST$Trgs
475	Sigvards/WGrid$ST
476	Sigvards/WGridSTPhCom
477	Sigvards/WGridSTPhCom$q
478	Sigvards/WGridSTPhCom$stVal
479	Sigvards/WGridSTPhCom$t
480	Sigvards/WNace$CF
481	Sigvards/WNaceCFAccX
482	Sigvards/WNaceCFAccX$db
483	Sigvards/WNaceCFAccX$s
484	Sigvards/WNaceCFAccX$u
485	Sigvards/WNaceCFAccY
486	Sigvards/WNaceCFAccY$db
487	Sigvards/WNaceCFAccY$s
488	Sigvards/WNaceCFAccY$u
489	Sigvards/WNaceCFClockTOD
490	Sigvards/WNaceCFNaclTemp
491	Sigvards/WNaceCFNaclTemp$db
492	Sigvards/WNaceCFNaclTemp$s
493	Sigvards/WNaceCFNaclTemp$u
494	Sigvards/WNaceCFPwrPnlTemp
495	Sigvards/WNaceCFPwrPnlTemp$db
496	Sigvards/WNaceCFPwrPnlTemp$s
497	Sigvards/WNaceCFPwrPnlTemp$u
498	Sigvards/WNaceCFVibXMax
499	Sigvards/WNaceCFVibXMax$db
500	Sigvards/WNaceCFVibXMax$s
501	Sigvards/WNaceCFVibXMax$u
502	Sigvards/WNaceCFVibXRMS
503	Sigvards/WNaceCFVibXRMS$db
504	Sigvards/WNaceCFVibXRMS$s
505	Sigvards/WNaceCFVibXRMS$u
506	Sigvards/WNaceCFVibYMax
507	Sigvards/WNaceCFVibYMax$db
508	Sigvards/WNaceCFVibYMax$s
509	Sigvards/WNaceCFVibYMax$u
510	Sigvards/WNaceCFVibYRMS
511	Sigvards/WNaceCFVibYRMS$db
512	Sigvards/WNaceCFVibYRMS$s
513	Sigvards/WNaceCFVibYRMS$u
514	Sigvards/WNaceCFWtrFrmClrTemp
515	Sigvards/WNaceCFWtrFrmClrTemp$db
516	Sigvards/WNaceCFWtrFrmClrTemp$s
517	Sigvards/WNaceCFWtrFrmClrTemp$u
518	Sigvards/WNaceCFWtrToClrTemp
519	Sigvards/WNaceCFWtrToClrTemp$db
520	Sigvards/WNaceCFWtrToClrTemp$s

521	Sigvards/WNaceCFWtrToClrTemp$u
522	Sigvards/WNace$LG
523	Sigvards/WNaceLGlcbMX
524	Sigvards/WNaceLGlcbMX$DatSetRef
525	Sigvards/WNaceLGlcbMX$IntgPd
526	Sigvards/WNaceLGlcbMX$LogEna
527	Sigvards/WNaceLGlcbMX$LogRef
528	Sigvards/WNaceLGlcbMX$TrgOps
529	Sigvards/WNaceLGlcbST
530	Sigvards/WNaceLGlcbST$DatSetRef
531	Sigvards/WNaceLGlcbST$IntgPd
532	Sigvards/WNaceLGlcbST$LogEna
533	Sigvards/WNaceLGlcbST$LogRef
534	Sigvards/WNaceLGlcbST$TrgOps
535	Sigvards/WNace$MX
536	Sigvards/WNaceMXAccX
537	Sigvards/WNaceMXAccX$mVali
538	Sigvards/WNaceMXAccX$q
539	Sigvards/WNaceMXAccX$t
540	Sigvards/WNaceMXAccY
541	Sigvards/WNaceMXAccY$mVali
542	Sigvards/WNaceMXAccY$q
543	Sigvards/WNaceMXAccY$t
544	Sigvards/WNaceMXNaclTemp
545	Sigvards/WNaceMXNaclTemp$mVali
546	Sigvards/WNaceMXNaclTemp$q
547	Sigvards/WNaceMXNaclTemp$t
548	Sigvards/WNaceMXPwrPnlTemp
549	Sigvards/WNaceMXPwrPnlTemp$mVali
550	Sigvards/WNaceMXPwrPnlTemp$q
551	Sigvards/WNaceMXPwrPnlTemp$t
552	Sigvards/WNaceMXVibXMax
553	Sigvards/WNaceMXVibXMax$mVali
554	Sigvards/WNaceMXVibXMax$q
555	Sigvards/WNaceMXVibXMax$t
556	Sigvards/WNaceMXVibXRMS
557	Sigvards/WNaceMXVibXRMS$mVali
558	Sigvards/WNaceMXVibXRMS$q
559	Sigvards/WNaceMXVibXRMS$t
560	Sigvards/WNaceMXVibYMax
561	Sigvards/WNaceMXVibYMax$mVali
562	Sigvards/WNaceMXVibYMax$q
563	Sigvards/WNaceMXVibYMax$t
564	Sigvards/WNaceMXVibYRMS
565	Sigvards/WNaceMXVibYRMS$mVali
566	Sigvards/WNaceMXVibYRMS$q
567	Sigvards/WNaceMXVibYRMS$t
568	Sigvards/WNaceMXWtrFrmClrTemp
569	Sigvards/WNaceMXWtrFrmClrTemp$mVali
570	Sigvards/WNaceMXWtrFrmClrTemp$q
571	Sigvards/WNaceMXWtrFrmClrTemp$t
572	Sigvards/WNaceMXWtrToClrTemp

573	Sigvards/WNaceMXWtrToClrTemp$mVali
574	Sigvards/WNaceMXWtrToClrTemp$q
575	Sigvards/WNaceMXWtrToClrTemp$t
576	Sigvards/WNace$RP
577	Sigvards/WNaceRPbrcbMX
578	Sigvards/WNaceRPbrcbMX$BufTim
579	Sigvards/WNaceRPbrcbMX$DatSet
580	Sigvards/WNaceRPbrcbMX$IntgPd
581	Sigvards/WNaceRPbrcbMX$OptFlds
582	Sigvards/WNaceRPbrcbMX$RBEPd
583	Sigvards/WNaceRPbrcbMX$RptEna
584	Sigvards/WNaceRPbrcbMX$RptID
585	Sigvards/WNaceRPbrcbMX$SeqNum
586	Sigvards/WNaceRPbrcbMX$TrgOps
587	Sigvards/WNaceRPbrcbMX$Trgs
588	Sigvards/WNaceRPbrcbST
589	Sigvards/WNaceRPbrcbST$BufTim
590	Sigvards/WNaceRPbrcbST$DatSet
591	Sigvards/WNaceRPbrcbST$IntgPd
592	Sigvards/WNaceRPbrcbST$OptFlds
593	Sigvards/WNaceRPbrcbST$RBEPd
594	Sigvards/WNaceRPbrcbST$RptEna
595	Sigvards/WNaceRPbrcbST$RptID
596	Sigvards/WNaceRPbrcbST$SeqNum
597	Sigvards/WNaceRPbrcbST$TrgOps
598	Sigvards/WNaceRPbrcbST$Trgs
599	Sigvards/WNace$ST
600	Sigvards/WNaceSTVentNac
601	Sigvards/WNaceSTVentNac$q
602	Sigvards/WNaceSTVentNac$stVal
603	Sigvards/WNaceSTVentNac$t
604	Sigvards/WNaceSTWtrPump
605	Sigvards/WNaceSTWtrPump$q
606	Sigvards/WNaceSTWtrPump$stVal
607	Sigvards/WNaceSTWtrPump$t
608	Sigvards/WRotor$CF
609	Sigvards/WRotorCFClockTOD
610	Sigvards/WRotorCFRotPos
611	Sigvards/WRotorCFRotPos$db
612	Sigvards/WRotorCFRotPos$s
613	Sigvards/WRotorCFRotPos$u
614	Sigvards/WRotorCFRotSpd
615	Sigvards/WRotorCFRotSpd$db
616	Sigvards/WRotorCFRotSpd$s
617	Sigvards/WRotorCFRotSpd$u
618	Sigvards/WRotor$LG
619	Sigvards/WRotorLGlcbMX
620	Sigvards/WRotorLGlcbMX$DatSetRef
621	Sigvards/WRotorLGlcbMX$IntgPd
622	Sigvards/WRotorLGlcbMX$LogEna
623	Sigvards/WRotorLGlcbMX$LogRef
624	Sigvards/WRotorLGlcbMX$TrgOps

625	Sigvards/WRotorLGlcbST
626	Sigvards/WRotorLGlcbST$DatSetRef
627	Sigvards/WRotorLGlcbST$IntgPd
628	Sigvards/WRotorLGlcbST$LogEna
629	Sigvards/WRotorLGlcbST$LogRef
630	Sigvards/WRotorLGlcbST$TrgOps
631	Sigvards/WRotor$MX
632	Sigvards/WRotorMXRotPos
633	Sigvards/WRotorMXRotPos$mVali
634	Sigvards/WRotorMXRotPos$q
635	Sigvards/WRotorMXRotPos$t
636	Sigvards/WRotorMXRotSpd
637	Sigvards/WRotorMXRotSpd$mVali
638	Sigvards/WRotorMXRotSpd$q
639	Sigvards/WRotorMXRotSpd$t
640	Sigvards/WRotor$RP
641	Sigvards/WRotorRPbrcbMX
642	Sigvards/WRotorRPbrcbMX$BufTim
643	Sigvards/WRotorRPbrcbMX$DatSet
644	Sigvards/WRotorRPbrcbMX$IntgPd
645	Sigvards/WRotorRPbrcbMX$OptFlds
646	Sigvards/WRotorRPbrcbMX$RBEPd
647	Sigvards/WRotorRPbrcbMX$RptEna
648	Sigvards/WRotorRPbrcbMX$RptID
649	Sigvards/WRotorRPbrcbMX$SeqNum
650	Sigvards/WRotorRPbrcbMX$TrgOps
651	Sigvards/WRotorRPbrcbMX$Trgs
652	Sigvards/WRotorRPbrcbST
653	Sigvards/WRotorRPbrcbST$BufTim
654	Sigvards/WRotorRPbrcbST$DatSet
655	Sigvards/WRotorRPbrcbST$IntgPd
656	Sigvards/WRotorRPbrcbST$OptFlds
657	Sigvards/WRotorRPbrcbST$RBEPd
658	Sigvards/WRotorRPbrcbST$RptEna
659	Sigvards/WRotorRPbrcbST$RptID
660	Sigvards/WRotorRPbrcbST$SeqNum
661	Sigvards/WRotorRPbrcbST$TrgOps
662	Sigvards/WRotorRPbrcbST$Trgs
663	Sigvards/WRotor$ST
664	Sigvards/WRotorSTHubHydrPump
665	Sigvards/WRotorSTHubHydrPump$q
666	Sigvards/WRotorSTHubHydrPump$stVal
667	Sigvards/WRotorSTHubHydrPump$t
668	Sigvards/WRotorSTHubHydrSole
669	Sigvards/WRotorSTHubHydrSole$q
670	Sigvards/WRotorSTHubHydrSole$stVal
671	Sigvards/WRotorSTHubHydrSole$t
672	Sigvards/WRotorSTHydPmpHub
673	Sigvards/WRotorSTHydPmpHub$q
674	Sigvards/WRotorSTHydPmpHub$stVal
675	Sigvards/WRotorSTHydPmpHub$t
676	Sigvards/WTurb$CF

677	Sigvards/WTurbCFActvFltCod
678	Sigvards/WTurbCFActvFltCod$db
679	Sigvards/WTurbCFActvFltCod2
680	Sigvards/WTurbCFActvFltCod2$db
681	Sigvards/WTurbCFActvFltCod3
682	Sigvards/WTurbCFActvFltCod3$db
683	Sigvards/WTurbCFActvFltCod4
684	Sigvards/WTurbCFActvFltCod4$db
685	Sigvards/WTurbCFClockTOD
686	Sigvards/WTurbCFRstLvl
687	Sigvards/WTurbCFRstLvl$db
688	Sigvards/WTurbCFStCod
689	Sigvards/WTurbCFStCod$db
690	Sigvards/WTurbCFTimeFltSt
691	Sigvards/WTurbCFTimeFltSt$db
692	Sigvards/WTurbCFTimeFltSt$s
693	Sigvards/WTurbCFTimeFltSt$u
694	Sigvards/WTurbCFTimeG1
695	Sigvards/WTurbCFTimeG1$db
696	Sigvards/WTurbCFTimeG1$s
697	Sigvards/WTurbCFTimeG1$u
698	Sigvards/WTurbCFTimeG2
699	Sigvards/WTurbCFTimeG2$db
700	Sigvards/WTurbCFTimeG2$s
701	Sigvards/WTurbCFTimeG2$u
702	Sigvards/WTurbCFTimeGridOk
703	Sigvards/WTurbCFTimeGridOk$db
704	Sigvards/WTurbCFTimeGridOk$s
705	Sigvards/WTurbCFTimeGridOk$u
706	Sigvards/WTurbCFTimeTotal
707	Sigvards/WTurbCFTimeTotal$db
708	Sigvards/WTurbCFTimeTotal$s
709	Sigvards/WTurbCFTimeTotal$u
710	Sigvards/WTurbCFTimeWndProd
711	Sigvards/WTurbCFTimeWndProd$db
712	Sigvards/WTurbCFTimeWndProd$s
713	Sigvards/WTurbCFTimeWndProd$u
714	Sigvards/WTurbCFWhConspt
715	Sigvards/WTurbCFWhConspt$db
716	Sigvards/WTurbCFWhConspt$s
717	Sigvards/WTurbCFWhConspt$u
718	Sigvards/WTurbCFWhG1
719	Sigvards/WTurbCFWhG1$db
720	Sigvards/WTurbCFWhG1$s
721	Sigvards/WTurbCFWhG1$u
722	Sigvards/WTurbCFWhG2
723	Sigvards/WTurbCFWhG2$db
724	Sigvards/WTurbCFWhG2$s
725	Sigvards/WTurbCFWhG2$u
726	Sigvards/WTurb$LG
727	Sigvards/WTurbLGlcbMX
728	Sigvards/WTurbLGlcbMX$DatSetRef

729	Sigvards/WTurbLGlcbMX$IntgPd
730	Sigvards/WTurbLGlcbMX$LogEna
731	Sigvards/WTurbLGlcbMX$LogRef
732	Sigvards/WTurbLGlcbMX$TrgOps
733	Sigvards/WTurbLGlcbST
734	Sigvards/WTurbLGlcbST$DatSetRef
735	Sigvards/WTurbLGlcbST$IntgPd
736	Sigvards/WTurbLGlcbST$LogEna
737	Sigvards/WTurbLGlcbST$LogRef
738	Sigvards/WTurbLGlcbST$TrgOps
739	Sigvards/WTurb$MX
740	Sigvards/WTurbMXTimeFltSt
741	Sigvards/WTurbMXTimeFltSt$mVali
742	Sigvards/WTurbMXTimeFltSt$q
743	Sigvards/WTurbMXTimeFltSt$t
744	Sigvards/WTurbMXTimeG1
745	Sigvards/WTurbMXTimeG1$mVali
746	Sigvards/WTurbMXTimeG1$q
747	Sigvards/WTurbMXTimeG1$t
748	Sigvards/WTurbMXTimeG2
749	Sigvards/WTurbMXTimeG2$mVali
750	Sigvards/WTurbMXTimeG2$q
751	Sigvards/WTurbMXTimeG2$t
752	Sigvards/WTurbMXTimeGridOk
753	Sigvards/WTurbMXTimeGridOk$mVali
754	Sigvards/WTurbMXTimeGridOk$q
755	Sigvards/WTurbMXTimeGridOk$t
756	Sigvards/WTurbMXTimeTotal
757	Sigvards/WTurbMXTimeTotal$mVali
758	Sigvards/WTurbMXTimeTotal$q
759	Sigvards/WTurbMXTimeTotal$t
760	Sigvards/WTurbMXTimeWndProd
761	Sigvards/WTurbMXTimeWndProd$mVali
762	Sigvards/WTurbMXTimeWndProd$q
763	Sigvards/WTurbMXTimeWndProd$t
764	Sigvards/WTurbMXWhConspt
765	Sigvards/WTurbMXWhConspt$mVali
766	Sigvards/WTurbMXWhConspt$q
767	Sigvards/WTurbMXWhConspt$t
768	Sigvards/WTurbMXWhG1
769	Sigvards/WTurbMXWhG1$mVali
770	Sigvards/WTurbMXWhG1$q
771	Sigvards/WTurbMXWhG1$t
772	Sigvards/WTurbMXWhG2
773	Sigvards/WTurbMXWhG2$mVali
774	Sigvards/WTurbMXWhG2$q
775	Sigvards/WTurbMXWhG2$t
776	Sigvards/WTurb$RP
777	Sigvards/WTurbRPbrcbMX
778	Sigvards/WTurbRPbrcbMX$BufTim
779	Sigvards/WTurbRPbrcbMX$DatSet
780	Sigvards/WTurbRPbrcbMX$IntgPd

781	Sigvards/WTurbRPbrcbMX$OptFlds
782	Sigvards/WTurbRPbrcbMX$RBEPd
783	Sigvards/WTurbRPbrcbMX$RptEna
784	Sigvards/WTurbRPbrcbMX$RptID
785	Sigvards/WTurbRPbrcbMX$SeqNum
786	Sigvards/WTurbRPbrcbMX$TrgOps
787	Sigvards/WTurbRPbrcbMX$Trgs
788	Sigvards/WTurbRPbrcbST
789	Sigvards/WTurbRPbrcbST$BufTim
790	Sigvards/WTurbRPbrcbST$DatSet
791	Sigvards/WTurbRPbrcbST$IntgPd
792	Sigvards/WTurbRPbrcbST$OptFlds
793	Sigvards/WTurbRPbrcbST$RBEPd
794	Sigvards/WTurbRPbrcbST$RptEna
795	Sigvards/WTurbRPbrcbST$RptID
796	Sigvards/WTurbRPbrcbST$SeqNum
797	Sigvards/WTurbRPbrcbST$TrgOps
798	Sigvards/WTurbRPbrcbST$Trgs
799	Sigvards/WTurb$ST
800	Sigvards/WTurbSTActvFltCod
801	Sigvards/WTurbSTActvFltCod$q
802	Sigvards/WTurbSTActvFltCod$stVal
803	Sigvards/WTurbSTActvFltCod$t
804	Sigvards/WTurbSTActvFltCod2
805	Sigvards/WTurbSTActvFltCod2$q
806	Sigvards/WTurbSTActvFltCod2$stVal
807	Sigvards/WTurbSTActvFltCod2$t
808	Sigvards/WTurbSTActvFltCod3
809	Sigvards/WTurbSTActvFltCod3$q
810	Sigvards/WTurbSTActvFltCod3$stVal
811	Sigvards/WTurbSTActvFltCod3$t
812	Sigvards/WTurbSTActvFltCod4
813	Sigvards/WTurbSTActvFltCod4$q
814	Sigvards/WTurbSTActvFltCod4$stVal
815	Sigvards/WTurbSTActvFltCod4$t
816	Sigvards/WTurbSTError
817	Sigvards/WTurbSTError$q
818	Sigvards/WTurbSTError$stVal
819	Sigvards/WTurbSTError$t
820	Sigvards/WTurbSTFreeRun
821	Sigvards/WTurbSTFreeRun$q
822	Sigvards/WTurbSTFreeRun$stVal
823	Sigvards/WTurbSTFreeRun$t
824	Sigvards/WTurbSTFreeToOp
825	Sigvards/WTurbSTFreeToOp$q
826	Sigvards/WTurbSTFreeToOp$stVal
827	Sigvards/WTurbSTFreeToOp$t
828	Sigvards/WTurbSTFreeToYaw
829	Sigvards/WTurbSTFreeToYaw$q
830	Sigvards/WTurbSTFreeToYaw$stVal
831	Sigvards/WTurbSTFreeToYaw$t
832	Sigvards/WTurbSTRemCtlInf
833	Sigvards/WTurbSTRemCtlInf$q
834	Sigvards/WTurbSTRemCtlInf$stVal
835	Sigvards/WTurbSTRemCtlInf$t
836	Sigvards/WTurbSTRstLvl
837	Sigvards/WTurbSTRstLvl$q
838	Sigvards/WTurbSTRstLvl$stVal
839	Sigvards/WTurbSTRstLvl$t
840	Sigvards/WTurbSTSafeChn
841	Sigvards/WTurbSTSafeChn$q
842	Sigvards/WTurbSTSafeChn$stVal
843	Sigvards/WTurbSTSafeChn$t
844	Sigvards/WTurbSTStCod
845	Sigvards/WTurbSTStCod$q
846	Sigvards/WTurbSTStCod$stVal
847	Sigvards/WTurbSTStCod$t
848	Sigvards/WTurbSTWarn
849	Sigvards/WTurbSTWarn$q
850	Sigvards/WTurbSTWarn$stVal
851	Sigvards/WTurbSTWarn$t
852	Sigvards/WYaw$CF
853	Sigvards/WYawCFCablTwst
854	Sigvards/WYawCFCablTwst$db
855	Sigvards/WYawCFCablTwst$s
856	Sigvards/WYawCFCablTwst$u
857	Sigvards/WYawCFClockTOD
858	Sigvards/WYawCFYawMalgmt
859	Sigvards/WYawCFYawMalgmt$db
860	Sigvards/WYawCFYawMalgmt$s
861	Sigvards/WYawCFYawMalgmt$u
862	Sigvards/WYawCFYawMalgmt2
863	Sigvards/WYawCFYawMalgmt2$db
864	Sigvards/WYawCFYawMalgmt2$s
865	Sigvards/WYawCFYawMalgmt2$u
866	Sigvards/WYawCFYawOilTemp
867	Sigvards/WYawCFYawOilTemp$db
868	Sigvards/WYawCFYawOilTemp$s
869	Sigvards/WYawCFYawOilTemp$u
870	Sigvards/WYawCFYawP
871	Sigvards/WYawCFYawP$db
872	Sigvards/WYawCFYawP$s
873	Sigvards/WYawCFYawP$u
874	Sigvards/WYawCFYawSpd
875	Sigvards/WYawCFYawSpd$db
876	Sigvards/WYawCFYawSpd$s
877	Sigvards/WYawCFYawSpd$u
878	Sigvards/WYaw$LG
879	Sigvards/WYawLGlcbMX
880	Sigvards/WYawLGlcbMX$DatSetRef
881	Sigvards/WYawLGlcbMX$IntgPd
882	Sigvards/WYawLGlcbMX$LogEna
883	Sigvards/WYawLGlcbMX$LogRef
884	Sigvards/WYawLGlcbMX$TrgOps

885	Sigvards/WYawLGlcbST
886	Sigvards/WYawLGlcbST$DatSetRef
887	Sigvards/WYawLGlcbST$IntgPd
888	Sigvards/WYawLGlcbST$LogEna
889	Sigvards/WYawLGlcbST$LogRef
890	Sigvards/WYawLGlcbST$TrgOps
891	Sigvards/WYaw$MX
892	Sigvards/WYawMXCablTwst
893	Sigvards/WYawMXCablTwst$mVali
894	Sigvards/WYawMXCablTwst$q
895	Sigvards/WYawMXCablTwst$t
896	Sigvards/WYawMXYawMalgmt
897	Sigvards/WYawMXYawMalgmt$mVali
898	Sigvards/WYawMXYawMalgmt$q
899	Sigvards/WYawMXYawMalgmt$t
900	Sigvards/WYawMXYawMalgmt2
901	Sigvards/WYawMXYawMalgmt2$mVali
902	Sigvards/WYawMXYawMalgmt2$q
903	Sigvards/WYawMXYawMalgmt2$t
904	Sigvards/WYawMXYawOilTemp
905	Sigvards/WYawMXYawOilTemp$mVali
906	Sigvards/WYawMXYawOilTemp$q
907	Sigvards/WYawMXYawOilTemp$t
908	Sigvards/WYawMXYawP
909	Sigvards/WYawMXYawP$mVali
910	Sigvards/WYawMXYawP$q
911	Sigvards/WYawMXYawP$t
912	Sigvards/WYawMXYawSpd
913	Sigvards/WYawMXYawSpd$mVali
914	Sigvards/WYawMXYawSpd$q
915	Sigvards/WYawMXYawSpd$t
916	Sigvards/WYaw$RP
917	Sigvards/WYawRPbrcbMX
918	Sigvards/WYawRPbrcbMX$BufTim
919	Sigvards/WYawRPbrcbMX$DatSet
920	Sigvards/WYawRPbrcbMX$IntgPd
921	Sigvards/WYawRPbrcbMX$OptFlds
922	Sigvards/WYawRPbrcbMX$RBEPd
923	Sigvards/WYawRPbrcbMX$RptEna
924	Sigvards/WYawRPbrcbMX$RptID
925	Sigvards/WYawRPbrcbMX$SeqNum
926	Sigvards/WYawRPbrcbMX$TrgOps
927	Sigvards/WYawRPbrcbMX$Trgs
928	Sigvards/WYawRPbrcbST
929	Sigvards/WYawRPbrcbST$BufTim
930	Sigvards/WYawRPbrcbST$DatSet
931	Sigvards/WYawRPbrcbST$IntgPd
932	Sigvards/WYawRPbrcbST$OptFlds
933	Sigvards/WYawRPbrcbST$RBEPd
934	Sigvards/WYawRPbrcbST$RptEna
935	Sigvards/WYawRPbrcbST$RptID
936	Sigvards/WYawRPbrcbST$SeqNum

937	Sigvards/WYawRPbrcbST$TrgOps
938	Sigvards/WYawRPbrcbST$Trgs
939	Sigvards/WYaw$ST
940	Sigvards/WYawSTAuRewCab
941	Sigvards/WYawSTAuRewCab$q
942	Sigvards/WYawSTAuRewCab$stVal
943	Sigvards/WYawSTAuRewCab$t
944	Sigvards/WYawSTHydPmpYaw
945	Sigvards/WYawSTHydPmpYaw$q
946	Sigvards/WYawSTHydPmpYaw$stVal
947	Sigvards/WYawSTHydPmpYaw$t
948	Sigvards/WYawSTYawCCW
949	Sigvards/WYawSTYawCCW$q
950	Sigvards/WYawSTYawCCW$stVal
951	Sigvards/WYawSTYawCCW$t
952	Sigvards/WYawSTYawCW
953	Sigvards/WYawSTYawCW$q
954	Sigvards/WYawSTYawCW$stVal
955	Sigvards/WYawSTYawCW$t
956	Sigvards/CustRP$CustRCB
957	Sigvards/CustRP$CustRCB$BufTim
958	Sigvards/CustRP$CustRCB$DatSet
959	Sigvards/CustRP$CustRCB$IntgPd
960	Sigvards/CustRP$CustRCB$OptFlds
961	Sigvards/CustRP$CustRCB$RBEPd
962	Sigvards/CustRP$CustRCB$RptEna
963	Sigvards/CustRP$CustRCB$RptID
964	Sigvards/CustRP$CustRCB$SeqNum
965	Sigvards/CustRP$CustRCB$TrgOps
966	Sigvards/CustRP$CustRCB$Trgs
967	DI$Class
968	DI$CommID
969	DI$CommID$CommAdr
970	DI$CommID$CommRev
971	DI$CommID$MAC
972	DI$CommID$Med
973	DI$CommID$Pro
974	DI$d
975	DI$Loc
976	DI$Name
977	DI$Own
978	DI$VndID
979	DI$VndID$DevMdls
980	DI$VndID$HwRev
981	DI$VndID$Mdl
982	DI$VndID$SerNum
983	DI$VndID$SftRev
984	DI$VndID$Vnd

C Tamarack client/HMI

C.1 The Tamarack MMS*d* software architecture

Introduction

The focus of the project is the application and analysis of the **server software**. The server contains the **information models**, the **service procedures**, the **monitoring** of data value changes, **filtering** of data, the **services**, the **communication buffers**, and **communication links** (e.g., TCP/IP). A very important issue is the interface between the application and the communication software.

To give some guidance in the understanding of the standard and the application of the standard, the following discussion of an example is intended to show what the standard covers compared with a real system.

The standard IEC 61850 does not constrain any implementation of the information model, service models, communication stacks, actions, and application program interfaces (APIs).

Note – The example is not representative. Many other possible interfaces on both sides are possible.

One major issue to be decided was:

> **Where shall the information models and the service models be located** (close to the communication software or in the application)?

With the current approach provided by NettedAutomation and Tamarack the **information models** and the **service models** are integrated into the communication software of the server. Two extreme and one mixed possibilities are depicted in **Figure 16**. The architecture shown at the top is applied in the project. The interface between the services (service models) and the information model is **hidden**. The information model is integrated in the provided server software. The interface to the application is realized as a "value interface" implemented as a DLL interface (the DLL provides all software on top of the Winsocket interface of Windows.

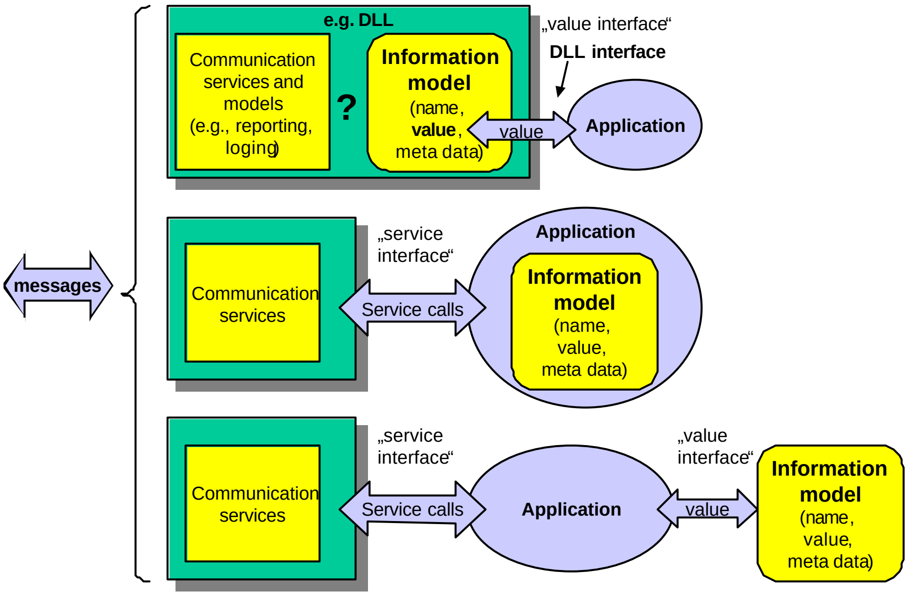

Figure 16 – Interface between communication and application (server)

The inter-process-communication (IPC) between the communication software and the application depends mainly on the requirements and operating system used. There is no way to find “the best approach that fits every size”.

NOTE – The approach how to interface with the communication software is independent of the protocols used to communicate between clients and server.

The approach of Dynamic Link Libraries (DLLs) has been chosen for the project.

See [5] for more details.

The server devices are usually less powerful than computers in control centers. Therefore the software architecture and the software running on servers are crucial with regard to real-time behavior and performance.

The server software architecture

The server covers the communication and the information model according to **Figure 17**.

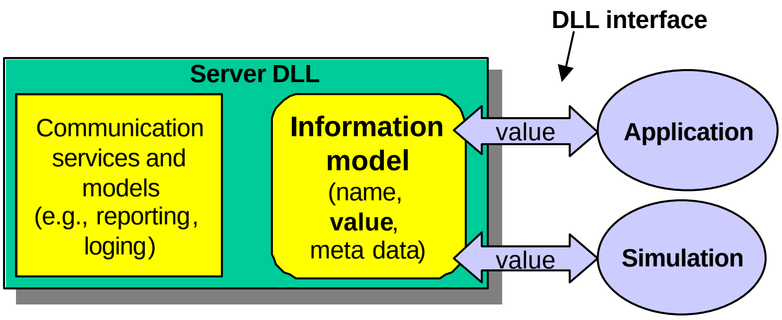

Figure 17 – DLL used as interface

Details on the communication of the Visual Basic application with the server can be found in the report from Anders Andersson

The Visual Basic simulation program <<WppTest.exe>> allows to run the server without connecting to the real application program. This simulation provides the same data values as the real application. The simulation allows to automatically increment the simulated values or to enter values manually (see Figure 18).

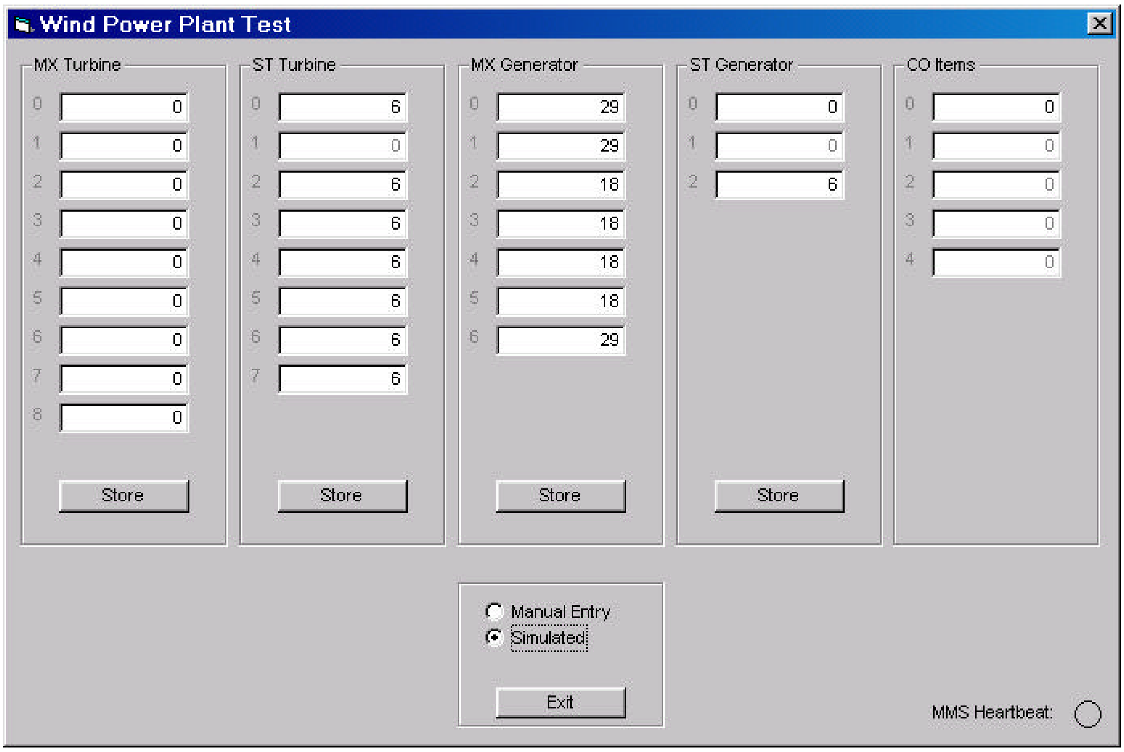

Figure 18 – Simulation window

The server DLL <<wpp_srv.dll>> implements the whole information model as well as the IEC 61850 service models and services. The server DLL communicates via TCP/IP (Winsocket interface in Windows) with the client.

The server DLL provides an additional feature: the implementation of a web server. The web server allows the exchange of all values of the information model in a non-real-time manner. The web server is included in the DLL <<WppTest.exe>>.

The server device provides two servers inside the DLL: The IEC 61850 (MMS) server and the Webserver (see Figure 19).

The webserver supports the “GET” service. The pages requested by a client are“*.html”, “*.xml”, and “*js” files. The HTML and XML files are created immediately after the webserver receives the “GET” request.

In case of “GET URL.html” (“GET URL.xml”) the server provides the data referenced by the URL and returns the values HTML (XML) coded. Examples are explained in the following paragraph.

The Javascript is used for the visualization of the HTML coded responses. The Javascript is retrieved at the beginning (first “GET URL.html”). The files “lib1.js” and “lib2.js” contain the Javascript for the visualization (two files because the webserver can handle maximum file size <10 KB).

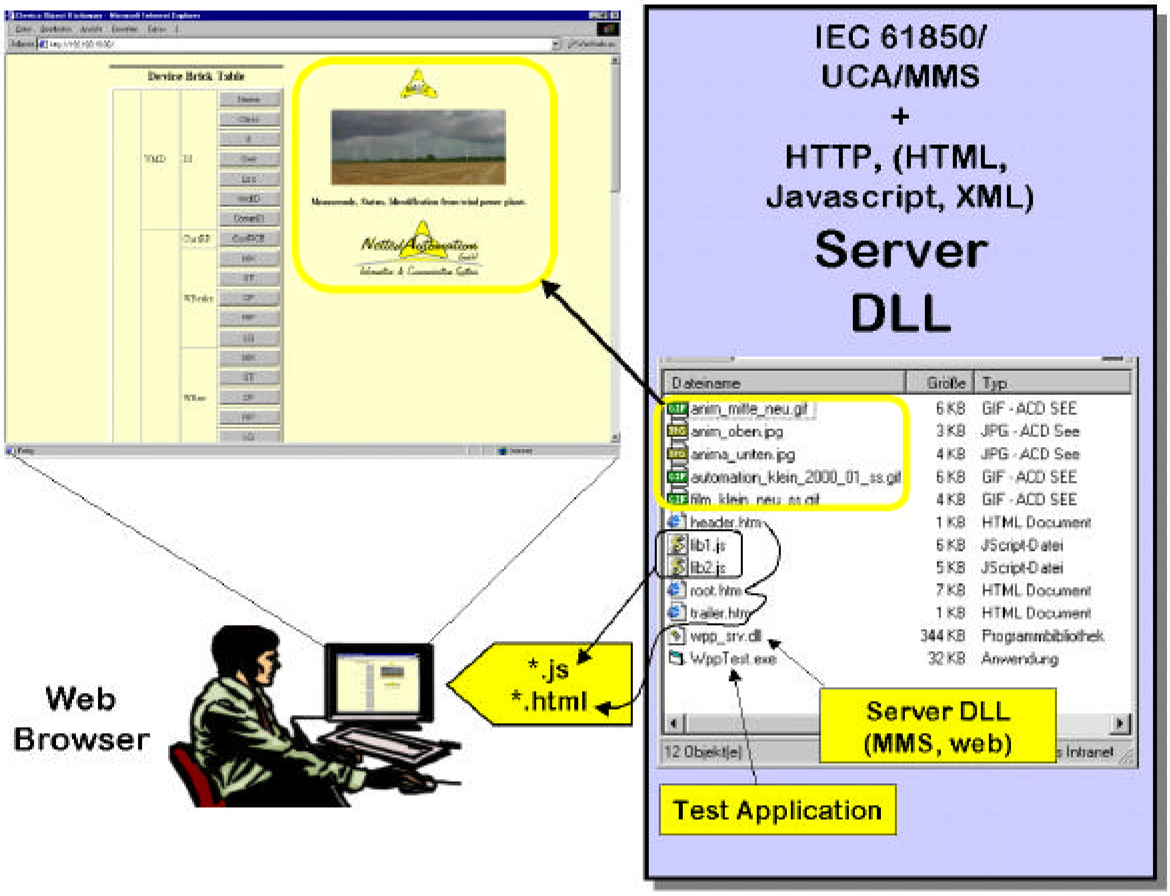

Figure 19 – Topology of server and clients

The “header.htm”, “root.htm”, and “trailer.htm” are needed to assemble the html response files containing the values.

The “wpp_srv.dll” (344 KB) is the complete server DLL containing the 61850 (MMS) server, information model (including the self-description), information exchange methods (e.g., reporting, logging, ...), and the webserver.

The “WppTest.exe” is the test application producing data and receiving control commands.

The web server also allows to retrieve all data values coded as in XML format. The XML code is transmitted from the server to the client on request y the client (HTTP GET).

Details about the application of HTTP, HTML and XML can be found in [6].

The whole server <<DLL Wpp_srv.dll>> (including the information model, service models, services, mapping to Winsocket, interfacing with the real application, HTTP web server, HTML and XML support).

The code size of the Wpp_srv.dll is 344 KB

The client software

The clients used to demonstrate the capabilities of the server are:

- Tamarack's general test client
- Tamarack's logging test client
- SISCO's MMS Object Explorer
- LifeCycle's MMS client with the access from Windows based applications (e.g., Excel, Word) via DDE channels.

Steps to create a server DLL based on a model description

The principle process of creating a server with the MMSd product is depicted in Figure 20. The process makes use of many already available parts like the MMS and CASM (ACSI) types. The first step in building the server is to create the model description (to write the .MDL file). The model file uses a syntax appropriate for the definition of the model. The notation is BNF based.

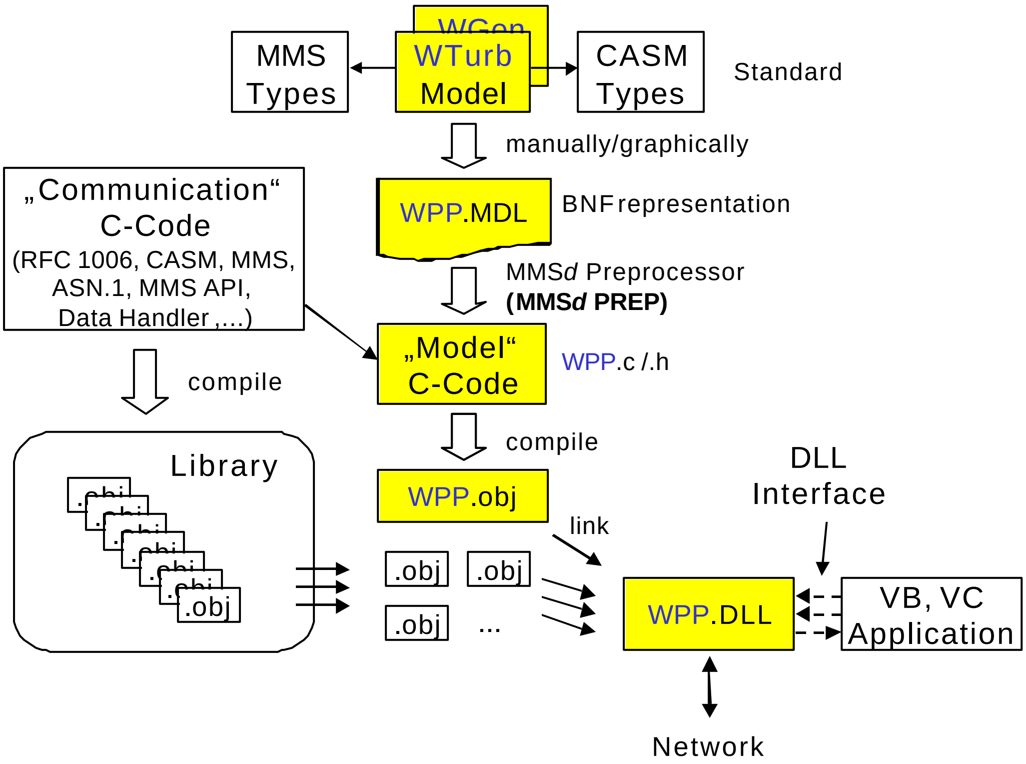

Figure 20 – The process of creating the server software

The basic idea of creating a server is to use a high level language to describe the content of the server (i.e. the model) The Server and all the content is described using a human and machine readable notation. The file that contains the model description has the extension ".MDL". The model represents the complete description of the server and all the instances that define the server (logical device, logical nodes, data objects and data attributes, ...).

The file with the model will then be processed by the MMSd Preprocessor (MMSd Prep). The MMSd Prep transforms the model into C-code and header files.

The MDL file (see [11]) starts with:

```
-- BNF File = Backus-Naur Form, language for standard
grammar definition technique
-- Global switches to define PREP processing
-- declare all variables in DOS RAM
%declare

-- This will cause all generated typedefs to be
written to this file
%typedefs wpp.h

-- Read in basic MMS foundation classes
%include ..\classlib\mmstyp9.mms

-- See MMSdPREP documentation
%construct_arrays

-- Now read in CASM basic class definitions
%include ..\classlib\casm9.mms

-- causes automatic reporting to be used
%reporting

-- More literal text to be included in output file
%text "#include <stdlib.h>\n"
%text "#include <stdio.h>\n"
%text "#include \"uca_time.h\"\n"
%text "#include \"tamvend.h\"\n"
%text "#include \"tam.h\"\n"
%text "#include \"tam_tp.h\"\n"
%text "#include \"wpp.h\"\n"
%text "#include \"packets.h\"\n"
...
```

The MDL file in the project is larger than MDLs for standardized models according to UCA/Gomsfe and 61850-7-4. The reason is that the WPP model has to be defined completely in the MDL file specific for this project. The WPP specific logical node models and the specific structures could be separated and moved to another file that contains the WPP model specific definitions. This file would be part of the software delivered by Tamarack for all applications in the wind power industry. The next time someone wants to create a server, he can just include the WPP specific definitions. The MDL file would just be 2 or 3 pages!

Next the interface to the server is defined.

The communication between the application and the server DLL is realized by a list of packets (two for each logical node: measurands (MX) and status (ST). The first two packets are for the MX and ST of the turbine logical node. The packets (ONLY the values (Int32) of the packets) exchanged between the application and the server DLL are “storeDLLpk01” and “storeDLLpk02”. These two store (write) the values of MX and ST into the server DLL. The structure of the two “packets” at the server DLL side is:

```
-- Define packet 01 (MX Turbine)
%text "long                    pk01[MAX_PK01];\n"  -- Points
received (32 bit!)
%text "unsigned char qu01;\n"                -- Quality of
points
%text "MMSd_time      ts01;\n"               -- Time last
received
%text "unsigned int  db01[MAX_PK01];\n" -- Deadbands
%text "float                sc01[MAX_PK01];\n" -- Scale of
points
%text "int                  un01[MAX_PK01];\n" -- Units of
points

-- Define packet 02 (ST Turbine)
%text "int                     pk02[MAX_PK02];\n"  -- Points
received
%text "unsigned char qu02;\n"                -- Quality of
points
%text "MMSd_time      ts02;\n"               -- Time last
received
%text "unsigned int  db02[MAX_PK02];\n" -- Deadbands
```

In addition to the values received via the “storeDLLpkxx” the structure contains also the quality information, the time stamp (set to the time the values have last received), and the deadband value to be applied for reporting of the values.

The quality information and the time stamp may also be provided by the application if available. In this case the “store” services have to carry theses values as well.

After the definition of the packets there are several WPP specific common classes defined.

The class for the measurement value of the turbine is defined as:

```
-- Measurement common classes
CLASS MVMX_Turb {        -- MX Attributes of MV
         AnalogInL  mVali                          +o
(params=&pk01[$$];

evalhand=MMSd_evalLongInteger;);
         AnalogInQ q              +o (params=&qu01;

evalhand=MMSd_evalBitString;);
         AnalogInT t              +o (params=&ts01;
                                      evalhand=;);
}
...
```

This class is applied in the class for the measurements (MX) of the logical node WTurb:

```
-- Wind Turbine Measurements
CLASS WTurb_MX {      -- Wind Turbine MX Attributes
    MVMX_Turb   WhG1         (params=0;);
    MVMX_Turb   WhG2         (params=1;);
    MVMX_Turb   WhConspt     (params=2;);
    MVMX_Turb   TimeG1       (params=3;);
    MVMX_Turb   TimeG2       (params=4;);
    MVMX_Turb   TimeFltSt    (params=5;);
    MVMX_Turb   TimeGridOk   (params=6;);
    MVMX_Turb   TimeWndProd  (params=7;);
    MVMX_Turb   TimeTotal    (params=8;);
}
```

The WTurb provides nine measurement data objects (.../WTurb.MX.WhG1, ...).

For each logical node two classes are defined: one for MX and one for ST.

Next the configuration specific classes for each logical node are defined.

For each logical node that is intended to provide reporting the **reporting control block** is defined. The reporting definition for the wind turbine is:

```
-- Wind Turbine Reporting
CLASS WTurb_RP {       -- Wind Turbine Report Control
Blocks
            BasRCB   brcbMX                      +orw
(params=MX;report=MX;);
            BasRCB   brcbST                      +orw
(params=ST;report=ST;);
}
```

There is one BasRCB for MX and one for ST. The BasRCB is a standard report control block in UCA2.0 (Gomsfe).

For each logical node that is intended to provide logging the **logging control block** is defined. The logging definition for the wind turbine is:

```
-- Wind Turbine Logging
CLASS WTurb_LG {       -- Wind Turbine Log Control
Blocks
            LCB61850   lcbMX                     +orw
(params=lcbMX;report=MX;);
            LCB61850   lcbST                     +orw
(params=lcbST;report=ST;);
}
```

There is one log control block for MX and one for ST.

After the definition of all components required for the definition of a logical node, the logical nodes can be build:

```
CLASS WTurb_Model {       -- Wind Turbine Model
    WTurb_MX MX;
    WTurb_ST ST;
    WTurb_CF CF;
    WTurb_RP RP;
    WTurb_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}
```

The WTurb model has five components: MX (meeasurements), ST (status), CF (configuration), RP (reporting), and LG (logging). Additionally there are two data sets defined: “MX” and “ST” for use by the report and log control blocks. The notation says that all measurements (^MX) and all status data objects (^ST) are include in the reports.

Before the logical device and server are specified, the following definition shows how any customized set of data objects can be defined to be reported with a separate (customized) report control block:

```
-- Custom Reporting
CLASS WCust_Report {          -- WCust Report Control
Block
    BasRCB    CustRCB                              +orw
(params=MX;report=CustList;);
    LIST         CustList         =         {/DI.Name,
Sigwards/WTurb.MX.WhG1,     Sigwards/WGen.MX.GenSpeed,
Sigwards/WGrid.MX.Power};
}
```

The CustRCB is a standard BasRCB wich reports values from the customized list of data objects defined in the CustList (4 data objects selected). Any other data object of the complete server can be included in the list.

Finally the complete logical device and the server can be defined:

```
SERVER
   DI_Model DI;
   JOURNAL WPP$Journal 32767;
   DEVICE Sigwards
      WTurb_Model      WTurb;
      WGen_Model       WGen;
      WGrid_Model      WGrid;
      WNace_Model      WNace;
      WGear_Model      WGear;
      WBrake_Model     WBrake;
      WRotor_Model     WRotor;
      WYaw_Model       WYaw;
      WEnv_Model       WEnv;
      WCust_Report     CustRP;
   END
END
```

The Server is composed of the device identification (DI), the journal for the logging control block, and the logical device with the name “Sigvards” including the customized report control (CustRP); any name of up to 32 characters can be used.

The logical device comprises all logical nodes defined in the model file above or in a file included. To remove a logical node it is just required to comment a line in the server definition out:

```
!!  WEnv_Model         WEnv;
```

The MDL file describes the complete server (including the logical device, logical nodes, ...). The model is still an abstract model. To produce C-code, the MDL file must be processed by the Tamarack MMS*d*PREP (MMS*d* preprocessor).

The preprocessor produces C source code and header files.

The last but one step is to compile the model source code. Finally the object code is linked together with other related code providing the communication and DLL interface related functions.

The DLL can now be used by a VB or C program (in the server) and by a client that communicates with the server over a network using the CASM/ACSI standard services based on the standard protocols like MMS, TCP/IP, Ethernet, ...

Adding new logical nodes or data classes for a given logical node requires that the DLL interface and the application have to be modified according top the additional definitions.

D The WPP.MDL file

```
-- BNF File = Backus-Naur Form, language for standard grammar definition
technique
-- Global switches to define PREP processing


-- declare all variables in DOS RAM
%declare


-- This will cause all generated typedefs to be written to this file
%typedefs wpp.h


-- Read in basic MMS foundation classes
%include ..\classlib\mmstyp9.mms


-- See MMSdPREP documentation
%construct_arrays


-- Now read in CASM basic class definitions
%include ..\classlib\casm9.mms


-- causes automatic reporting to be used
%reporting


-- More literal text to be included in output file
%text "#include <stdlib.h>\n"
%text "#include <stdio.h>\n"
%text "#include \"uca_time.h\"\n"
%text "#include \"tamvend.h\"\n"
%text "#include \"tam.h\"\n"
%text "#include \"tam_tp.h\"\n"
%text "#include \"wpp.h\"\n"
%text "#include \"packets.h\"\n"


-- Define packet 01 (MX Turbine)
%text "long          pk01[MAX_PK01];\n" -- Points received (32 bit!)
%text "unsigned char qu01;\n"           -- Quality of points
%text "MMSd_time     ts01;\n"           -- Time last received
%text "unsigned int  db01[MAX_PK01];\n" -- Deadbands
%text "float         sc01[MAX_PK01];\n" -- Scale of points
%text "int           un01[MAX_PK01];\n" -- Units of points


-- Define packet 02 (ST Turbine)
%text "int           pk02[MAX_PK02];\n" -- Points received
```

```
%text "MMSd_time     ts02;\n"            -- Time last received
%text "unsigned int  db02[MAX_PK02];\n" -- Deadbands
%text "unsigned char qu02;\n"            -- Quality of points

-- Define packet 03 (MX Generator)
%text "int           pk03[MAX_PK03];\n" -- Points received
%text "unsigned char qu03;\n"            -- Quality of points
%text "MMSd_time     ts03;\n"            -- Time last received
%text "unsigned int  db03[MAX_PK03];\n" -- Deadbands
%text "float         sc03[MAX_PK03];\n" -- Scale of points
%text "int           un03[MAX_PK03];\n" -- Units of points


-- Define packet 04 (ST Generator)
%text "int           pk04[MAX_PK04];\n" -- Points received
%text "unsigned char qu04;\n"            -- Quality of points
%text "MMSd_time     ts04;\n"            -- Time last received
%text "unsigned int  db04[MAX_PK04];\n" -- Deadbands


-- Define packet 05 (MX Grid)
%text "int           pk05[MAX_PK05];\n" -- Points received
%text "unsigned char qu05;\n"            -- Quality of points
%text "MMSd_time     ts05;\n"            -- Time last received
%text "unsigned int  db05[MAX_PK05];\n" -- Deadbands
%text "float         sc05[MAX_PK05];\n" -- Scale of points
%text "int           un05[MAX_PK05];\n" -- Units of points


-- Define packet 06 (ST Grid)
%text "int           pk06[MAX_PK06];\n" -- Points received
%text "unsigned char qu06;\n"            -- Quality of points
%text "MMSd_time     ts06;\n"            -- Time last received
%text "unsigned int  db06[MAX_PK06];\n" -- Deadbands


-- Define packet 07 (MX Nacelle)
%text "int           pk07[MAX_PK07];\n" -- Points received
%text "unsigned char qu07;\n"            -- Quality of points
%text "MMSd_time     ts07;\n"            -- Time last received
%text "unsigned int  db07[MAX_PK07];\n" -- Deadbands
%text "float         sc07[MAX_PK07];\n" -- Scale of points
%text "int           un07[MAX_PK07];\n" -- Units of points


-- Define packet 08 (ST Nacelle)
%text "int           pk08[MAX_PK08];\n" -- Points received
%text "unsigned char qu08;\n"            -- Quality of points
%text "MMSd_time     ts08;\n"            -- Time last received
%text "unsigned int  db08[MAX_PK08];\n" -- Deadbands


-- Define packet 09 (MX Gear)
%text "int           pk09[MAX_PK09];\n" -- Points received
%text "unsigned char qu09;\n"            -- Quality of points
%text "MMSd_time     ts09;\n"            -- Time last received
%text "unsigned int  db09[MAX_PK09];\n" -- Deadbands
%text "float         sc09[MAX_PK09];\n" -- Scale of points
```

```
%text "int           un09[MAX_PK09];\n" -- Units of points
-- Define packet 10 (ST Gear)
%text "int           pk10[MAX_PK10];\n" -- Points received
%text "unsigned char qu10;\n"           -- Quality of points
%text "MMSd_time     ts10;\n"           -- Time last received
%text "unsigned int  db10[MAX_PK10];\n" -- Deadbands


-- Define packet 11 (MX Brake)
%text "int           pk11[MAX_PK11];\n" -- Points received
%text "unsigned char qu11;\n"           -- Quality of points
%text "MMSd_time     ts11;\n"           -- Time last received
%text "unsigned int  db11[MAX_PK11];\n" -- Deadbands
%text "float         sc11[MAX_PK11];\n" -- Scale of points
%text "int           un11[MAX_PK11];\n" -- Units of points


-- Define packet 12 (ST Brake)
%text "int           pk12[MAX_PK12];\n" -- Points received
%text "unsigned char qu12;\n"           -- Quality of points
%text "MMSd_time     ts12;\n"           -- Time last received
%text "unsigned int  db12[MAX_PK12];\n" -- Deadbands


-- Define packet 13 (MX Rotor)
%text "int           pk13[MAX_PK13];\n" -- Points received
%text "unsigned char qu13;\n"           -- Quality of points
%text "MMSd_time     ts13;\n"           -- Time last received
%text "unsigned int  db13[MAX_PK13];\n" -- Deadbands
%text "float         sc13[MAX_PK13];\n" -- Scale of points
%text "int           un13[MAX_PK13];\n" -- Units of points


-- Define packet 14 (ST Rotor)
%text "int           pk14[MAX_PK14];\n" -- Points received
%text "unsigned char qu14;\n"           -- Quality of points
%text "MMSd_time     ts14;\n"           -- Time last received
%text "unsigned int  db14[MAX_PK14];\n" -- Deadbands


-- Define packet 15 (MX Yaw)
%text "int           pk15[MAX_PK15];\n" -- Points received
%text "unsigned char qu15;\n"           -- Quality of points
%text "MMSd_time     ts15;\n"           -- Time last received
%text "unsigned int  db15[MAX_PK15];\n" -- Deadbands
%text "float         sc15[MAX_PK15];\n" -- Scale of points
%text "int           un15[MAX_PK15];\n" -- Units of points


-- Define packet 16 (ST Yaw)
%text "int           pk16[MAX_PK16];\n" -- Points received
%text "unsigned char qu16;\n"           -- Quality of points
%text "MMSd_time     ts16;\n"           -- Time last received
%text "unsigned int  db16[MAX_PK16];\n" -- Deadbands


-- Define packet 17 (MX Environment)
```

```
%text "unsigned char qu17;\n"           -- Quality of points
%text "MMSd_time     ts17;\n"           -- Time last received
%text "int           pk17[MAX_PK17];\n" -- Points received
%text "unsigned int  db17[MAX_PK17];\n" -- Deadbands
%text "float         sc17[MAX_PK17];\n" -- Scale of points
%text "int           un17[MAX_PK17];\n" -- Units of points


-- Define packet 18 (ST Environment)
%text "int           pk18[MAX_PK18];\n" -- Points received
%text "unsigned char qu18;\n"           -- Quality of points
%text "MMSd_time     ts18;\n"           -- Time last received
%text "unsigned int  db18[MAX_PK18];\n" -- Deadbands


-- Define packet 20 (MX Meteorological station)
%text "int           pk20[MAX_PK20];\n" -- Points received
%text "unsigned char qu20;\n"           -- Quality of points
%text "MMSd_time     ts20;\n"           -- Time last received
%text "unsigned int  db20[MAX_PK20];\n" -- Deadbands
%text "float         sc20[MAX_PK20];\n" -- Scale of points
%text "int           un20[MAX_PK20];\n" -- Units of points


-- Define controls CO data (GS - 01/06/01)
%text "int           ctls[MAX_CTLS];\n" -- Controls written by client (32 bit)
%text "CCtlCallBack  ctlCallBackFunctionC = NULL;\n"
%text "VbCtlCallBack ctlCallBackFunctionVb = NULL;\n"


-- Default array subscript range
%text "#define n 5\n"


-- These are used to interface with the CASM reporting model
%text "#include \"rpt_if.h\"\n"
%text "#include \"evalfunc.h\"\n"
%text "unsigned char LTC_conditions = 0;\n"
%text "\n"


-- Now read in GOMSFE basic class definitions
%include ..\classlib\gomsfe9.mms


-- Causes the default end processing code to be included
%append ..\classlib\full7.end

-- Added controls by GS - 01/06/01 */
%typetext "typedef struct {\n"
%typetext "    int            ctlValue;\n"
%typetext "    int            stValue;\n"
%typetext "    unsigned char  q[2];\n"
%typetext "    MMSd_time      t;\n"
%typetext "    int            ctlModel;\n"
%typetext "} WPP_Control;\n"

PRIMITIVE MMSd_AnalogInL (
    size=4;
```

```
    sizekind=bytes;
    readhand=MMSd_readLongInteger;
    params=$$;
    writehand=MMSd_writeLongInteger;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrLongInteger;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexLongInteger;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef MMSd_LongInteger MMSd_AnalogInL;\n"


CLASS AnalogInL    :MMSd_AnalogInL     (params=baseparams;)


-- Measurement common classes
CLASS MVMX_Turb {        -- MX Attributes of MV
         AnalogInL mVali           +o (params=&pk01[$$];
                                       evalhand=MMSd_evalLongInteger;);
         AnalogInQ q               +o (params=&qu01;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts01;
                                       evalhand=;);
}


CLASS MVCF_Turb {        -- CF Attributes of MV
                  s s              +o (params=&sc01[$$];);
                  u u              +o (params=&un01[$$];);
                 db db             +o (params=&db01[$$];);
}


CLASS MVMX_Gen {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk03[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu03;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts03;
                                       evalhand=;);
}


CLASS MVCF_Gen {        -- CF Attributes of MV
                  s s              +o (params=&sc03[$$];);
                  u u              +o (params=&un03[$$];);
                 db db             +o (params=&db03[$$];);
}


CLASS MVMX_Grid {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk05[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu05;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts05;
```

```
}

                                       evalhand=;);
CLASS MVCF_Grid {        -- CF Attributes of MV
                 s s               +o (params=&sc05[$$];);
                 u u               +o (params=&un05[$$];);
                db db              +o (params=&db05[$$];);
}


CLASS MVMX_Nace {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk07[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu07;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts07;
                                       evalhand=;);
}


CLASS MVCF_Nace {        -- CF Attributes of MV
                 s s               +o (params=&sc07[$$];);
                 u u               +o (params=&un07[$$];);
                db db              +o (params=&db07[$$];);
}


CLASS MVMX_Gear {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk09[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu09;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts09;
                                       evalhand=;);
}


CLASS MVCF_Gear {        -- CF Attributes of MV
                 s s               +o (params=&sc09[$$];);
                 u u               +o (params=&un09[$$];);
                db db              +o (params=&db09[$$];);
}


CLASS MVMX_Brake {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk11[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu11;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts11;
                                       evalhand=;);
}


CLASS MVCF_Brake {        -- CF Attributes of MV
                 s s               +o (params=&sc11[$$];);
                 u u               +o (params=&un11[$$];);
                db db              +o (params=&db11[$$];);
```

```
}
CLASS MVMX_Rotor {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk13[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu13;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts13;
                                       evalhand=;);
}


CLASS MVCF_Rotor {        -- CF Attributes of MV
                 s s               +o (params=&sc13[$$];);
                 u u               +o (params=&un13[$$];);
                db db              +o (params=&db13[$$];);
}


CLASS MVMX_Yaw {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk15[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu15;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts15;
                                       evalhand=;);
}


CLASS MVCF_Yaw {        -- CF Attributes of MV
                 s s               +o (params=&sc15[$$];);
                 u u               +o (params=&un15[$$];);
                db db              +o (params=&db15[$$];);
}


CLASS MVMX_Env {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk17[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu17;
                                       evalhand=MMSd_evalBitString;);
         AnalogInT t               +o (params=&ts17;
                                       evalhand=;);
}


CLASS MVCF_Env {        -- CF Attributes of MV
                 s s               +o (params=&sc17[$$];);
                 u u               +o (params=&un17[$$];);
                db db              +o (params=&db17[$$];);
}


CLASS MVMX_Met {        -- MX Attributes of MV
         AnalogInI mVali           +o (params=&pk20[$$];
                                       evalhand=MMSd_evalInteger;);
         AnalogInQ q               +o (params=&qu20;
                                       evalhand=MMSd_evalBitString;);
```

```
                                        evalhand=;);
}
         AnalogInT t               +o (params=&ts20;

CLASS MVCF_Met {        -- CF Attributes of MV
                 s s               +o (params=&sc20[$$];);
                 u u               +o (params=&un20[$$];);
                db db              +o (params=&db20[$$];);
}



-- This introduces a WPP-specific primitive type. The methods (see HANDLERS.C)
-- for this picks out a single bit of a word in the packet.  The void *param
-- passed to the handlers have (instead of a pointer) the word index and bit
-- offset to use.
PRIMITIVE WTurb_StatusInB1 (
    size=1;
    params=$$;
    sizekind=bits;
    readhand=WTurb_readBitOfString;
    writehand=WTurb_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WTurb_StatusInB1;\n"


NEW CLASS   StatusInB1_Turb   :WTurb_StatusInB1   (params=baseparams;)


PRIMITIVE WGen_StatusInB1 (
    size=1;
    params=$$;
    sizekind=bits;
    readhand=WGen_readBitOfString;
    writehand=WGen_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WGen_StatusInB1;\n"


NEW CLASS   StatusInB1_Gen    :WGen_StatusInB1    (params=baseparams;)


PRIMITIVE WGrid_StatusInB1 (
    size=1;
```

```
    sizekind=bits;
    readhand=WGrid_readBitOfString;
    params=$$;
    writehand=WGrid_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WGrid_StatusInB1;\n"


NEW CLASS   StatusInB1_Grid    :WGrid_StatusInB1    (params=baseparams;)


PRIMITIVE WNace_StatusInB1 (
    size=1;
    params=$$;
    sizekind=bits;
    readhand=WNace_readBitOfString;
    writehand=WNace_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WNace_StatusInB1;\n"


NEW CLASS   StatusInB1_Nace    :WNace_StatusInB1    (params=baseparams;)


PRIMITIVE WGear_StatusInB1 (
    size=1;
    params=$$;
    sizekind=bits;
    readhand=WGear_readBitOfString;
    writehand=WGear_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WGear_StatusInB1;\n"


NEW CLASS   StatusInB1_Gear    :WGear_StatusInB1    (params=baseparams;)


PRIMITIVE WBrake_StatusInB1 (
```

```
    params=$$;
    sizekind=bits;
    size=1;
    readhand=WBrake_readBitOfString;
    writehand=WBrake_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WBrake_StatusInB1;\n"


NEW CLASS   StatusInB1_Brake    :WBrake_StatusInB1    (params=baseparams;)


PRIMITIVE WRotor_StatusInB1 (
    size=1;
    params=$$;
    sizekind=bits;
    readhand=WRotor_readBitOfString;
    writehand=WRotor_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WRotor_StatusInB1;\n"


NEW CLASS   StatusInB1_Rotor    :WRotor_StatusInB1    (params=baseparams;)


PRIMITIVE WYaw_StatusInB1 (
    size=1;
    params=$$;
    sizekind=bits;
    readhand=WYaw_readBitOfString;
    writehand=WYaw_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WYaw_StatusInB1;\n"


NEW CLASS   StatusInB1_Yaw    :WYaw_StatusInB1    (params=baseparams;)
```

```
PRIMITIVE WEnv_StatusInB1 (
    size=1;
    params=$$;
    sizekind=bits;
    readhand=WEnv_readBitOfString;
    writehand=WEnv_writeBitOfString;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBitString;
    evalhand=NULL_HANDLER;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WEnv_StatusInB1;\n"


NEW CLASS   StatusInB1_Env    :WEnv_StatusInB1    (params=baseparams;)


-- Added controls by GS - 01/06/01
PRIMITIVE WPP_CtlVal (
    size=4;
    params=$$;
    sizekind=bytes;
    readhand=WPP_readControl;
    writehand=WPP_writeControl;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrInteger;
    evalhand=;
    nameparam=NULL;
    index=MMSd_indexInteger;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef int WPP_CtlVal;\n"

-- Added controls by GS - 01/06/01
PRIMITIVE WPP_CtlStatus (
    size=4;
    params=$$;
    sizekind=bytes;
    readhand=WPP_readControlStatus;
    writehand=MMSd_writeFailure;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrInteger;
    evalhand=WPP_evalControlStatus;
    nameparam=NULL;
    index=MMSd_indexInteger;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef int WPP_CtlStatus;\n"

-- Added controls by GS - 01/06/01
PRIMITIVE WPP_CtlQuality (
    size=11;
    params=$$;
    sizekind=bits;
```

```
    writehand=MMSd_writeFailure;
    namehand=NULL_HANDLER;
    readhand=WPP_readControlQuality;
    attrhand=MMSd_attrBitString;
    evalhand=WPP_evalControlQuality;
    nameparam=NULL;
    index=MMSd_indexBitString;
    array=NOARRAY;
    userData=(void*)NULL;
)
        DEFINED {
                        invalid[0],
                        notTopical[1],
                        substituted[2],
                        operatorBlocked[3],
                        overFlow[4],
                        outOfRange[5],
                        badReference[6],
                        commFailure[7],
                        commBlocked[8],
                        inputBlocked[9],
                        defaultValue[10]
                }
%typetext "typedef unsigned char WPP_CtlQuality[2];\n"

-- Added controls by GS - 01/06/01
PRIMITIVE WPP_CtlTime (
    size=6;
    params=$$;
    sizekind=bytes;
    readhand=WPP_readControlTime;
    writehand=MMSd_writeFailure;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrBtime;
    evalhand=;
    nameparam=NULL;
    index=MMSd_indexBtime;
    array=NOARRAY;
    userData=(void*)NULL;
)
%typetext "typedef unsigned char WPP_CtlTime;\n"

-- Added controls by GS - 01/06/01
PRIMITIVE WPP_CtlConfig (
    size=4;
    params=$$;
    sizekind=bytes;
    readhand=WPP_readControlConfig;
    writehand=WPP_writeControlConfig;
    namehand=NULL_HANDLER;
    attrhand=MMSd_attrInteger;
    evalhand=;
    nameparam=NULL;
    index=MMSd_indexInteger;
    array=NOARRAY;
    userData=(void*)NULL;
)

-- Added controls by GS - 01/06/01
```

```
CLASS WGen_CtlValue    :WPP_CtlVal
CLASS WGen_CtlStatus   :WPP_CtlStatus
%typetext "typedef int WPP_CtlConfig;\n"
CLASS WGen_CtlQuality  :WPP_CtlQuality
CLASS WGen_CtlTime     :WPP_CtlTime
CLASS WGen_CtlConfig   :WPP_CtlConfig


-- Status common classes
CLASS SPSST_Turb {       -- ST Attributes of SP
            StatusInB1_Turb stVal  +o (params=$$;
                                       evalhand=WTurb_evalBitOfString;);
            StatusInQ  q           +o (params=&qu02;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts02;
                                       evalhand=;);
}


CLASS ISIST_Turb {       -- ST Attributes of ISI
            AnalogInI  stVal       +o (params=&pk02[$$];
                                       evalhand=MMSd_evalInteger;);
            StatusInQ  q           +o (params=&qu02;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts02;
                                       evalhand=;);
}


CLASS ISICF_Turb {        -- CF Attributes of ISI
                db db              +o (params=&db02[$$];);
}


CLASS SPSST_Gen {       -- ST Attributes of SP
            StatusInB1_Gen stVal   +o (params=$$;
                                       evalhand=WGen_evalBitOfString;);
            StatusInQ  q           +o (params=&qu04;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts04;
                                       evalhand=;);
}


CLASS ISIST_Gen {       -- ST Attributes of ISI
            AnalogInI  stVal       +o (params=&pk04[$$];
                                       evalhand=MMSd_evalInteger;);
            StatusInQ  q           +o (params=&qu04;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts04;
                                       evalhand=;);
}


CLASS ISICF_Gen {        -- CF Attributes of ISI
                db db              +o (params=&db04[$$];);
}
```

```
CLASS SPSST_Grid {       -- ST Attributes of SP
            StatusInB1_Grid stVal  +o (params=$$;
                                       evalhand=WGrid_evalBitOfString;);
            StatusInQ  q           +o (params=&qu06;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts06;
                                       evalhand=;);
}


CLASS SPSST_Nace {       -- ST Attributes of SP
            StatusInB1_Nace stVal  +o (params=$$;
                                       evalhand=WNace_evalBitOfString;);
            StatusInQ  q           +o (params=&qu08;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts08;
                                       evalhand=;);
}


CLASS SPSST_Gear {       -- ST Attributes of SP
            StatusInB1_Gear stVal  +o (params=$$;
                                       evalhand=WGear_evalBitOfString;);
            StatusInQ  q           +o (params=&qu10;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts10;
                                       evalhand=;);
}


CLASS SPSST_Brake {       -- ST Attributes of SP
            StatusInB1_Brake stVal +o (params=$$;
                                       evalhand=WBrake_evalBitOfString;);
            StatusInQ  q           +o (params=&qu12;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts12;
                                       evalhand=;);
}


CLASS SPSST_Rotor {       -- ST Attributes of SP
            StatusInB1_Rotor stVal +o (params=$$;
                                       evalhand=WRotor_evalBitOfString;);
            StatusInQ  q           +o (params=&qu14;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts14;
                                       evalhand=;);
}


CLASS SPSST_Yaw {       -- ST Attributes of SP
            StatusInB1_Yaw stVal   +o (params=$$;
                                       evalhand=WYaw_evalBitOfString;);
            StatusInQ  q           +o (params=&qu16;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts16;
                                       evalhand=;);
}
```

```
CLASS SPSST_Env {       -- ST Attributes of SP
            StatusInB1_Env stVal   +o (params=$$;
                                       evalhand=WEnv_evalBitOfString;);
            StatusInQ  q           +o (params=&qu18;
                                       evalhand=MMSd_evalBitString;);
            StatusInT  t           +o (params=&ts18;
                                       evalhand=;);
}


-- Added controls by GS - 01/06/01
CLASS ISCCO_Gen {        -- CO Attributes of ISC
     WPP_CtlVal ctlVal      +m (params=$$;);
}
CLASS ISCST_Gen {        -- ST Attributes of ISC
  WPP_CtlStatus stVal       +m (params=$$;);
 WPP_CtlQuality q           +m (params=$$;);
    WPP_CtlTime t           +m (params=$$;);
}
CLASS ISCCF_Gen {        -- CF Attributes of ISC
  WPP_CtlConfig ctlModel    +m (params=$$;);
}


-- Wind Turbine Measurements
CLASS WTurb_MX {      -- Wind Turbine MX Attributes
    MVMX_Turb   WhG1         (params=0;);
    MVMX_Turb   WhG2         (params=1;);
    MVMX_Turb   WhConspt     (params=2;);
    MVMX_Turb   TimeG1       (params=3;);
    MVMX_Turb   TimeG2       (params=4;);
    MVMX_Turb   TimeFltSt    (params=5;);
    MVMX_Turb   TimeGridOk   (params=6;);
    MVMX_Turb   TimeWndProd  (params=7;);
    MVMX_Turb   TimeTotal    (params=8;);
}


-- Wind Generator Measurements
CLASS WGen_MX {      -- Wind Generator MX Attributes
    MVMX_Gen   GenSpeed     (params=0;);
    MVMX_Gen   Slip         (params=1;);
    MVMX_Gen   GenA         (params=2;);
    MVMX_Gen   GenBeTemp    (params=3;);
    MVMX_Gen   GenTemp      (params=4;);
    MVMX_Gen   Gen2Temp     (params=5;);
    MVMX_Gen   DFacSToGen   (params=6;);
}


-- Wind Grid Measurements
CLASS WGrid_MX {      -- Wind Grid MX Attributes
    MVMX_Grid   Power        (params=0;);
    MVMX_Grid   APhsA0       (params=1;);
    MVMX_Grid   CosPhi       (params=2;);
    MVMX_Grid   VPhsA        (params=3;);
    MVMX_Grid   VPhsB        (params=4;);
    MVMX_Grid   VPhsC        (params=5;);
```

```
    MVMX_Grid   APhsB        (params=7;);
    MVMX_Grid   APhsC        (params=8;);
    MVMX_Grid   APhsA        (params=6;);
    MVMX_Grid   VAr          (params=9;);
    MVMX_Grid   Hz           (params=10;);
}


-- Wind Nacelle Measurements
CLASS WNace_MX {      -- Wind Nacelle MX Attributes
    MVMX_Nace   AccX         (params=0;);
    MVMX_Nace   AccY         (params=1;);
    MVMX_Nace   VibXMax      (params=2;);
    MVMX_Nace   VibYMax      (params=3;);
    MVMX_Nace   VibXRMS      (params=4;);
    MVMX_Nace   VibYRMS      (params=5;);
    MVMX_Nace   NaclTemp     (params=6;);
    MVMX_Nace   PwrPnlTemp   (params=7;);
    MVMX_Nace   WtrToClrTemp (params=8;);
    MVMX_Nace   WtrFrmClrTemp (params=9;);
}


-- Wind Gear Measurements
CLASS WGear_MX {      -- Wind Gear MX Attributes
    MVMX_Gear   GeaOilTemp   (params=0;);
    MVMX_Gear   GeaOil2Temp  (params=1;);
}


-- Wind Brake Measurements
CLASS WBrake_MX {      -- Wind Brake MX Attributes
    MVMX_Brake  Calip1       (params=0;);
    MVMX_Brake  Calip2       (params=1;);
}


-- Wind Rotor Measurements
CLASS WRotor_MX {      -- Wind Rotor MX Attributes
    MVMX_Rotor  RotSpd       (params=0;);
    MVMX_Rotor  RotPos       (params=1;);
}


-- Wind Yaw Measurements
CLASS WYaw_MX {      -- Wind Yaw MX Attributes
    MVMX_Yaw    YawP         (params=0;);
    MVMX_Yaw    YawMalgmt    (params=1;);
    MVMX_Yaw    YawMalgmt2   (params=2;);
    MVMX_Yaw    CablTwst     (params=3;);
    MVMX_Yaw    YawSpd       (params=4;);
    MVMX_Yaw    YawOilTemp   (params=5;);
}


-- Wind Environment Measurements
CLASS WEnv_MX {      -- Wind Environment MX Attributes
    MVMX_Env    WindSpd      (params=0;);
    MVMX_Env    WindSpd2     (params=1;);
```

```
    MVMX_Env    AirPres      (params=3;);
}
    MVMX_Env    OutdrTemp    (params=2;);

-- Meteorological station Measurements
CLASS WMet_MX {
    MVMX_Met   WindSpd10    (params=0;);
    MVMX_Met   WindSpd38    (params=1;);
    MVMX_Met   WindSpd40    (params=2;);
    MVMX_Met   WindSpd54    (params=3;);
    MVMX_Met   WindSpd56    (params=4;);
    MVMX_Met   WindSpd75    (params=5;);
    MVMX_Met   WindSpd77    (params=6;);
    MVMX_Met   WindSpd96    (params=7;);
    MVMX_Met   WindSpd98    (params=8;);
    MVMX_Met   WindSpd120   (params=9;);
    MVMX_Met   WindSpd122   (params=10;);
    MVMX_Met   WindSpd145   (params=11;);
    MVMX_Met   WindDir40    (params=12;);
    MVMX_Met   WindDir56    (params=13;);
    MVMX_Met   WindDir77    (params=14;);
    MVMX_Met   WindDir98    (params=15;);
    MVMX_Met   WindDir122   (params=16;);
    MVMX_Met   Temp1        (params=17;);
    MVMX_Met   Temp10       (params=18;);
    MVMX_Met   Temp38       (params=19;);
    MVMX_Met   Temp54       (params=20;);
    MVMX_Met   Temp75       (params=21;);
    MVMX_Met   Temp96       (params=22;);
    MVMX_Met   Temp120      (params=23;);
    MVMX_Met   Temp145      (params=24;);
    MVMX_Met   AirPres      (params=25;);
    MVMX_Met   Rain         (params=26;);
}



%text "#define SPS(a,b) (void*)((a<<8)|b)\n"
-- Wind Turbine Status
CLASS WTurb_ST {      -- Wind Turbine Status Attributes
    SPSST_Turb  Error           (params=SPS(0,15););
    SPSST_Turb  Warn            (params=SPS(0,14););
    SPSST_Turb  FreeToYaw       (params=SPS(0,13););
    SPSST_Turb  FreeToOp        (params=SPS(0,12););
    SPSST_Turb  FreeRun         (params=SPS(0,11););
    SPSST_Turb  SafeChn         (params=SPS(0,10););
    SPSST_Turb  RemCtlInf       (params=SPS(0,9););

    ISIST_Turb  RstLvl          (params=2;);
    ISIST_Turb  StCod           (params=3;);
    ISIST_Turb  ActvFltCod      (params=4;);
    ISIST_Turb  ActvFltCod2     (params=5;);
    ISIST_Turb  ActvFltCod3     (params=6;);
    ISIST_Turb  ActvFltCod4     (params=7;);
}


-- Wind Generator Status
```

```
CLASS WGen_ST {      -- Wind Generator Status Attributes
    SPSST_Gen  TyrOpen          (params=SPS(0,15););
-- Added controls by GS - 01/06/01
    SPSST_Gen  GenCon           (params=SPS(0,14););
    SPSST_Gen  HeatGen          (params=SPS(0,13););

    ISIST_Gen  SWW              (params=2;);
    ISCST_Gen  Mode             (params=0;);
}

-- Added controls by GS - 01/06/01
CLASS WGen_CO {      -- Wind Generator Control Attributes
    ISCCO_Gen Mode              (params=0;);
}

-- Wind Grid Status
CLASS WGrid_ST {      -- Wind Grid Status Attributes
    SPSST_Grid  PhCom           (params=SPS(0,15););
}


-- Wind Nacelle Status
CLASS WNace_ST {      -- Wind Nacelle Status Attributes
    SPSST_Nace  WtrPump         (params=SPS(0,15););
    SPSST_Nace  VentNac         (params=SPS(0,14););
}


-- Wind Gear Status
CLASS WGear_ST {      -- Wind Gear Status Attributes
    SPSST_Gear  OilPump         (params=SPS(0,15););
}


-- Wind Brake Status
CLASS WBrake_ST {      -- Wind Brake Status Attributes
    SPSST_Brake Brake50         (params=SPS(0,15););
    SPSST_Brake Brake75         (params=SPS(0,14););
    SPSST_Brake Brake199        (params=SPS(0,13););
    SPSST_Brake Brake200        (params=SPS(0,12););
    SPSST_Brake DiskBrk1        (params=SPS(0,11););
    SPSST_Brake DiskBrk2        (params=SPS(0,10););
    SPSST_Brake SoftBrk         (params=SPS(0,9););
    SPSST_Brake HydPmpBrk       (params=SPS(0,8););
}


-- Wind Rotor Status
CLASS WRotor_ST {      -- Wind Rotor Status Attributes
    SPSST_Rotor HydPmpHub       (params=SPS(0,15););
    SPSST_Rotor HubHydrPump     (params=SPS(0,14););
    SPSST_Rotor HubHydrSole     (params=SPS(0,13););
}


-- Wind Yaw Status
CLASS WYaw_ST {      -- Wind Yaw Status Attributes
    SPSST_Yaw   YawCCW          (params=SPS(0,15););
    SPSST_Yaw   YawCW           (params=SPS(0,14););
```

```
    SPSST_Yaw   HydPmpYaw       (params=SPS(0,12););
}
    SPSST_Yaw   AuRewCab        (params=SPS(0,13););

-- Wind Environment Status
CLASS WEnv_ST {      -- Wind Environment Status Attributes
    SPSST_Env   HeatWndGau      (params=SPS(0,15););
}



-- Wind Turbine Configuration
CLASS WTurb_CF {      -- Wind Turbine Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Turb   WhG1             +orw (params=0;);
    MVCF_Turb   WhG2             +orw (params=1;);
    MVCF_Turb   WhConspt         +orw (params=2;);
    MVCF_Turb   TimeG1           +orw (params=3;);
    MVCF_Turb   TimeG2           +orw (params=4;);
    MVCF_Turb   TimeFltSt        +orw (params=5;);
    MVCF_Turb   TimeGridOk       +orw (params=6;);
    MVCF_Turb   TimeWndProd      +orw (params=7;);
    MVCF_Turb   TimeTotal        +orw (params=8;);

    ISICF_Turb  RstLvl           +orw (params=2;);
    ISICF_Turb  StCod            +orw (params=3;);
    ISICF_Turb  ActvFltCod       +orw (params=4;);
    ISICF_Turb  ActvFltCod2      +orw (params=5;);
    ISICF_Turb  ActvFltCod3      +orw (params=6;);
    ISICF_Turb  ActvFltCod4      +orw (params=7;);
}


-- Wind Generator Configuration
CLASS WGen_CF {      -- Wind Turbine Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Gen   GenSpeed          +orw (params=0;);
    MVCF_Gen   Slip              +orw (params=1;);
    MVCF_Gen   GenA              +orw (params=2;);
    MVCF_Gen   GenBeTemp         +orw (params=3;);
    MVCF_Gen   GenTemp           +orw (params=4;);
    MVCF_Gen   Gen2Temp          +orw (params=5;);
    MVCF_Gen   DFacSToGen        +orw (params=6;);

    ISICF_Gen  SWW               +orw (params=2;);
    ISCCF_Gen  Mode              +orw (params=0;);
}


-- Wind Grid Configuration
CLASS WGrid_CF {      -- Wind Grid Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Grid   Power            +orw (params=0;);
    MVCF_Grid   APhsA0           +orw (params=1;);
    MVCF_Grid   CosPhi           +orw (params=2;);
```

```
    MVCF_Grid   VPhsB            +orw (params=4;);
    MVCF_Grid   VPhsC            +orw (params=5;);
    MVCF_Grid   VPhsA            +orw (params=3;);
    MVCF_Grid   APhsA            +orw (params=6;);
    MVCF_Grid   APhsB            +orw (params=7;);
    MVCF_Grid   APhsC            +orw (params=8;);
    MVCF_Grid   VAr              +orw (params=8;);
    MVCF_Grid   Hz               +orw (params=8;);
}


-- Wind Nacelle Configuration
CLASS WNace_CF {      -- Wind Nacelle Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Nace   AccX             +orw (params=0;);
    MVCF_Nace   AccY             +orw (params=1;);
    MVCF_Nace   VibXMax          +orw (params=2;);
    MVCF_Nace   VibYMax          +orw (params=3;);
    MVCF_Nace   VibXRMS          +orw (params=4;);
    MVCF_Nace   VibYRMS          +orw (params=5;);
    MVCF_Nace   NaclTemp         +orw (params=6;);
    MVCF_Nace   PwrPnlTemp       +orw (params=7;);
    MVCF_Nace   WtrToClrTemp     +orw (params=8;);
    MVCF_Nace   WtrFrmClrTemp    +orw (params=8;);
}


-- Wind Gear Configuration
CLASS WGear_CF {      -- Wind Gear Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Gear   GeaOilTemp       +orw (params=0;);
    MVCF_Gear   GeaOil2Temp      +orw (params=1;);
}


-- Wind Brake Configuration
CLASS WBrake_CF {      -- Wind Brake Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Brake  Calip1           +orw (params=0;);
    MVCF_Brake  Calip2           +orw (params=1;);
}


-- Wind Rotor Configuration
CLASS WRotor_CF {      -- Wind Rotor Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Rotor  RotSpd           +orw (params=0;);
    MVCF_Rotor  RotPos           +orw (params=1;);
}


-- Wind Yaw Configuration
CLASS WYaw_CF {      -- Wind Yaw Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);
```

```
    MVCF_Yaw    YawMalgmt        +orw (params=1;);
    MVCF_Yaw    YawMalgmt2       +orw (params=2;);
    MVCF_Yaw    YawP             +orw (params=0;);
    MVCF_Yaw    CablTwst         +orw (params=3;);
    MVCF_Yaw    YawSpd           +orw (params=4;);
    MVCF_Yaw    YawOilTemp       +orw (params=5;);
}


-- Wind Environment Configuration
CLASS WEnv_CF {      -- Wind Environment Configuration Attributes
    BTIME6 ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Env    WindSpd          +orw (params=0;);
    MVCF_Env    WindSpd2         +orw (params=1;);
    MVCF_Env    OutdrTemp        +orw (params=2;);
    MVCF_Env    AirPres          +orw (params=3;);
}


-- Meteorological station Configuration
CLASS WMet_CF {
    BTIME6     ClockTOD         +cr  (params=@&ClockTod;);

    MVCF_Met   WindSpd10        +orw (params=0;);
    MVCF_Met   WindSpd38        +orw (params=1;);
    MVCF_Met   WindSpd40        +orw (params=2;);
    MVCF_Met   WindSpd54        +orw (params=3;);
    MVCF_Met   WindSpd56        +orw (params=4;);
    MVCF_Met   WindSpd75        +orw (params=5;);
    MVCF_Met   WindSpd77        +orw (params=6;);
    MVCF_Met   WindSpd96        +orw (params=7;);
    MVCF_Met   WindSpd98        +orw (params=8;);
    MVCF_Met   WindSpd120       +orw (params=9;);
    MVCF_Met   WindSpd122       +orw (params=10;);
    MVCF_Met   WindSpd145       +orw (params=11;);
    MVCF_Met   WindDir40        +orw (params=12;);
    MVCF_Met   WindDir56        +orw (params=13;);
    MVCF_Met   WindDir77        +orw (params=14;);
    MVCF_Met   WindDir98        +orw (params=15;);
    MVCF_Met   WindDir122       +orw (params=16;);
    MVCF_Met   Temp1            +orw (params=17;);
    MVCF_Met   Temp10           +orw (params=18;);
    MVCF_Met   Temp38           +orw (params=19;);
    MVCF_Met   Temp54           +orw (params=20;);
    MVCF_Met   Temp75           +orw (params=21;);
    MVCF_Met   Temp96           +orw (params=22;);
    MVCF_Met   Temp120          +orw (params=23;);
    MVCF_Met   Temp145          +orw (params=24;);
    MVCF_Met   AirPres          +orw (params=25;);
    MVCF_Met   Rain             +orw (params=26;);
}



-- WPP Reporting
-- CLASS WPP_RP {      -- WPP Report Control Blocks
--            BasRCB brcbMX       +orw   (params=MX;report=MX;);
--            BasRCB brcbST       +orw   (params=ST;report=ST;);
```

```
== }Wind Turbine Reporting
CLASS WTurb_RP {      -- Wind Turbine Report Control Blocks
            BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
            BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Wind Generator Reporting
CLASS WGen_RP {      -- Wind Generator Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Wind Grid Reporting
CLASS WGrid_RP {      -- Wind Grid Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Wind Nacelle Reporting
CLASS WNace_RP {      -- Wind Nacelle Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Wind Gear Reporting
CLASS WGear_RP {      -- Wind Gear Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Wind Brake Reporting
CLASS WBrake_RP {      -- Wind Brake Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Wind Rotor Reporting
CLASS WRotor_RP {      -- Wind Rotor Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Wind Yaw Reporting
CLASS WYaw_RP {      -- Wind Yaw Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}
```

```
CLASS WEnv_RP {      -- Wind Env Report Control Blocks
           BasRCB brcbMX       +orw   (params=brcbMX;report=MX;);
-- Wind Env Reporting
           BasRCB brcbST       +orw   (params=brcbST;report=ST;);
}


-- Meteorological station Reporting
CLASS WMet_RP {
           BasRCB brcbMX       +orw   (params=MX;report=MX;);
}



-- Wind Turbine Logging
CLASS WTurb_LG {      -- Wind Turbine Log Control Blocks
            LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
            LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


-- Wind Generator Logging
CLASS WGen_LG {      -- Wind Generator Log Control Blocks
           LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
           LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


-- Wind Grid Logging
CLASS WGrid_LG {      -- Wind Grid Log Control Blocks
           LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
           LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


-- Wind Nacelle Logging
CLASS WNace_LG {      -- Wind Nacelle Log Control Blocks
           LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
           LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


-- Wind Gear Logging
CLASS WGear_LG {      -- Wind Gear Log Control Blocks
           LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
           LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


-- Wind Brake Logging
CLASS WBrake_LG {      -- Wind Brake Log Control Blocks
           LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
           LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


-- Wind Rotor Logging
CLASS WRotor_LG {      -- Wind Rotor Log Control Blocks
           LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
```

```
}

            LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
-- Wind Yaw Logging
CLASS WYaw_LG {      -- Wind Yaw Log Control Blocks
            LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
            LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


-- Wind Env Logging
CLASS WEnv_LG {      -- Wind Env Log Control Blocks
            LCB61850 lcbMX       +orw   (params=lcbMX;report=MX;);
            LCB61850 lcbST       +orw   (params=lcbST;report=ST;);
}


CLASS WTurb_Model {       -- Wind Turbine Model
    WTurb_MX MX;
    WTurb_ST ST;
    WTurb_CF CF;
    WTurb_RP RP;
    WTurb_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}


CLASS WGen_Model {       -- Wind Generator Model
    WGen_MX MX;
    WGen_ST ST;
    WGen_CF CF;
    WGen_CO CO;
    WGen_RP RP;
    WGen_LG LG;
    LIST    MX = {^MX};
    LIST    ST = {^ST};
}


CLASS WGrid_Model {       -- Wind Grid Model
    WGrid_MX MX;
    WGrid_ST ST;
    WGrid_CF CF;
    WGrid_RP RP;
    WGrid_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}

CLASS WNace_Model {       -- Wind Nacelle Model
    WNace_MX MX;
    WNace_ST ST;
    WNace_CF CF;
    WNace_RP RP;
    WNace_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
```

```
CLASS WGear_Model {        -- Wind Gear Model
}   WGear_MX MX;
    WGear_ST ST;
    WGear_CF CF;
    WGear_RP RP;
    WGear_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}

CLASS WBrake_Model {        -- Wind Brake Model
    WBrake_MX MX;
    WBrake_ST ST;
    WBrake_CF CF;
    WBrake_RP RP;
    WBrake_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}

CLASS WRotor_Model {        -- Wind Rotor Model
    WRotor_MX MX;
    WRotor_ST ST;
    WRotor_CF CF;
    WRotor_RP RP;
    WRotor_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}

CLASS WYaw_Model {          -- Wind Yaw Model
    WYaw_MX MX;
    WYaw_ST ST;
    WYaw_CF CF;
    WYaw_RP RP;
    WYaw_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}

CLASS WEnv_Model {        -- Wind Environment Model
    WEnv_MX MX;
    WEnv_ST ST;
    WEnv_CF CF;
    WEnv_RP RP;
    WEnv_LG LG;
    LIST     MX = {^MX};
    LIST     ST = {^ST};
}


CLASS WMet_Model {        -- Meteorological station Model
    WMet_MX MX;
    WMet_CF CF;
    WMet_RP RP;
    LIST   MX = {^MX};
--    LIST   L1 = {^Meteor/WMet.MX};
```

```
--      LIST     L2  =  {MX.WindSpd54,  MX.WindSpd56,  MX.WindSpd75,  MX.WindSpd77,
MX.WindSpd96,    MX.WindSpd98,    MX.WindSpd120,    MX.WindSpd122,    MX.WindSpd145,
}
MX.WindDir40, MX.WindDir56, MX.WindDir77, MX.WindDir98, MX.WindDir122, MX.Temp1,
MX.Temp10,  MX.Temp38,  MX.Temp54,  MX.Temp75,  MX.Temp96,  MX.Temp120,  MX.Temp145,
MX.AirPres, MX.Rain};

-- Custom Reporting
CLASS WCust_Report {      -- WCust Report Control Block
    BasRCB CustRCB       +orw   (params=MX;report=CustList;);
    LIST CustList = {/DI.Name, Sigvards/WTurb.MX.WhG1, Sigvards/WGen.MX.GenSpeed,
Sigvards/WGrid.MX.Power};
}



SERVER
   DI_Model DI;
   JOURNAL WPP$Journal 32767;       -- Added by GS 04/07/01

   DEVICE Sigvards
      WTurb_Model WTurb;
      WGen_Model WGen;
      WGrid_Model WGrid;
      WNace_Model WNace;
      WGear_Model WGear;
      WBrake_Model WBrake;
      WRotor_Model WRotor;
      WYaw_Model WYaw;
      WEnv_Model WEnv;

      WCust_Report CustRP;
   END

   DEVICE Meteor
      WMet_Model WMet;
   END
END
```

E The use of Web technologies (HTTP, HTML, XML, and Javascript) in Tamarack's MMS*d* server

Specification of the wind power plant information model based on IEC 61850 (UCA™2.0) and The use of Web technologies (HTTP, HTML, XML, and Javascript) in Tamarack's MMS*d* server

by Karlheinz Schwarz, NettedAutomation GmbH

karlheinz.schwarz@nettedautomation.com

06 April 2002

Summary

This report presents the use of the following Internet technologies integrated into the Wind Power Plant server realized with the Tamarack MMS*d* software:

- Webserver
- Protocol: HTTP (Get service)
- Notation: HTML and XML
- Javascript

These additional possibilities are described and discussed. The modularity of the software allows for easy integration of the additional server, protocol, service, and notations.

NOTE – These specifications are not yet part of the coming standard IEC 61850. The use of XML as a means to visualize the ASN.1 coded MMS messages is under discussion in ISO TC 184 SC5 WG2.

These Internet technologies (HTTP, HTML, XML, and Javascript) implemented in the Tamarack MMS*d* server can be used by a **standard Internet browser** in the client to easily access and visualize the information defined and made visible in the server (current values of the process information and the Self-Description of the values – the Meta-Data). The access of the process data for real-time exchange is realized by the MMS services.

NOTE – This report does not discuss the use of XML in the IEC 61850-6 “Substation configuration language”.

The crucial issues specified in IEC 61850 are:

- Application information models
- Information exchange methods
- Mapping of models and exchange methods to application layer protocols
- Communication stacks.

The Internet technologies applied and described in this report have an impact on the third issue only! The core specification (information models and exchange methods) are totally independent of the protocols and syntax notations applied.

The use of the Internet technologies described in this report are a useful enhancement of the mapping to MMS. The mapping to HTTP, HTML, XML, and Javascript can be used for many non-time-critical applications.

Contents

1 Applying HTTP, HTML, and XML for exchanging information

1.1 General

IEC 61850-7-x defines information models and exchange services, and 61850-8-1 defines the mapping of the information models and exchange services to MMS (ISO 9506).

This paper shows the MMS mapping as well as the HTTP, HTML, and XML mapping and discusses the advantages of either mapping.

Additionally the use of Javascript will be demonstrated for the visualization of the information exchanged with HTTP/HTML. The examples use the IE 5.0.

This report discusses many general issues that are independent of the use of any of these technologies in the application area of the wind power plant communications.

1.2 What to be mapped?

HTTP provides a few services. Two of the services are of primary interest: GET and POST. To allow a simple HTTP server the GET service has been implemented into the Tamarack MMS server.

As shown in Figure 1 the mapping of the models and services of IEC 61850 100% of the defined models and services can be mapped to MMS. Just a small percentage can be mapped to HTTP “Get”, HTML and XML.

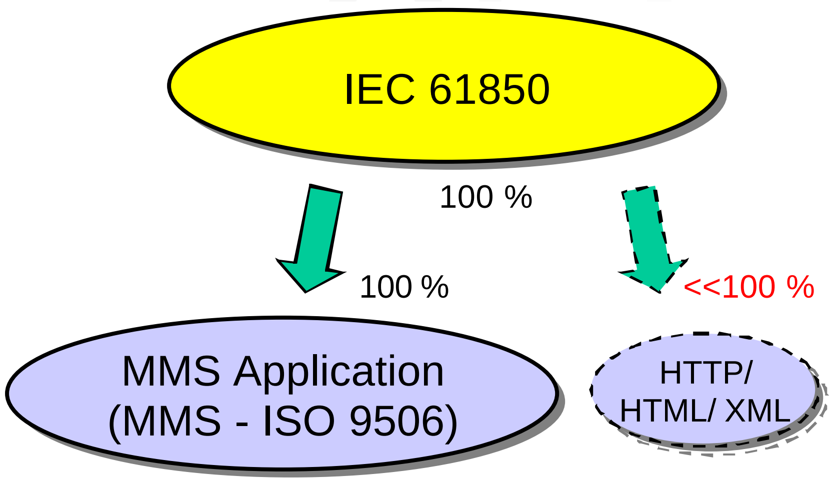

Figure 1 – Two mappings compared (in principle)

The mapping to HTTP/XML comprises just a small number of services to be mapped. The main services provided by HTTP are GET (Read, Polling) and POST (Write).

In this report just the “GET” service is discussed.

Figure 2 depicts the differences in principle.

	61850-8-1	HTTP/HTML*	HTTP/XML*	HTTP/XML**
Wind Power Plant Device information model	MMS Classes (Domain, Named Var., ...)	Text	Structured and name tagged text	Structured and name tagged text
Wind Power Plant Basic Types	MMS Named Var. and Types	Text	Structured and name tagged text	XML Schema
Wind Power Plant Services: Get, Set, Report, Log Query, ..	MMS Services: Read, Write, InfoReport, Read Journal, ..	HTTP Services: GET, -- -- --	HTTP Services: GET, -- -- --	HTTP Services: GET, -- -- --
Syntax schema (appl./services)	MMS Classes/ MMS Syntax	--/ GET Syntax	DTD**/ GET Syntax	XML Schema/ GET Syntax
Syntax notation (appl./services)	ASN.1/ ASN.1	Text/ Text	XML/ Text	XML/ Text
Encoding (appl./services)	ASN.1 BER/ ASN.1 BER	Text/ Table	Text/ Table	Text/ Table

* realized by Tamarack ** not yet implemented

Figure 2 – Comparison

- 61850-8-1 maps the complete information model and all information exchange services to MMS (and ASN.1, ASN.1/BER).
- XML is used as the syntax notation (like ASN.1) for the syntax of the application data.
- XML Schema can be used in addition for basic types and application syntax specification.
- XML does not specify any Service.

  NOTE – XML Schema can be used as a notation to specify the MMS services, replacing ASN.1 and ASN.1 BER. In this case, we just replace one syntax notation (ASN.1, ASN.1 BER) by another notation (XML Schema, XML document). The semantic (the different services and the information carried by the services) is still the same. The syntax is independent of the services and information content.

- XML could not replace the MMS mapping (information and services are mapped).
- XML could be used for simple retrieval (GET) of the information model and data values (XML document).
- HTTP GET and XML document support can easily be integrated into the Tamarack MMS*d* Server.

Figure 3 describes what can be mapped to MMS and to HTML, XML, and HTTP.

A comprehensive discussion of the use of ASN.1 and ASN.1 BER can be found under: www.nettedautomation.com/standardization/ISO/TC184/SC5/WG2/mms_intro/intro6.html

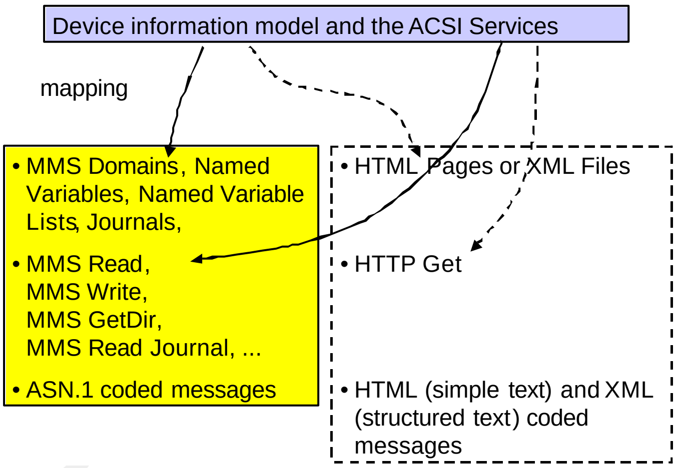

Figure 3 – Different mappings

The categories of the IEC 61850 services are listed in Figure 4.

Operational Services

- Get Values of Data (Data Set) Objects
- Set Values of Data (Data Set) Objects,
- Control,
- Substitute,
- Report, Log and Query Data Values, ...

Self-Description

- Get Directory of Objects (Data of Logical Node, ...),
- Get Definition of Objects (Type, Unit, Scale, ...),
- Get Definition of Communication Objects, ...

Remote Configuration

- Set Communication Objects (En/Dis Report (Log), ...),
- Define Data Sets, …

Figure 4 – IEC 61850 exchange services

These services have to mapped in a specific communication service mapping (see Figure 5). The mapping shown is specific to the Tamarack MMS*d* implementation. All "Set" services have not been implemented by September 2001.

Mappings	
	Operational Services
HTTP, MMS	• Get Values of Data (Data Set) Objects
MMS	• Set Values of Data (Data Set) Objects,
MMS	• Control,
MMS	• Substitute,
MMS	• Report, Log and Query Data Values, ...
	Self-Description
HTTP, MMS	• Get Directory of Objects (Data of Logical Node, ...),
HTTP, MMS	• Get Definition of Objects (Type, Unit, Scale, ...),
HTTP, MMS	• Get Definition of Communication Objects, ...
	Remote Configuration
MMS	• Set Communication Objects (En/Dis Report (Log), ...),
MMS	• Define Data Sets, …

Figure 5 – Service mappings

2 Examples for HTTP, HTML, XML, and Javascript

2.1 Topology

The topology of the devices for the example are shown in Figure 6. The server communicates with the application via a DLL interface. The server provides the IEC 61850 (MMS mapping as well as the web server (HTTP, HTML, XML, Javascript).

Two clients (running on the same PC) communicate via TCP/IP (Winsocket) with the server. On client is the IEC 61850 (MMS) client and the other is a standard Internetbrowser (IE 5.5).

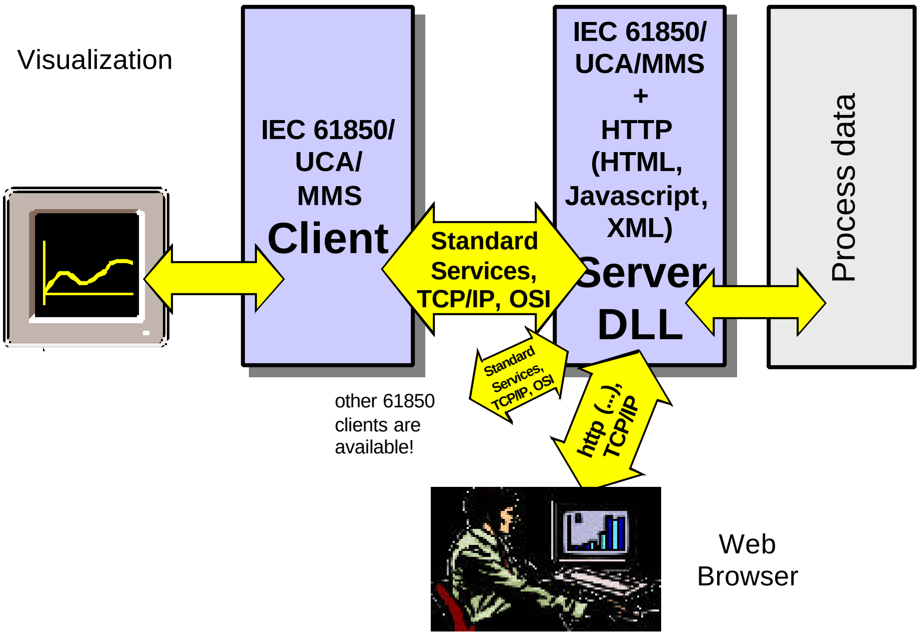

Figure 6 – Topology of server and clients

Details about the DLLs can be found under the following URL:

www.nettedautomation.com/solutions/uca/products/dll/demo/

2.2 The architecture of the server device

The server device provides two servers inside the DLL: The IEC 61850 (MMS) server and the Webserver.

NOTE – The 61850 (MMS) server is described in the report “ Specification of the wind power plant information model based on IEC 61850 (UCA™2.0) and Description of the implementation with Tamarack's MMS*d*”

The webserver supports the “GET” service. The pages requested by a client are“*.html”, “*.xml”, and “*js” files. The HTML and XML files are created immediately after the web-server receives the “GET” request.

In case of “GET URL.html” (“GET URL.xml”) the server provides the data referenced by the URL and returns the values HTML (XML) coded. Examples are explained in the following paragraph.

The Javascript is used for the visualization of the HTML coded responses. The Javascript is retrieved at the beginning (first “GET URL.html”). The files “lib1.js” and “lib2.js” contain the Javascript for the visualization (two files because the webserver can handle maximum file size <10 KB).

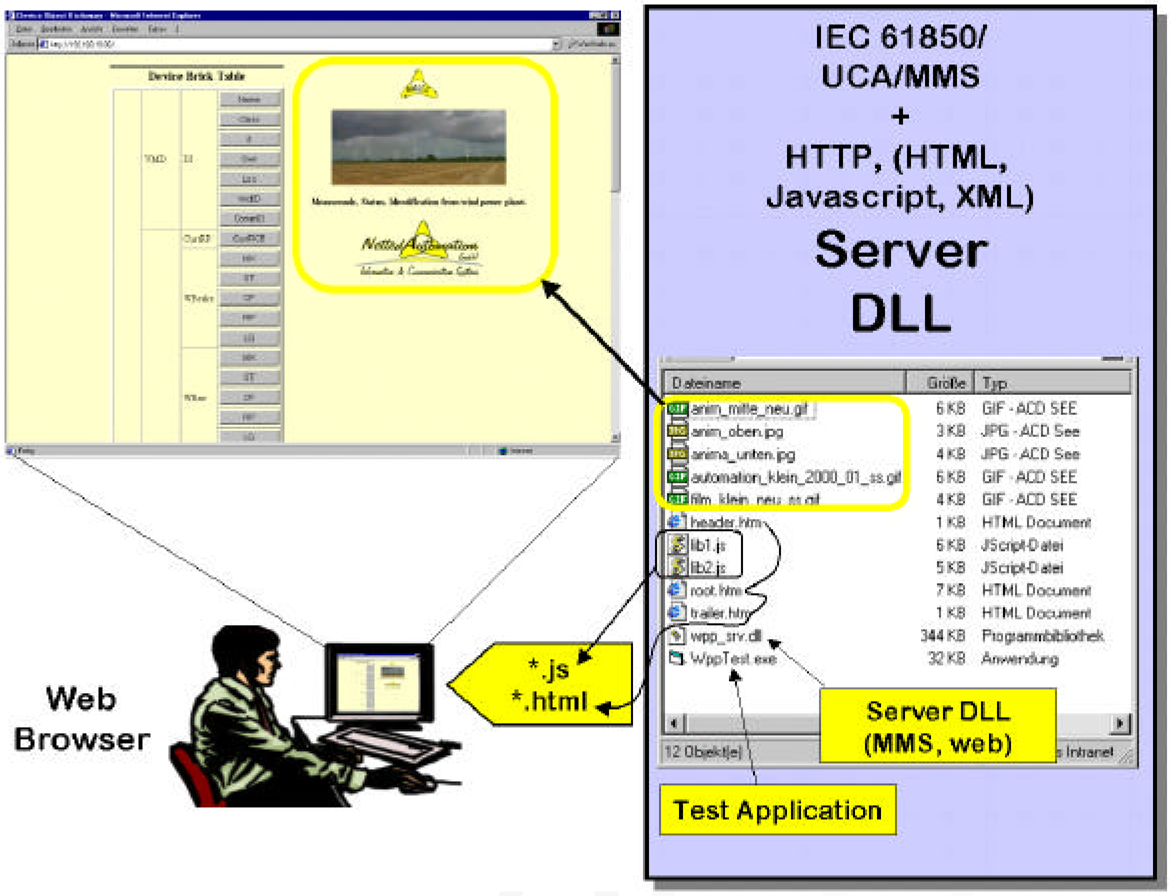

Figure 7 – Topology of server and clients

The “header.htm”, “root.htm”, and “trailer.htm” are needed to assemble the html response files containing the values.

The “wpp_srv.dll” (344 KB) is the complete server DLL containing the 61850 (MMS) server, information model (including the self-description), information exchange methods (e.g., reporting, logging, ...), and the webserver.

The “WppTest.exe” is the test application producing data and receiving control commands.

2.3 The web browser client

The web browser client is depicted in Figure 8.

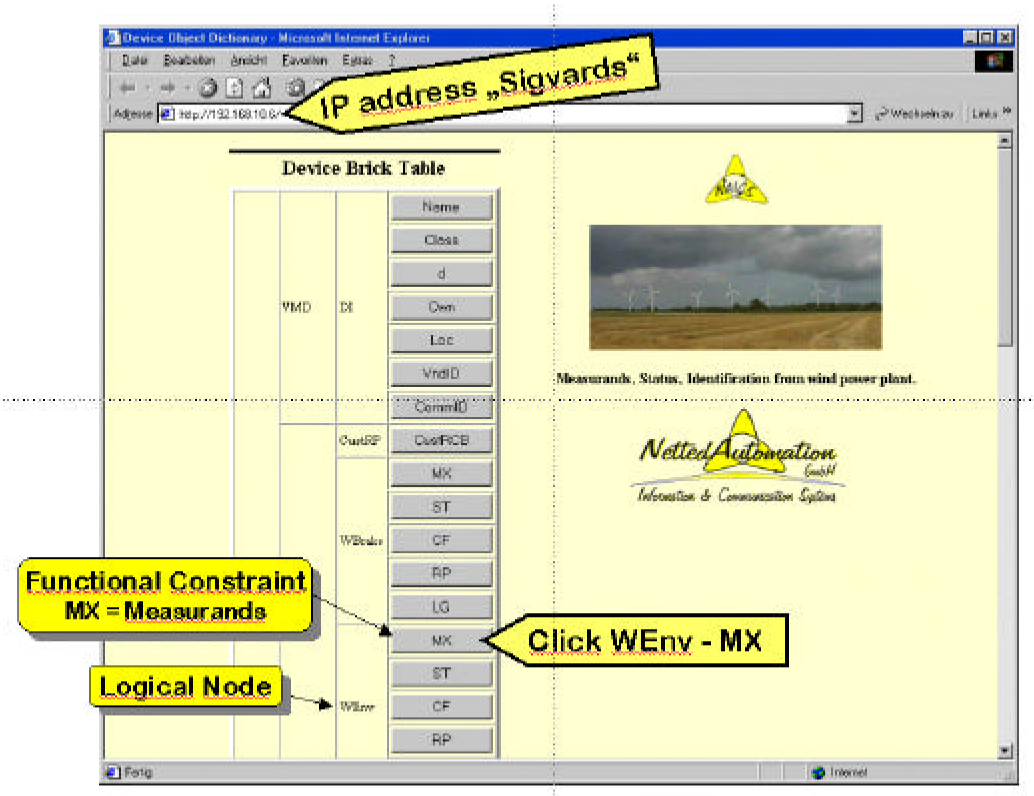

Figure 8 – Web page for wind power plant „Sigvards“

The browser provides the capability to request the servers information model (Wind Power Plant model). The “click” at the “MX” button in Figure 8 opens the window in Figure 9.

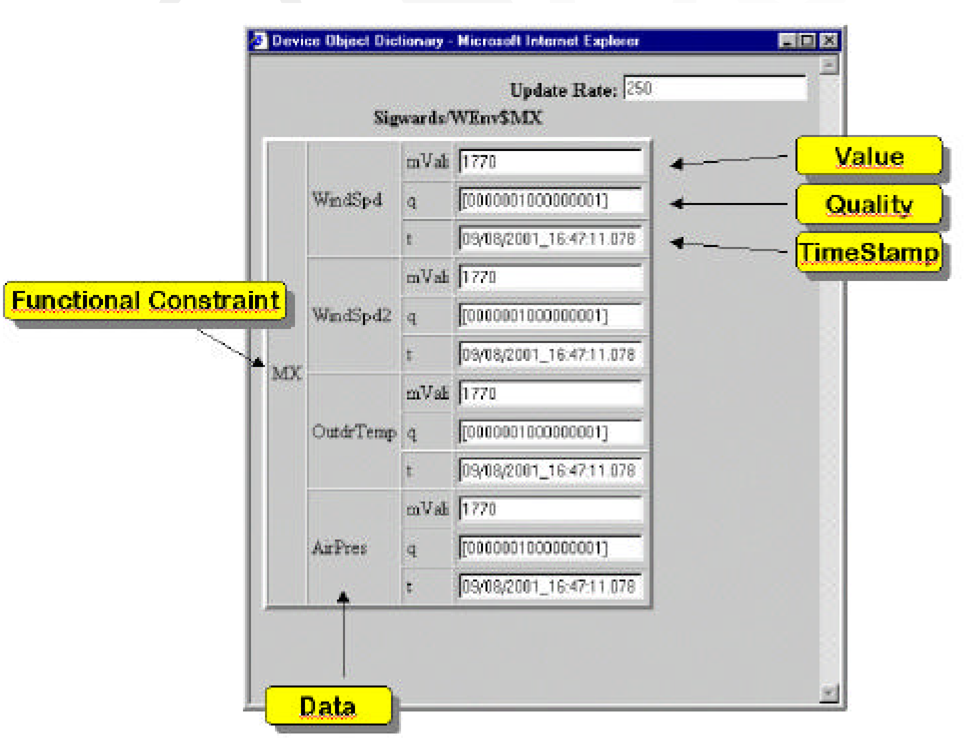

Figure 9 – Sigvards/WEnv$MX

The reference to the measurements (MX) is “Sigvards/WEnv$MX”. That means that all measurements of the Wind-Environment logical node “WEnv” are shown. Each data object (e.g., WindSpd) has a value (mVali), a quality code (q), and a timestamp (t).

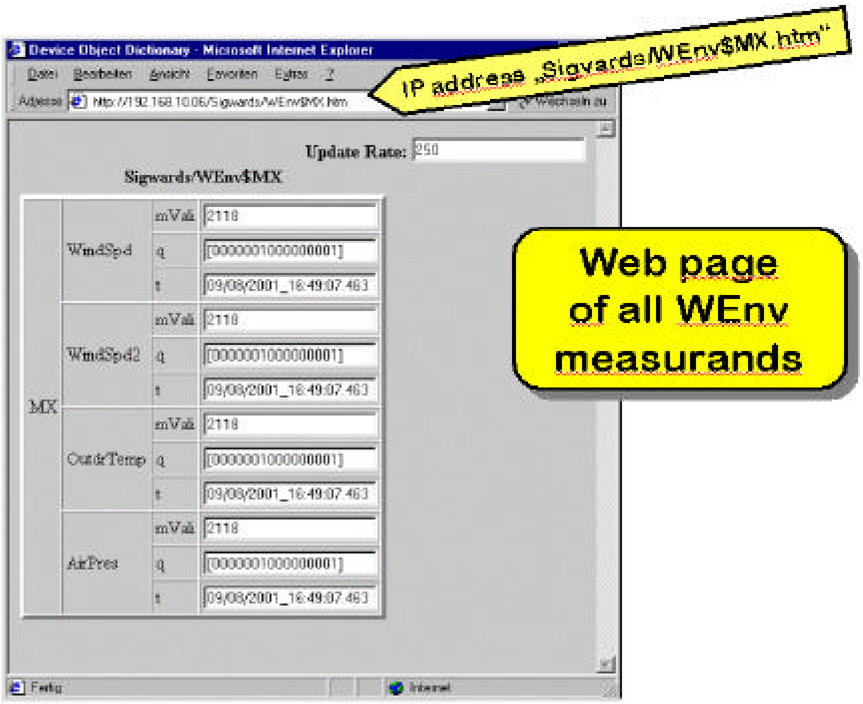

Figure 10 – Sigvards/WEnv$MX.htm

The MX values can also be accessed by a "GET Sigvards/WEnv$MX.htm" as shown in Figure 10 or by a "GET Sigwards/WEnv$MX.xml" as depicted in Figure 11. The XML structure can be seen – all branches of the tree are open.

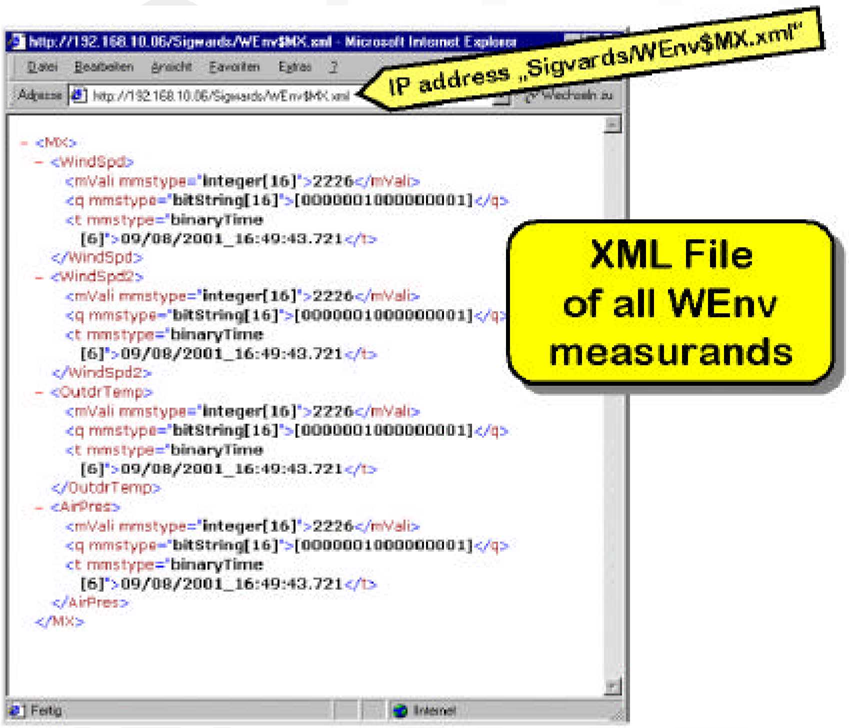

Figure 11 – GET Sigvards/WEnv$MX.xml

The standardized names of the logical device ("Sigvards"), the logical node ("WEnv"), and the functional constraint ("MX") are directly used as tags in the elements (element names) of the XML file. Two other examples with additional explanations are depicted in Figure 12 and Figure 13.

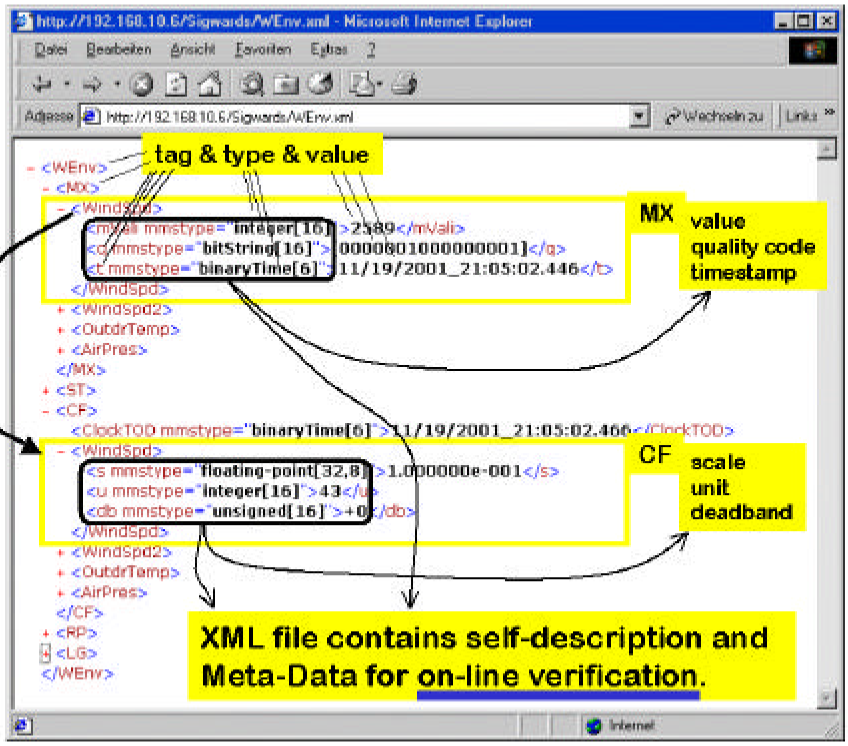

Figure 12 – Reporting and logging

The element tag names, the type, and the values are all contained in the XML file received from the server. The very same information is carried in the MMS services (in the Report messages).

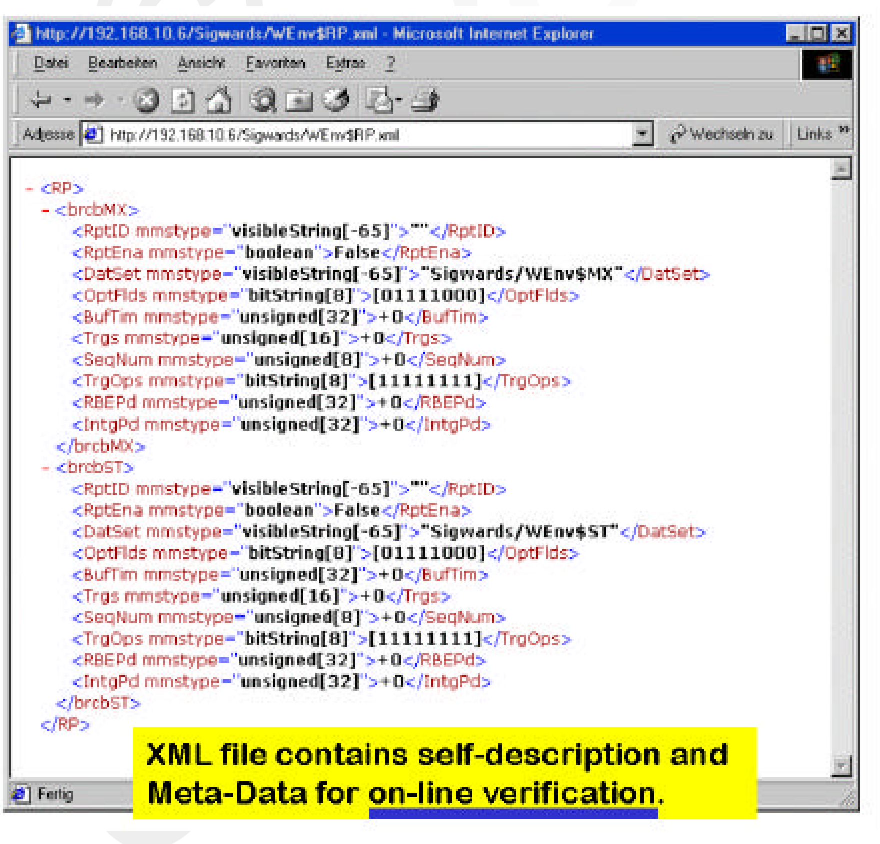

Figure 13 – Reporting and logging

The Meta-Data that are carried in the messages (XML or MMS message) can be used for on-line verification of the received information.

The information model of the wind power plant can easily be mapped to a XML file!

*The following story may help to understand the benefit of Meta-Data. At the time of writing of this report the editor got a phone call from his bank. The bank informed him that a customer has wired an amount of D-Mark X*thousand . This seemed to be very high. What had happened? The invoice was in D-Mark. But the customer had wired X*thousand EURO which is about twice as much as expected (1 EURO is about 2 D-Mark). The Meta-Data "Currency" in the invoice and in the customers bookkeeping system was different! An employee of the customer had not taken the difference into account. So it was a real benefit for the customer that we figured out very soon that there was a mismatch in the values of the Meta-Data "Currency". The "on-line" check was possible because the invoice and the bookkeeping system showed the "Currency" Meta-Data.*

The functional constraint "CF" (Configuration) provides for each data object the configuration attributes (scale, unit, and deadband) as shown in Figure 14.

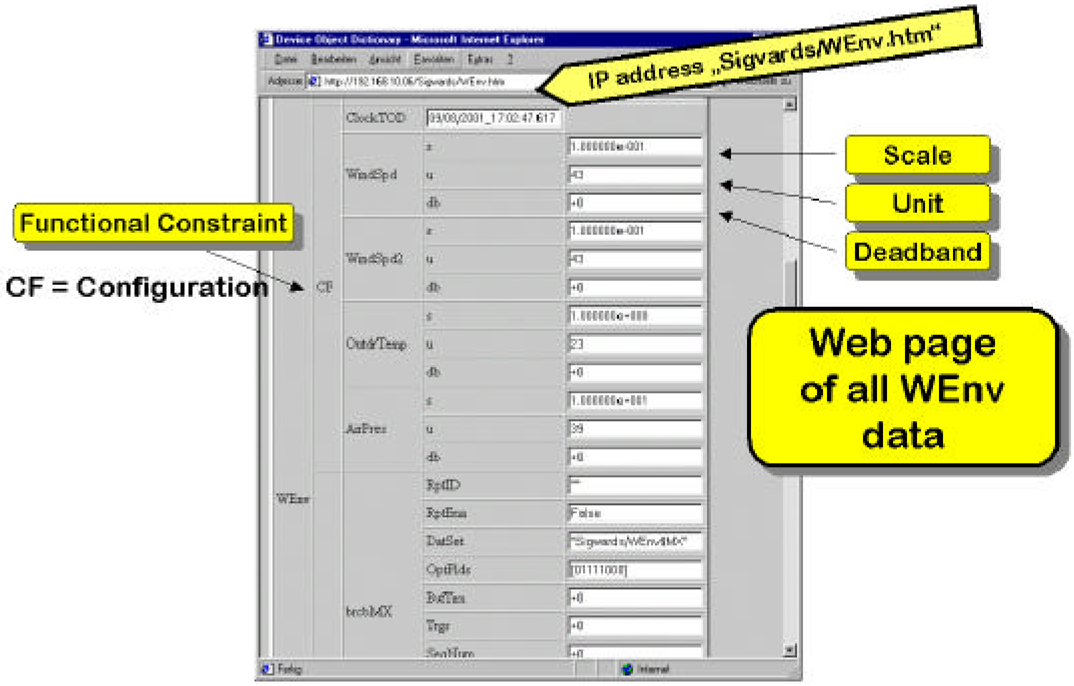

Figure 14 – Configuration attributes

The functional constraint "RP" (report control) as exposed in Figure 15 represents two control blocks: basic report control block "brcbMX" and "brcbST".

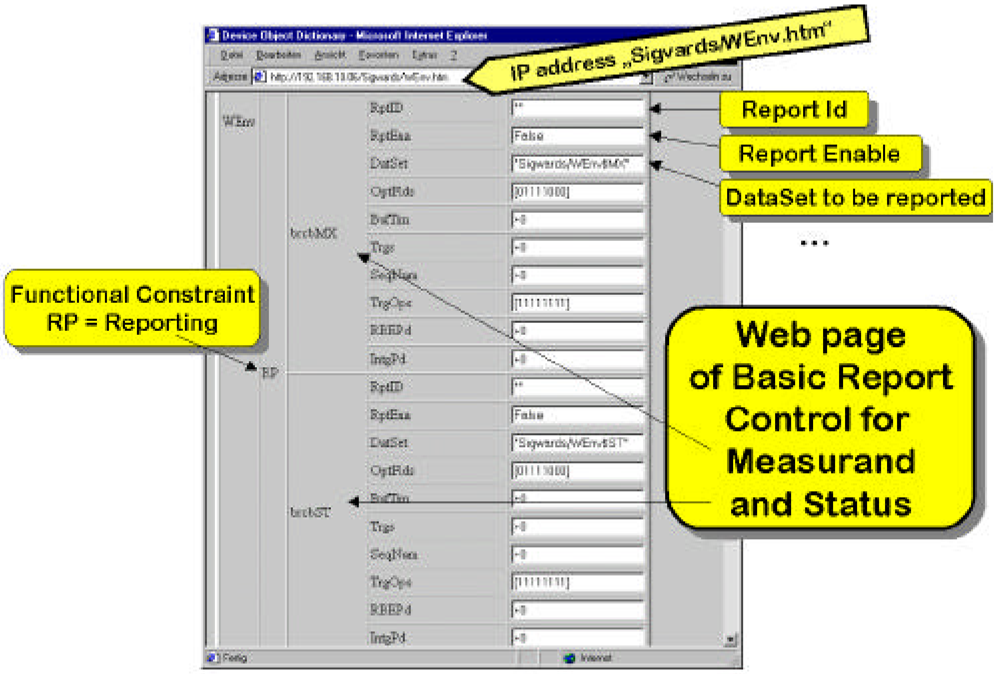

Figure 15 – Report control blocks

The report control blocks represent the current attribute values that control the reporting process. The same values that are reported “Sigvards/WEnv$MX” and “Sigvards/WEnv$ST” can be logged. The attribute values of the log control block are shown in Figure 16.

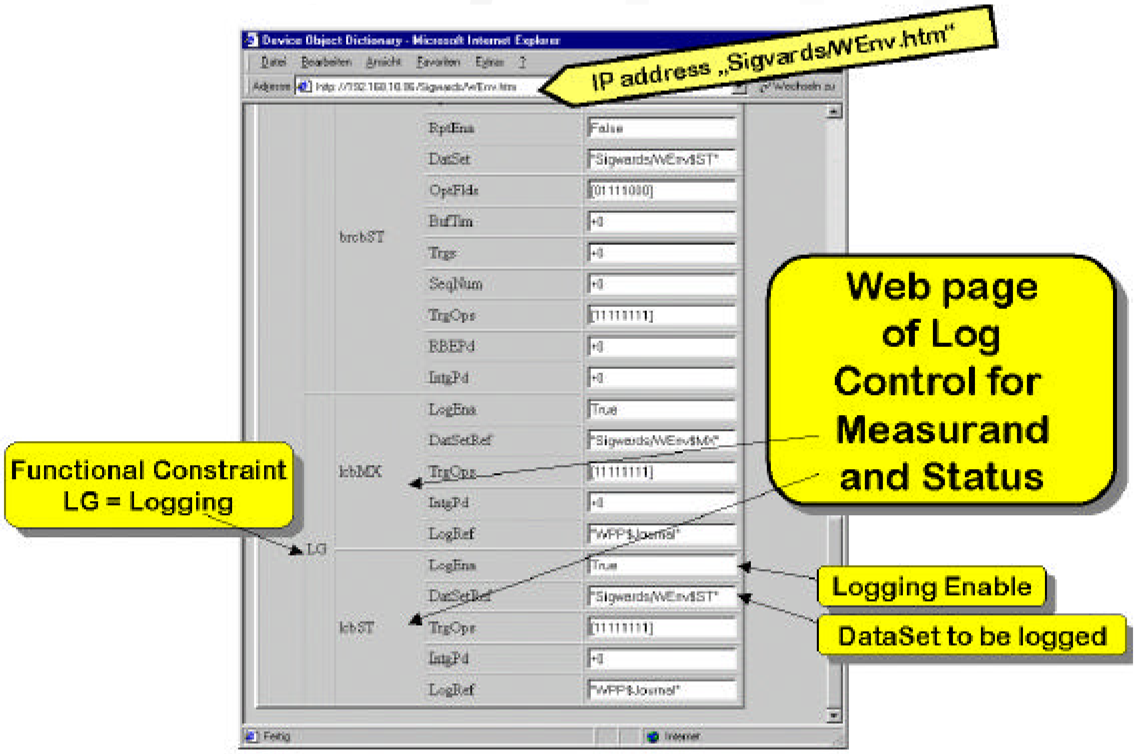

Figure 16 – Log control block

2.4 The Tamarack test client

The Tamarack test client is depicted in Figure 17. The client provides an interface to the 61850 (MMS) services and the configuration. The test client provides full access to all services etc. The test client is shown here to see the difference between the Internet Browser and a test client with comprehensive services.

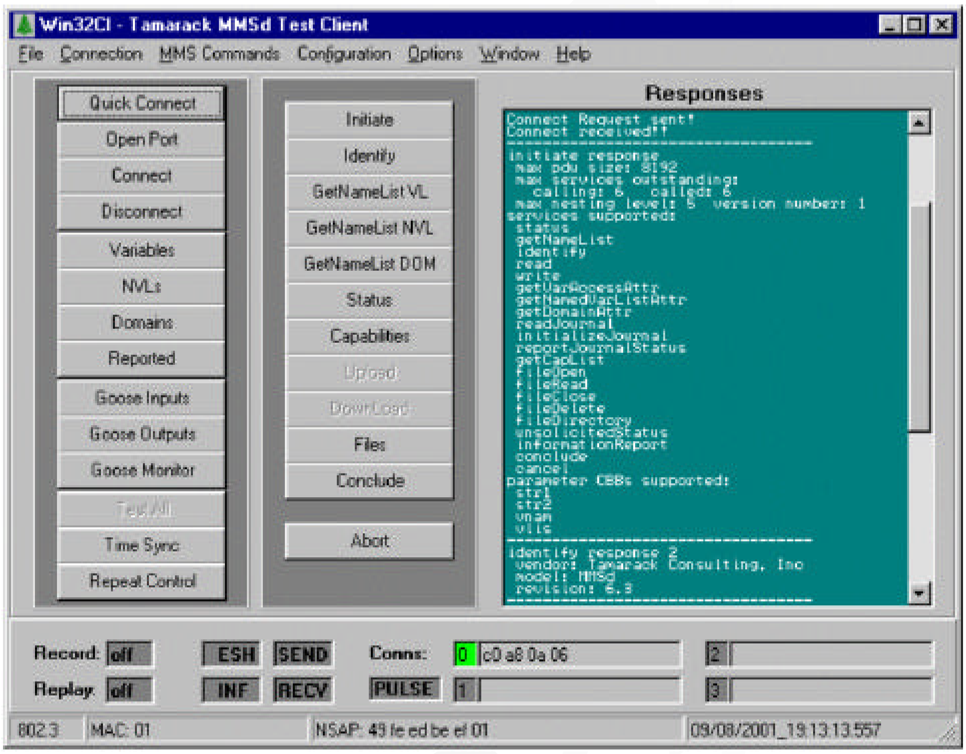

Figure 17 – Test client

Figure 18 shows the reporting window.

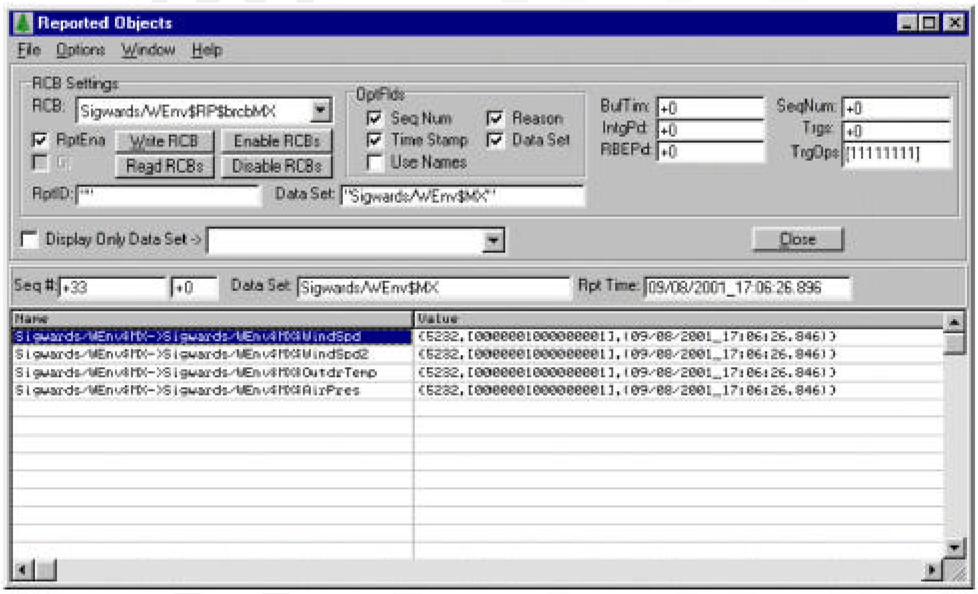

Figure 18 – Test Server (Reporting)

Reporting with a standard Internet Browser is possible in principle but would require a lot of high efficient software (not provided by the browser). The message rate for receiving reports may be several hundred per second.

The logging test client window is depicted in Figure 19.

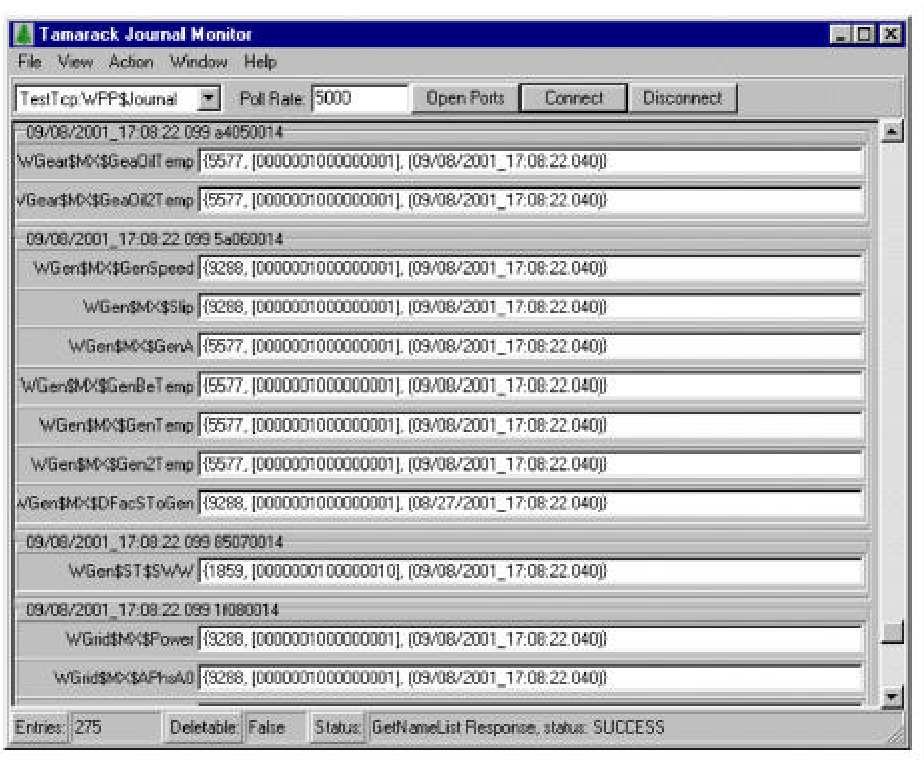

Figure 19 – Test client (Logging)

2.5 XML PRO tool

The XML tool PRO has been used to visualize the XML files that contain the values received from the server. Figure 20 through Figure 22 show the content of the XML files as trees.

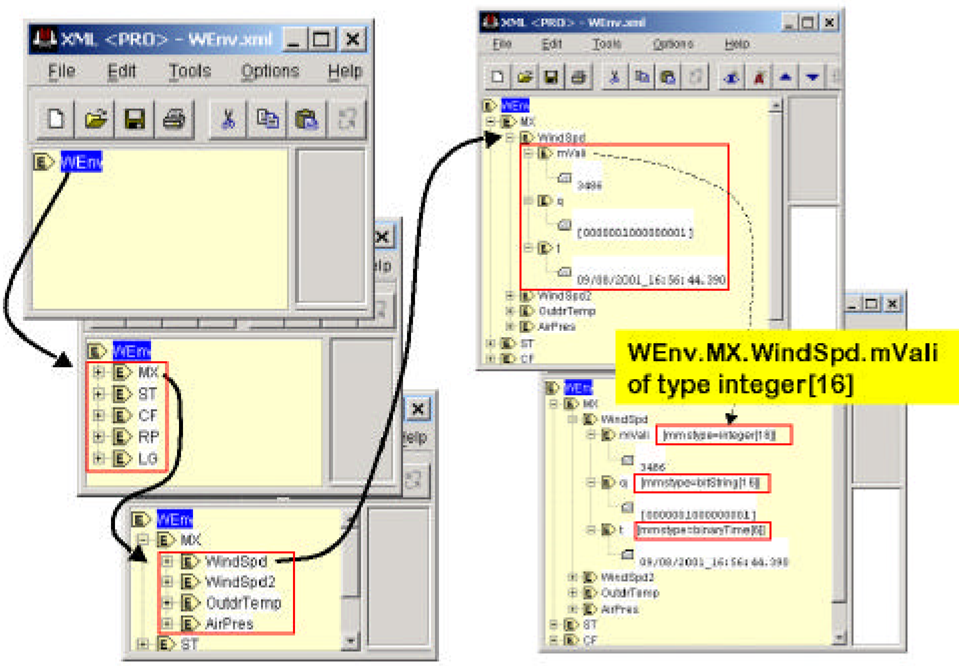

Figure 20 – Browsing the „WEnv Tree“

The information of the tree shown here is carried in the XML file as well as in the MMS messages. The XML tree could be visualized with a “standard” tool while one needs a special ASN.1 tool to make the show the tree carried with ASN.1 BER.

NOTE – It seems that ASN.1 tools will be available for free soon because ASN.1 is used in more and more applications. The wireless phone market requires more efficient encoding schemas than plain XML ASCII text. ASN.1 is a important option.

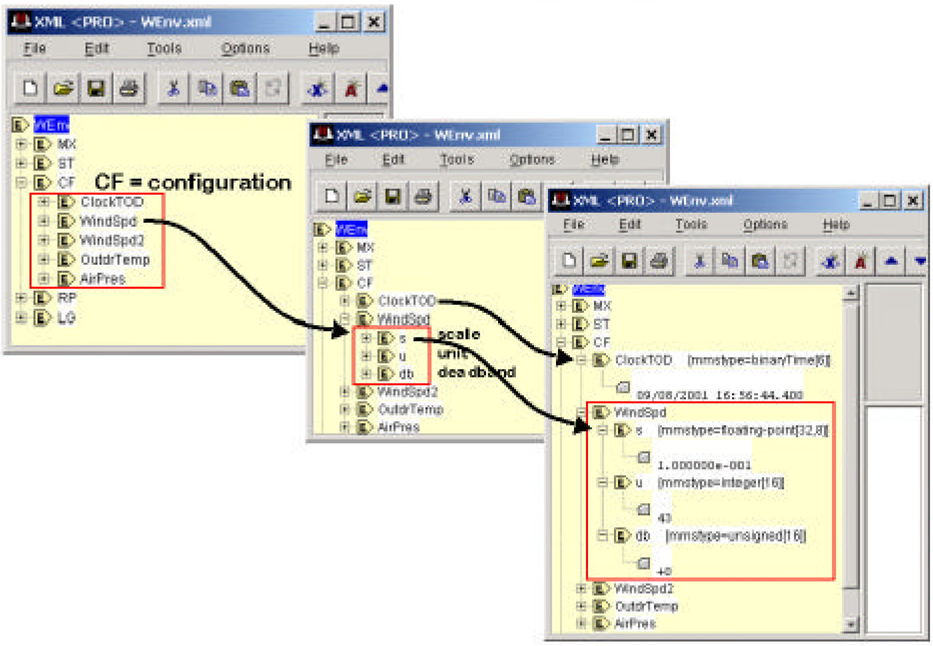

Figure 21 – Configuration

The complete information model including all control objects (report, logging, ...) can be made visible with a XML tool.

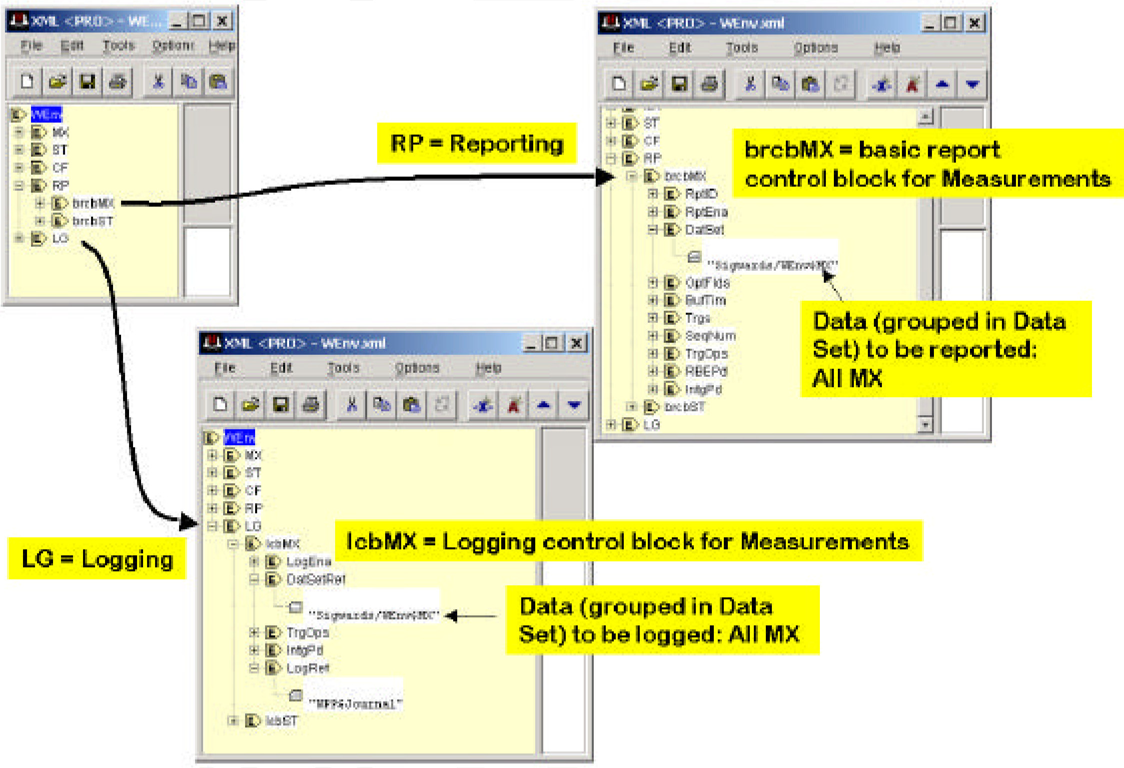

Figure 22 – Reporting and logging

ELFORSK

SVENSKA ELFÖRETAGENS FORSKNINGS- OCH UTVECKLINGS – ELFORSK – AB

Elforsk AB, 101 53 Stockholm. Besöksadress: Olof Palmes Gata 31

Telefon: 08-677 25 30. Telefax 08-677 25 35

www.elforsk.se