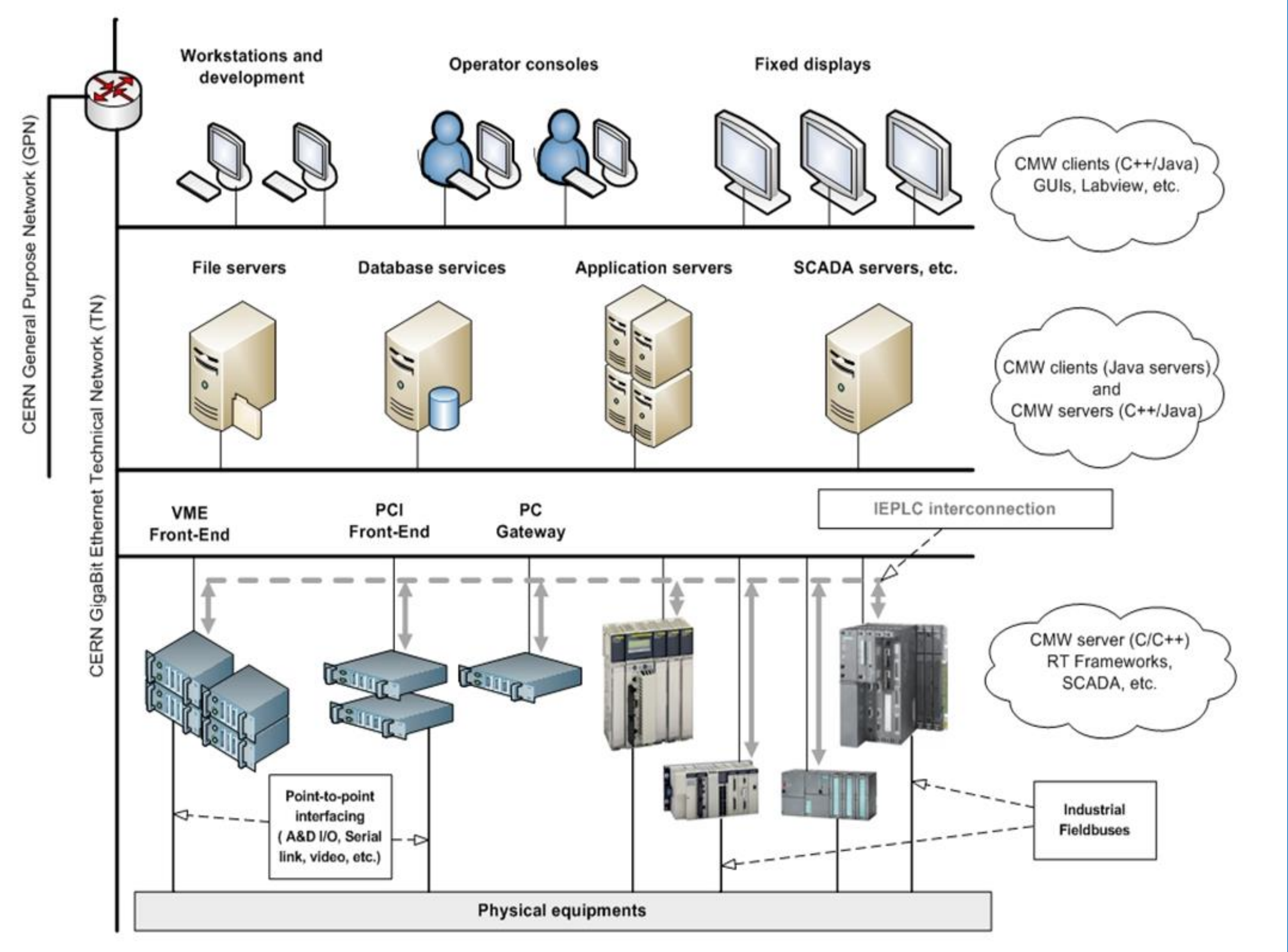
IEPLC framework

Automated communication in a heterogeneous control system environment

F.Locci, S.Magnoni - CERN Beam department/ Control group, Geneva, Switzerland

Purpose of the IEPLC framework

The Programmable Logic Controller s (PLCs) and other micro-controller families are essential components of the CERN control system. Together with their weaknesses ant their strengths they typically present custom communication protocols and it is therefore difficult to federate them into the control system using a single communication strategy.
The purpose of IEPLC is **to mitigate the communication issues** by providing communication interfaces in a **hardware independent manner**. In addition it **automatically generates all the resources** needed on client and controller side to implement a common and **generic Ethernet communication**.
The IEPLC framework is composed of a set of tools, scripts and a C++ library. The configuration tool allows the definition of the data to be exchanged and their instantiation on different controllers within the control system. The scripts generate the resources necessary to the final communication while the library eventually allows the application on the client side to send and receive data to/from the different controllers. Finally, **a diagnostic tool** is proposed to validate the whole configuration without implementing a single line of code.


CERN General Purpose Network (GPN)
CERN GigaBit Ethernet Technical Network (TN)
Workstations and development
Operator consoles
Fixed displays
CMW clients (C++/Java) GUIs, Labview, etc.
File servers
Database services
Application servers
SCADA servers, etc.
CMW clients (Java servers) and CMW servers (C++/Java)
VME Front-End
PCI Front-End
PC Gateway
IEPLC interconnection
CMW server (C/C++) RT Frameworks, SCADA, etc.
Point-to-point interfacing (A&D I/O, Serial link, video, etc.)
Industrial Fieldbuses
Physical equipments

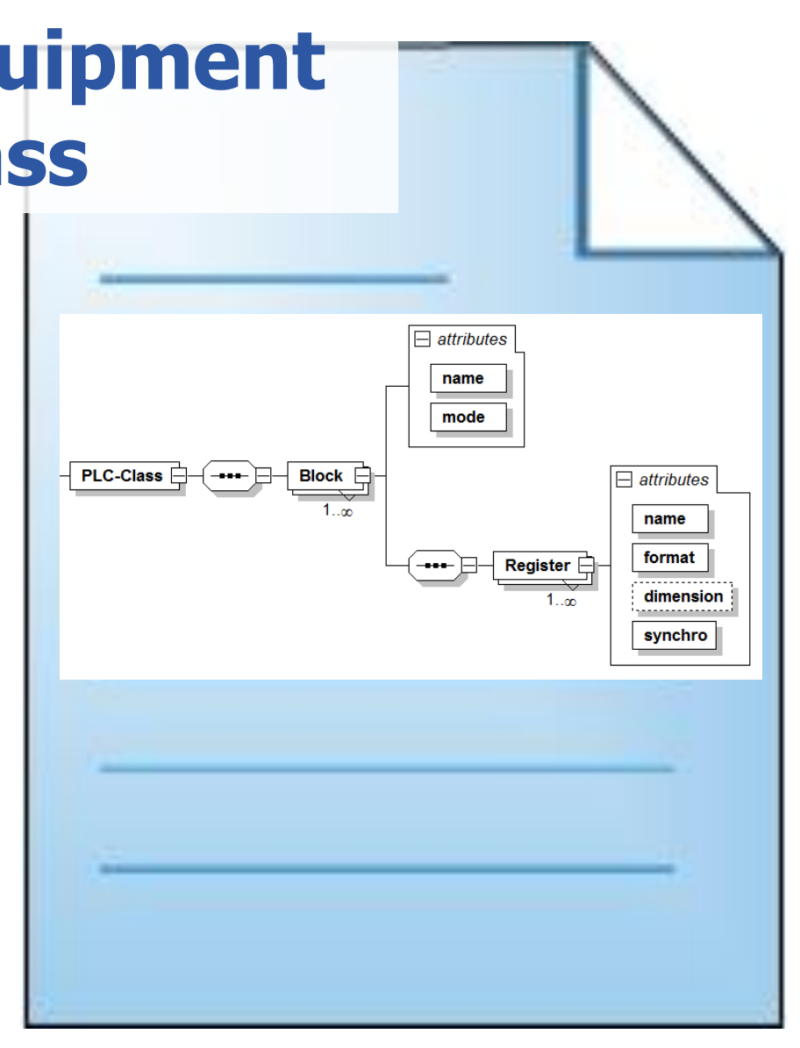
Context: The Three-tiers control system 1

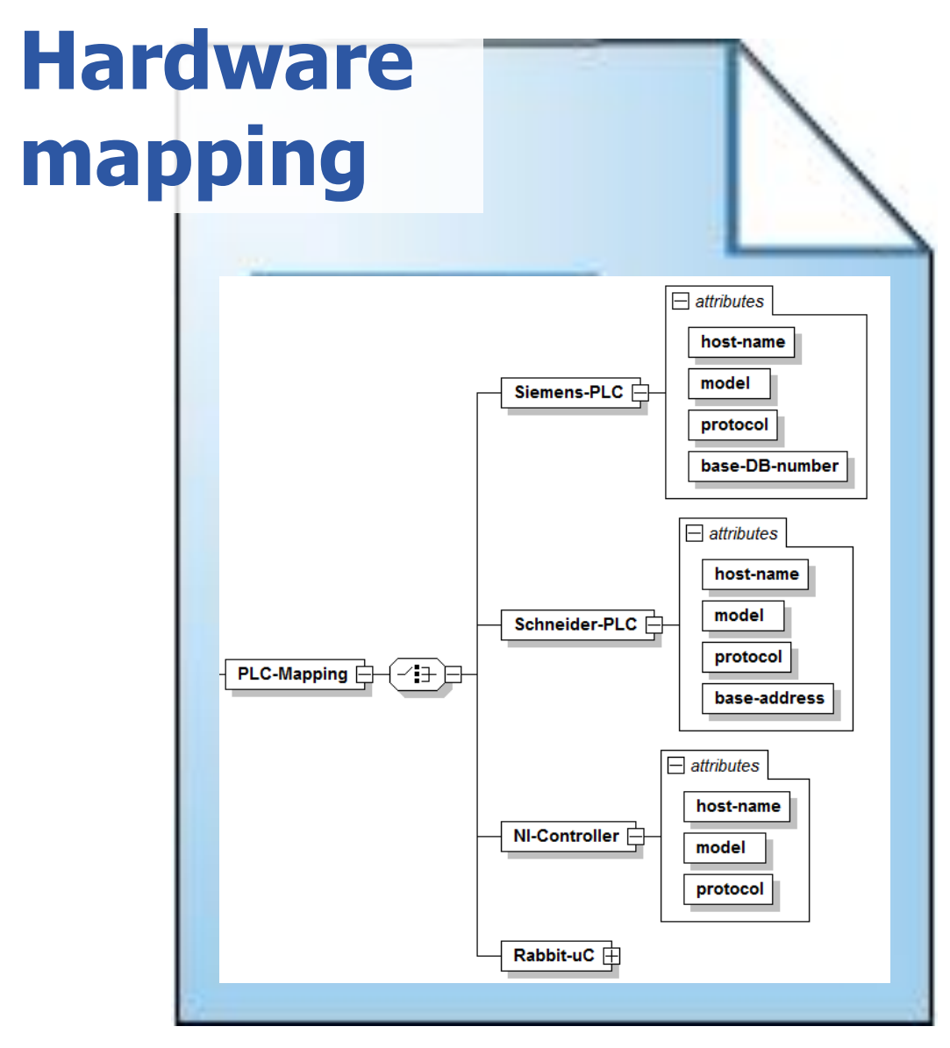
The control system splits over three different responsibility tiers which communicate among them via TCP-IP protocol over Gigabit Ethernet connection:

- the **presentation tier**, the **middle tier** and the **resource tier**

Both equipment controllers and FECs belong to the resource tier. In the IEPLC model, FEC acts as bridges between the presentation tier and the physical layer by standardising the integration of industrial components within CERN's global infrastructure.

Keywords:
Front-end computer (FEC), VME&PCI platforms, 64bits RT-Linux, FESA framework





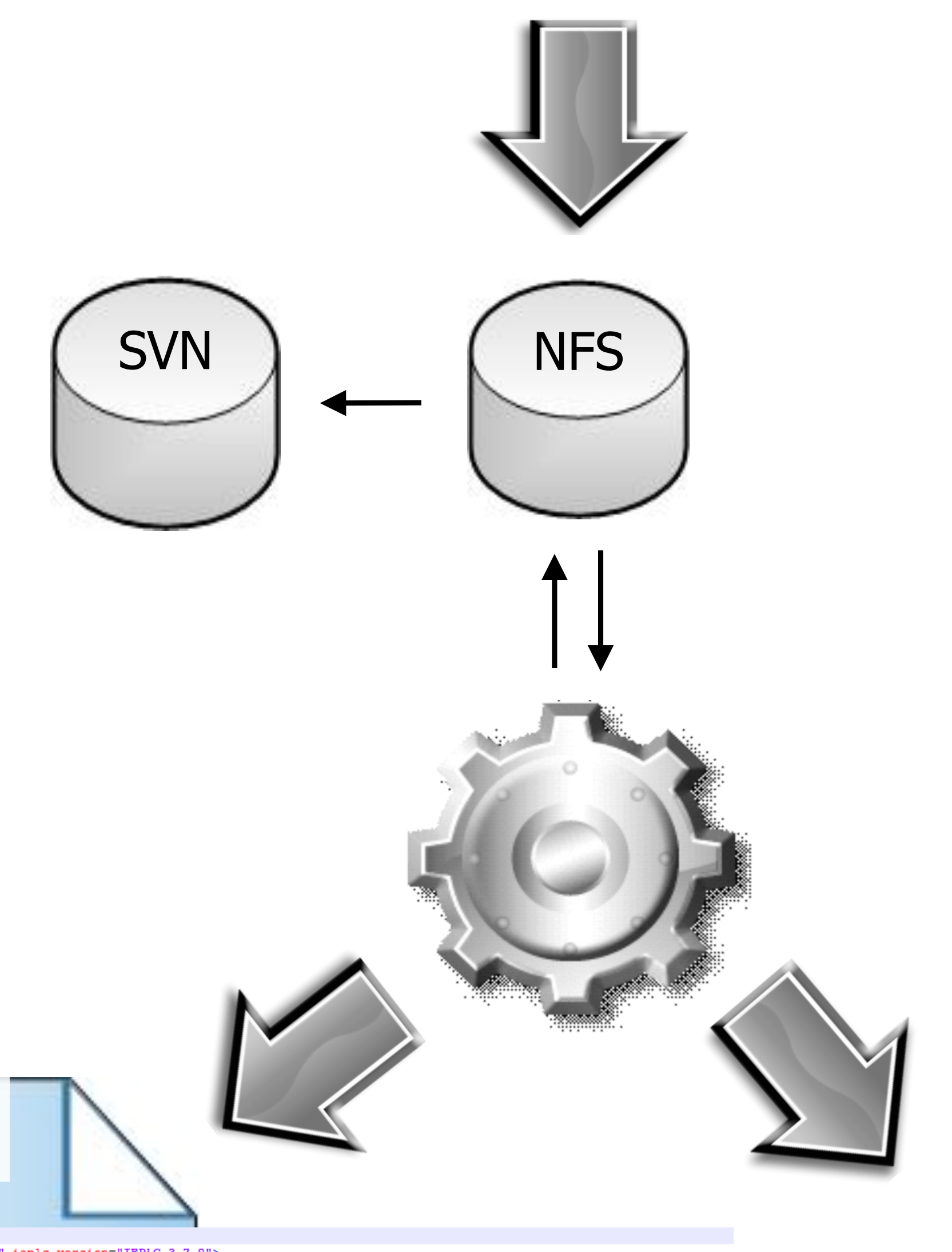
The device data model 2

A dedicated configuration tool is used to create the two documents required for the generation process:

- **The Class document**: structure of the exchanged data
- **The Mapping document**: binding between the physical device and the class instances

A centralised file system allows users to cooperate reusing and editing documents created by other people.

Keywords:
Java XML editor, XML schema, Constraint language (CliXML)



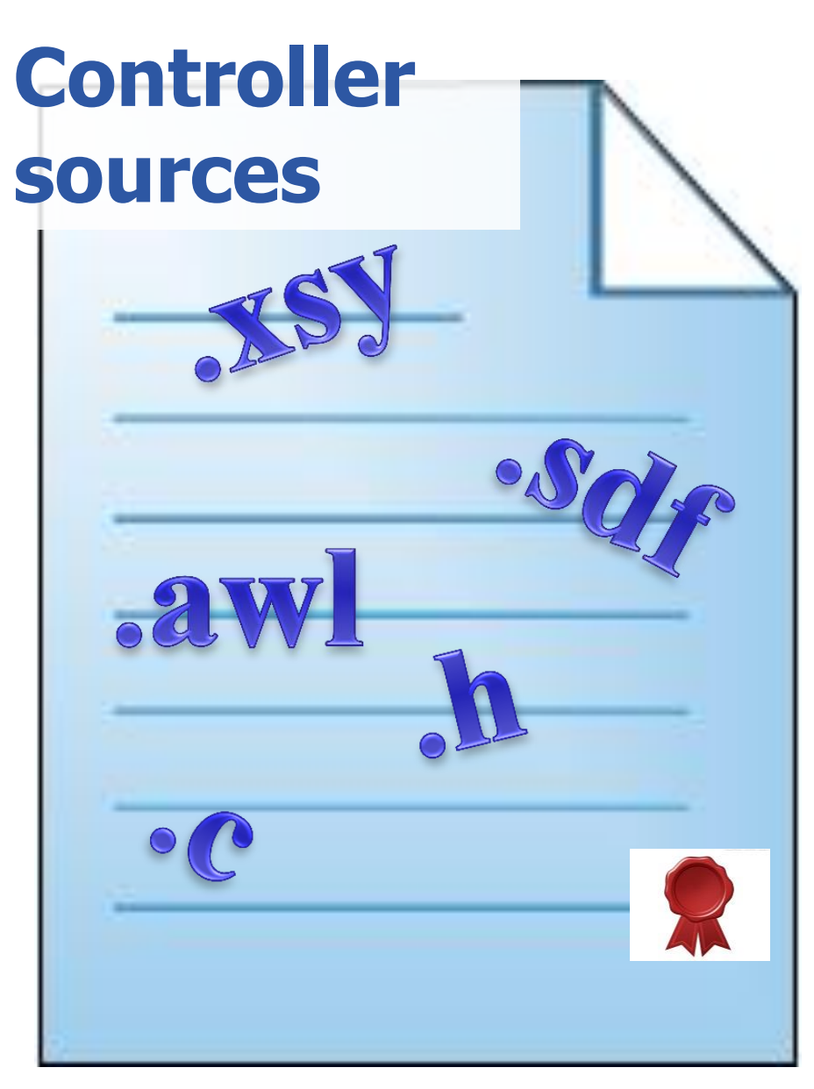
The generation process 3

The Generation, fully automated by a set of python scripts, generates two kind of resources:

- **The parameters file**: an XML document used to provide the library with all information required to communicate with the predefined controller (hostname, protocol, mapping, …)
- **The controller sources**: the code to be uploaded on the controller itself

A checksum mechanism ensures the consistency of each IEPLC configuration.

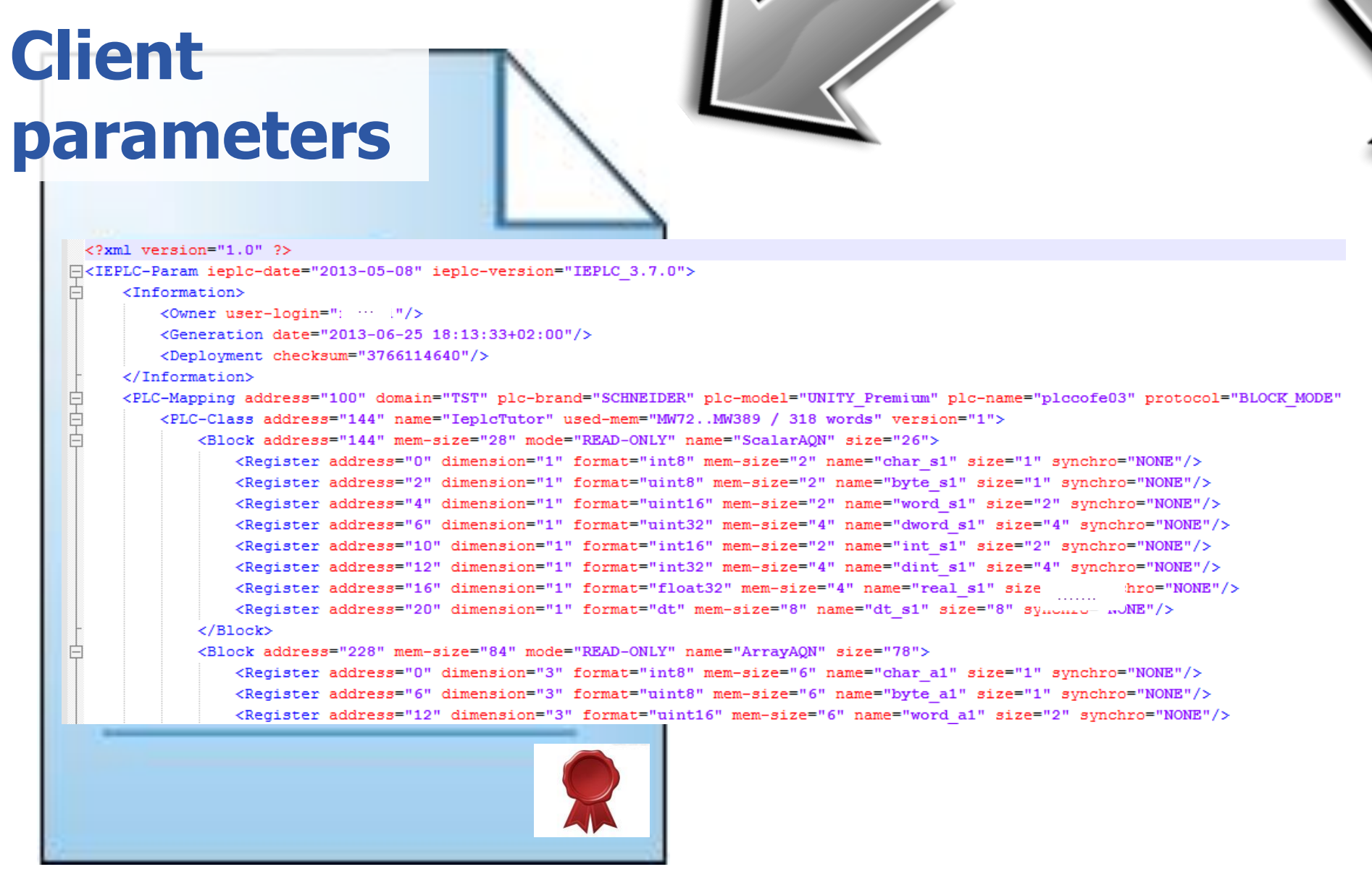
Keywords:
Python, CRC32, NFS, SVN



Controller upload 4

IEPLC provides a non-intrusive solution by generating only data structures and no process code, not even for the communication tasks. Different memory mapping can be adopted to optimize the network traffic. The user relies on the manufacturers software and tools to upload the controllers with the generated code.

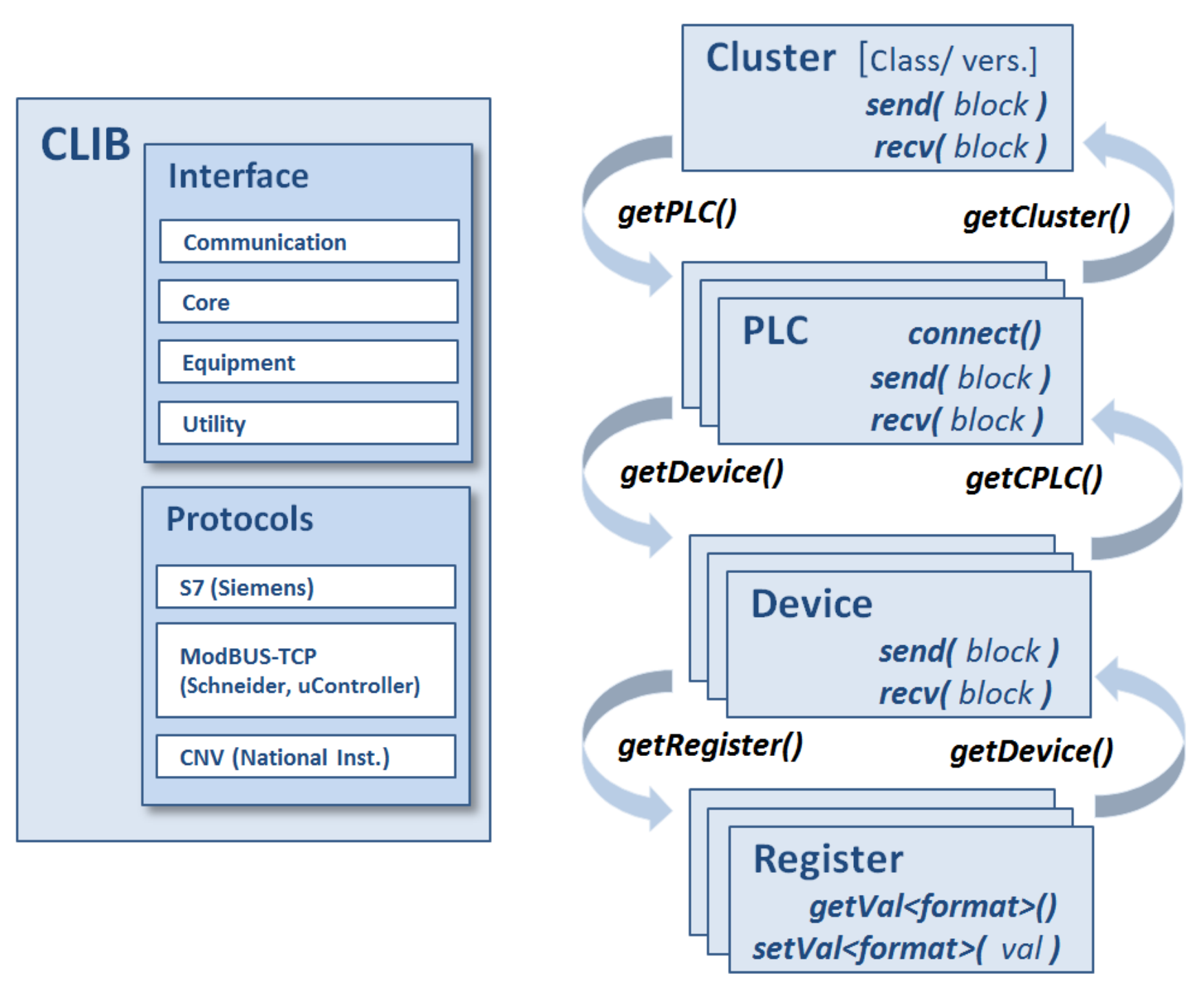
Keywords:
Step-7 STL/SCL, Unity-Pro XSY, C-code
NI CVI Network Variable, Labview



```
<?xml version="1.0" ?>
<IEPLC-Param ieplc-date="2013-05-08" ieplc-version="IEPLC_3.7.0">
    <Information>
        <Owner user-login=": ... ."/>
        <Generation date="2013-06-25 18:13:33+02:00"/>
        <Deployment checksum="3766114640"/>
    </Information>
    <PLC-Mapping address="100" domain="TST" plc-brand="SCHNEIDER" plc-model="UNITY_Premium" plc-name="plccofe03" protocol="BLOCK_MODE"
        <PLC-Class address="144" name="IeplcTutor" used-mem="MW72..MW389 / 318 words" version="1">
            <Block address="144" mem-size="28" mode="READ-ONLY" name="ScalarAQN" size="26">
                <Register address="0" dimension="1" format="int8" mem-size="2" name="char_s1" size="1" synchro="NONE"/>
                <Register address="2" dimension="1" format="uint8" mem-size="2" name="byte_s1" size="1" synchro="NONE"/>
                <Register address="4" dimension="1" format="uint16" mem-size="2" name="word_s1" size="2" synchro="NONE"/>
                <Register address="6" dimension="1" format="uint32" mem-size="4" name="dword_s1" size="4" synchro="NONE"/>
                <Register address="10" dimension="1" format="int16" mem-size="2" name="int_s1" size="2" synchro="NONE"/>
                <Register address="12" dimension="1" format="int32" mem-size="4" name="dint_s1" size="4" synchro="NONE"/>
                <Register address="16" dimension="1" format="float32" mem-size="4" name="real_s1" size ....... hro="NONE"/>
                <Register address="20" dimension="1" format="dt" mem-size="8" name="dt_s1" size="8" sy[illegible]NONE"/>
            </Block>
            <Block address="228" mem-size="84" mode="READ-ONLY" name="ArrayAQN" size="78">
                <Register address="0" dimension="3" format="int8" mem-size="6" name="char_a1" size="1" synchro="NONE"/>
                <Register address="6" dimension="3" format="uint8" mem-size="6" name="byte_a1" size="1" synchro="NONE"/>
                <Register address="12" dimension="3" format="uint16" mem-size="6" name="word_a1" size="2" synchro="NONE"/>
```

The parameters file 5

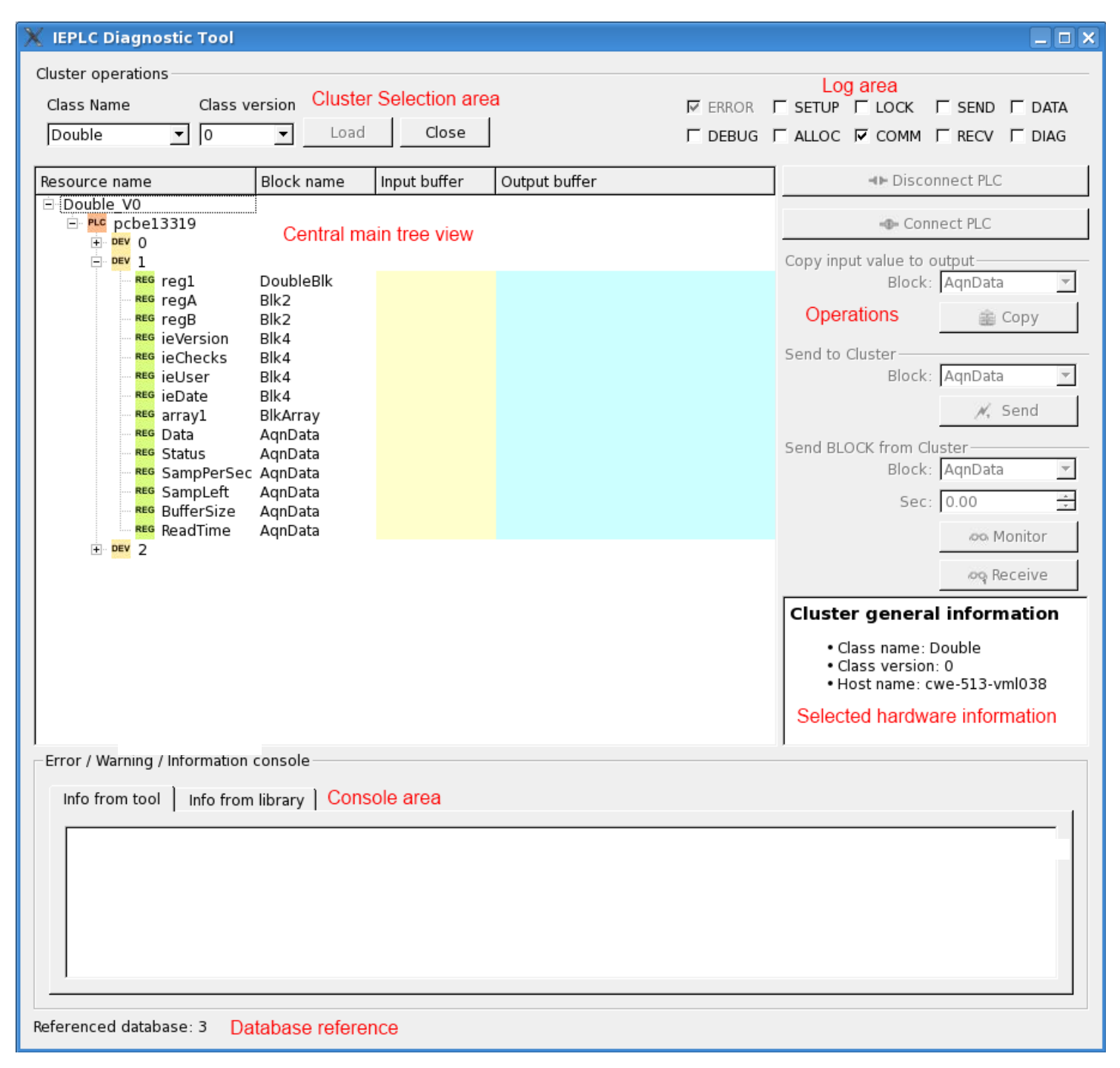
The library that is instantiated by a client application running on the FEC (FESA server, python scripts, C/C++ software, etc.) first loads the parameters file(s) which reflects the exact mapping of the target controller. IEPLC allows the same equipment class to be instantiated over different heterogeneous controllers as well as deploying different classes on the same controller.



The client library 6

The C++ library handles the communication between the FEC and the controller(s) hiding all the communication details through a very simple and high-level API. It implements the low-level protocols of the supported PLCs and microcontrollers including the (re)connection mechanism, data transport, communication parallelism, diagnostic, etc. Use of native and standard protocols of the industrial ethernet coupler and polling mode for data transmission simplifies considerably the communication process and makes the service fully generic, and very flexible and scalable.

Keywords:
Native protocols: S7 put/get, Modbus-TCP, NI-CNV



Class Name: Double Class version: 0

Resource name	Block name	Input buffer	Output buffer
Double_V0			
PLC pcbe13319			
DEV 0			
DEV 1			
REG reg1	DoubleBlk		
REG regA	Blk2		
REG regB	Blk2		
REG ieVersion	Blk4		
REG ieChecks	Blk4		
REG ieUser	Blk4		
REG ieDate	Blk4		
REG array1	BlkArray		
REG Data	AqnData		
REG Status	AqnData		
REG SampPerSec	AqnData		
REG SampLeft	AqnData		
REG BufferSize	AqnData		
REG ReadTime	AqnData		
DEV 2			

Block: AqnData
Block: AqnData
Block: AqnData
Sec: 0.00

Cluster general information
- Class name: Double
- Class version: 0
- Host name: cwe-513-vml038

Referenced database: 3

The diagnostic tool 7

A graphical tool provides the user the possibility of interoperating with hardware adopting the IEPLC service for diagnostic and testing purposes. Full control is given in sending and receiving data from the tool, so that specialist can validate his complete configuration before operational delivering.

What about the future?

Both IEPLC and FPGA technologies could be used to speed-up and standardize the integration of controls' custom hardware within the front-end system. Proposal is to extend the IEPLC framework in order to be able to communicate with generic WISHBONE or IPbus slave cores. FPGA will be used as a bridge between the specific IO peripheral and the Ethernet communication using projects developed by GSI, University of Bristol and CERN, respectively called Etherbone and IPbus bridge.