


Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		


Monitor And Control Architectural Design Document

Verified:			
Name	Signature	Date	Rev.nr.

Accepted:		
Work Package Manager	System Engineering Manager	Program Manager
Ruud Overeem	Andre Gunst	Jan Reitsma

© ASTRON 2007
All rights are reserved. Reproduction in whole or in part is prohibited without written consent of the copyright owner.




Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Distribution list:

Group:	For Information:
ASTRON	

Document history:

Revision	Date	Chapter / Page	Modification / Change
1.0	29 September, 2001	-	Creation
2.0	11 October, 2002	All	Special issue for SSSR
2.1	25 November, 2002	All	Major rework
2.2	18 February, 2003	All	Added comments R. Kleefman
2.3	20 February, 2003	GCF layer	Major update
2.4	21 February, 2003	All	Prepare for internal review
2.5	12 March, 2003	All	Added review comments MAC-team.
2.6	21 October, 2003	GCF storage layer	Added chapters 5.5.1 and 5.5.2
2.7	4 January, 2004	GCF	Added integration PVSS.
3.0	25 March, 2007	All	Rewrote document to reflect current design ideas.
3.1	30 March 2007	All	Update after internal review.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Abstract

Doing an observation with the LOFAR telescope requires that several thousand programs work precisely together. Not only are these programs distributed over almost hundred physical different locations, they also run on different hardware platforms. Robust control is necessary to guarantee correct cooperation between these programs.

The Monitor And Control package is responsible for the coordination between all these programs and reporting the problems to the operator of the instrument. The MAC package offers the operator full insight in the state of all hardware- and software parts of the instruments and allows the operator to control the instrument.

This document describes the architectural concepts and design choices of the MAC package¹. It shows how the package is able to control an instrument as complex as LOFAR, how the package is made robust to minimize failures and how the operator can interfere with the observations.

¹ The MAC software was originally designed and developed by a third party. The GCF framework they delivered turned out to be very good and is still in use after some minor changes. The controller concept turned out to be too unstable and unusable for production software. The controller concept was redesigned and rebuild in 2006. This document describes this new concept.



Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Table of contents:

1	Introduction.....	5
1.1	Purpose of this document.....	5
1.2	LOFAR subsystem overview	5
1.3	Document reference	6
1.4	Definitions.....	6
1.5	Acronyms and Abbreviations.....	6
1.6	Document overview	7
2	Introduction MAC architecture.....	8
2.1	MAC hardware architecture.....	10
2.2	MAC software architecture	11
2.3	MAC control architecture	12
3	The SCADA package PVSS	13
3.1	Deployment of PVSS in MAC.....	13
3.2	Database setup and maintenance.....	14
3.3	Synchronization with SAS	14
3.4	Maintenance sessions of the database	14
3.5	Runtime maintenance of the LOFAR instrument.....	14
4	Generic Control Framework	16
4.1	Layer architecture	16
4.2	Property concept.....	17
4.3	Datapoints and applications	18
4.4	Datapoint hierarchy.....	20
5	Monitor and Control concept	22
5.1	Monitor and Control chains.....	23
5.2	Main control over TCP/IP	23
5.3	Monitoring via PVSS.....	24
5.4	Controller template	24
5.5	Control commands.....	25
6	Subsystem relations	27
6.1	SAS.....	27
6.2	SHM.....	28
6.3	Station software	28
6.4	WAN.....	29
6.5	CEP.....	29
	Appendix A: PVSS manual abstract.....	31
	Appendix B: Requirements compliance.....	38

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

1 Introduction

1.1 Purpose of this document

When designing a Monitor And Control package three major design issues come in mind: distributed, robust and (near) real-time. This document gives an answer how these issues are solved in the MAC package. The document starts with explaining the lower- and mid-layers of MAC. After the reader is familiar with some terms and mechanisms the top-layer of MAC, containing the controllers and monitor programs, is described.

1.2 LOFAR subsystem overview

Because the Monitor and Control subsystem manages the whole LOFAR telescope it has interactions with all other subsystems of LOFAR. The functional block diagram below shows the environment of the MAC subsystem.

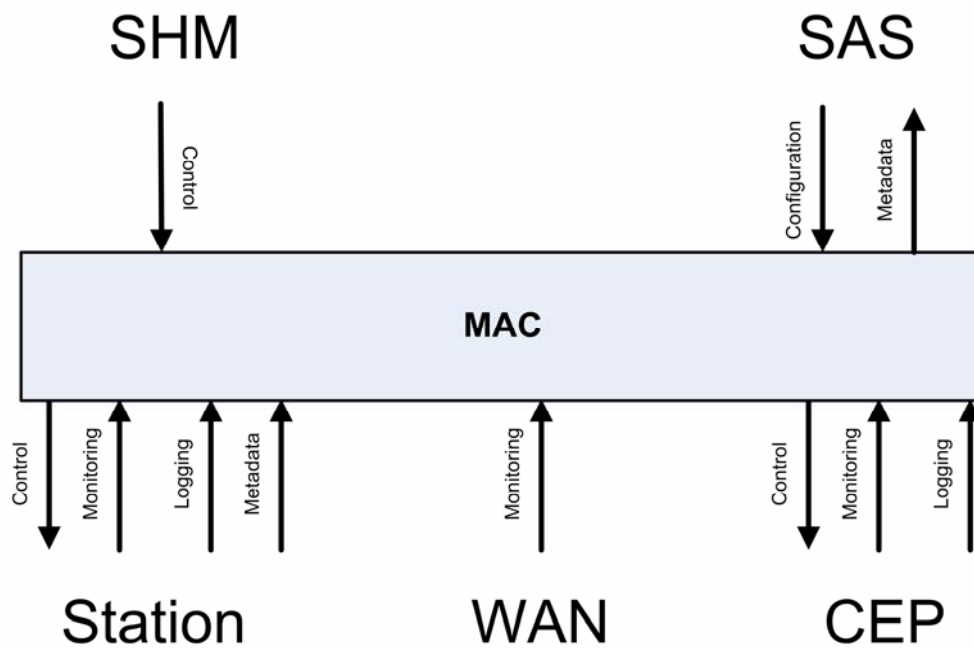



Figure 1: System overview of the environment of the MAC package.

The SAS (Specification, Administration and Scheduling) subsystem contains the start-up parameters for every observation-related program of LOFAR. It is used for specifying observations and for storing runtime metadata of the observations. The Station consists of the receivers, hardware drivers and signal processing (beam-forming) components. The CEP (central processing) subsystem provides further signal processing and contains functionality for image-, pulsar post-processing and much more.

The Monitoring and control segment (MAC) contains the monitoring and control function. This function ensures that all parts of the LOFAR system work together coherently and failures in the hardware, software of signal transport are detected. MAC supports transparent access to all parts of the LOFAR system to the operators.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

1.3 Document reference

1. LOFAR Software architecture (LOFAR-ASTRON-SDD-049)
Author: K. v.d. Schaaf; dd. 2006-07-07; Rev. 1.0
2. LOFAR, System Requirement Specification (LOFAR-ASTRON-SRS-001)
Authors: M. van Haarlem, H. Kollen; dd. 2007-03-28; Rev. 4.1
3. Reflections of the MAC-SAS interface meetings (LOFAR-ASTRON-SER-005)
Authors: A. van der Hoeven, E. Lawerman, H. de Wolf; dd. 2002-02-15; Rev. 1.0
4. SAS, Architectural Design Document (LOFAR-ASTRON-SDD-041)
Authors: R. Overeem, dd. 2007-03-30; Rev. 1.2
5. GCF, Detail Design Document (LOFAR-ASTRON-SDD-007)
Author: T. Muller; dd. 2003-09-19; Rev. 5.2


1.4 Definitions

Term	Description
Alert	An event that could not be solved by the MAC controllers and that is passed to the operator for human intervention.
Controller	Process that controls a part of the instrument in order to perform an observation.
Control Database	The real-time database on a control unit.
Control Unit	A computer containing an instance of a control database.
Metadata	Information that is relevant for processing the received signal data.
Monitor data	Status information of the instrument that is not relevant for processing the received signal data.
Observation	A description that specifies a set of measurements and subsequent processing steps to be made by the telescope.
Operator	A user that verifies the execution of the observations. He is allowed to modify the instrument in order to get better observation results.
Monitor process	A process that monitors hardware and/or software and stores the relevant information in a database.
Property	A Property represents a status-field of the instrument or a status-field of a process. It can be used for monitor and control purposes.
Property Set	A C++ instance of a property.
Terminal Unit	A computer that contains a user interface program that can connect to a control unit. It allows the user to monitor and control the programs that are connected to the same control database.

1.5 Acronyms and Abbreviations

ACC	Application Configuration and (lifecycle) Control
API	Application Programming Interface
CEP	CEntral Processing
CCU	Coordination Control Unit
CPU	Central Processing Unit
CU	Control Unit
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
HMI	Human-Machine Interface
MAC	Monitoring And Control
MCU	Main Control Unit
PA	Property Agent
PIC	Physical Instrument Components
PML	Property Management Layer




Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

PVSS	Prozessvisualisierungs- und Steuerungs-System
RFI	Radio Frequency Interference
RTC	RealTime Control
SAS	Specification, Administration And Scheduling
SCADA	Supervisory Control And Data Acquisition
SCU	Station Control Unit
SHM	System Health Management
SNMP	Simple Network Management Protocol
STS	STation Subsystem
TU	Terminal Unit
VLAN	Virtual Local Area Network
WAN	Wide Area Network

1.6 Document overview

The document is organized in the following way:

- Chapter 2 gives an overview of several views on the MAC architecture.
- Chapter 3 describes how this package is used in the MAC package.
- The Generic Control Framework that forms the basis of MAC is described in chapter 4.
- Chapter 5 zooms in on the concept of 'monitor' and 'control'.
- Chapter 6 describes the (interface) relations MAC has with all other subsystems.
- In Appendix A an abstract of the PVSS manual is given to give the reader an idea of the possibilities of this SCADA package.
- Appendix B summarizes the compliancy of MAC with the requirements.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

2 Introduction MAC architecture

The function of the Monitor and Control package can be symbolized in one picture:

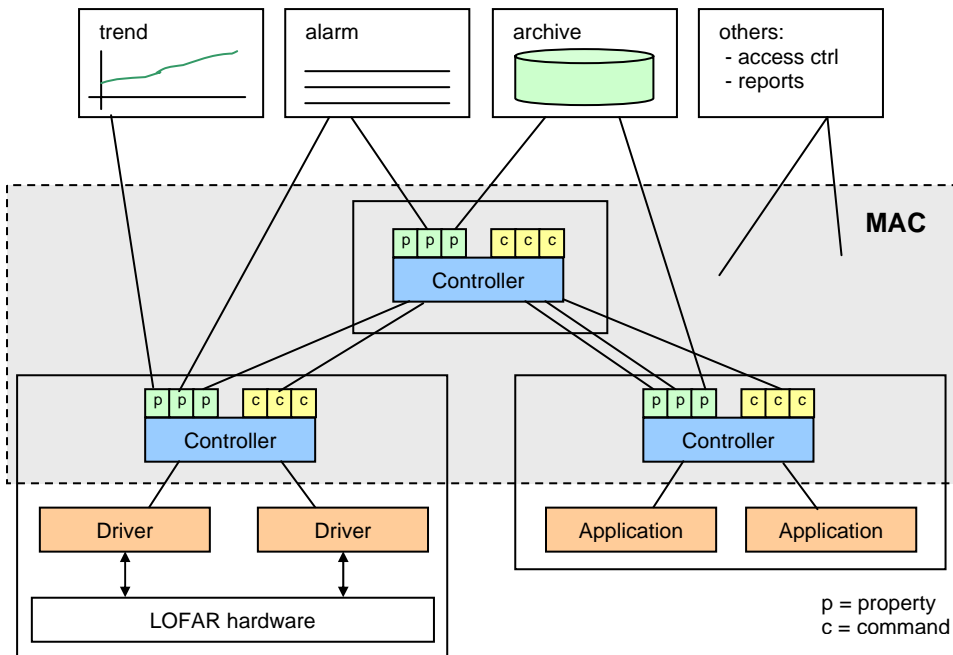



Figure 2: Functional view on the MAC package.

MAC contains hierarchical chains of controllers that are used to control the LOFAR instrument and software. These controllers, drivers and applications can be deployed on different computers. Each controller has his own set of properties. The properties are addressed in a global name-space so that every controller can also access the properties of other controllers and applications

Properties can be used to export the internal state of a control application to the outside world or to change the internal settings of a control application. But the properties are of course pre-eminently suitable for registering the status of the LOFAR instrument.

On top of the controller chains are User Interface services as trends, alerts, archiving, access-control, HMI and others. These facilities are used by the User Interface Applications to implement their tasks.

It is important to notice that these services have properties as input. It is possible to connect a service to a certain property. As soon as the property changed, the connected service is updated (re-calculated).

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

When we look at the MAC package in a more detailed way we can distinguish the following modules in and around the MAC package:

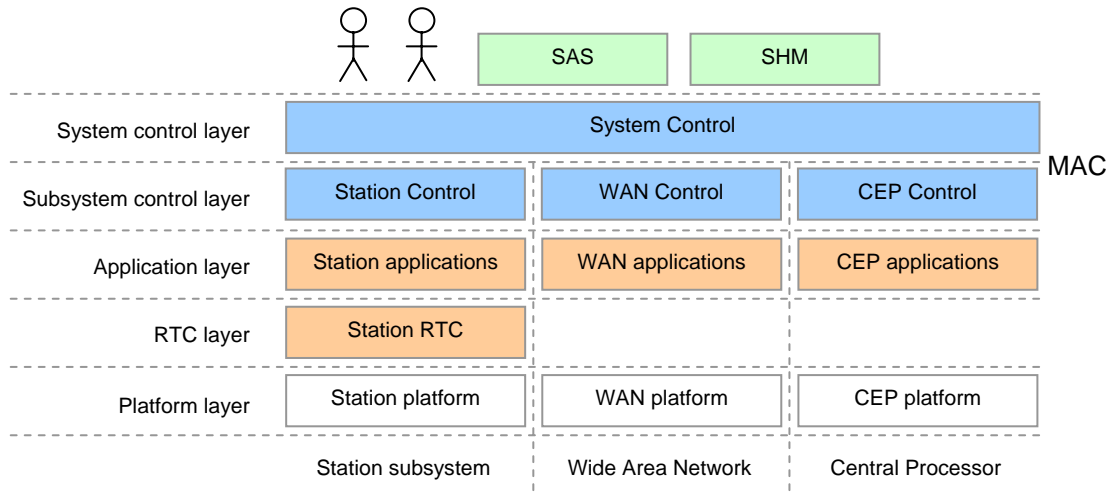


Figure 2: Module view of the MAC package environment.

Platform layer: LOFAR instrument hardware such as antennas, receptors, CPU's, DSP's, FPGA's, network transport equipment and Linux clusters.


RTC layer: direct (real-time) control of the LOFAR instrument. This layer can be seen as an "embedded software" layer, in which very instrument specific control software is implemented. Software in this layer has a very tight coupling with the instrument. Platform functionality is controlled in the RTC layer.

Application Layer: LOFAR application software. This software is used for the analysis and processing of measurement data. Example application software: beam forming, RFI analysis and mitigation algorithms, self-calibration algorithms, data production software. There are no WAN applications yet.

Subsystem control layer: in this layer, (non-real-time) subsystem control functionality is implemented (control of platform and application functions). Also supervisory control is implemented (monitoring and control functionality which is directly related to operator interaction).

System control layer: this layer supports system control activities: control of the system platform hardware and control of system wide applications (called observations). Also supervisory control functionality is available here. This layer supports transparent access to all application and platform functions of the whole LOFAR system.

The system control and subsystem control layers are part of the MAC system. In this document, the architectural design of these layers is described. The following paragraphs start with an introduction of the various views on the MAC architecture.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

2.1 MAC hardware architecture

The figure below shows the MAC hardware architecture.

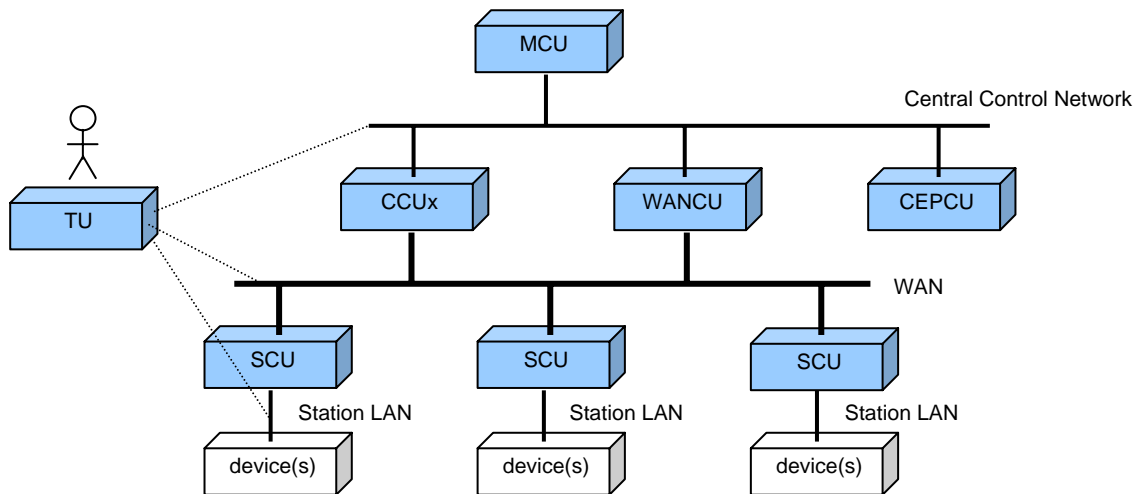


Figure 3: Top-view of the MAC hardware architecture.

To control the LOFAR instrument MAC uses many Control Units that are connected by various networks. These Control Units and networks are described in the following paragraphs.

2.1.1 Control Units


A Control Unit is a PC containing an instance of a control database. The contents of the database and the functions of the CU depend on the local control tasks that must be managed. Every type of CU has got its own name that reflects its local tasks.

Stations Control Units: All the station-software is located on a Station Control Unit (SCU)². The main task of the SCU is controlling the dedicated LOFAR hardware. MAC uses the drivers of the RTC software for this task. Controllers running on a station are for example StationControl, BeamControl, CalibrationControl, etc.

Coordinating Control Units: A CCU coordinates the activities of a group of SCU's. It has to be determined how the grouping of the SCU's will be done: per ring (core, inner-, outer-ring) or per arm. The main reason for using CCU's is spreading the load on the control-databases. There will be about 100 LOFAR stations and connecting 100 station-databases to 1 central database will probably overload the central control database.

WAN Control Unit: On the WAN Control Unit all information about the WAN is collected. This is data like average load on lines, status off the lines, status of the switches, etc. This information is collected at regular bases using the SNMP interfaces of the network components.

² In other documentation the SCU is often still called LCU (Local Control Unit). Since 'Local' is an ambiguous term the name Station Control Unit is used in the more recent MAC documents.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

CEP Control Unit: On this machine the control of the data-processing applications is done. The applications can be divided into the online applications that must keep-up with the life data-stream and the offline applications that are less time-bound. It is likely that both types of applications can be managed with one control unit.

Main Control Unit(s): In the database on the Main Control Unit all control databases come together. This is typically the place where the operators and other LOFAR users connect to with their Terminal Unit. Once their Terminal Unit is connected to this database they are able to monitor and control the whole LOFAR instrument.

Terminal Unit: A Terminal Unit is a machine that contains a User Interface program (called Navigator) that allows the user to monitor and control the programs that are connected to the same control database. A TU can be plugged in on any network of LOFAR but the scope of the monitor and control rights will be limited to that section of LOFAR the TU is connected to.

Hardware requirements

All Control Units can be built with industrial PC's running Linux and a SCADA package. The requirements of the various CU's will be different and depend on the (peak) load of the software. For the TU's a machine with Windows is required.

2.1.2 Networks

Station LAN: On the LOFAR stations the instrument devices are connected to the SCU through an Ethernet-switch. To separate the several data-streams VLANs can be created on this switch.

Wide Area Network: The LOFAR stations are connected to the central site via a WAN that is based on a private network infrastructure. This WAN is primary used to transport the immense amount of measurement-data from the stations to CEP but has enough bandwidth left to use it as a control network too.

Central Control Network: Control network located at the central site. It will probably be implemented as a VLAN to assure safety and bandwidth.

2.2 MAC software architecture

All the MAC software will be build using the same software architecture:

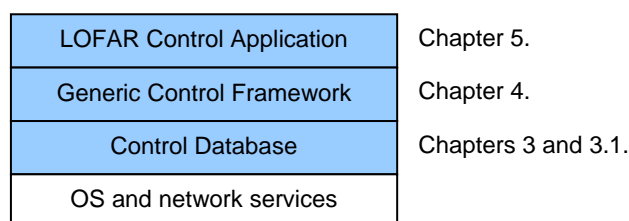



Figure 4: *Top-view of the MAC software architecture.*

OS and Network services: Operating system layer (Linux, NT and others) and specific network services (e.g. multicast support, IPv4 services).

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Control Database: All MAC controllers make use of a distributed control database. This database is used for storing the monitored values, passing control values and alerting the operator(s). A more generic name for such a control database is a SCADA package. (Supervisory Control And Data Acquisition). For MAC the SCADA package PVSS from the company ETM will be used.

Generic Control Framework: Application independent framework for building real-time applications like controllers. This layer contains no LOFAR specific concepts. The Generic Control Framework is not only used by MAC software but parts of it are also used by the RTC software. The GCF is described in more detail in chapter 4.

LOFAR Control Application: Specific Monitoring and Control software for the LOFAR system. The Control Applications (or simply Controllers) are running on many machines since they control all (signal data related) programs. The Controllers are described in chapter 5.

2.3 MAC control architecture

MAC uses a hierarchical control chain to be able to control the LOFAR instrument as a whole but also every detail of the instrument. This hierarchy can be divided into several logical layers :

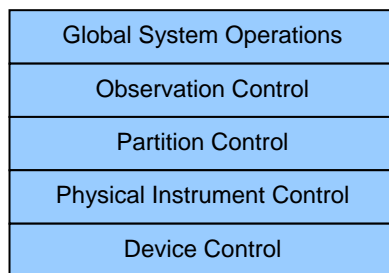


Figure 5: Top-view of the MAC control architecture.


Global System Operations: Operations that have a full system view. In this layer the MAC scheduler is located. The MAC scheduler receives schedules from the Scheduling Administration and Specification subsystem. The MAC scheduler executes this schedule: it start and stops observations. Platform Management activities are also scheduled here (e.g. maintenance tasks).

Observation Control: Control and monitoring of observations. In this layer, observations are defined, controlled and evaluated. Multiple parallel observations can be controlled here.

Partition Control: Implementation and control of partitions. In this layer partitions, like subarrays, are defined and implemented. Multiple parallel partitions can be activated here.

Physical Instrument Control: Control of the physical instrument. Platform management tasks are implemented here (e.g. power control, start/stop/reset control, software distribution, ...).

Device Control: Control of the devices. Device drivers, connection drivers and Device servers are located here.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

3 The SCADA package PVSS

One of the major elements of the MAC package is the SCADA package that provides a real time distributed database environment. For MAC PVSS is chosen to fulfill this task. Although PVSS is wrapped in a SCADA abstraction layer to make the SCADA interface more generic it is important to know what facilities PVSS offers to MAC. Readers that are not familiar with SCADA packages are advised to read Appendix A first.

This chapter describes how PVSS is used within MAC.

3.1 Deployment of PVSS in MAC

Applying the available managers of PVSS to the various Control Units gives the following picture:

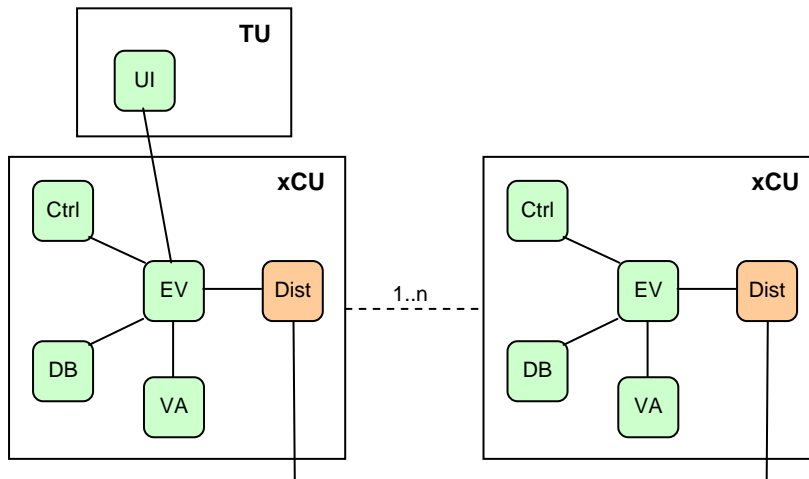



Figure 6: *Deployment of PVSS modules in MAC.*

Since all control units are more or less the same the deployment of PVSS can be the same on every control unit. Each control unit should get:

- An Event Manager which is the kernel of the PVSS system.
- A DataBase manager for saving data-changes and making reports.
- A Control manager that makes it possible to run PVSS scripts as independent tasks.
- A Value Archive to store history of data changes.
- A Distribution manager to connect the PVSS databases of different control units to each other.

The Terminal Unit where the User Interface Application is running only needs the User Interface module. This keeps the installation on these TU's simple and has the benefit that no license is required on these machines.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

3.2 Database setup and maintenance

Usually all database maintenance can be performed by the standard PVSS tools such as the PARA module and the ASCII manager. These tools are sufficient to perform maintenance on the PVSS databases when there are no relations with other languages or systems. However, in our case we have to deal with C++ layers that are built on top of PVSS and a SAS database that also wants to know the status of the LOFAR hardware. Therefore another approach is chosen to maintain the PVSS contents.

Every controller needs to know the names of the properties he uses and of the (shared) properties of other controllers. The names should of course match exactly with the names of the datapoints in the database. The only way to guarantee this is that the C++ names and the PVSS names origin from the same source.

With some very simple tables like station-names, number of RSPboards per station, cluster-names, number of nodes per cluster, etc it is possible to write a script that generates header-files for the C++ code and datapointtype- and datapoint- files for the PVSS ASCII manager that contain all the information that is needed to create PVSS databases that are consistent with the C++ headerfiles.

During the development phase of LOFAR this solution will do fine, there is no difficulty in recreating the PVSS databases since they do not contain important data. For final LOFAR however we need some extra tooling to guarantee that only the differences between the current database and the new database are applied to the database.

The ASCII manager of PVSS will do the trick in this case. With the ASCII manager an export can be made of the current datapointtypes and datapoints. These files can be compared with the new generated files and a insert- and delete-file can be made. Processing these files with the ASCII manager will add the new parts to the database and delete the obsolete parts.

3.3 Synchronization with SAS

The scheduler of SAS also needs to know what hardware elements are currently in the physical instrument. It is not necessary that this is the actual state of the physical instrument since the scheduling is done in advance and there will always be a certain average percentage of the system unavailable. It is therefore sufficient to update the SAS database only when the PVSS database is updated.


In SAS there is a separate PIC (Physical Instrument Components) table that can store the hardware configurations. These configurations, called PIC trees, can be created from an export-file from the ASCII manager from PVSS (or the datapoint-files created by the maintenance script). Several PIC trees can exist in SAS but only one PIC tree can be the 'current' tree.

3.4 Maintenance sessions of the database

Most programs determine during their setup how the hardware of LOFAR is configured. Deleting and adding hardware (and the corresponding datapoints) while these programs are running will have unpredictable results. It is therefore recommended that maintenance on the databases is done in regular (planned) periods, e.g. the monthly maintenance sessions. During such periods no observations are running so it is save to stop and start databases and controllers.

3.5 Runtime maintenance of the LOFAR instrument

Every hardware component that has a datapoint in the PVSS database will have a state field that can have the following states:


Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

State	Description
Off	Component is switched off or is removed.
Operational	Component is available for observations.
Maintenance	Component can not be used for observations because maintenance will be applied.
Test	Component can not be used for production observations but can be scheduled for observation of the type 'test'. This mode will be used when testing new stations are doing regression tests.
Suspicious	The health of the component is degrading, it will not be used in future observations. Manual intervention is needed to test or replace the device.
Broken	The component is malfunctioning and needs to be replaced.

The normal state of each component will of course be 'Operational', but the System Health Management subsystem will be able to set the state to 'suspicious' whenever it thinks the component has an abnormal behavior. This will not interfere with the running observations but the component will not be used anymore in future observations.

An operator or maintenance engineer can use the other states to flag the usage of the component.

Major maintenance, e.g. maintenance of station power-supplies or Ethernet-switches, that will result in 'loss' of large(r) partitions of the instrument can be scheduled in SAS as a maintenance 'observation'. These observation-types have the highest priority and will prevent that the real observations can use the related part of the instrument.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

4 Generic Control Framework

The Generic Control Framework is an application independent framework for building control applications. This framework is a middleware layer between the LOFAR control application and the MAC hardware architecture.

4.1 Layer architecture

The GCF consists of the following layers:

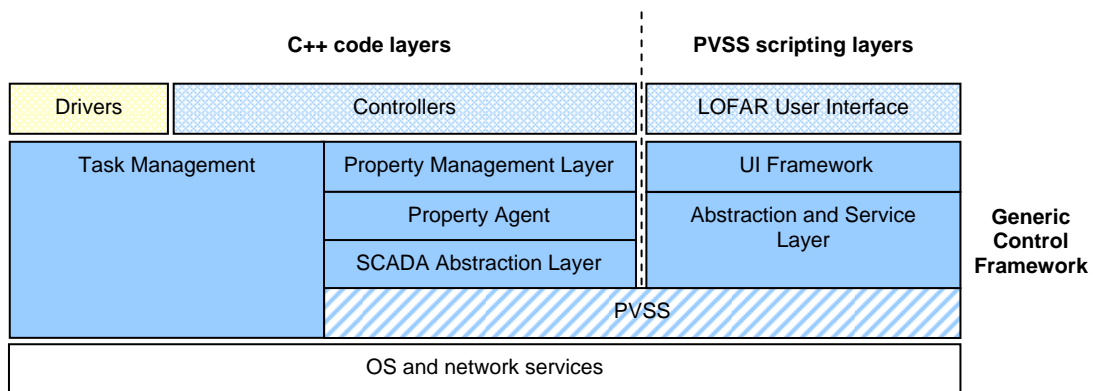


Figure 7: Logical layers of the MAC GCF.


The Generic Control Framework offers support for C++ code and for the scripting language that comes with PVSS. The layers for C++ are true layers so the C++ programs only have access to the Task Management module and the Property Management Layer. The layers for the user interface are relaxed so each layer can be used in the scripting, this allows fast and easy access to the database.

The GCF C++ libraries allow the applications to access datapoints and access network ports. The classes are designed in such a way that it is possible to design so-called reactive, event-driven applications. For example, a task responds immediately to messages that are arrived on its connected ports and reacts immediately to changes in (its) datapoints.

A short description of the modules is given below.

PVSS: This is the chosen package to fulfill the SCADA functionality. The current state of the instrument-elements are stored in this database, e.g. voltage levels, temperatures, clock-settings, power-states, etc. The database is also used to represent the status of internal control structures like beam-direction, observation settings, actual spectral RFI mitigation and many more. Per property it is possible to configure the needed alert-handling, archive handling, sample (trend) handling and access-control handling.

Notice that the only connection between the User Interface application and the C++ programs are the datapoints in the PVSS database. So when a C++ program wants to communicate with the User Interface Program, or vice versa, they have to use data-points to exchange information.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

4.1.1 C++ support

For C++ the following modules are defined:

Task Management: This module offers all kind of basic services as timers, TCP/IP communication, state machines and tasks. It is only based on the services that the core OS offers, so it is independent from PVSS. The LOFAR device drivers make use of this layer.

SCADA Abstraction Layer: This layer is a wrapper around the PVSS package. The main reasons for building this layer are:

- Implement extra (missing) services/features for handling control applications.
- Create a (more) generic interface for accessing the SCADA product.
- Loosen the dependencies with the SCADA product.

Property Agent: The PA is responsible for the synchronisation of the datapoint instances used by the C++ programs and the real datapoints in the PVSS database. To limit the number of database changes the PA will only update those datapoints that are used by others.³

Property Management Layer: The PML is the only interface to the PVSS database for the C++ programs. It offers functions for using PropertySets and Properties (see paragraph 4.2).

4.1.2 User Interface support

For the LOFAR User Interface the following modules are defined:

Abstraction and Service Layer: This layer simplifies the use of PVSS datapoints and makes sure the (C++) Property Agent gets the right information to do its job by telling him which data-points the user is monitoring.

Framework: The framework implements the main functions and layout for the LOFAR User Interface, like a hierarchical representation of the datapoints, standard alert windows, user and access management, etc.

4.2 Property concept

The core concept of GCF is the property concept. To be able to understand the design of GCF we first zoom in on the property definitions.

Property: A property represents a status-field of the instrument or a status-field of an internal control process. Properties can be read-only or read/write. If a read/write property cannot read its value back (for example it is associated to a write only physical device), it caches the last written value and returns this upon read request.


Properties can represent values using a limited set of basic data types:

Property ::= <Name>, <Type>

Type ::= <Simple-Type> | <Complex-Type> | <Structure-Type>

Simple-Type ::= "unsigned-integer" | "integer" | "character" | "boolean" | "bit-pattern" | "float" | "text" | "date/time"

³ Another reason for limiting the number of data-points in the PVSS is that the licenses are based on the amount of datapoints you want to manage in your database.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Complex-Type ::= “text“ | “array“ | “dynamic-array“
Structure-Type ::= { <Property> }

PropertySet: An C++ instance of a property. In most cases this will be a property of the type Structure-Type. Propertysets are used by the MAC applications but are managed by the PML and the PA. Each property of a propertyset is mapped to a datapoint of the PVSS database.

Property type: PVSS type definition of a property.

Property tree: It is possible to configure a tree of properties. In each node of this tree, one or more properties are available. Normally at the leafes of this tree, properties are configured which are a subset of the controlled devices.

Property functions: a property functions converts a set of properties to a new (more abstract) set of properties. E.g. $p_1, p_2, p_3 = f(p_4, p_5, p_6)$. A property function can be a transition function or a control function.

Characteristic: static data associated with a property, including meta-data. Each property has 0..n characteristics. The values of the characteristics are stored in the configuration database that is also part of PVSS.

The following characteristics are possible:

- name, description, version
- system value range
- user value range
- alert handling: alert ranges can be defined.
- default value handling: if a value is invalid, a default value can be set in various ways.
- archiving
- authorization: authorization level settings to check user authorizations and control user activity.

4.3 Datapoints and applications

As described in the previous chapters, the control applications (or simply Controllers) use datapoints to exchange information. These datapoints are accessible via PropertySets that are managed by the Property Management Layer and the Property Agent. The following picture shows these elements:

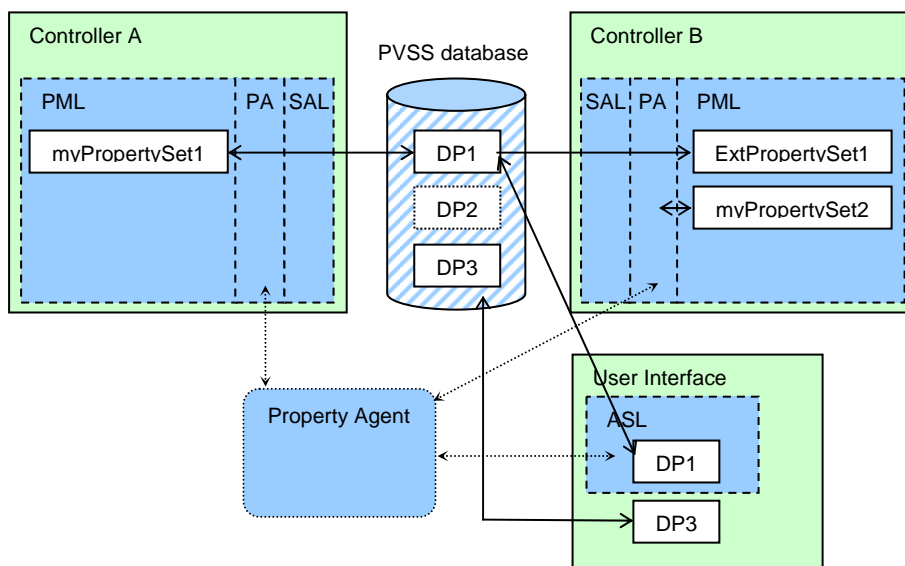



Figure 8: Datapoint usage in applications.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

In the figure above, the typical usage of datapoints (property-sets) is illustrated. We see two control applications and one user interface program. An application can own a propertyset (myPropertySet) or take a subscription on a propertyset of another application (extPropertySet). The property sets of an application are created in the PML layer. The property values have to pass the PA layer and the SAL layer before the values reach the database. The PA layer of the applications is connected to the Property Agent process that runs on every control unit. The ASL layer of the User Interface can communicate with the Property Agent process also.

Three typical situations exists:

Situation1: MyPropertySet1 refers to datapoint 1 in the database and at least one other application is using the datapoint. If a propertyset is used by another application the Property Agent assures that all modifications made to the propertyset will be transported to the datapoint (and vice versa). When the datapoint or the propertyset is changed very frequently this can slow down the related programs since they receive a trigger for every value change.

Situation2: MyPropertySet2 refers to datapoint 2 in the database and nobody else is using this datapoint. The Property Agent will not create a datapoint for this propertyset in the database so all the changes made by Application B in the MyPropertySet2 object will only exist in Application B. The advantage of this construction is a lower load on the database.


Situation3: Datapoint3 from a User Interface Application is linked to datapoint3 in the database and nobody else is using this datapoint. When this datapoint is used for internal tasks the UIA can decide to access the datapoint directly by calling the functions of the PVSS API. In that case the Property Agent is not informed about the existence of the datapoint and it will not take any action. Maximum speed is assured. This situation is drawn in the figure above.

When datapoint3 is a datapoint that the UIA will share with other applications it will access the datapoint via the ASL. In that case the ASL informs the Property Agent about the datapoint so that other applications are informed whenever the value is changed.

The Property Agent is a GCF task that manages the above situations. When an application creates an instance of a myPropertySet, the PA is automatically informed about the existence of the datapoint. When an application creates an instance of an extPropertySet, the PA will create the datapoint and make a link between the database and the corresponding myPropertySet so that value-changes are stored in the database.⁴

Note: The datapoints and propertysets described in situations above are called *temporary propertysets*: the database is only updated when other applications are interested in the datapoint values. On the other hand you sometimes want that the value changes of a propertyset are always stored in the database, regardless of some other application is watching it. E.g. when you want to register the trend of a temperature. In that case the application can create a *permanent propertyset*. These sets are always linked to the datapoints of the database.

⁴ The construction with the PropertyAgent is one of the last elements of the original control concept we use nowadays. The concept of the PropertyAgent is not stable enough in its current implementation. Analysis of the code is necessary to see if minor changes can fix the problems or that a redesign is needed,

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

4.4 Datapoint hierarchy

The module User Interface Framework offers hierarchical representation of the datapoints. The datapoints themselves are not hierarchical so a special naming convention is necessary to realize this. By using underscores in the name of the datapoints this can be achieved. Each underscore starts a new level in the hierarchical representation. The example illustrates the instrument model of a station.

Example datapoint types:

```
DPTYPE station
    status          [byte]
    maintenance     [byte]

DPTYPE rack
    status          [byte]
    maintenance     [byte]

DPTYPE fpga
    status          [byte]
    firmwareversion [string]

DPTYPE antenna
    band           [byte]
    status         [byte]
    maintenance    [byte]
```

Example datapoint instances:

```
S1 [station]
S1_Rack1 [rack]
S1_Rack1_FPGA1 [fpga]
S1_Rack1_FPGA1_ant1 [antenna]
S1_Rack1_FPGA1_ant2 [antenna]
S1_Rack1_FPGA2 [fpga]
S1_Rack1_FPGA2_ant1 [antenna]
S1_Rack1_FPGA2_ant2 [antenna]
S1 [station]
S1_Rack2 [rack]
S1_Rack2_FPGA1 [fpga]
S1_Rack2_FPGA1_ant1 [antenna]
S1_Rack2_FPGA1_ant2 [antenna]
S1_Rack2_FPGA2 [fpga]
S1_Rack2_FPGA2_ant1 [antenna]
S1_Rack2_FPGA2_ant2 [antenna]
```

These datapoints will be shown as a tree like:

```
- S!
  - Rack1
    + FPGA1
    - FPGA2
      + ant1
      + ant2
  + Rack2
```


Each 'level' of the tree can be folded out by clicking on the plus-sign or folded in by clicking on the min-sign.

In order to access the elements of the data points (which are the basic types) a dot is used to separate elements from the datapoints, e.g.:

- S1_Rack2_FPGA2_ant2.band
- S1_Rack2.status

Notice that multiple dots can be used in case the Datapoint type contains sub-structures.


In order to access remote data points which are located on a remote database, the system name of the remote database must be entered before the instance name of the data point.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

The following naming convention is used for remote names:

- Data points on SCUs: SCU_{xx} : A_B_C.
- Data points on CCUs: CCU_{xx} : A_B_C.
- etc.

With the above naming convention, it is possible to create a containment relations. Containment relations are perfect to describe physical structures, such as instrument models and resource models.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

5 Monitor and Control concept

The following picture shows an example of a simplified Monitor and Control chain.

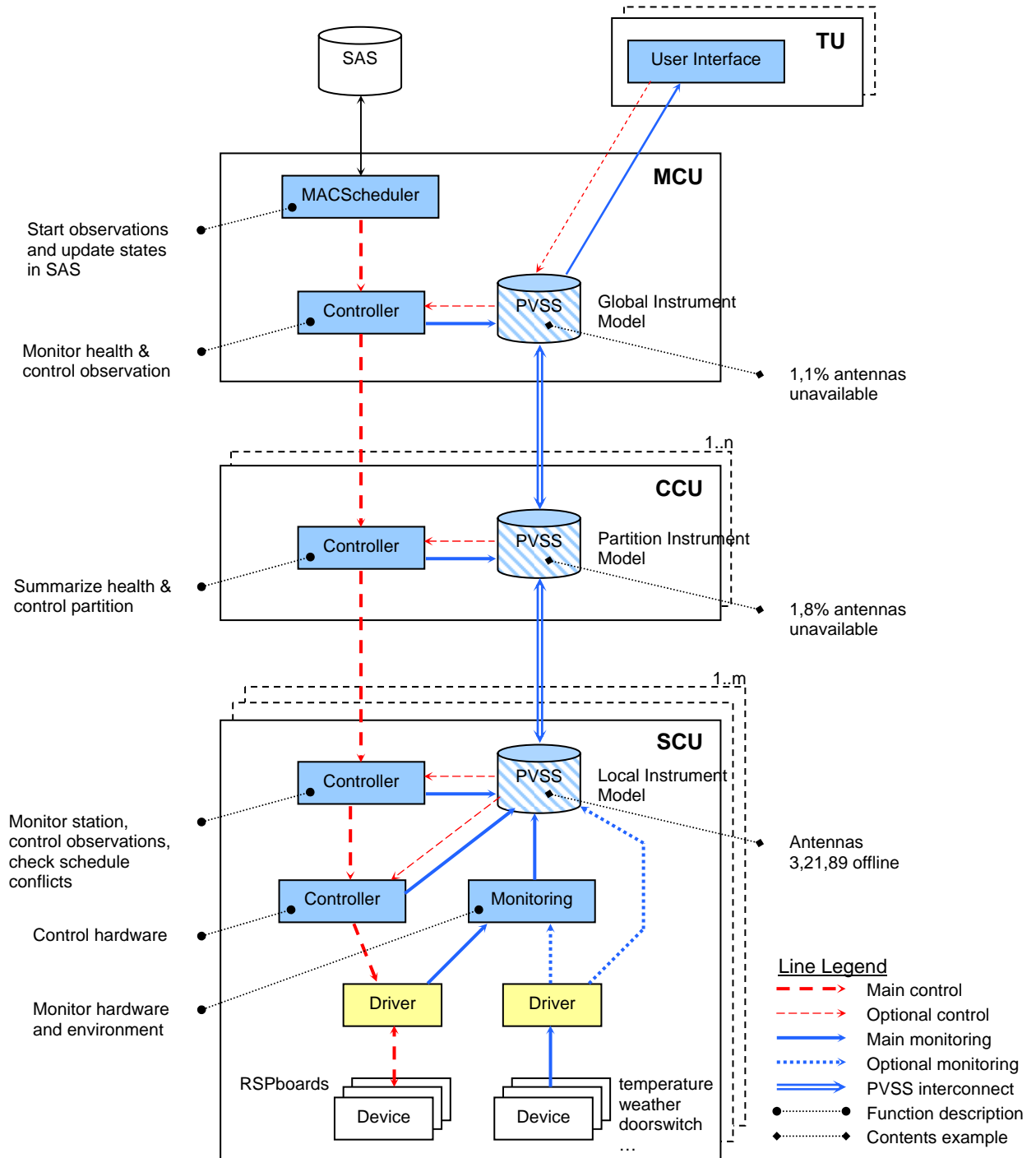



Figure 9: Schema of the Monitor and Control chains.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

The Monitor and Control concept is based on the following basic rules:

1. Instrument monitoring and Instrument control are two separate chains.
2. Main control is only using TCP/IP, but control via PVSS is still possible.
3. Monitoring information and controller-states are stored in the PVSS database.
4. Controllers use a control-template that supports chains of controllers.

These rules are explained in the following paragraphs:

5.1 Monitor and Control chains

The main idea behind the separation of the monitoring and control chains is that the control chain should be as simple and robust as possible. Monitoring of the instrument, which is a secondary task, should never interfere with the primary task 'control'.

The control-cycle of MAC consists of 8 states (see 5.5). Mainly during a state-switch many instrument values might change. When instrument monitoring and instrument control would be combined into one application the application may be so busy handling all these value changes that it cannot respond in time to new state-changes.

Of course the controllers do use PVSS for their status information and their statistical information.

5.2 Main control over TCP/IP

To make the control chain as robust as possible the main control of the controllers is passed over TCP/IP connections. Normally the top-controller of a control-chain passes a command to its child controllers via TCP/IP connections. These controllers, on their turn, pass the command to their child controllers, etc. In this way the top-controller can manage all controllers of an observation.

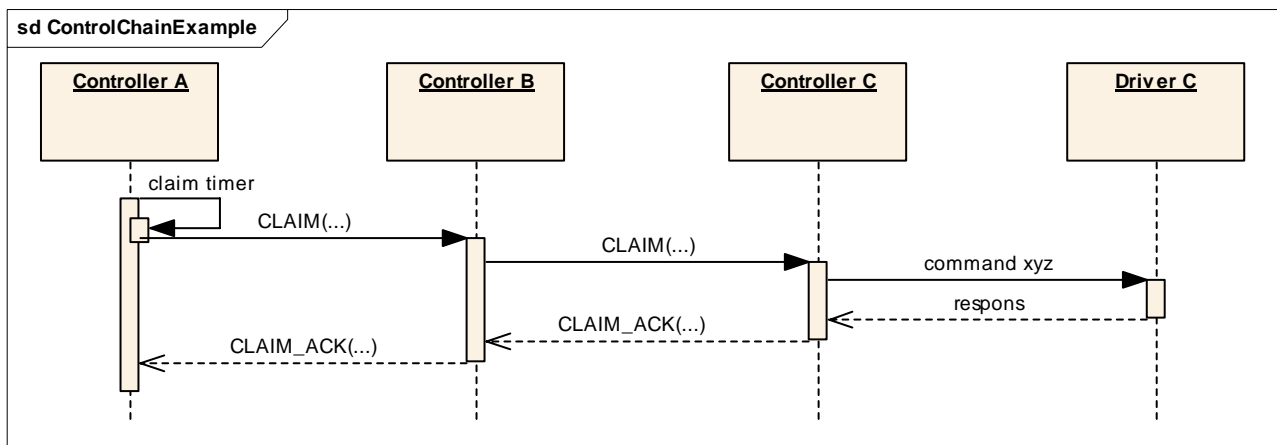



Figure 10: Propagation of a command in a controller chain.

The top-controller uses a timer to start a command, when the timer expires it sends the command to its child controller(s). This controller passes it to its child controller(s) until it reaches a 'leaf' controller. The 'leaf' controller sends the right command(s) to its driver. When the driver is set the result propagates back to the top controller.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

There are however two exceptions to this rule: the start and stop of the observation itself (not including preparing and release states etc) is initiated by timers in each controller separately. This has two major advantages:

- There is a better guarantee that the observation is started at the same time in every part of the instrument.
 - Whenever the communication with a part of the instrument is lost, the observation will not run forever.
- Starting and stopping the observation via the TCP/IP connection or via PVSS still remains possible.

The controllers can also be controlled via a 'command' datapoint in the PVSS database. Writing the desired state in this datapoint will have the same effect as giving the corresponding command over TCP/IP. In this way a user who is connected to a PVSS database via his User Interface Application is able to control the instrument also.

5.3 Monitoring via PVSS

Naturally the monitoring of the instrument is done in PVSS. The SCADA package is not only capable of distributing the data but it can also raise alarms, calculate statistics, etc.

There are two ways to get the information in the database: Write a monitoring Application that at regular interval questions the drivers and stores the information in the database. This is the preferred way for time-critical drivers. Non-critical drivers, e.g. for monitoring the weather conditions, can store the information directly in the database.

5.4 Controller template

When zooming in on a controller, three separate tasks can be distinguished:

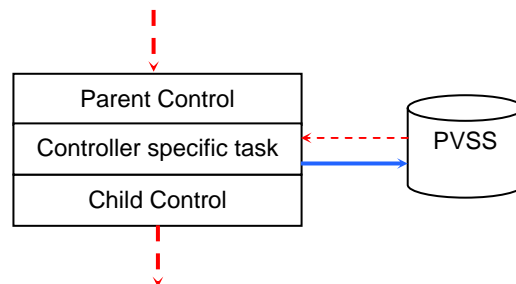



Figure 11: *Tasks of a controller.*

Parent Control: This task receives the commands from the parent controller and is responsible for maintaining the connection with the parent controller. The observation start- and stop-timer are located here. When a controller is restarted, this task will be informed by its parent controller that it is a restart instead of a start and it will try to resynchronise the controller specific task to the assumed state.


Controller specific task: This is the real controller task. It is different for every controller.

Child Control: The child control task is responsible for the communication with all child controllers. As soon as the connection with one of them is lost it tries to restart the child controller. The child control task reports the states of the child controllers to the controller specific task, this latter task has to decide what to do in case things go wrong.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

The SAS scheduler is responsible for calculating these timing aspects and storing the expected state durations in the metadata of the observation.

There are also commands that do not result in state changes. E.g. it is possible to change the start- and stop-time of the 'active' state. These commands can be received in every state of a controller and are not drawn in the picture above.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

6 Subsystem relations

MAC has interfaces with many other subsystems. In this chapter the relations between those subsystems are described.

6.1 SAS

Between MAC and SAS the following interfaces exists:

1. Poll SAS database to start observations.
2. Ask SAS to make a parameterfile.
3. Update observation state in SAS
4. Export physical instrument layout to SAS.
5. Copy alerts and actions to SAS.

6.1.1 Poll the SAS database.

The top-process of MAC is the MACScheduler (see Figure 9). The MACScheduler has a connection with the SAS database and executes every 5 seconds a stored procedure in the SAS database. This procedure delivers a list of the first 10 observations that are ready for execution. This means that the observation has valid start- and stop-times that lay in the future and that the state of the observation is 'scheduled'.

The MACScheduler keeps an administration of the running observations so it won't start an observation for a second time.

6.1.2 Create parameterfile

When its time to start an observation the MACScheduler calls a stored procedure that delivers the configuration file for the whole observation, it starts a new ObservationController and passes this configuration file to the ObservationController. This configuration file contains enough information for all programs that are involved in the observation to do their job.

6.1.3 Update observation state

Each observation in SAS has a state-field that reflects the current state of the observation. During the configure process SAS uses this field to manage the access rights to the observation. Once it is set to 'scheduled' SAS won't touch it any more and it becomes the task of the MACScheduler to keep this field up to date. The MACScheduler can set the state field to 'queued', 'active', 'finished', 'aborted'.


6.1.4 Export physical instrument layout

As described in paragraph 3.3 the PIC-trees in the SAS database are derived from the PVSS contents. When the ASCII manager is used to make a dump of the database, SAS is able to read in this file and construct a new PIC tree.

6.1.5 Copy alerts and actions

When serious problems arise during control or monitoring these problems are stored as 'alerts' in the MAC database. The operator must respond to these alerts otherwise they will be on his screen forever. The action(s) he takes on these alert are also registered in PVSS.

Because these alerts and action can also be important for the processing of the signal-data they are send from MAC to SAS as KVT triples.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

6.2 SHM

The System Health Management package has the following cycle: analyze (historic) metadata, find abnormalities, register abnormalities in PVSS and in the future perform automatic tests between observations to improve the analysis.

The historic metadata is available in SAS. The actual metadata like Array Correlation Matrices is available in the drivers and services of the station software. The only interfaces that remain are the registration of the abnormalities in PVSS and the execution of the tests. A dedicated MAC controller will be able to perform these tasks.

6.3 Station software

The following tasks must be managed on station level:

1. Control of the RSPBoards.
2. Control of the calibration of the antenna arrays.
3. Control of the beam-directions.
4. Control of the station clock
5. Control of the Transient Buffer Board

The station software delivers for each of these tasks a driver or server that is capable of doing their dedicated tasks. MAC only has to instruct these drivers and servers. For every driver and every server MAC will develop a specific controller that has knowledge about that part of the process.

By placing these controllers in the right hierarchy an observation can be controlled:

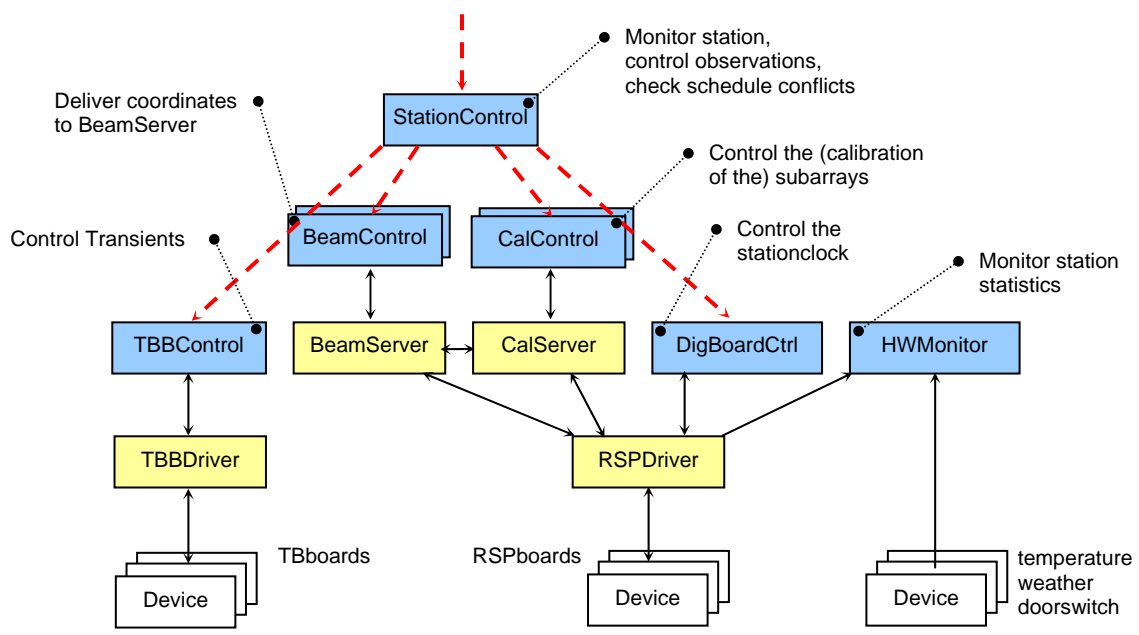



Figure 13: *Controller hierarchy on the stations.*

Notice that some controllers are shared: there is only one instance of them, like StationControl and DigBoardControl, while of others there is a controller per observation. That has to do with the kind of information they should handle, sometimes this is global information sometimes this is more observation based (e.g. beam-direction).

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Assigning a separate controller to each driver and server has the benefit the every controller only needs to 'talk' one dedicated protocol. Replacement of a controller also becomes much easier.

6.4 WAN

The WAN will be built with standard network components. These components all support the SNMP protocol to query the status of the equipment. A WAN Controller can use this protocol to keep the status of the WAN up to date in PVSS.

6.5 CEP

All software that is developed to run on the CEntral Processing platform is designed to run under control of ACC. ACC is a multiplexing/demultiplexing layer the has its own state-machine. When MAC controls an CEP application it has to interface with ACC in stead of the specific application:

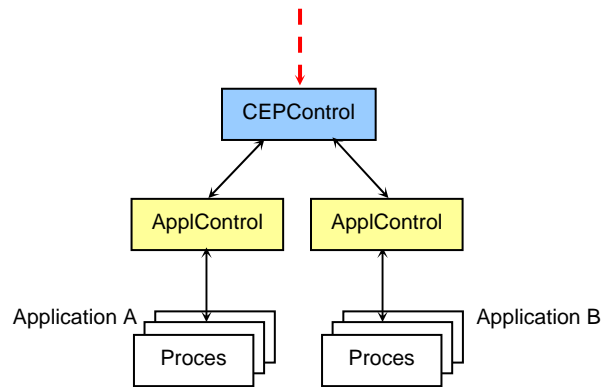

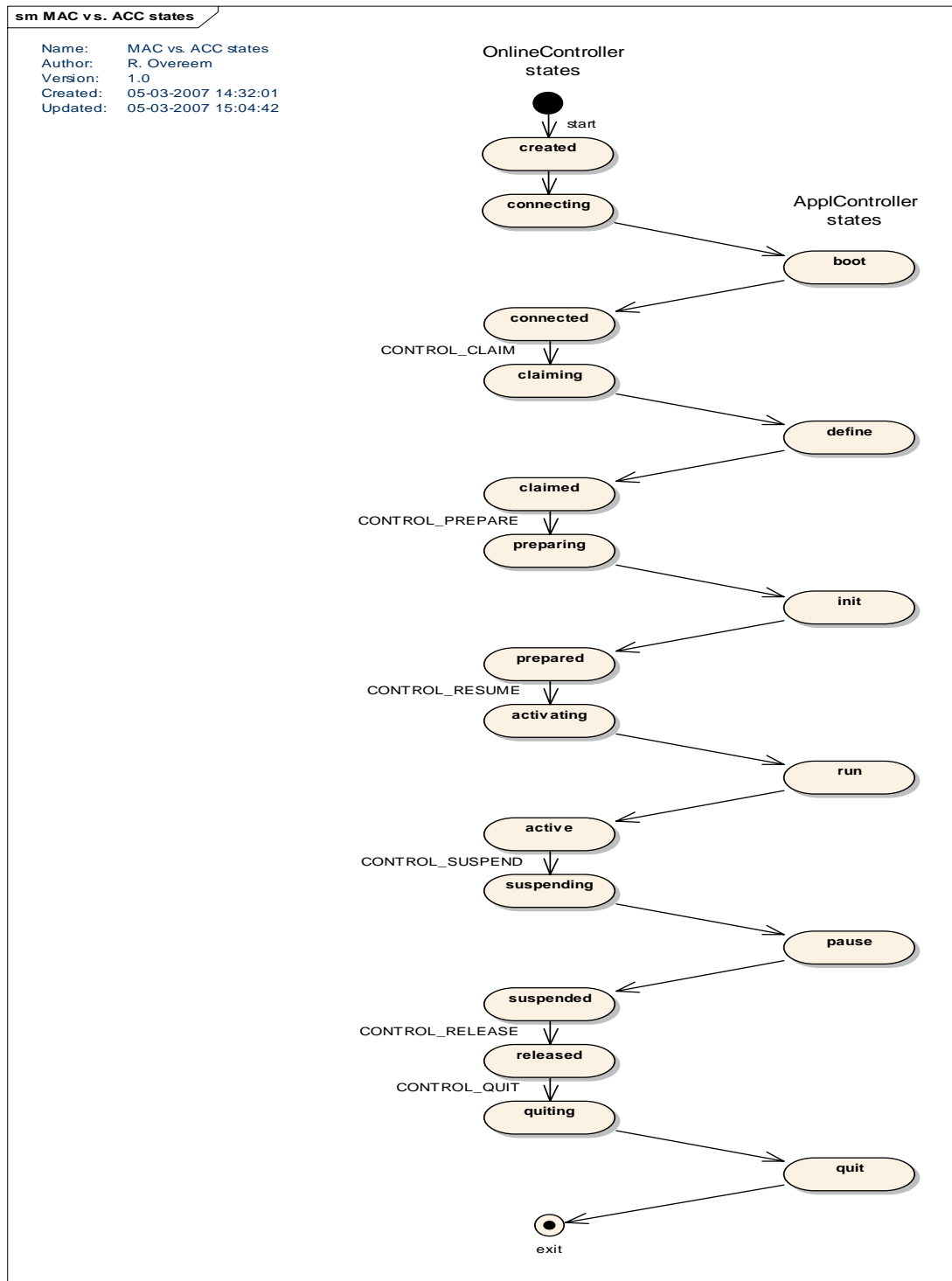



Figure 14: *Controller hierarchy on CEP.*

The states of the MAC controllers can easily be mapped on the states of the ApplController of ACC (see picture on the next page).

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		



There is only one MAC state that is not supported by the ACC states and that is 'release'. So going to the release state will not involve any action with the ACC controller.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Appendix A: PVSS manual abstract

Database structure

The datapoint structures in PVSS are reality-oriented. Process variables that belong together from a logical point of view are combined to form hierarchically structured datapoints. This means that in PVSS a valve is also addressed internally as a valve, an engine remains an engine. A datapoint is not just an individual piece of information but rather a related package of individual information. The individual process variables are grouped in an object-oriented approach and describe a particular component in its entirety.

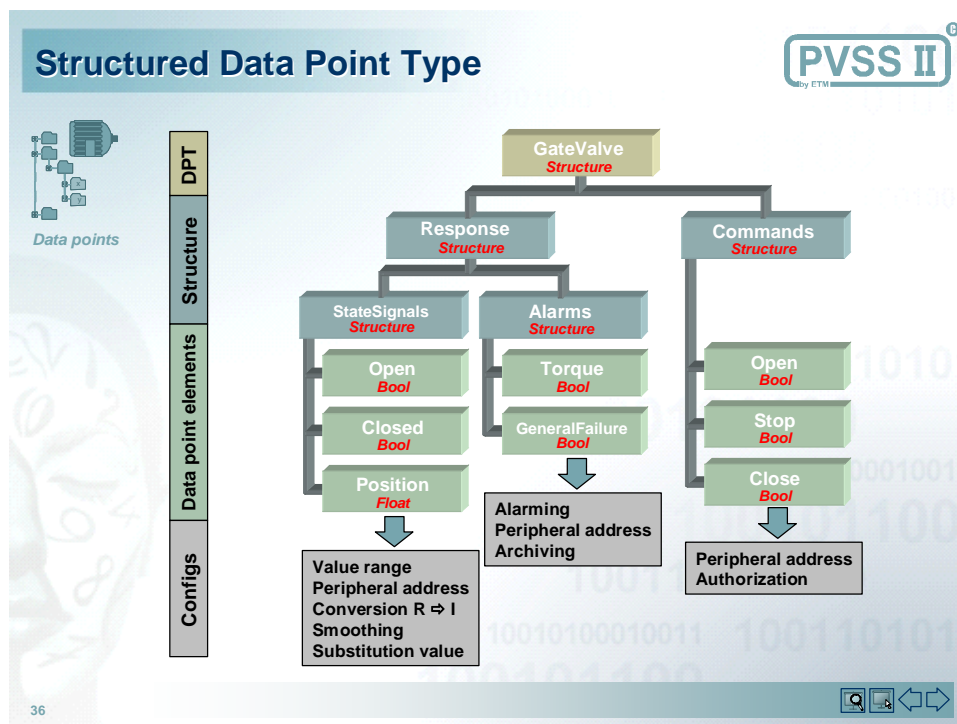



Figure 15: Structured datapoint types of PVSS.

Datapoint type	Structure of information that belongs together. E.g. a GateValve, a Receptor or a Correlator.
Datapoint	Instance (of a certain Datapoint type).
Datapoint elements	Basic types, forming the leafs of a datapoint.
Configs	Attributes of the datapoint elements. Via these configs, automatic transformations can be performed.

The PVSS database contains a set of datapoints as illustrated in the following picture.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

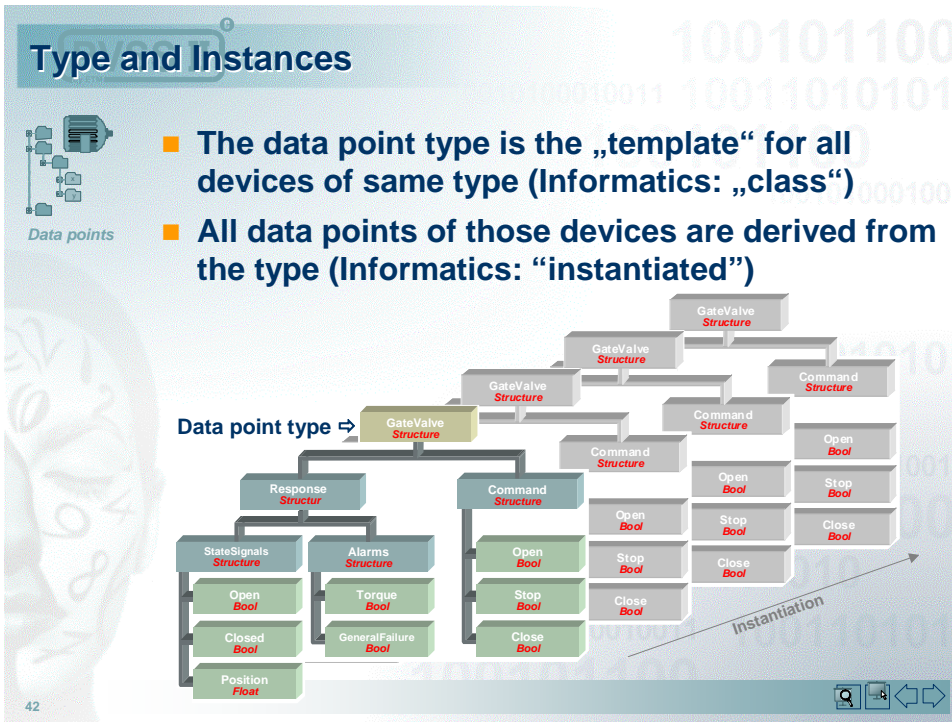


Figure 16: The PVSS database contains instances of datapointtypes.


Datapoint configs

Configs are items that may be assigned to datapoint elements in PVSS. These items are defined with the aid of config attributes that carry the actual datapoint values.

With the aid of configs, you can convert the original values of a datapoint variable, replace them with predefined values or add an alert handler. You can smooth out and archive changes in values of variables. Addresses with which datapoint elements are connected with the periphery are also assigned to variables using a config.

The following configs are available in PVSS:

Peripheral address	Links a PVSS datapoint variable to the peripheral device. The expression for the peripheral address depends on the driver used.
Alert class	Configurable properties of the alert ranges of datapoint variables are summarized in an alert class. There are five predefined alert classes available.
Alert handling	You can define alert ranges that are subject to configurable alert handling.
Archiving	Settings for archiving process statuses, datapoint values, etc.
Authorization	Authorization level settings for supervising user permissions and controlling actions.
Command conversion	Algorithms for converting engineering values into raw values
Default value	Default value handling: if a value is invalid then a default value can be set in various ways.
DP function	Mathematical functions that are run automatically for recalculating results when values are changed. Includes statistical functions for calculating means, minima, maxima etc.
Message conversion	Algorithms for converting alert values from raw into engineering values
PVSS value range	Global value range of a DP variable. A value must always lie in the PVSS value range.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Filtering	Parameters can be set for various filtering algorithms (time-based, value-related, edge-dependent, ...). Reduces the communications time and the amount of data.
User value range	A personal value range can be defined for each user.

Manager structure

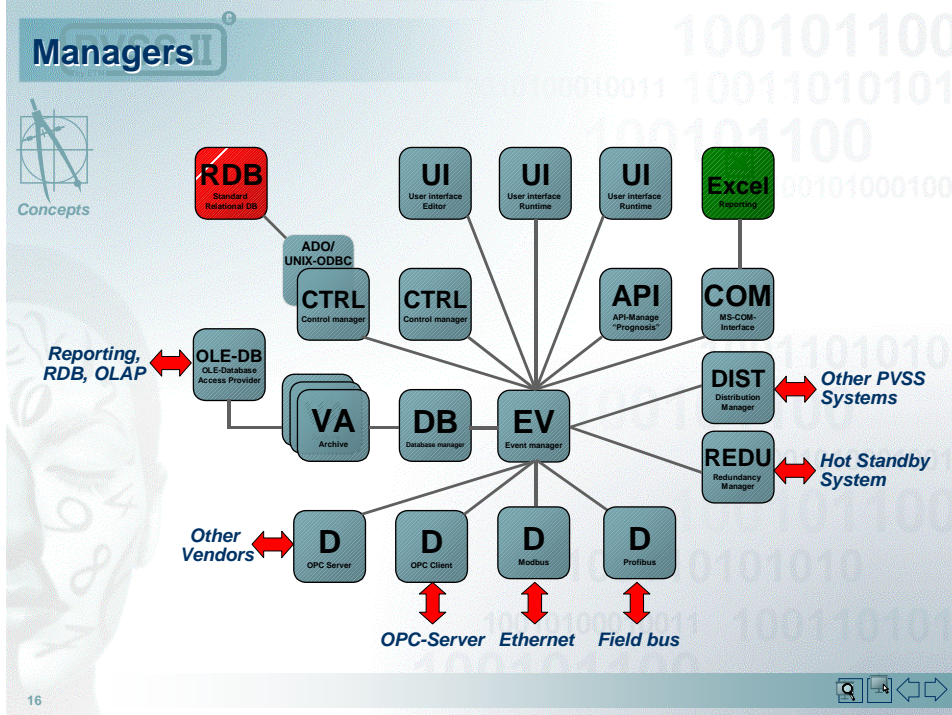



Figure 17: PVSS consists of many different manager-modules.

PVSS is designed as a distributable system. The individual tasks are carried out by special program modules, the "managers". Communication is carried out in accordance with the client/server principle.

EV	Event Manager	Central part, receives telegrams, evaluates and distributes them.
D	Driver Manager	Interface of PVSS to the periphery (PLC, remote control system).
DB	Database Manager	Saving process changes in a high-speed database. Archival of data and creation of evaluations and reports.
CTRL	Control Manager	Is PVSS's own programming language that can concurrently process several functional blocks event-driven and with multitasking capability. It is used amongst other things for the creation of more complex control functions or for testing and simulation tasks.
API	Application Program Interface	The interface for the integration of external programs. Existing software can be integrated via class libraries. Implementation of own managers for industry-specific solutions.
UI	User Interface Manager	Takes care of the visualization of process statuses and forwarding of user input.
DIST	Distribution Manager	Takes care of communication between multiple distributed PVSS systems.

Distributed systems

Distributed systems in PVSS allow to connect two or more autonomous PVSS systems via a network. Each subsystem of a distributed system can be configured either as single-station system or multiple-station system in each case redundant or not redundant. A sub system means in this connection a server on which

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

an Event manager is running on (this means not necessarily a complete project). In a redundant system both redundant running servers are considered as one system.

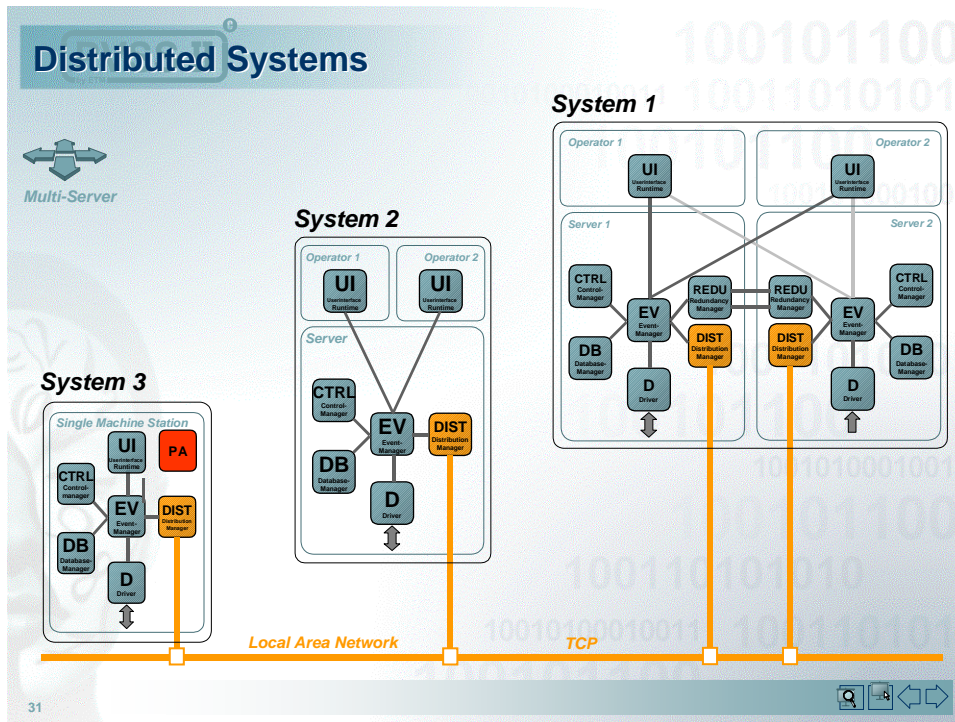



Figure 18: Example of distributed PVSS systems.

The figure shows in detail three systems. System 3 is a single station system, system 2 a multiple-station system and system 1 is configured redundant. All three systems are connected via the local network. The network connection to the three computers used in this distributed system can be redundant. The three systems may have an own process connection.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

Database parameterization and maintenance

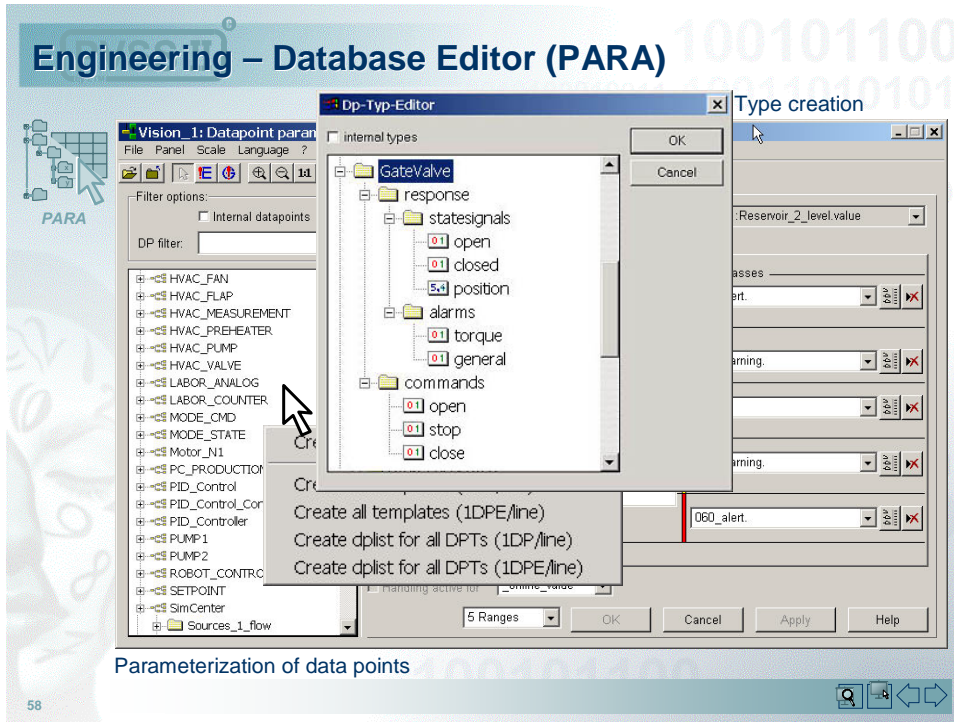


Figure 19: Screenshot of the database editor of PVSS.

The module Native PARA of the PVSS User Interface Manager is a graphical interface for editing datapoint types and datapoints. This interface constitutes a tool. With the tool you can access the internal database and make modifications simultaneously. The datapoint type with its structure serves as a template for the similar datapoints derived from it. Datapoints in process-control system conform to requirements of concrete counterparts of a system. This means that an engine or a valve contains a structure, which conforms to requirements of a particular engine. The structure for a particular engine type or valve type is the datapoint type.


Creating datapoint types in the PARA module means creating a tree structure. Nodes and leaves can be easily added to and removed from the structure by clicking the mouse. The datapoint type is the "mother" of all the datapoints derived from it.

For setting config attributes, standard panels are displayed. All possible alternatives of an allocation are prepared via radio- and checkboxes. The parameterization of single configs on the datapoint elements takes place via these panels

Mass-parameterization

Mass parameterization in PVSS redefines the fundamental principles of setting up a project and subsequent parameterization of the huge range of datapoints is redefined by. It demonstrates perfectly the full advantage of the flexible datapoint concept of PVSS with its object-oriented approach.

The requirement to implement large systems on the master computer quickly and with the minimum number of parameterization steps is also a major factor in automation engineering, reflected in the mass parameterization facility.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

From the user viewpoint, mass parameterization offers in many situations a significant simplification in the creation of datapoints in the system. Working with the mass parameterization feature should not present any major difficulties to the experienced or novice PVSS user alike. Clear dialog windows, tips and warnings help the parameterizer when working with PVSS and mass parameterization.

The master datapoints form the central element of the mass parameterization facility, holding the complete parameterization settings (Configs containing the set parameters). These settings made for the master datapoint are automatically transferred to each new datapoint of the given datapoint type. Changes to the master datapoint are also transferred to the datapoints at runtime.

By defining templates, one can specify which parameterization settings subsequently need to be made for every instance, and which are always adopted irrespective of the master datapoint (datapoint type).

The use of PowerConfigs - user-definable "virtual" Configs allowing the combination or specialization of a number of Configs - is a further step towards modern "mass engineering". All the settings for two or more Configs can be made using e.g. a parameterization screen. Parameter calculations enable, for example, automatic generation of the correct peripheral address or descriptive texts on the leaf elements. A number of predefined PowerConfigs have been supplied with the software.

The Native PARA module also provides a facility for creating, deleting or copying multiple data points (the full parameterization is derived from the master datapoint).

ASCII-Manager

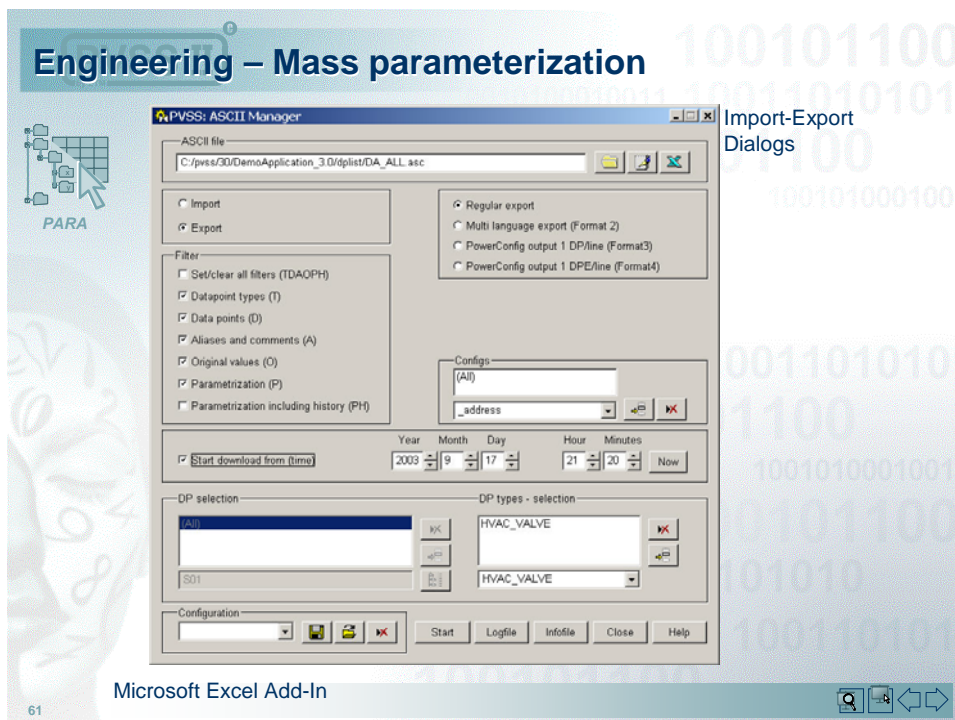



Figure 20: Screenshot of mass parameterization.


Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

The ASCII Manager allows you to input and output and thus edit datapoints and datapoint types and to parameterize datapoints. The data are passed to the ASCII Manager in an ASCII file (hence the name).

There are a huge amount of filter-options which can be used to control the import and export of data.

Examples are:

- Export only database modifications which are younger than a certain date/time.
- Export only datapoints with a certain name (wild-card can be used).
- Export only datapoint types
- Export only parameterization.

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		


Appendix B: Requirements compliance

This chapter summarizes to what requirements [2] the MAC subsystem is compliant. In the tables below the following codes are used to indicate the compliance:

Code	Meaning
Y	MAC is already compliant.
y	MAC will be compliant.
?	Requirement is not clear
N	MAC is not compliant.
...	Solved in another subsystem.

LO-3.08.1 Monitoring and Control function

Nr.	Requirement	Compliant
01	LOFAR shall provide a single distributed monitoring and control function.	Y
02	The monitoring and control function shall ensure that all parts of the system work together coherently.	Y
03	The monitoring and control function shall exclude control dependencies that are not necessary from a functional point of view; in particular remote stations should be capable of autonomous operation for periods of at least an hour.	Y
04	The monitoring and control function shall ensure that failures in hardware, software or signal transport are detected.	y
05	The monitoring and control function shall take autonomous action to compensate for failures where possible.	y
06	The monitoring and control function shall give users transparent and hierarchical access to the instruments functions and parameters.	Y
07	The monitoring and control function shall provide layers of security and access control.	Y
08	The monitoring and control function shall be designed to operate the instrument fully remotely, with options to grant full access to part of the instrument (incl. central processor) to sufficiently qualified users.	Y
09	The monitoring and control function shall provide a subfunction that will calculate and report performance monitoring data to users.	y
10	All LOFAR subsystems shall provide monitoring data to the monitoring and control function (for performance monitoring and closed-loop control functions)	Y
11	The monitoring and control function shall provide for a long-term logging subfunction with workflow support for the Operational Team and with sufficient information to relate system events to artefacts in the data.	SAS
12	It shall be possible to monitor and control the execution of multiple observations in parallel.	Y
13	In case of failure of the principal monitoring and control network, those functions required to analyze the network failure and recover from the failure, shall continue to work using an alternative communication path.	y
14	It shall be possible to change the value of parameters during the acquisition provided these parameters do not have impact on the required acquisition or processing resources (e.g. changing beamdirections shall be possible, changing the number of beams shall not be possible).	y
15	The monitoring and control function shall provide statistical information (on what timescales and in how many directions) on the RFI environment, weather conditions and ionospheric behavior to other system functions (in particular to the specification	y

Author: Ruud Overeem Edzer Lawerman	Date of issue: 2007-03-30 Kind of issue: public	Scope: MAC Doc.id: LOFAR-ASTRON-ADD-005	
	Status: draft Revision nr: 3.1		

	and scheduling function).	
16	It shall be possible to abort an observation if monitor parameters exceed user specified limits (including RFI mitigation performance indication parameters).	Y
17	It shall be possible to control RFI mitigation processes in which multiple stations have to cooperate or in which stations and the central site have to cooperate.	?
18	The monitoring and control function shall provide the communication, storage and processing resources required for the station level RFI mitigation processes.	?
19	The monitoring and control function shall provide sufficient information on the digitization thresholds and offsets for all systems (TBD) to enable reconstruction of the noise characteristics.	Y
20	On the fly calibration information shall be available to the monitoring and control function.	Y

LO-3.08.2 Control requirements

Nr.	Requirement	Compliant
01	LOFAR shall have a control system that actively controls all system settings in the instrument.	Y
02	The control system shall be capable of autonomously calculating system settings in response to changes in instrument status, environment or measurement results.	Y
03	It shall be possible to activate the calculated system settings either automatically (autonomous control) or after explicit confirmation by the operator (manual control).	Y
04	It shall be possible to specify when settings should be activated automatically and when they need to be confirmed by the operator.	SAS
05	It shall be possible to synchronize all timing reference equipment of stations cooperating for an observation.	Y
06	It shall be possible to control the flow of measurement data in the data transport network.	WAN?
07	The control system shall manage the allocation of system resources (acquisition, processing, storage e.g. to avoid conflicts between simultaneous observations).	SAS
08	It shall be possible to receive and accept updated schedules before the end-time of the currently active schedule has expired.	Y
09	The control system shall provide a hierarchical view on the physical system.	Y
10	The control system shall provide a hierarchical view on designated logical control concepts, like observational modes, beams, subbands.	Y

LO-3.08.3 Monitoring requirements

Nr.	Requirement	Compliant
01	It shall be possible to consolidate monitoring information to produce high-level monitoring information from low-level monitoring information.	Y
02	Subsystems shall report completion of actions to MAC	Y
03	It shall be possible for all user roles to produce summarized historical monitoring information.	y
04	The measurement data flow shall be augmented with the result of control decisions that have influenced the data flow at the position in the data stream where the control decision comes into effect	not <i>in</i> the data flow
05	It shall be possible to consolidate monitoring information both on the physical instrument status and on designated logical concepts like observation, correlator.	Y