



Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## Specification Administration & Scheduling Architectural Design Document

<b>Verified:</b>			
Name	Signature	Date	Rev.nr.

<b>Accepted:</b>		
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	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## Distribution list:


---

<b>Group:</b>	<b>For Information:</b>
ASTRON	

## Document history:

---

Revision	Date	Section	Page(s)	Modification
1.0 draft 1	2005-10-04		-	Creation
1.0 draft 2	2005-10-07	All		Incorporated review comments from K.v.d.Schaaf and E.Lawerman.
1.0 draft 3	2005-10-13			Add paragraph about database size.
1.1	2007-03-25	All		Update for CDR2007.
1.2	2007-03-30	All		Update after internal review.


Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

### Abstract

To be able to do measurements with the LOFAR telescope every part of the telescope needs the right settings at the right time. Some settings like the telescope pointing must be known in many parts of the telescope, other settings are specific for one process or one antenna.


The Specification, Administration & Scheduling package is responsible for managing these settings from design stage when the developer designs a part of the telescope software until post-observation time when the astronomer interprets the observation results.

This document describes the architectural concepts and design choices for the Specification, Administration & Scheduling package. It gives the reader insight in the kind of information that is handled by the package, how it is managed and how this is interfaced to the users of the SAS package.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

**Table of contents:**

1	Introduction.....	5
1.1	Purpose of this document.....	5
1.2	LOFAR subsystem overview.....	5
1.3	Document reference.....	5
1.4	Definitions.....	6
1.5	Acronyms and Abbreviations.....	7
1.6	Document overview.....	7
2	Top-level functional view.....	8
3	Chronological view.....	10
3.1	The configuration data.....	11
3.2	Result data.....	12
3.3	Offline dataflow KVT data.....	14
4	SAS database architecture.....	15
4.1	Observation specification.....	15
4.2	Physical Instrument Configuration.....	16
4.3	Metadata changes.....	16
4.4	User rights.....	16
5	Observation trees.....	18
5.1	Value validation.....	19
6	SAS software architecture.....	21
6.1	OTDB architecture.....	21
6.2	OTB architecture.....	23
6.3	Scheduler.....	25
7	Deployment.....	28
8	Subsystem relations.....	29
8.1	NorthStar/MoM.....	29
8.2	MAC.....	29
8.3	SHM.....	30
8.4	CEP.....	30
9	Open ends.....	31
	Appendix A Requirements compliance.....	32

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 1 Introduction

### 1.1 Purpose of this document

This document describes the architectural concepts and design choices for the Specification, Administration & Scheduling subsystem. It gives the reader insight in the kind of information that is handled by the package, how it is managed and how this is interfaced to the other subsystems and the users of the SAS package.

### 1.2 LOFAR subsystem overview

The Specification, Administration and Scheduling subsystem manages all metadata of LOFAR that is required to specify and execute observations. During the observations it collects the statistics and records the value changes of the metadata.

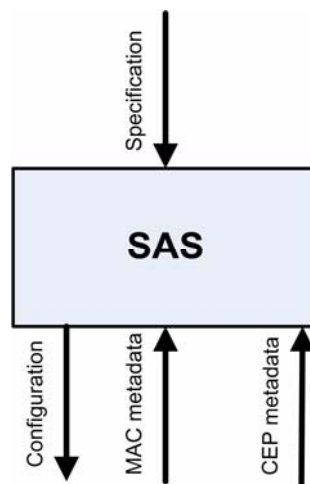



Figure 1: System overview of the environment of the SAS package.

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. SAS gets the basic observation information from the MoM (Management of Measurements) subsystem that is used for other astronomical instruments as well. Once an observation is scheduled the MAC (Monitor And Control) subsystem assures that the observation is executed and the signal data is sent to the CEP (central processing) subsystem that provides further signal processing and contains functionality for image, pulsar post-processing and much more. System Health Management (SHM) needs the metadata of SAS for e.g. trend-analysis.

### 1.3 Document reference


1. Specification and Scheduling Subsystem, Architectural Design (LOFAR-ASTRON-ADD-008)  
Author: H. de Wolf; dd. 2002-09-27; 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. LOFAR SAS, use-case overview (LOFAR-ASTRON-UCD-008)  
Authors: E. Lawerman, dd. 2005-08-26, Draft 0.4

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

4. LOFAR Common Database Service, Comparing database products  
Author: R. Overeem; dd. 2003-07-23; Rev. 1.1
5. MAC Architectural Design Document (LOFAR-ASTRON-ADD-005)  
Authors: R. Overeem, E. Lawerman, dd. 2007-03-30; Rev 3.1
6. MoM-OTDB, Protocol description  
Author: R. Overeem; dd. 2005-11-21, Draft 1.0
7. Observation Tree Database, Global Design (LOFAR-ASTRON-SDD-044)  
Author: R. Overeem; dd 2005-10-10, Draft 1.0

## 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.
Board	See Review Board.
Instrument Scientist	A user with detailed technical knowledge of the telescope. He is responsible for producing the operational schedules and maintaining the observation templates.
Key-Value-Time	A combination of three values that defines a value-change of a parameter on a certain time.
Management of Measurements	A browser-based package that is used to assist the scientist in the process of specifying an observation request at an astronomical level and tracing the progress of observation projects.
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.
NorthStar	A browser-based application that is used by the scientists to prepare proposals containing observation requests. It also supports the board in the assessment of observation requests and allocation of the LOFAR resources.
Observation	A description that specifies a set of measurements and subsequent processing steps to be made by the telescope. An observation description is an instantiation of an observation template.
Observation Parameter	See Parameter.
Observation Request	An observation proposal that has been submitted to the review board for approval.
Observation Tree	A collection of parameters and placeholders for parameters that describe how the LOFAR telescope must be initialised in order to do an observation.
Observation Type	Type of observation, e.g. EOR or All Sky Monitoring. Determines to a large extent what kind of processing must be done on the received signal data.
Operator	A user that verifies the execution of the observations. He is allowed to modify the instrument in order to get better observation results.
Parameter	Placeholder for a value, used in generic descriptions.
Parameter Set	A collection of parameters that contains the initial values for the parameters of a software module or application.
Component Description File	A file that contains the definition of the parameters for one software module or application.
Review Board	A body that controls the allocation telescope resources. It reviews the observation requests submitted by the scientist and determines whether these are to be executed on the telescope and their priority.
SAS Scheduler	A module that is capable of planning observations taking into account all the constraints for these observations.
Scientist	End user of the telescope. A scientist is thought to be familiar with the scientific

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

	concepts of the telescope, but not with its exact inner workings.
Stored Procedure	A piece of code that is kept in a database. The code can be called from inside the database or outside the database. Under certain circumstances the code can be stored in a compiled form.
Template tree	A minimized observation Tree dedicated for one specific observation type.
Virtual Instrument	A specification that describes which part of the physical instrument must be used during an observation.


## 1.5 Acronyms and Abbreviations

CDef	Component Definiton
CEP	CEntRAL Processing
COTS	Common Of The Shelve
DMZ	De-Militarized Zone
HTTP	HyperText Transfer Protocol
JNI	Java Native Interface
KVT	Key-Value-Timestamp triple
MAC	Monitoring And Control
MoM	Management of Measurements
OTB	Observation Tree Browser
OTDB	Observation Tree DataBase
PIC	Physical Instrument Configuration
PVSS	Prozessvisualisierungs- und Steuerungs-System
RFI	Radio Frequency Interference
RMI	Remote Method Invocation
SAS	Specification, Administration And Scheduling
SHM	System Health Management
SQL	Simple Query Language
SR	Schedule Request
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
VIC	Virtual Instrument Configuration

## 1.6 Document overview

The document is organized in the following way:

- Chapter 2 gives a functional description of the components that make up SAS. It gives the reader a rough idea about the interactions SAS has with other (sub)systems and users.
- In chapter 3 the document describes the scope of the data of the SAS database and explains the dataflows in SAS.
- In chapter 4 the architecture of the SAS database is described.
- Chapter 5 gives an example of how the program parameters and metadata are structured in a hierarchical tree.
- The software architecture of the SAS modules is described in chapter 6.
- Chapter 7 gives a diagram of the deployment of the modules.
- Chapter 8 finally summarizes the interfaces SAS has with the other subsystems.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 2 Top-level functional view

Before we zoom in to the architecture of SAS it is important that we know the basic definition of a LOFAR program. Every program receives during its start-up a parameterfile that contains all relevant information for that program to perform its task. During its execution the program produces logging of at least five severity levels of which four are meant for the operator. Every program can also produce metadata that is relevant for the processing of the received signal-data.

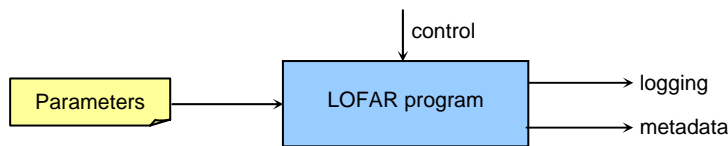


Figure 2: Basics of every LOFAR program.

The main target for the SAS subsystem is to make it possible for the users to specify an observation together with a 'virtual' instrument and finally to schedule it. In doing so SAS gathers enough information to create the parameterfiles for every program needed in the observation.

During runtime these programs produce metadata that might be important for the processing of the signal data. This metadata consists of three values: parametername, value and timestamp. Since SAS already knows most of parameternames and has definitely already most of the functionality to store this metadata, this information is also stored in SAS.

Keeping in mind that SAS is responsible for (startup)parameters, scheduling and metadata we can make the following (non-UML) picture of SAS.

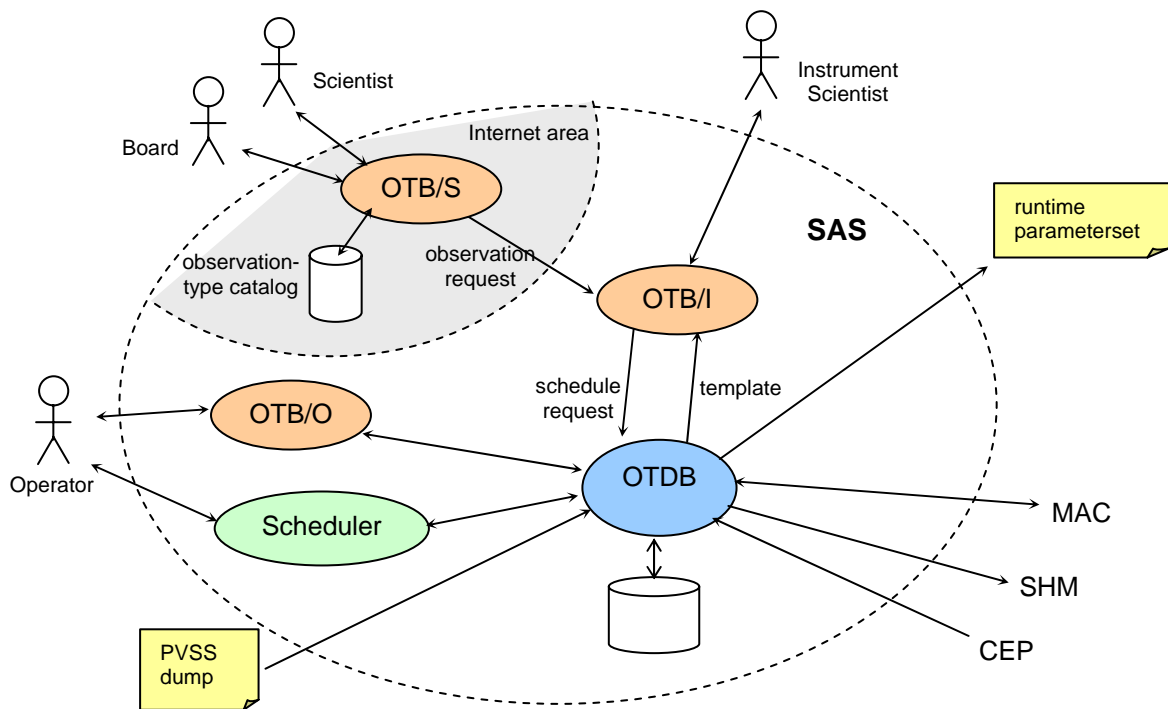



Figure 3: Components of the SAS package.



Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

SAS contains three major modules:


- The heart of the SAS package is the Observation Tree DataBase (OTDB) that contains the configuration data and the collected metadata of the whole LOFAR instrument. Almost all interactions of the users will affect the contents of the OTDB. The OTDB acts as an intermediary between the different users.
- All interfacing to human users is defined in the Observation Tree Browser (OTB). Since every user needs its own specialized view on the configuration-data several flavours of OTB are defined: OTB/S for scientific view, OTB/I for instrument view and OTB/O for operational view. Notice that all these flavours of views can be realized in one program that offers several views on the same data.

Note: It is very likely that the interfacing with the scientist and review board will be done with NorthStar and MoM. Till then it is also possible to specify the observations with OTB.

- The scheduler is the most complex module in SAS, it maps the observation requests to the real hardware of the instrument assuring that the hardware is available at that time.

SAS has interactions with CEP, SHM and MAC who need access to the configuration- and metadata for operating the instrument and processing the signal data.

So the database of SAS contains various kinds of data like observation specifications, Instrument layout, startup parameters, metadata and of course access rights to manage the access to the information. Except for the latter all this information is presented to the user as tree-structures, e.g. VIC trees contain the observation specifications and the used virtual instrument; PIC trees represent the Physical Instrument Components. See chapter 5 for an example.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

### 3 Chronological view

Since the information in the OTDB guides the complete process from observation-request till data processing there is a certain chronological dependency in the data of the OTDB. In the following picture the process is split into four stages: development, scheduling, observation and analysis.

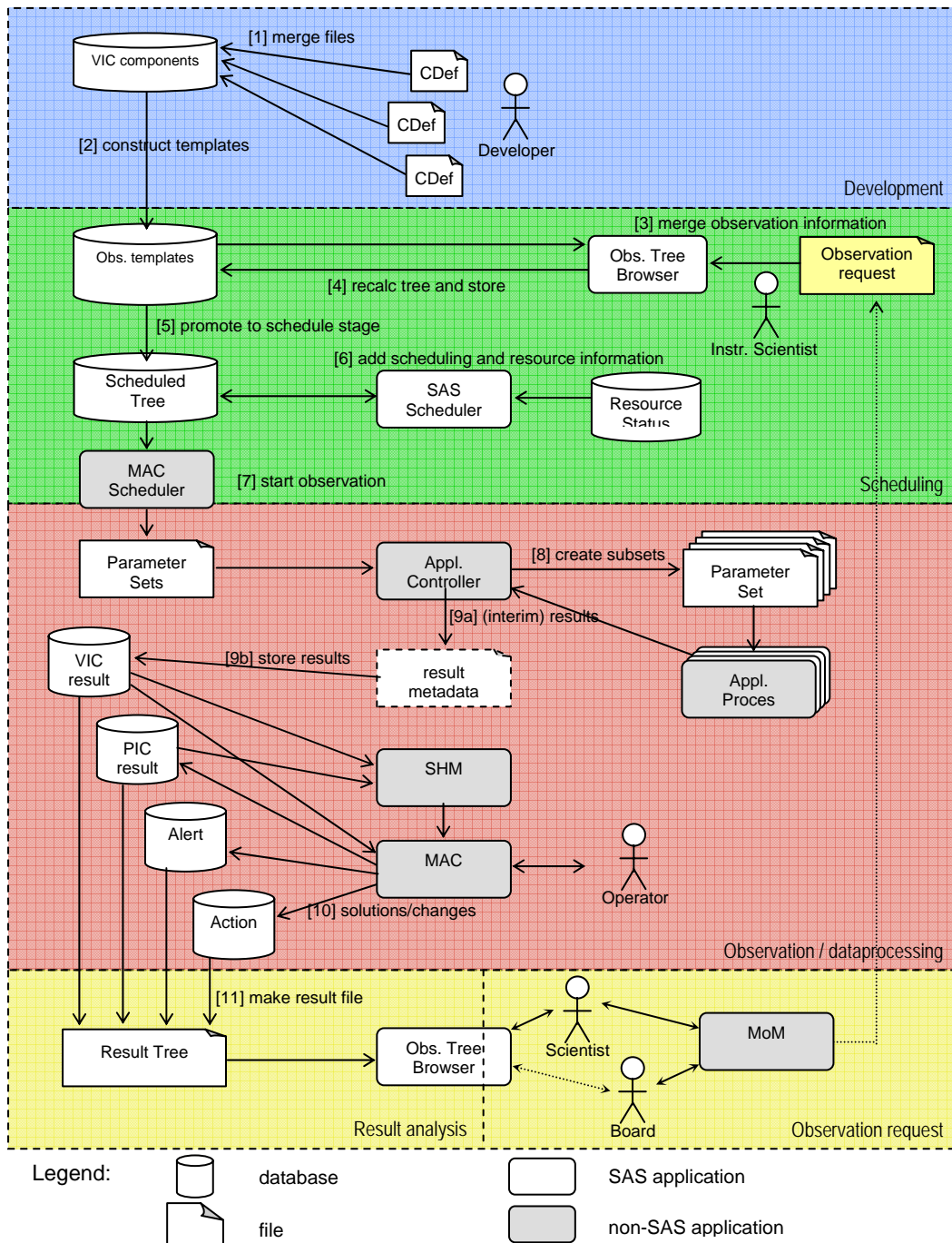



Figure 4: Simplified chronological view on the configuration- and result-data.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

The SAS package has to support the follow chain of interactions:


1. The developer makes one or more Component Definition files in which he specifies which values are observation dependent in his program. This can be a start-up variable for a program or the metadata datapoints the program will deliver during runtime. These CDef files are loaded in a database.
2. Using the building blocks from the developer the instrument scientist can build observation templates that reflect the configuration of the instrument for a specific type of measurement (e.g. pulsar, EOR, etc).
3. The instrument scientist uses these observation templates to fill in the observation request information from the scientist.
4. The results are written back into in the database.
5. When the instrument scientist is ready he releases the tree for the SAS scheduler.
6. The SAS scheduler tries to find the optimal observation time for each observation and fills in the start- and stop-time of the observation and the resources that will be used. When the scheduler is ready the observation tree is marked 'scheduled'.
7. When an observation tree is marked 'scheduled' it will be visible for the MAC scheduler. MAC will ask SAS to create the parameter sets for (sub)trees of the configuration, distribute these to the right machines of the LOFAR infrastructure and finally start the observation.
8. When an observation starts, the parameter set files are split into parameter subsets that contain the settings for one process or module. This is done by the Application Controller from the ACC package.
9. During the observation the various parts of the LOFAR telescope produce interim results (metadata) and alerts. This information is also stored in the observation database. These results consist always of the triple: parametername, value and timestamp.
10. Some value changes result in an alert. The operator who is monitoring the observation can react to these alerts e.g. by modifying parameters in MAC. This information can also be important for processing the signal data so it is stored as metadata in SAS as well.  
Note: An 'alert' in MAC is a serious event to which the operator *must* respond.
11. Finally, after the observation is finished, all observation configuration information together with the metadata is made available to the scientist who can use it in its analysis of the results.

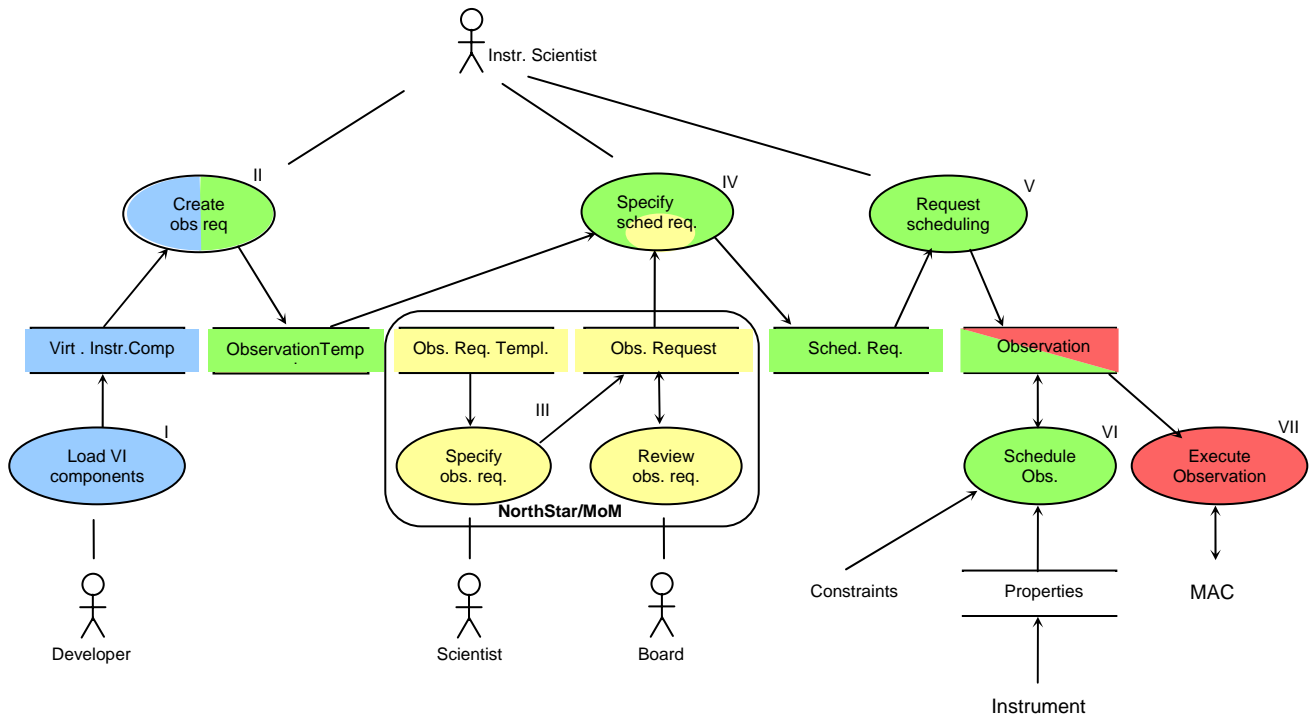
From Figure 4 it is clear that there are many data-flows running through the databases of SAS. The following paragraphs zoom in onto the interactions on these database types.

### 3.1 The configuration data

The configuration data are the parameter settings entered by the scientist, the instrument scientist and the scheduler. Their purpose is to specify how the instrument must be tuned and how the signal data must be processed to obtain the desired results. The data covers the whole instrument including the central processing units. The parameternames and types are compatible with the instrument- and process- trees used by MAC to simplify the exchange of information.

The following diagram shows a more detailed dataflow of the specification part of an observation:

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		




Note: The colours are in accordance with those used in Figure 4.

Figure 5: Dataflow diagram of the configuration data.

- I. The developer designs the definitions of the buildingblocks (=components) and loads them into the component-storage.
- II. The instrument scientist builds (reusable) virtual instrument / observation templates for specific types of observations from the building blocks.
- III. The scientist creates an observation request to specify the observation he wants to do with the instrument. After the board has approved the observation request the information is sent to SAS. Note: It is also possible the create observation requests with the user interface of SAS in stead of NorthStar/MoM. This is to provide required flexibility in scheduling during testing.
- IV. The instrument scientist takes an approved observation requests merges it with an observation template and creates a schedule request.
- V. When the instrument scientist is ready with filling in the schedule request he instructs the scheduler to schedule this observation or a group of observations.
- VI. The scheduler assigns a time-window to the observation and assigns the required hardware to the software when the schedule request passes all constraint tests.
- VII. Finally MAC is signalled that an observation is ready to be executed.

### 3.2 Result data

During the execution of an observation another dataflow becomes active. This dataflow is concentrated around values changes(=metadata) of the instrument and in the software. Most of the data will be status-changes like going on- or offline from a piece of equipment but it can also be a regular report from an application about e.g. the quality of the signal data.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

The result data is passed through a filter when it is entered into the database. For certain serious status changes this filter can create alerts that will be shown on the screen of the operator in MAC. The operator *must* respond on these events. The actions he takes on these alert are again stored in the OTDB database to get the metadata complete.

Note that the only relation between the result data and the configuration data is that the parametername of a result datapoint must exist in the configuration tree since otherwise it would be impossible to couple the metadata values to a parameter. When browsing through an observation-tree the registered metadata is available on a separate tab on the screen.

To allow the hardware(drivers) to report changes in the physical instrument the configuration of the physical instrument must be present in SAS. This info exists in the PVSS database of MAC and can be loaded into the SAS database.

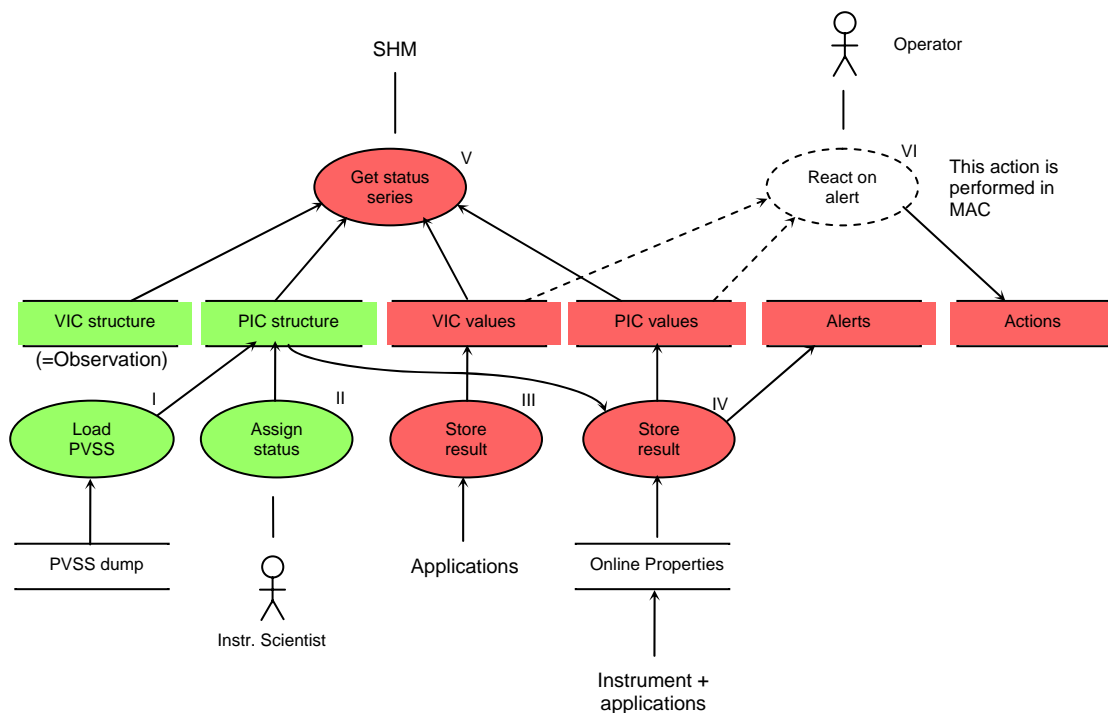



Figure 6: Dataflow diagram during observation.

- I. A dump of the PVSS database containing the structure of the hardware is loaded in the SAS system to make the SAS system familiar with the parameter names.
- II. The instrument scientist must assign the status 'operational' to a new loaded PVSS dump before it can be used in operational observations.

Once the structure of the hardware (Physical Instrument) and the software (Virtual Instrument) are known to SAS the observation can be scheduled and executed:

- III. When applications report results (metadata) to SAS, these are stored in the Virtual Instrument Configuration (VIC) value-storage (Key-Value-Timestamp triples).

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

- IV. In the same way the instrument and controller applications will report status and value changes to the SAS system. These are stored in the Physical Instrument Configuration (PIC) value-storage.
- V. The System Health Management (SHM) subsystem continuously queries these VIC and PIC value-storage in order to detect trend changes and other possible leads to malfunctioning modules.
- VI. Certain value changes will result in an alert the operator should react on, e.g. when a parameter gets outside its defined range and the MAC controllers cannot fix this problem. The Actions the operator takes are also stored in the SAS system.

### 3.3 Offline dataflow KVT data

To complete the interactions with the database the remaining actions are shown in the following diagram:

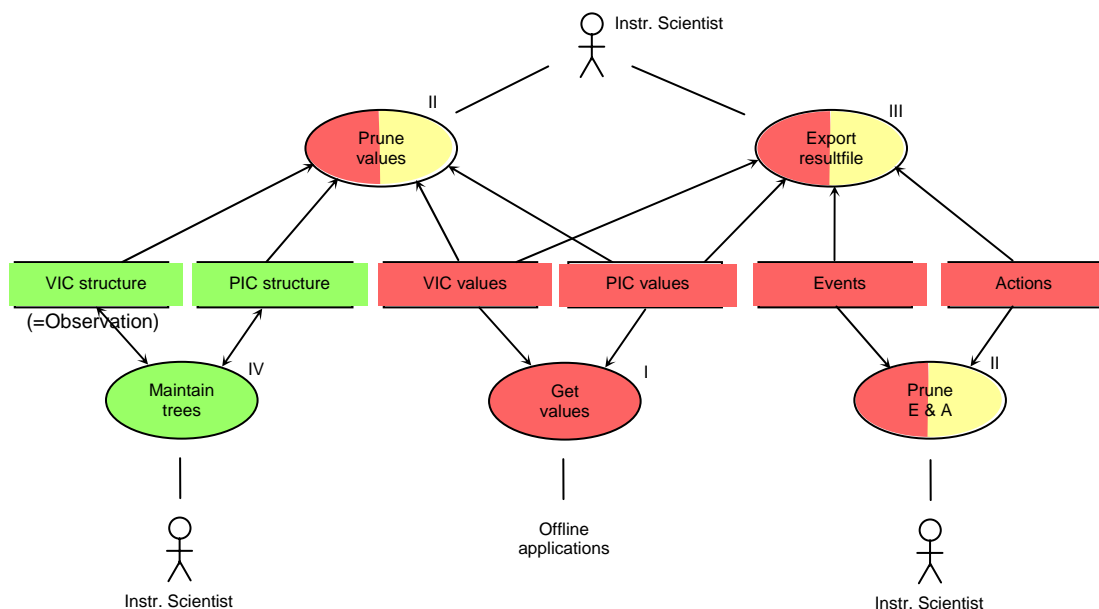



Figure 7: Dataflow diagram after observation.

When the Observation is finished, the configured and collected metadata is still needed during the post-processing phase. Several actions are distinguished:

- I. The imaging process, which runs after the observation is done, may use configuration- and result-data for its calculations.
- II. The instrument scientist can prune the collected metadata when e.g. some kind of information is no longer needed.
- III. The instrument scientist can export the configured and collected data to e.g. a file so that it can be used at a different location.
- IV. The instrument scientist can maintain the VIC and PIC trees. He can make them obsolete, archive them or even remove them.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 4 SAS database architecture

From the previous chapters it is clear that SAS has to be able to store the following types of data:

1. Observation specification
2. Physical Instrument Configuration
3. Metadata
4. User rights

### 4.1 Observation specification

The information of an observation is stored in a hierarchical tree structure (see chapter 5). To maintain such a tree the following four tables are needed:

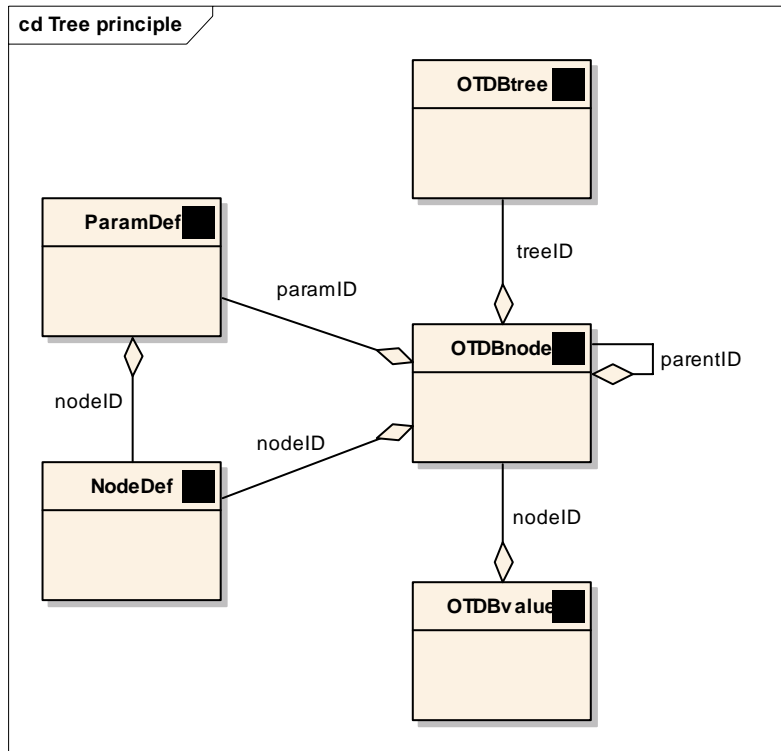


Figure 8: Simplified data-model for storing hierarchical trees.


**Tree:** Table that contains properties about the tree like treeID, owner, schedule times, etc.

**Node:** Represents an element of the tree. It can represent a node (e.g RSPBoard) or a parameter (e.g. temperature). When it represents a parameter it contains the startup value of the parameter, when it represents a node it is just a placeholder for the name of the node. Each node has a reference to its parent-node to be able to register relations.

**Value:** The Key-Value-Timestamp triples are stored here.

**Parameter definition:** The definition of a parameter, contains description, constraints, unit, type, a (reference to a) validation script, etc. Does NOT contain the actual value of a parameter because it is the *definition* of the parameter, the actual value is stored in the node that refers to the parameter definition.



Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

**Node definition:** The definition of the node, contains description and constraints.

## 4.2 Physical Instrument Configuration

The physical instrument hardware is monitored in the PVSS database of MAC which is a real-time database system. So the actual layout of the hardware is always known in the PVSS database. Once LOFAR is fully operation this information will not change much, only replacements can change the PVSS tree, but during the construction phase of LOFAR the PVSS tree will change weekly or monthly.

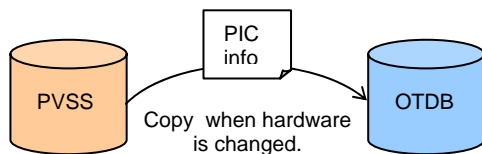


Figure 9: PVSS is leading for physical instrument configuration.

The information in the PVSS database is considered to be up to date. Therefore it is important that SAS uses this same information: The PVSS information can be exported to a PVSS dump file that can be imported into the SAS database so all information originates from one point.

The table model for the PVSS information is the same as the table model for the observation specification that is described in the previous paragraph.

## 4.3 Metadata changes

For various reasons it has been decided that value changes of the metadata that are important for the data-processing pipelines are stored in SAS and not in PVSS:

- SHM will continuously query the historic metadata
- The other metadata information is already in SAS
- Limiting value archives in PVSS keeps PVSS fast and small.
- Single place where metadata is stored.

Storing metadata changes is very simple: storing the parameter-name (Key), Value and Time is enough. To simplify the *queries* on the metadata each KVT triple gets the referenceID of the parameter it belongs to when it is stored in the database.

## 4.4 User rights

As shown in Figure 3 many different types of users access the data in SAS. It's obvious that not every type of user may access all data. For instance, an instrument engineer has more rights than the scientists, and often you don't want that scientist A can access the data of scientist B.

SAS supports the following authorisation model:



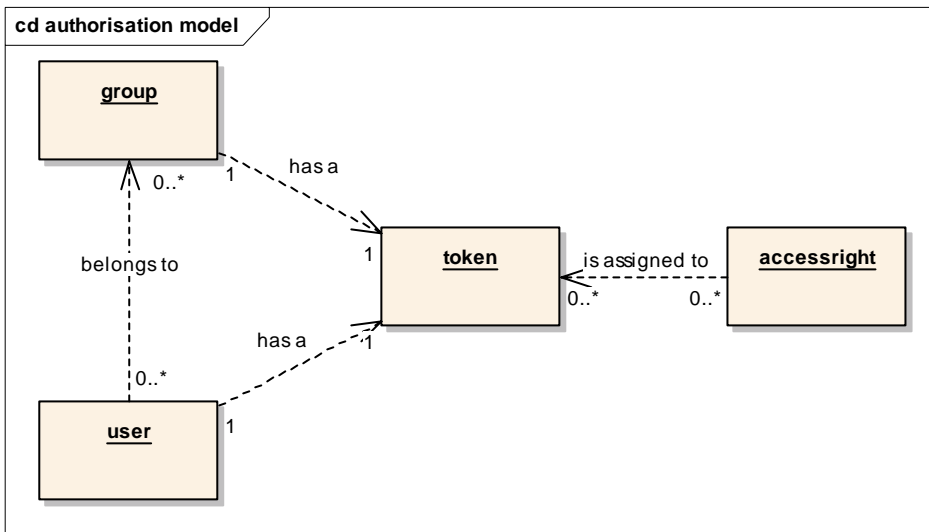



Figure 10: Authorisation model.

Every user can be assigned to zero or more groups. Both users and groups have a unique token to which access rights can be coupled. When a 'critical' stored procedure is executed, it will check with the authorisation module if it is allowed to execute this task for this user on the selected tree.

Beside the user- and group-based protection there is another protection used that is based on the state of a tree. During its lifecycle each tree gets various states[7]. Depending on this state access to the trees is managed.

State \ User	scientist	Board	operator/IS	SAS-Scheduler	MAC-Scheduler
idle	-	-	R/W	-	-
being specified	R/W	R	R/W	-	-
specified	R	R/W	R/W	-	-
approved	R	R	R/W	R/W	-
scheduled	R	R	R/W	R/W	R/W
queued	R	R	R/W	-	R/W
active	R	R	R/W	-	R/W
aborted	R	R	R/W	-	-
finished	R	R	R/W	-	-

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 5 Observation trees

This chapter describes what information is stored in an observation tree and how the validation of the information is done. A very simplified representation of a software- and hardware configuration is used to explain the concept. The parameters used in this example are far from complete, they are exclusively meant as example.

An observation tree (or VIC tree) contains all information to schedule and execute an observation. This means it assigns software and hardware. Assume the following simplified software chain for an observation running on several stations.

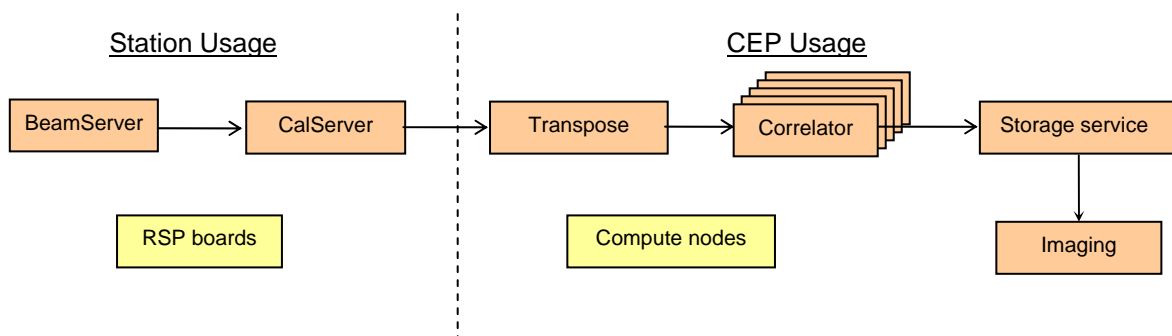


Figure 11: Simplified software chain needed for an observation.

A BeamServer together with a CalServer make a beam with a set of antennas. This beam is transferred to the CEP clusters where it is combined with similar beams from the other remote stations. After the correlation is calculated, the imaging software creates images from the observation. On the stations we need some RSPboards (=antennas) and on CEP we need an amount of compute nodes.

For each element we can describe what resources they will need and what parameters they need to do their job.

Program	BeamServer	CalServer	Station Hardware
Parameters	source (direction)	subbands sampleclock nr antennas <i>calibration_timestamp</i>	k x RSPboard

Program	Transpose	Correlator	Storage service	Imaging	CEP Hardware
Parameters	nr stations subbands beamlets sampleclock <i>%bad_data</i>	nr stations subbands beamlets sampleclock <i>processor_load</i>	MB /sec capacity <i>average_write_speed</i> <i>average_read_speed</i>	subbands source duration	q x Transpose nodes r x Correlator nodes s x Storage nodes t x Imaging nodes

As an example there are also some metadata KVT datapoints in the table above, these are in italic. These 'parameters' are only used using runtime to store the interim metadata results.

The components of the example above can also be presented in a hierarchical tree. It will look like this:

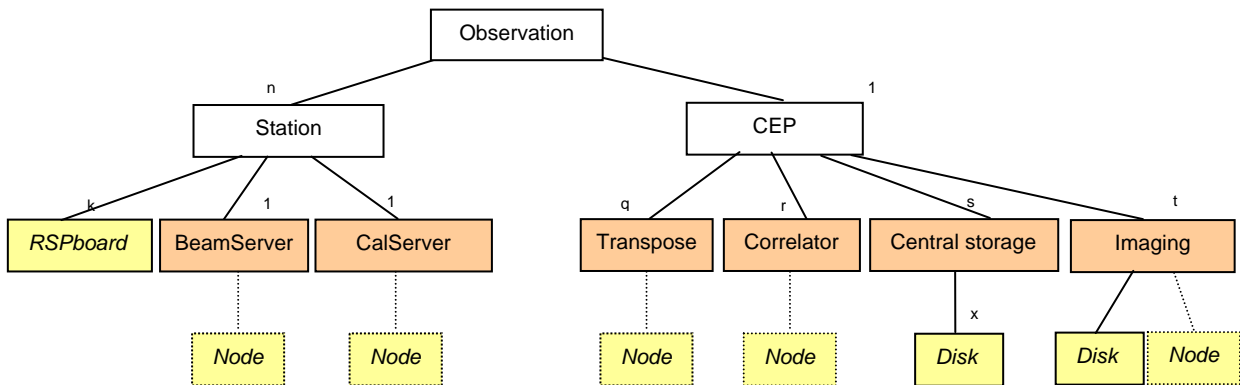


Figure 12: Way to represent the observation as a tree.

Beside the software and hardware components shown in Figure 11 three 'container' nodes appear in the tree (Observation, Station and CEP). These nodes are just for convenience: they make it possible to structure the information in a logical way.

This tree structure still has one shortcoming: the quantities shown by the relations ( $q, r, s$  etc) should not be numbers but should be references to/names of the elements that are used. So instead of  $n$  stations there should be a list of used stations. This is solved by putting all resource allocation in a node called 'virtual instrument'.

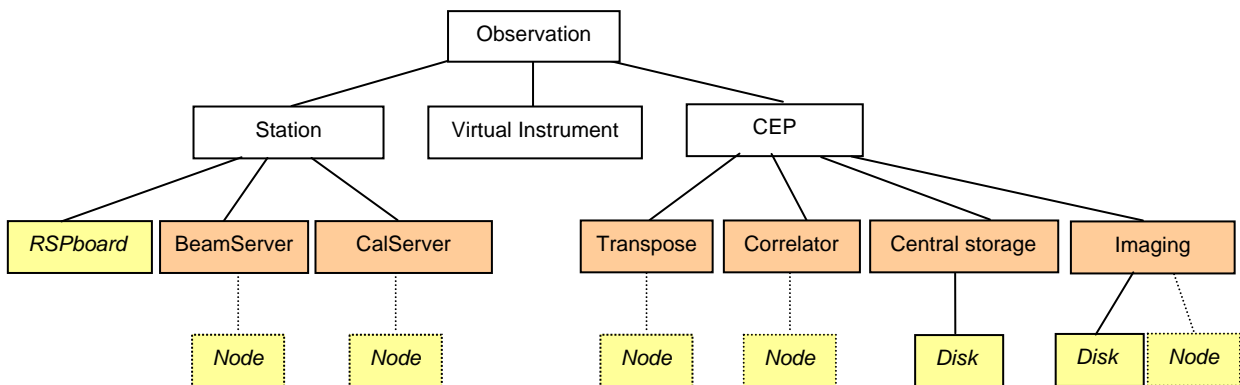



Figure 13: Tree representation of the observation in SAS.

The virtual instrument node will have parameters like: stationlist, RSPboardIDs, Transpose-nodelist, etc. Storing all this resource information together has a great benefit: the SAS scheduler who is responsible for the resource allocation only needs one node of an observation tree to allocate the resources.

## 5.1 Value validation

To guard the input process of the scientist and instrument scientist the input values they enter must be validated. This validation can be split into three different validation-levels:

- parameter level
- node level
- tree level

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

### 5.1.1 Parameter validation

Validating the value of a single parameter is done in the OTB. For every parameter a value range or value-set is defined when the parameter is added to a tree (development/maintenance phase). The OTB gets these restrictions from the OTDB when it asks for the parameter characteristics. When the parameter value is not within range OTB will reject the value.

### 5.1.2 Node validation

It becomes more difficult when there is a dependency between several parameters of one node. It is not desirable to register the same dependency more than once, so multi-parameter validations are stored on node-level. When saving a parameter value the node-validation will also be executed so that the value of the parameter can be checked against the value of the other parameters. This validation can only report a result when all related parameters have a value.


### 5.1.3 Tree validation

The values entered in a node may also influence the integrity of the nodes above. So when a parameter of a node is changed, the node validation is executed. This will result in 'OK', 'ERROR' or 'UNKNOWN'. The result 'UNKNOWN' means that not all validations could be checked because several parameters are not filled in yet.

When the result is 'OK' the node validation of the parent node is called. As long as each validation gives 'OK' the execution is propagated upwards in the tree. When the top-node is 'OK' the tree is OK. The propagation stops as soon as a node validation returns 'ERROR' or 'UNKNOWN'. [7]

### 5.1.4 Validation language

The node- and the tree-validations will be performed on the database server to speed up the validation process. The node validations will be small python [TBC] scripts stored in the database.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 6 SAS software architecture

The architecture of the software layers is not very complex, SAS is just a database with some access layers around it to guarantee data integrity and manage access rights.

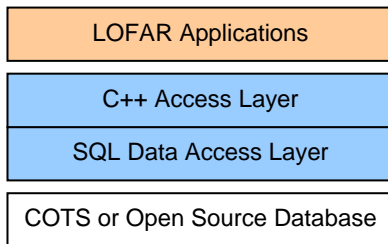


Figure 14: Top-view of the SAS software architecture.

**COTS or Open Source database:** Standard relational database that support user managements, multiple simultaneous queries and stored procedures. After doing tests with the most used open source packages MySQL and PostgreSQL [4] it turned out that PostgreSQL performs better during multiple simultaneous sessions, so PostgreSQL was chosen as database package for SAS.

**SQL Data Access Layer:** This is the most important layer of SAS since it is responsible for data integrity and access management. It consists of a collection of stored procedures to hide the implementation details and guarantee data access.

**C++ Access Layer:** This is the C++ interface for all LOFAR software. Almost every function that this layer offers has a SQL equivalent in the SQL Data Access Layer.

**LOFAR applications:** Applications of LOFAR that need access to the SAS data, e.g. MAC, SHM and CEP.

All modules of SAS are built on top these layers.


### 6.1 OTDB architecture

Figure 15 on the next page shows the main building blocks of the OTDB.

The main functionality of OTDB is implemented in stored procedures in the database using the PL/SQL language. In this way the knowledge of the database tables is isolated in the database. The PL/SQL language is supported by several database providers and is closely related with SQL. Based on the tests described in [4] PostgreSQL is chosen as non-commercial database server.

To access the stored procedures from a C++ program a thin libOTDB library is made that acts as a wrapper. Subsystems like MAC and SHM use this layer to access the SAS data. Based on the same libOTDB interface it is possible to define a JNI layer so that the functions also become available within Java, which is necessary for the OTB application.

LibOTDB itself lays on libpq and libpqxx. Libpq is a C based interfacing library for PostgreSQL. Libpqxx is a small wrapper to offer a C++ interface.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

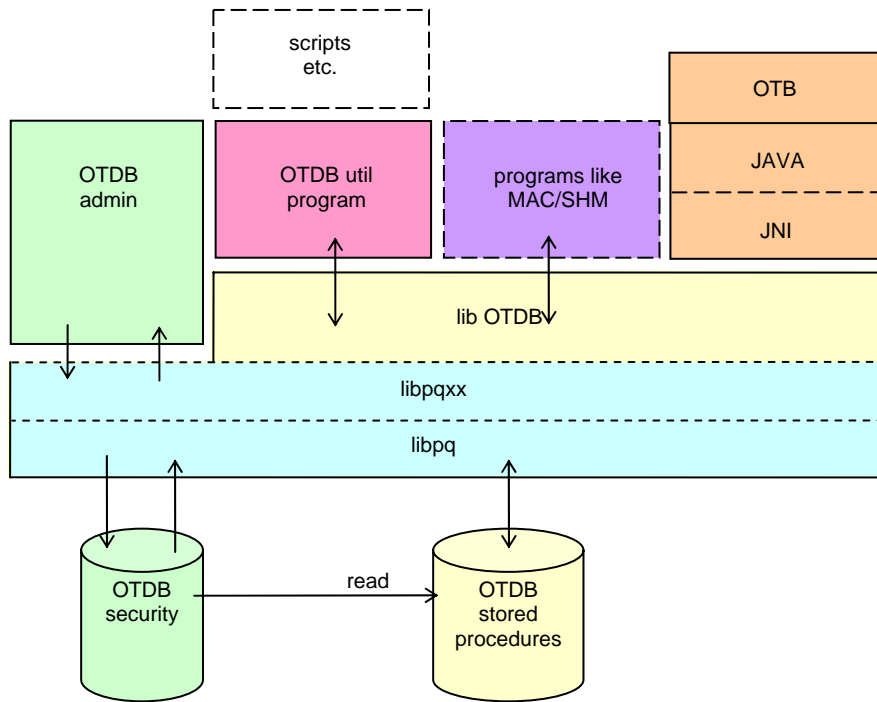


Figure 15: Layer view on the OTDB architecture.

When access to the OTDB is needed from a scripting language an 'OTDButil' program can be written that makes the necessary functions available in the scripting environment.

To manage the authorisation of OTDB (see 4.4) a separate security database is used. All relevant stored procedures first verify with the security database if they are allowed to perform their task for this user on the selected tree. The advantage of building this authorisation check into the stored procedures is that it becomes quite difficult to access the data when you are not authorised. Building it in on any other level would make it easier to break into the database.


To maintain the security database a separate program (OTDB admin) is required that will be available to the LOFAR 'superuser(s)' only.

**Architectural risk:** Whenever another database server product is chosen, the libOTDB layer and OTDB admin must be rewritten. Although libOTDB is a thin layer, it is based on libpqxx that is dedicated for PostgreSQL. The stored procedures for the database are written in PLPG/SQL that is almost compatible with e.g. PL/SQL from Oracle.

### 6.1.1 Backup facility

Since the OTDB is the heart of the SAS module, it is important that the database content is assured some way. When LOFAR becomes operational it is advisable to run the OTDB on a high availability cluster so that maximal uptime is guaranteed.

Making a backup of the database contents once every 24 hours to a standby machine seems sufficient for the test period. [TBC]

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

### 6.1.2 OTDB security

To manage the access of all user(group)s a simple user administration will be used. This administration will be based on four main entities (see paragraph User rights4.4):

- users: having a name and password for authentication and an unique generated user-token for authorisation.
- groups: a user belongs to zero or more groups. Each group will also have an unique generated group-token.
- access-rights: access is granted to a combination of: token, functioncall-ID, tree-type, tree-classification and optional an extra value.
- functioncall-IDs: Every critical stored procedure gets its own unique functioncall-ID.

When a user logs in into the database he receives a unique user token, this token is stored in the connection-object and is automatically added to each function-call in the libOTDB. When a 'critical' stored procedure is called the procedure checks if the combination user token/group token, functioncall-ID, tree-type, tree-classification exists in the security database. If the combination is not in the database, the function is aborted.

### 6.1.3 Database size

To get a rough indication about the size of the database we assume there are 100.000 nodes in each tree and there are also 100.000 metadata changes(KVTs) per tree. One node is about 200 bytes, one KVT is about 100 bytes. So for each tree:

100.000 nodes of 200 bytes = 20 Mb  
100.000 logs per observation of 100 bytes = 10 Mb

This is 30 Mb per observation. To calculate with 'nice round' numbers: 50 Mb per observation.

When there are 1000 trees in the actual database, the database will be about :1000 trees x 50Mb = 50 Gb large. The use of indices may double this size.

This gives us the following numbers for a database:

number of trees	1000
nodes per tree	100.000
loggings per tree	100.000
total record count	200 million
total database size	100 Gb


This amount of data is manageable by PostgreSQL.

## 6.2 OTB architecture

The Observation Tree Browser (OTB) is the main GUI for users to get access to the SAS information. The OTB is aimed at many different users: the scientist who wants to use the instrument to prepare an observation, the instrument scientist who translates the observation request into system parameters and finally the operator who can monitor the observation and the parameter trends.

Each type of user is provided with its own set of screens he may use.

The runtime environment of the OTB is also very diverse. The scientist and the board will use the OTB over the internet. The instrument scientist and the operator will use the OTB from a local LOFAR machine. Finally the OTB will also be delivered as a standalone program to the scientist so that he is able to view the

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

metadata of his observations locally on his own machine. Therefore the OTB is written in Java that provides the most flexibility in runtime environment. Furthermore it makes it possible to use the OTB as an applet on the internet with a webbrowser.

Note: It is very likely that the interfacing with the scientist and review board will be done using NorthStar and MoM. Until then it is also possible to specify the observations with OTB. Specifying observations using the OTB will in any case remain to be of use for operational and test observations, as well as an “expert level” interface for scientists who require access to non standard options and/or who want to prepare and schedule observations on the fly.

Since the OTB is designed to be used locally, remotely and over the internet it is based on RMI. For local use the observation tree is read from a file, for remote and internet access the OTB uses an OTB server program that translates the Java calls to libOTDB calls.

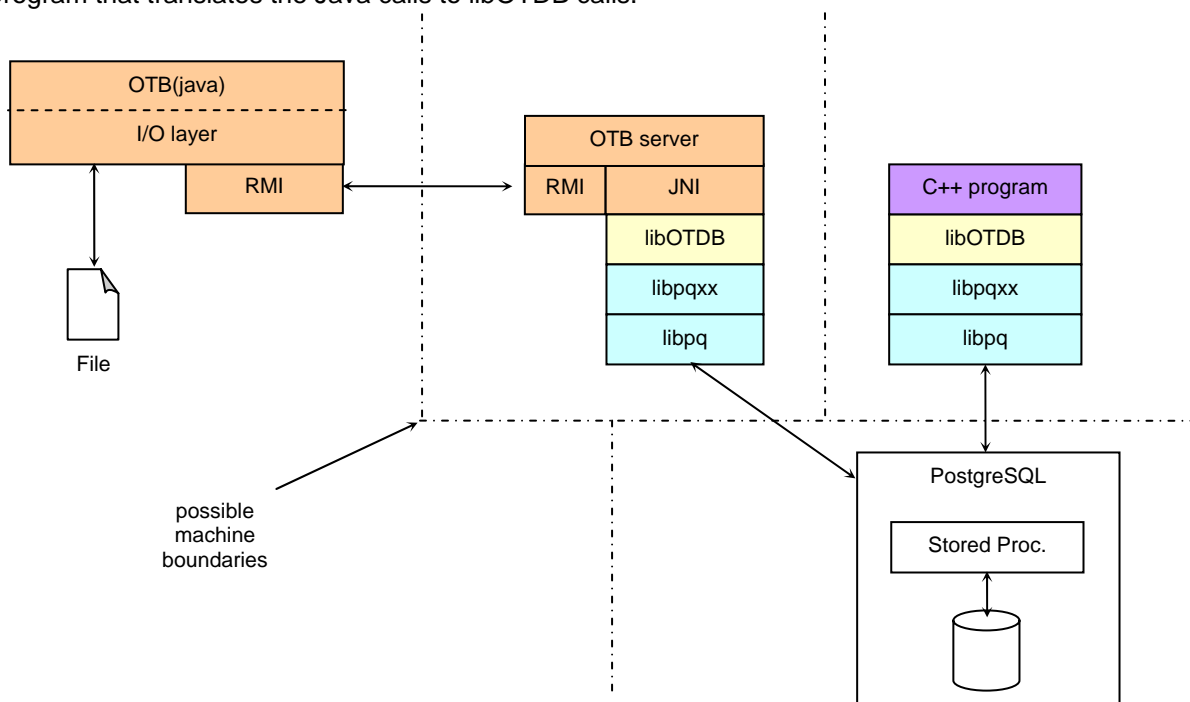



Figure 16: Architecture around the OTB application

The metadata of the observation will (partially) also be delivered to the scientist with the exported dataproducts. To be able for the scientist to browse through this metadata the OTB is capable of switching between file input and database input.

When the OTB is used in database-mode it uses the Remote Method Invocation protocol for accessing the database. For these RMI-calls an OTB-server is required that translates these RMI calls to calls to the libOTDB. A JNI layer is used for the conversion from Java to C++ and vice versa. The OTB server is based on the same C++ library stack as the C++ programs that access the OTDB.

The OTB server can be placed in a DMZ to improve the security for internet access.



Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

#### Architectural risks:

- When OTB is used over internet the firewalls on both side of the connection must be enabled for RMI. On the client-side this may require an extra action for the system administrators. On the other hand if NorthStar and MoM are used for the Proposal phase OTB will rarely be used over the internet.
- The OTB server brings an extra step into the communication chain. Besides a very small extra delay this might hamper the debugging of the OTB application. Adequate logging in the OTB server will be necessary.

### 6.3 Scheduler

The scheduler is an autonomous module that tries to schedule observations as efficient as possible. It takes into account that some parts of the instrument can be shared between several observations and other parts cannot. Also it assures that the target is within the view-area of the instrument, that the RFI is below a required value, etc.


When the scheduler finds one or more observations that can be executed it sets the start- and stoptime of the observations and releases the observations for execution. The execution of the observation will be handled by the Monitoring And Control (MAC) subsystem.

The major functions of the scheduler are:

- long term scheduling: predict if a set of observation requests can be executed within a given period (several weeks). This scheduling will be based mainly on the astronomical, environmental and instrumental constraints.
- short term scheduling: secure scheduling of the observations, all constraints (including resource availability) are taken into account. The output of this scheduler will be used for normal operation of the instrument.
- emergency scheduling: when an observation is aborted during or just before execution the emergency scheduler will attempt to fit in another observation in the time-slice that becomes available.

Since all three schedules use slightly different constraints and timescales it might be advisable to create three separate modules that operate autonomously. This also improves replaceability.

Note: The company CQM has built a demo scheduler in 2006 to prove the concept described in this document.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

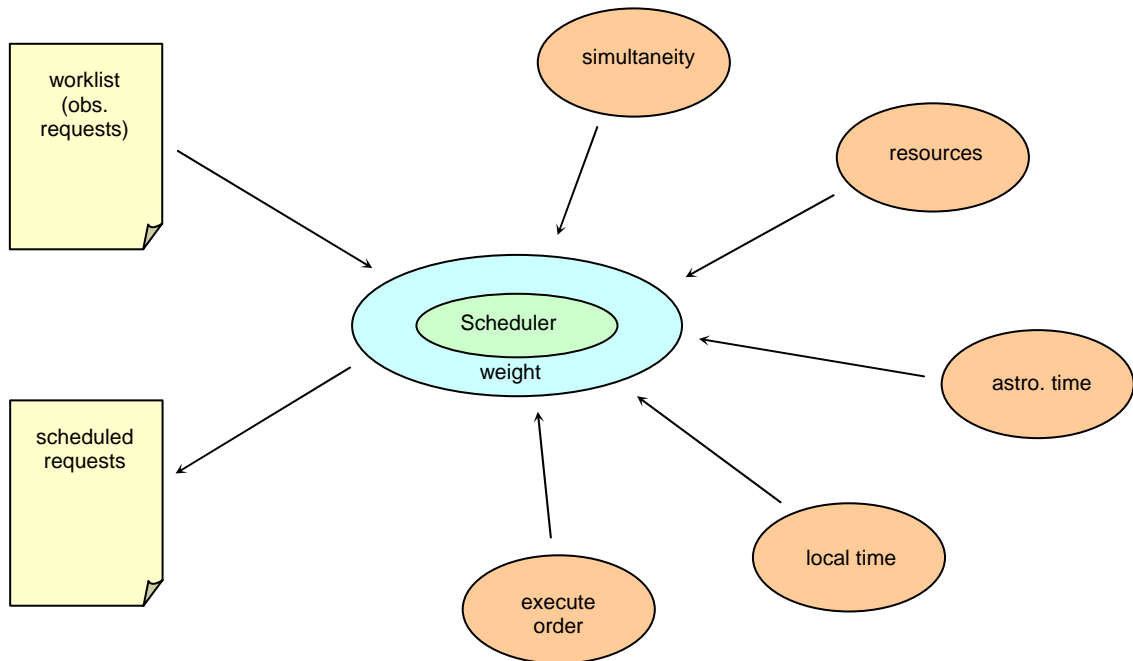


Figure 17: Logical view on a scheduler module; the constraints affect the decisions the scheduler has to make..


The constraints are separated from the scheduler modules. A set of functions will be made available to the scheduler modules. These functions are independent of the schedulers and contain all astronomical-, environmental- and instrument knowledge.

Note: By splitting the scheduler functionality and the constraint functionality, the scheduler functionality can be acquired from a company specialized in that field.

### 6.3.1 Synchronisation between scheduler and data-storage.

The scheduler has a (predictive) knowledge of the available storage in the storage cluster. With this knowledge, the scheduler is able to avoid scheduling two data intensive observations simultaneously (long term scheduling).

For the short term scheduling the scheduler (through a constraint function) will ask the storage system how much storage will be available in the period the scheduler is trying to fill in. In this way the scheduler does not need to keep detailed information about available disks itself. The available storage may differ from the predicted storage because the execution time from the offline processing can vary. By making the storage system an 'autonomous' system this capacity information can be managed at the source.


Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

### 6.3.2 Resource Maintenance

For maintenance a distinction should be made between planned maintenance and unplanned maintenance.

For planned maintenance the operator selects the resources and enters the expected maintenance period. This action will result in a scheduled 'observation' entry in the SAS database. The scheduler will take this 'observation' into account when scheduling the scientific observations. By making the operator the owner of the 'observation' the scheduler is not allowed to move or cancel this observation.

For unplanned (immediate) maintenance the operator directly sets the resource in maintenance state. This state change will be detected by the related running observations (if any) and they will take account of this 'failing' resource. When the maintenance on the resource has finished, the operator sets the resource in the 'idle' state (assuming the resource is operational again). It is not set in the on-line mode as that would result in another resource change of the working instrument that would again disturb the running observation.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 7 Deployment

The OTDB will be placed within the LOFAR local network like all other applications such as MAC and SHM. Therefore, these applications can access the OTDB over the local network.

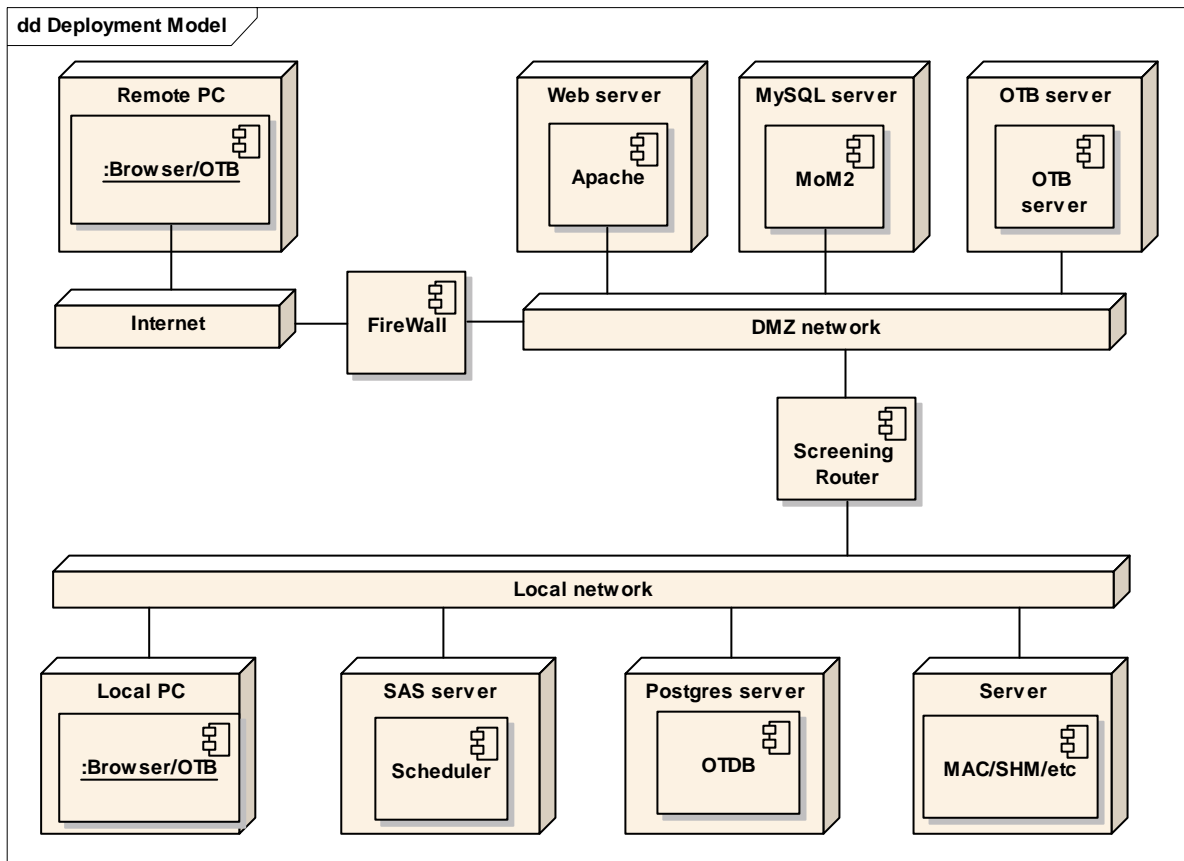



Figure 18: *Deployment of SAS components.*

The access for the human users is handled by a web-server, e.g. Apache, which is placed in the DMZ network. Remote users will have to pass a firewall to reach this server. In-house users like operator and instrument scientist can reach the server via the local network (and probably a screening router or similar equipment).

The OTB server can be placed within the DMZ network or within the local network. This choice will mainly depend on how often the OTB will be accessed by internal users and external users.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 8 Subsystem relations

In this chapter the interfaces that SAS has with the outside world are summarized (see also Figure 1).

### 8.1 NorthStar/MoM

As stated before it is possible to use NorthStar for the proposal phase of an observation and MoM as a observation project management application (possibly offering a high level observation preparation interface for standard observations). This implies that there is an interface between NorthStar/MoM and the OTDB.

There are only a few different interface-moments between OTDB and MoM:

- a. A new (set of) observation(s) is created in OTDB from the information received from MoM.
- b. Status changes in OTDB are reported back to MoM.

If NorthStar is used without MoM, only the first interface is required as NorthStar is purely a phase I application.

MoM and OTDB use a different mechanism as their core interface mechanism. Currently MoM is a web application with no knowledge of the OTDB. It accepts HTTP requests and can in reply return XML-formatted answers. The interface model of OTDB is based on a C++ library that relies on stored procedures in the database. This is not an ideal situation for connecting the two systems.

As a proof of concept an HTTP adapter has been developed. In this way, MoM can keep its original interface that is expanded to support OTDB messages. The OTDB HTTP adapter uses the standard C++ interface of OTDB so no changes are necessary at the OTDB interface either. Ultimately an interface without the HTTP adapter is preferred for security.

### 8.2 MAC

There are many interface actions between MAC and SAS:


1. MAC polls SAS for observations that are ready for execution
2. MAC asks SAS to make a parameterfile of an observation tree
3. MAC keeps the state of the running observations up to date
4. SAS imports a PVSS dump from MAC containing the configuration of the physical instrument.
5. MAC delivers alert and action information to SAS as metadata

#### 8.2.1 Deliver executable observations

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 are in the future and that the state of the observation is 'scheduled'.

#### 8.2.2 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 the configuration file to the ObservationController. This configuration file contains sufficient information for all programs that are involved in the observation to do their job.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

### 8.2.3 Allow tree-state updates

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'.

### 8.2.4 Import physical instrument layout

As described in paragraph 4.2 the PIC-trees in the SAS database are derived from the PVSS contents. The MAC subsystem has an import- and export utility, the ASCII manager, which can be used for database maintenance. When this ASCII manager of MAC is used to make a dump of the database, SAS is able to read in this file and construct a new PIC tree.

### 8.2.5 Import alerts and actions

When serious problems arise during control or monitoring these problems are stored as 'alerts' in the MAC database. These alerts will remain on the operator screen until the operator has responded to them. The action(s) he takes on these alerts are also registered in PVSS.

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

## 8.3 SHM


System Health Management looks for (changing) trends in metadata. It will do queries like 'give me the metadata values of parameter(type) X in the period Y till Z. When SHM detects a serious change in the trend it can instruct MAC to perform some tests with the equipment.

Note: typical, the period length SHM is looking at is much longer than the period for which an observation is kept in SAS. When removing an observation out of SAS, the metadata for SHM should not be deleted. One of the flags of the parameter-definitions in SAS is used for this purpose.

## 8.4 CEP


Every LOFAR application may produce metadata information that is important for the processing of the signal data. To store this information in the SAS database libOTDB offers functions like addKVT and addKVTcollection for storing a single Key-Value-Timestamp triple or storing a group of triples.

The name of the 'Key' should match a parameter name in SAS.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## 9 Open ends

- The company CQM has built a prototype of the SAS scheduler to prove that the scheduling problem can be solved. A decision must be made how to continue with this issue.

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

## Appendix A Requirements compliance

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

Code	Meaning
Y	SAS is already compliant.
y	SAS will be compliant.
CQM	The demo-scheduler built by CQM was compliant.
TDB	To be defined.
...	Solved in another subsystem.

### LO-3.12 Specification and Scheduling function


Nr.	Requirement	Compliant
01	LOFAR shall provide a specification function allowing users to initiate observation requests, to detail such requests and to check the status of these requests.	Y
02	The specification function shall be responsible for the translation of astronomical specification into instrumental settings and configurations (e.g. from a required sky coverage to a collection of beams with certain properties).	TBD
03	The specification function shall provide optimized entry functions for the observational modes listed in 3.09.1.	Y
04	LOFAR shall provide a scheduling function allowing staff at Scientific Exploration Centers and members of the Operational Team to create an integrated instrument schedule based on specified observation requests and maintenance requirements.	Y
05	The scheduling function shall provide assistance in the scheduling process by suggesting computer generated schedules, allowing user interaction.	CQM
06	The scheduling function shall be capable to dynamically change the schedule in case active observations are aborted by the monitoring and control function and in case external triggers with sufficient priority (e.g. gamma ray bursts) demand a change in the schedule.	CQM <sup>1</sup>

#### LO-3.12.1 Observation Specification

Nr.	Requirement	Compliant
01	LOFAR shall be able to receive observation requests submitted by scientists.	Y
02	It shall be possible to request post processing of archived data products	y
03	It shall be possible to request archiving and export of data products	y
04	LOFAR shall be able to evaluate observation requests received from scientists on their technical feasibility.	TBD
05	All requests that require significant LOFAR resources (both for acquisition and for processing of archived data) shall be subject to approval by a scientific review board [TBD].	MoM?
06	The approved observation request shall be available TBD days/hrs prior to the	Y

<sup>1</sup> To support observations that can become immediately active after an external trigger, 'standby' observations were defined. The demo-scheduler was capable of scheduling these observations simultaneously with active observations when several constraints were met. It is the task of MAC to stop the active observation(s) and 'resume' the standby observation(s) when an external trigger occurs.




Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

	observation execution time.	
07	It shall be possible to derive an observation schedule from the information contained in an approved observation request.	Y
08	The observation request shall be specified such that no human interaction is required during execution of the observation.	Y
09	LOFAR shall maintain a record containing all specified observations with all observational requirements.	y
10	LOFAR shall include the administrative process of handling observation requests, including approval, rejection and status reporting.	NorthStar/MoM
11	It shall be possible for scientists to provide supporting and specifying information with their request.	NorthStar/MoM
12	It shall be possible to specify parameter values in terms that are generally accepted in the user community (note: must be specified in more detail). This includes common units, time and date information and co-ordinate systems.	y
13	It shall be possible to specify maintenance activities to be included in the LOFAR observation schedule.	Y
14	It shall be possible to specify preventive maintenance activities at regular intervals including health checks.	y
15	It shall be possible to specify corrective maintenance activities (repairs).	Y
16	It shall be possible to specify observations ('triggered observations') that depend on an internal or external trigger.	y
17	It shall be possible to specify a fixed period in time for an observation (e.g. for synchronization with other telescopes).	Y
18	It shall be possible to specify criteria that determine when an observation should stop.	y
19	It shall be possible to specify, as one observation, measurements of a sequence of observation targets.	Y
20	LOFAR shall be able to check and consequently to notify the user that a requested observation has been performed at an earlier point in time.	MoM
21	LOFAR shall provide a database containing reference information of scientific users and observation proposal reviewers.	MoM

### LO-3.12.2 Observation Scheduling

Nr.	Requirement	Compliant
01	It shall be possible to create a consistent schedule of multiple simultaneous observations and maintenance activities.	CQM
02	It shall be possible to create a consistent schedule of only observations or only maintenance activities.	CQM
03	The LOFAR schedule shall only contain approved observations and approved maintenance activities.	Y
04	LOFAR shall be controlled on basis of the generated schedule.	Y
05	It shall be possible to create a schedule report that shows the available remaining capacity of the instrument over a period of at least 6 months (in order to support the approval process).	CQM
06	All generated schedules shall be compliant with the constraints of observations and maintenance.	CQM
07	The scheduling process shall take preferences of observations and maintenance into account.	CQM
08	All generated schedules shall be compliant with the availability of resources.	CQM
09	It shall be possible for the operator to modify any schedule.	Y

Author: Ruud Overeem	Date of issue: 2007-03-30 Kind of issue: public	Scope: SAS Doc.id: LOFAR-ASTRON-SDD-041	
	Status: final Revision nr: 1.2		

10	It shall be possible to generate an operational schedule (i.e. a schedule that is to be transferred to the monitoring and control function for execution) for a period of at least 2 weeks.	CQM
11	It shall be possible to inform a scientist automatically about when his/her observation is scheduled.	y
12	It shall be possible to reschedule observations that have been interrupted.	Y
13	It shall be possible to schedule a remaining part of an observation if it has been interrupted, but only if it is suitable to split an observation.	Y
14	It shall be possible to reschedule observations that have not been completed	Y
15	It shall be possible to schedule observations that will be started through a pre-defined event (trigger). The occurrence of such an event should lead to the activation of a well defined alternative schedule (a so-called "priority" or "stand-by" schedule).	y
16	It shall be possible to schedule re-processing of archived data.	y
17	It shall be possible to schedule post processing jobs on request.	y
18	It shall be possible to schedule archiving and export of data products.	y
19	The scheduling function shall generate consistent (lower level) schedules for both data flow processing and dataset processing on the central processing system.	CQM