European Organisation for Astronomical Research in the Southern Hemisphere

Atacama Large Millimeter/submillimeter Array

ESO ALMA Support Centre

Integrated Alarm System Design

Document Number: ESO-299387

Document Version: 1

Document Type: Design Report (DER)

Released On: 2017-06-26

Document Classification: Public

Prepared by: Caproni, Alessandro

Validated by: Schmid, Erich

Approved by: Schmid, Erich

Name

Authors

Name	Affiliation
Alessandro Caproni	ESO Directorate of Engineering
Erich Schmid	ESO ALMA Support Centre

Change Record from previous Version

Affected Section(s)	Changes / Reason / Remarks
All	New document

Contents

1 Introduction

1.1 Scope

This document defines the high-level design and deployment of the various components that make up the Integrated Alarm System. Detailed design will be evaluated during the development process and documented in the project Wiki pages [RD3].

1.2 Definitions, Acronyms and Abbreviations

This document employs several abbreviations and acronyms to refer concisely to an item, after it has been introduced. The following list is aimed to help the reader in recalling the extended meaning of each short expression:

ACS	ALMA Common Software
ADC	ALMA Department of Computing
ALMA	Atacama Large Millimeter/submillimeter Array
APE	ALMA Production Environment
ASCE	Alarm System Computing Element
BSDB	Backstage Database
CSV	Comma Separated Values
DASU	Distributed Alarm System Unit
ESO	European Southern Observatory
GUI	Graphic User Interface
IAS	Integrated Alarm System
IASIO	Integrated Alarm System Input/Output
I/F	Interface
JAO	Joint ALMA Observatory
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LTDB	Long Term Database
RDB	Relational Database
RDBMS	Relational Database Management System
R/O	Read-only
R/W	Read-Write
STE	Standard Test Environment
TF	Transfer Function

2 Related Documents

2.1 Reference Documents

The following documents, of the exact version shown herein, are listed as background references only. They are not to be construed as a binding complement to the present document.

RD1 Integrated Alarm System for the ALMA observatory
ESO-287159, Version 1

RD2 Integrated Alarm System Architecture
ESO-293482, Version 2

RD3 Integrated Alarm System GitHub Project Wiki
https://github.com/IntegratedAlarmSystem-Group/ias/wiki

3 Introduction

The Integrated Alarm System (IAS) is a tool to improve the situation awareness of operators in the control room and engineers sitting at the executives showing the operational state of the observatory composed of several heterogeneous monitored systems.

Each monitored system is modelled by an acyclic graph whose nodes represent the components of the systems and are mapped into elements in the panels shown in the control room. Each component can be operational or in a non-nominal state, depending not only on the values of the monitor points and alarms in input, but also on the current operational phase. Each component's inputs are elaborated against a user provided Transfer Function (TF) to determine its state and finally mapped in the panels with proper and consistent colour coding.

RD4 *As described in the* Integrated Alarm System Architecture

ESO-293482, *Version 2*

RD5 Integrated Alarm System GitHub Project Wiki

https://github.com/IntegratedAlarmSystem-Group/ias/wiki

, the monitor point values and alarms produced by each monitored system are routed to the IAS by one or more IAS Plugins. Such Plugins are either running in the workspace of the monitored system or in the IAS server, when the data can be accessed remotely. The values are sent to the IAS core on change or after a time interval expires to ensure that the values owned by the IAS are up-to-date and the connection alive. Such values sent by the Plugins are converted into Integrated Alarm System Input/Output (IASIO), that is the unique data structure recognized by the IAS, before being elaborated.

The components of the monitored systems correspond to the Distributed Alarm System Units (DASU) that are the deployable units of the IAS. They elaborate the IASIOs in input and produce an output (another IASIOs) that represent their states and are finally represented in the operator and engineers' panels. The output produced by the DASUs can be sent as input to other DASUs, modelling the hierarchical graph of the monitored systems.

The DASUs are internally composed of a set of Alarm System Computing Elements (ASCE) that take a subset of the inputs of the DASU itself and produce an output. They are in turn organized as an acyclic graph whose topmost element is the output of the DASU. The ASCE elaborates the inputs against a user provided Transfer Function (TF) that returns the IASIO in output, considering also the actual operational phase.

The IASIOs produced by the DASUs are finally mapped into the panels shown in the control room or remotely and are also sent to the Long Term Database (LTDB)

4 Deployment

Figure 1**Error! Reference source not found.** shows the deployment of high-level IAS components in the case of three monitored systems.

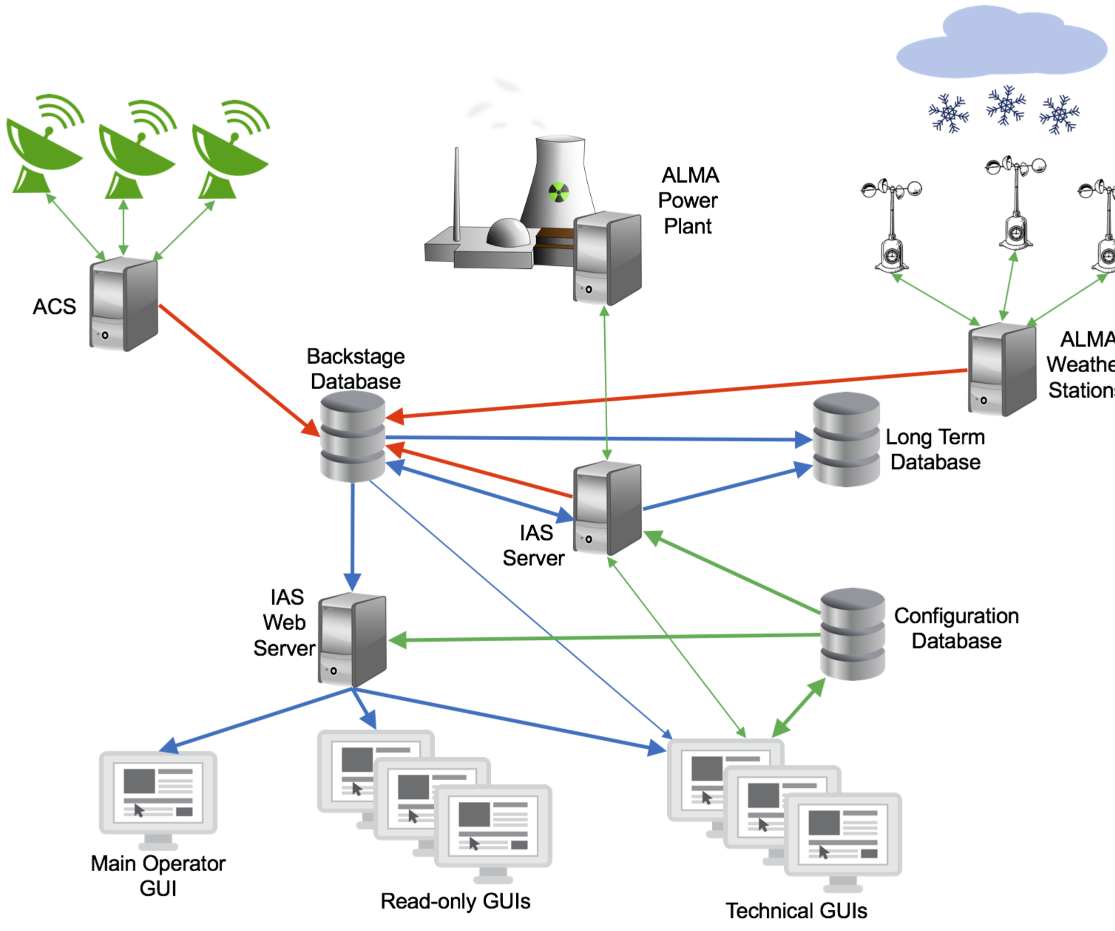

Figure 1 - Integrated Alarm System Deployment

The ALMA Power Plant in the example is a closed system where it is not possible to deploy additional software components. It provides an API to get monitor point values and alarms. This is a typical use case for all third-party systems that need to be integrated into the alarm system. In the worst case, such systems do not provide an API, but as a minimum requirement for a system to be integrated, they must provide structured log files that can be parsed to get the status of the monitored in system. In any case, to get values from a system like that, a dedicated software component needs to access the data through its API or logs and send them to the Backstage Database (BSDB). It could run for example in the IAS Server, as shown in this example, but this is not the only possibility as the only constraint it has is to be able access the Power Plant data and send values to the BSDB.

The other two systems in this example are ALMA proprietary systems that offer other possibilities or have different limitations.

ACS is the ALMA Common Software, the software infrastructure for the ALMA software that controls the array elements to run observations. ACS is a distributed system based on a component container paradigm with CORBA as middleware. It provides access to monitor point values and alarms produced or collected by the ALMA software. Because the ALMA software for a particular array runs in an isolated environment called *ALMA Production Environment (APE),* or quite misleadingly an *STE*, and there can be more than one APE active at the same time, it is not possible for the IAS to interrogate ACS directly from the IAS Server. It will therefore be required to deploy a dedicated component, an IAS plugin, into every APE that collects the monitor points and alarms from ACS and sends them to the IAS Backstage Database. There is no communication from the IAS with this component: when the component is alive it sends data to the BSDB; when the component is not running, all the values and monitor point values that the IAS expects to receive from it are marked as invalid. Exact deployment details, e.g. one component collecting all the data from ACS, or several components (for example one component for each antenna), or if this will be deployed as a proper ACS component or an independent piece of software, still need to be decided in discussion with other stakeholders.

The last example is the Weather Station control software, which is an independent tool developed by ADC. Here it is possible to deploy a dedicated software application that runs inside the weather station workspace, but not inside the control software: as in the other cases, it collects monitor point values and alarms from the weather station through its API or from log files.

The three example of plugins represent different plugin types:

- A dedicated software application that runs in an IAS server (ALMA Power Plant);
- A component that is part of the monitored system (ACS case);
- A dedicated software application that runs in the monitored system's workspace (Weather Station case)

All these examples are possible IAS plugins that perform the same task: collect data from the monitored system and send them to the BSDB.

The IAS server gets data from the monitored systems in the BSDB and processes them by means of DASUs and their contained ASCEs. The alarms generated by ASCEs are, in turn, also stored in the BSDB, so they can be consumed and processed by other DASUs.

The BSDB, by design, keeps only a "live" status of all the monitored components that gets updated frequently.

The configuration and deployment of the IAS components is stored in the Configuration Database (CDB). Dedicated Technical GUIs that are part of the IAS are used to define and maintain the configuration.

The IAS Web Server reads the alarms and monitoring information generated by the IAS, combines them with the configuration information stored in the CDB and makes them available to the Main Operator and Read-only (or engineering) GUIs. It takes the alarms directly from the BSDB, which is the place where the IAS server stores the results of the computations made by the ASCEs.

In principle, DASUs, ASCEs, CDB, LTDB, BSDB and the IAS Web Server can all be deployed in the same server but it is probably more convenient a distributed deployment to improve performances and robustness.

In Figure 1**Error! Reference source not found.**, red arrows represent monitor point values and alarms sent by plugins to the BSDB. As we said, the plugin of the power plant of our example runs in the IAS server and sends data to the BSDB as represented by the red arrow from the IAS server to the BSDB.

Monitor point values and alarms from the remote software systems are converted into IASIOs as well as the results of the computation of the ASCEs, and propagated through the IAS components: they are represented by blue arrows.

Other types of communications are represented by green arrows.

Figure 2 shows the conceptual deployment of IAS components:

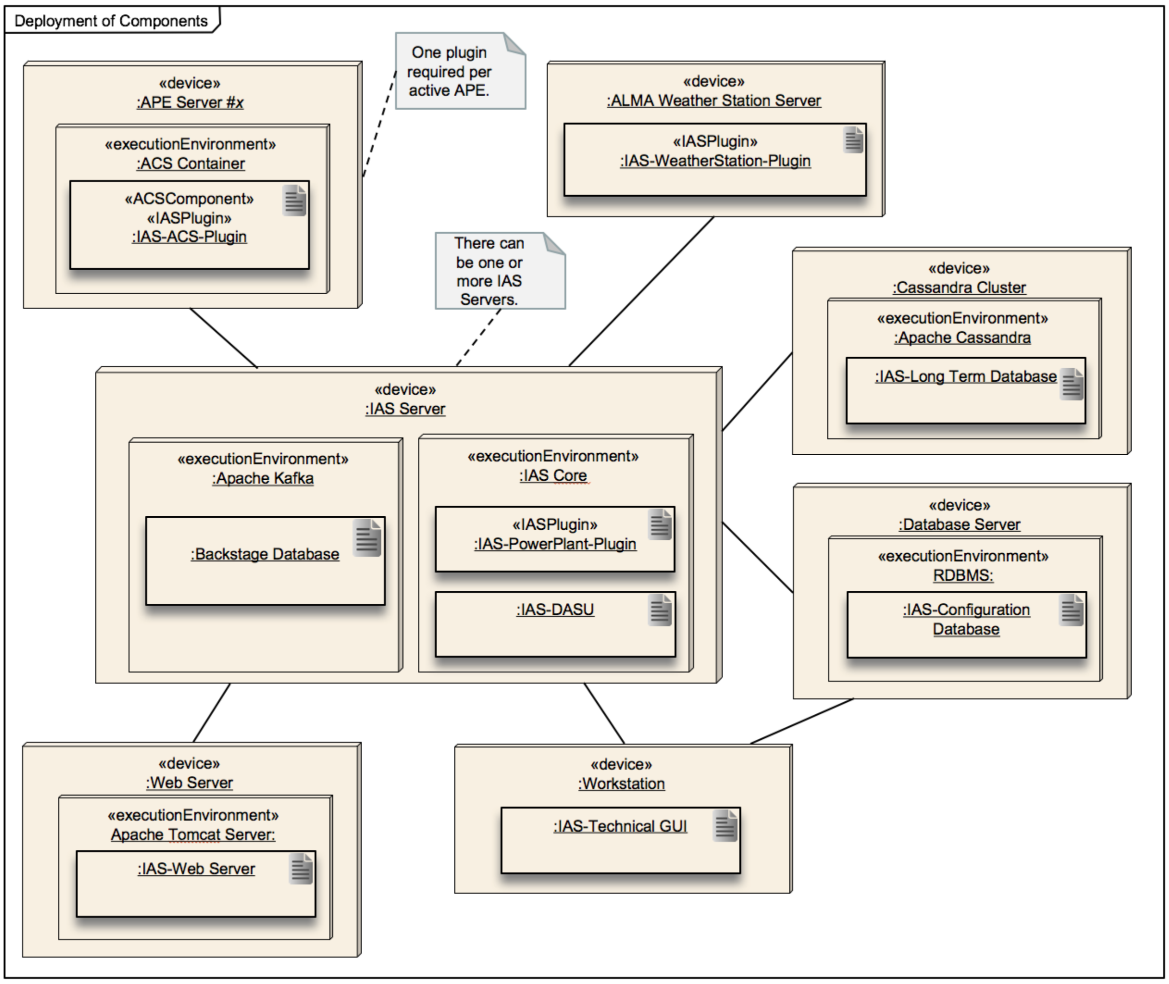
Deployment of Components
One plugin required per active APE.
«device» :APE Server #x
«executionEnvironment» :ACS Container
«ACSComponent» «IASPlugin» :IAS-ACS-Plugin
«device» :ALMA Weather Station Server
«IASPlugin» :IAS-WeatherStation-Plugin
There can be one or more IAS Servers.
«device» :Cassandra Cluster
«executionEnvironment» :Apache Cassandra
:IAS-Long Term Database
«device» :IAS Server
«executionEnvironment» :Apache Kafka
:Backstage Database
«executionEnvironment» :IAS Core
«IASPlugin» :IAS-PowerPlant-Plugin
:IAS-DASU
«device» :Database Server
«executionEnvironment» RDBMS:
:IAS-Configuration Database
«device» :Web Server
«executionEnvironment» Apache Tomcat Server:
:IAS-Web Server
«device» :Workstation
:IAS-Technical GUI

Figure 2 - Integrated Alarm System Deployment Diagram

5 Design Description of IAS Components

This section describes the conceptual design of the various IAS components introduced in the previous section. Only a very high level design is provided in component diagrams. Detailed design will be developed and documented as part of the implementation.

5.1 Configuration Database (CDB)

The Configuration Database contains configuration and deployment information for the IAS. The CDB is edited using one or more so-called Technical GUIs, which are Java desktop applications that are provided by IAS.

The configuration information is stored in a relational database. The IAS provides an API to access the CDB, which requires a network connection to the database server. The CDB implementation does not require a dedicated process. The relational database can either be deployed on a dedicated server, as shown in Figure 2, or in one of the IAS servers.

The following diagram depicts the CDB components and the main interactions with other IAS components.

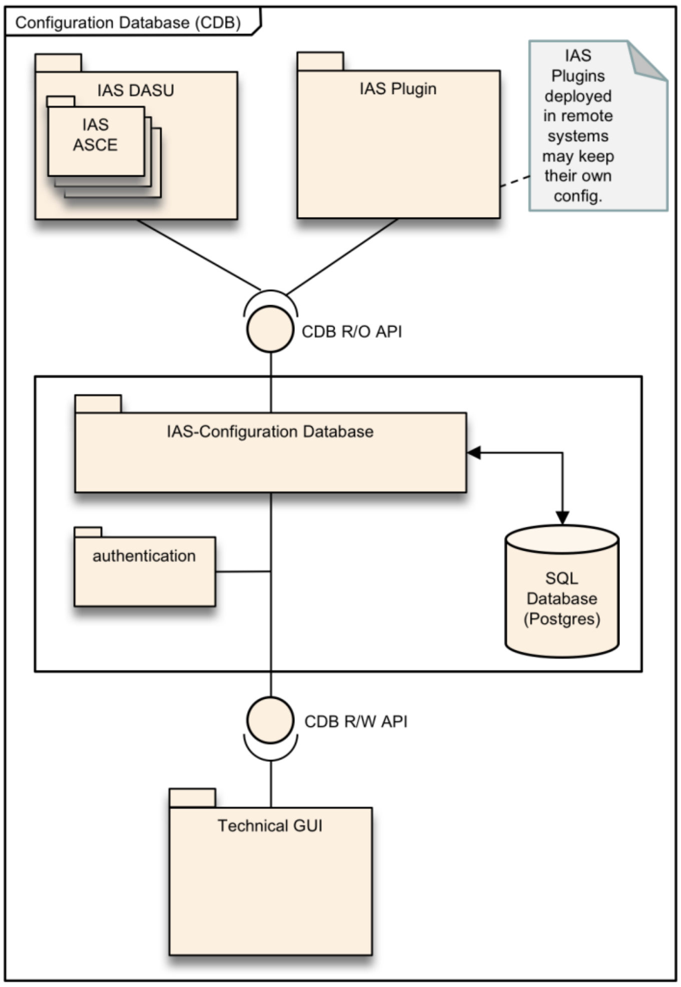
Configuration Database (CDB)
IAS DASU
IAS ASCE
IAS Plugin
IAS Plugins deployed in remote systems may keep their own config.
CDB R/O API
IAS-Configuration Database
authentication
SQL Database (Postgres)
CDB R/W API
Technical GUI

Figure 3 - IAS Configuration Database

The CDB component provides read-only access to the configuration information stored in the database via an open interface to all other IAS components, not requiring authentication. IAS-Plugins deployed in remote system (e.g. ACS component or Weather Station) may keep their own configuration information locally. The Technical GUIs are the only components that can be used to alter the content of the CDB. In order to protect the content from unauthorised or unintentional modification, any write access requires user authentication.

5.2 Backstage Database (BSDB)

The Backstage Database is central to the functioning and robustness of the IAS: this is the place where the Plugins send alarms and monitor point values and where the DASUs obtain this information for processing and store the results of their computations to be propagated between the various IAS components, to the Long Term Database and to the IAS Web Server (and from there finally to the user GUIs). This requires that the BSDB is deployed in a network/server accessible by all the IAS components.

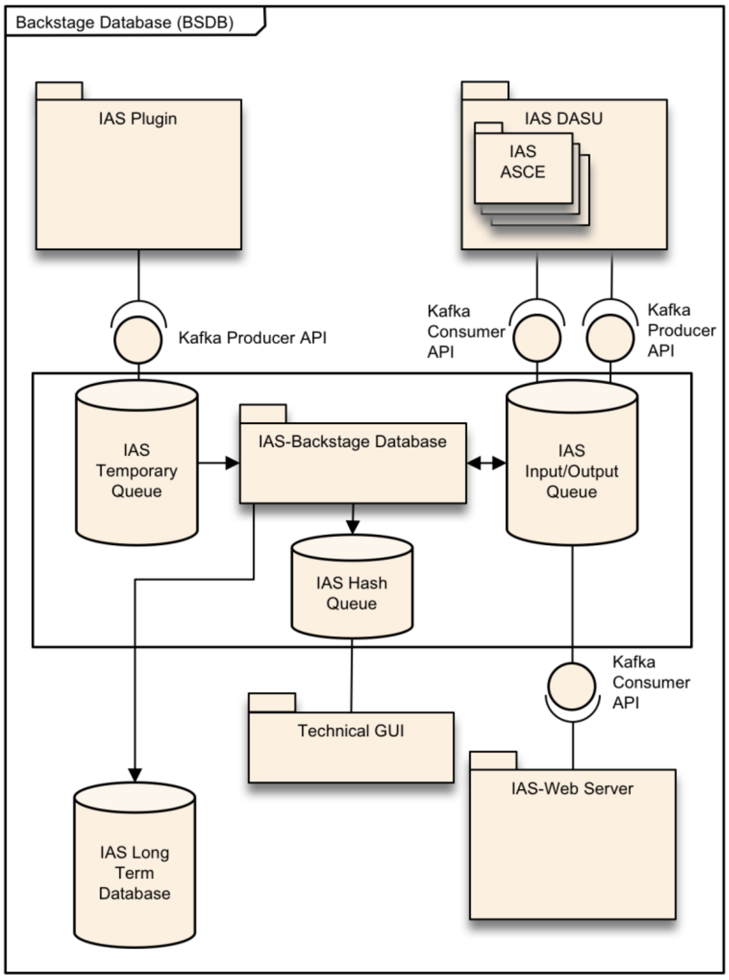

Figure 4 - IAS Backstage Database

Plugins send alarms and monitor point values to the Temporary Queue in the form of triplets that contain the *identifier*, *value* and *timestamp.* This helps keeping the Plugins simple and generic. The BSDB converts these triplets into IAS Input/Output (IASIO) objects and propagates them to the IAS Input/Output Queue. DASUs and their contained ASCEs consume these initial inputs, perform the defined Transfer Functions and send resulting alarms and monitor point values back into the Input/Output Queue, where they can be once again serve as inputs to other DASUs, or be directly used by the IAS Web Server to update the status of the corresponding entities in the operator GUIs.

All Plugins and DASUs update all values on change and at defined time intervals into the BSDB. The BSDB therefore does not need to persist its contents. If the BSDB is restarted, all the entities shown in the user panels will initially be invalid, meaning that no current and reliable information is available. They are then automatically refreshed as soon as the new values are sent again to the restarted BSDB instance.

The unavailability of the BSDB does not affect the other components of the IAS that continue sending monitor point values and alarms to the BSDB as if it would be running. So, for example, the Plugins run and send data to the BSDB without taking care if it is running or not.

The Hash Queue contains a snapshot of all the monitored entities, their status and associated alarms. Its main purpose is to quickly retrieve entities based on their identifiers. This will mostly be used by the Technical GUIs, while the Web Server feeding the operator and engineering GUIs will receive updates in a *live* fashion from the Input/Output Queue.

The BSDB components can be deployed in one or more servers. Initially it is however planned to deploy the entire BSDB and the IAS Core, which includes all the DASUs and certain Plugins, on a single server. Replication to improve the robustness, or scalability requirements could require deploying more instances in different servers. The configuration of the BSDB is a task of the alarm system administrator. Deployment details of the IAS components and the Plugins will be evaluated as part of the commissioning of the IAS.

5.3 Long Term Database (LTDB)

The Long Term Database stores all the values generated by the computation of the IAS components, which is essentially every IASIO that flows through the Input/Output Queue of the BSDB (see Figure 4). Browsing the data in the LTDB allows to reconstruct the computation of the IAS offline for after the facts investigation or to debug the IAS itself.

Monitor point values and alarms sent by plugins to the BSDB will be stored in the LTDB only after conversion to IASIOs. Monitor point values and alarms generated by remote software system are not stored in the LTDB.

A copy of IAS logs is also saved in the LTDB for debugging purposes.

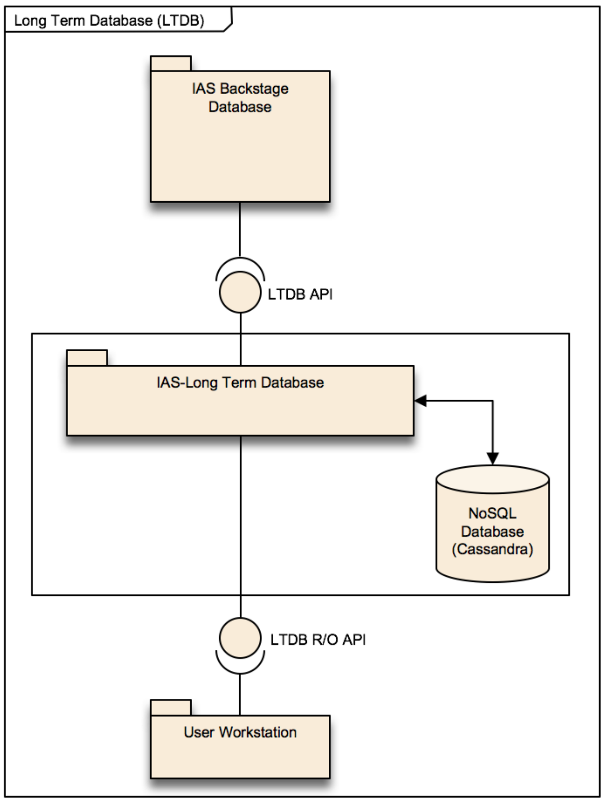

Figure 5 - IAS Long Term Database

A non-relational database like Cassandra seems the best solution for the LTDB. Cassandra is also used by the ALMA software for the so-called Monitor Store. Note that even if the technology is the same, and the IAS LTDB can be co-hosted on the Cassandra cluster used for the Monitor Store, there will be no interference between the two databases.

The IAS will provide a read-only API for accessing the data stored in the LTDB using generic software tools. Specific tools for this are not foreseen.

5.4 IAS Core

IAS Core is not an IAS component per se, but a term we use to describe the central execution environment of the IAS. As such IAS Core includes IAS components such as all the DASUs, each of which is a self-contained, deployable application, and IAS Plugins that serve remote systems, where the Plugin cannot be deployed directly.

All these components are deployed in one or an arbitrary number of IAS Servers, which are in the same network. Initially it is envisioned that all DASUs, Plugins, and the Backstage Database are deployed in the same server. Depending on the number of DASUs, it could be required to have more than one server but this is a deployment choice. The name of the server where components are deployed, is part of its configuration stored in the CDB.

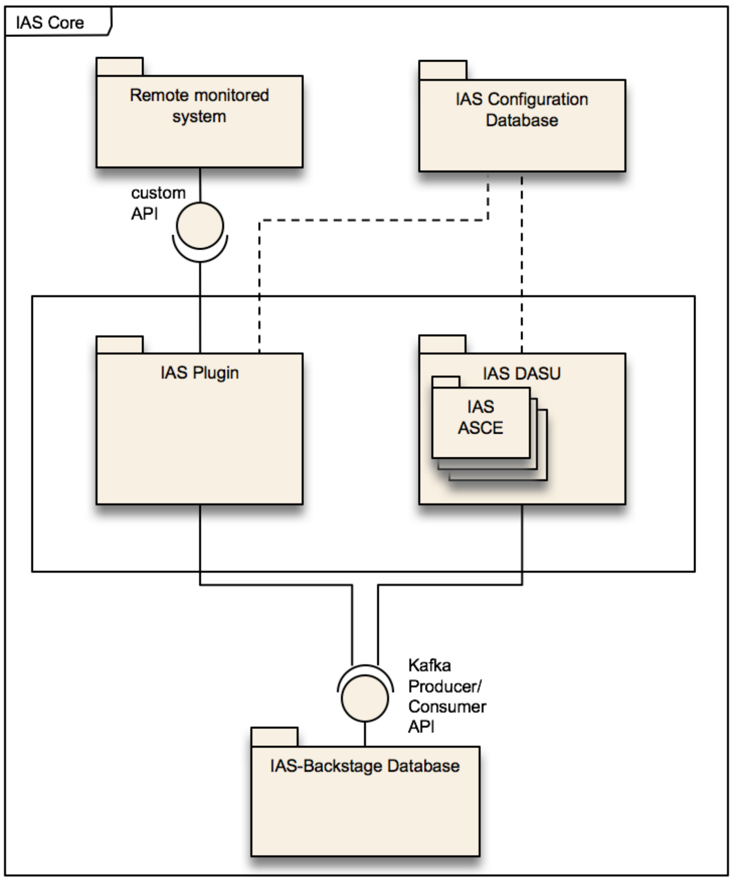

Figure 6 - IAS Core

Using the Backstage Database described before, the interactions of the main components of the IAS Core become remarkably simple. Plugins use a custom interface (or log files) provided by the remote system to obtain data and feed this data, after proper conversion into the Temporary Queue of the BSDB. The BSDS takes care of feeding these raw data packets into its own Input/Output Queue. All deployed DASUs consume data from this queue and write again the calculated, so-called synthetic, values back into the BSDB.

The Configuration Database is only used during start-up to obtain the deployment and configuration information.

The simplest deployment option is to run all the IAS components in the same server: the relational database for the CDB, the backstage database, the web server, DASUs. Exact deploy needs will be determined during the course of the project.

5.5 IAS Web Server

The IAS Web Server will be based on a standard web application environment, e.g. Apache Tomcat, to serve single page web applications to any number of user clients. User client machines must only have a standard modern web browser (e.g. Google Chrome, Firefox or Safari). No IAS specific software needs to be installed, but the IAS will officially only support one of these browsers, which will be either Firefox or Google Chrome.

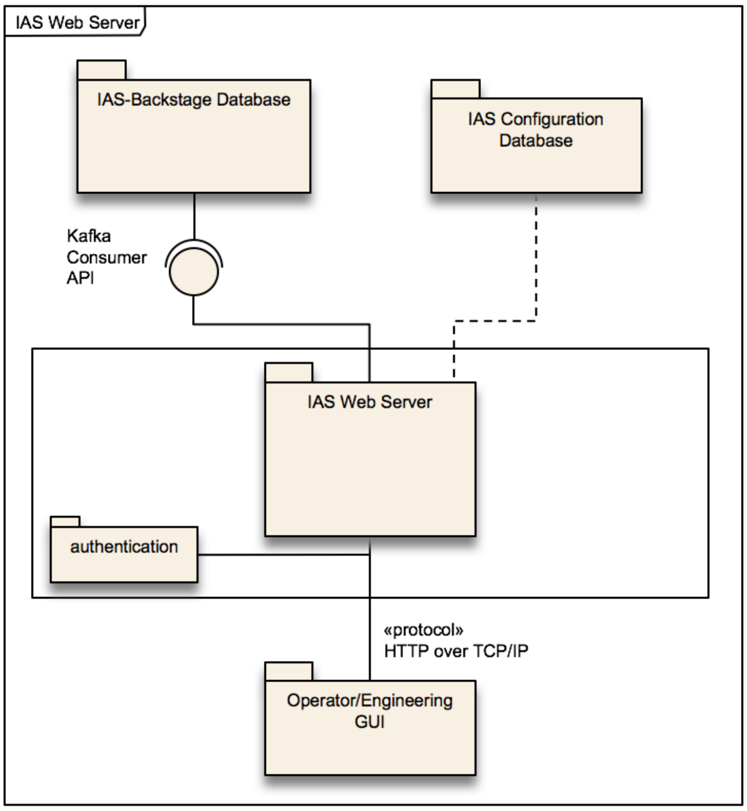

Figure 7 - IAS Web Server

At start-up, the IAS Web Server reads configuration information from the CDB. It then subscribes to all updates of the IAS BSDB's consumer interface, maps the information to any of the provided user panels and updates these panels accordingly in the users' web browsers.

Note that the Web Server does not persist any status or alarm information. After a restart of the Web Server, entities shown in any of the panels will only be updated once the first corresponding event is received from the BSDB. Initially, also if the Web Server becomes unresponsive for a prolonged period, all entities will be shown as invalid, Given the short refresh intervals of all entities this does not pose a problem.

5.6 IAS Technical GUI

IAS Technical GUIs will be implemented as standalone Java desktop applications, most likely using Java FX.

Its main purposes are the editing of the Configuration Database and giving administrators access to the contents of the Long Term Database, namely log files and time series of events flowing through the IAS. Other technical GUIs allow to monitor the functioning of the IAS and identify issues like cheating alarms, malfunctioning transfer functions and so on.

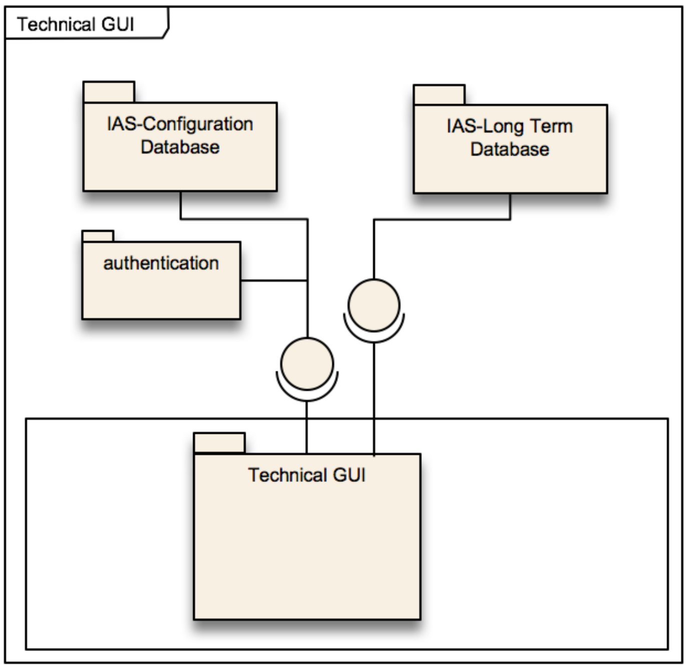

Figure 8 - IAS Technical GUI

IAS Technical GUIs can be installed and deployed on any user machine that has access to the OSF network where the IAS Core is deployed. Updates to the configuration database will require proper authentication, and only one instance can update the CDB at any one time. The LTB can be accessed from this closed network without authentication.

Exact details of the Technical GUIs will be evaluated at a later stage.

6 Deployment and Start-up

6.1 IAS_ROOT

IAS_ROOT is an environment variable used to identify the installation folder of all IAS components. Third party tools, scripts, binary executables, libraries and configuration files are installed in a folder relative to the *IAS_ROOT* environment variable: a script, *ias-bash-profile.sh*, generates the environment on top of that folder.

If IAS components are deployed on multiple servers, the *IAS_ROOT* content must be replicated or even better shared to ensure that all the executables run with the same configuration and the same versions of the libraries and the tools.

By default, the log files are written in a set folder relative to *IAS_ROOT*, but can be redirected to a user defined location by setting an environment variable.

6.2 Start-up

Starting the IAS means to read the deployment configuration and then run the DASUs that, in turn, load and activate the ASCEs. The configuration must then be already available before starting up the system. With the CDB implementation in a relational database, we assume that the CDB server is already running, either on a dedicated machine or in one of the IAS servers.

Although the life span of the BSDB could be the same than of the IAS, i.e. the BSBD is started before the first IAS component and shut down after the last component, a simpler solution is to have the BSDB always running as a service and ready to accept values.

Details will be evaluated at a later stage of the project.

7 Communications

IAS Plugins send monitor point values and alarms to the BSDB queues as formatted strings (CSV or JSON, for example). The strings are taken from the Temporary Queues, converted into IASIO data structures, serialised and ingested into another queue, the Input/Output Queue. Clients of the IAS, namely the IAS Web Server and other DASUs, receive IASIO data structures directly from this queue. The distribution of IASIOs to IAS components is done with a publisher/subscriber mechanism: the client that wants to be notified of IASIOs subscribes for updates for certain *topics*. Subscribing to a topic, e.g. a DASU modelling an antenna, will also subscribe to all its contained subsystems, such as frontend receivers, HVAC systems, etc..

A filtering mechanism must be put in place to control the amount of information circulating for each topic. Each client can subscribe to a limited number of events, either by requesting updates at specific time intervals only, only if they reach certain boundaries, or a combination thereof. Some of the clients must be notified of all the IASIOs. like the LTDB or the Web Server: in this case, no filtering will be applied but the clients must be fast enough digesting events to not affect the performance of the BSDB.

Publishing and subscribing to the IAS topics is available through an API provided by the IAS Core that allows to change the publisher/subscriber implementation transparently if there is the need to change from one middleware to another.

8 Commands

In general, IAS components passively receive events, IASIOs, through the publisher/subscriber paradigm.

Operators send commands to the IAS to ACK or shelve alarms: such commands are sent from the web browser hosting the operator panels, to the application in the Web Server and from there to the DASUs and ASCEs.

There will be a limited number of operational commands to DASUs, like for example a command to shut down or restart or dump their status.

We expect that the number of commands generated by operators plus the operational ones will be very infrequent at intervals of a minute or more.

Sending and receiving of commands will be implemented by an API that hides the low-level implementation and allows to quickly replace the technology and/or the used middleware with another one, if convenient.

Considering the small amount of commands and if request/reply is not needed, a possible solution is to use again the publisher/subscriber mechanism, avoiding the introduction of another communication paradigm. A dedicated topic will be used for all the commands to which all the IAS components able to receive commands subscribe. Commands are represented by formatted string (CSV, JSON, for example). A subscriber checks each command published in the topic immediately discarding the ones that do not require an action.

If the request/reply is needed, as it will probably be the case, a MQ middleware could be used instead. The identifier of the destination of the command allows to get the host where it runs and to which send the command and wait for the reply.

9 Requirements

IAS requirements are those reported on RD1.

The followings are additional requirements

Requirement Id	Description	Qualifier
REQ-CORE-001	The IAS must clearly report malfunctions at run-time	Mandatory
Comment		
The IAS core must publish alarms in case of malfunctions in one or more of its components: • Failure, crash of the BSDB • BSDB queues full or consumers too slow • DASU not operative or malfunctioning • ASCE transfer function too slow or broken • Remote plugin unavailable		
Justification		
Operators in control room must be able to recognize problems of the IAS core components. A panel shall display core alarms allowing the operator to browse them the same way they do for other alarms, identifying the problems and being able to quickly start the counter action.		

--- End of document ---