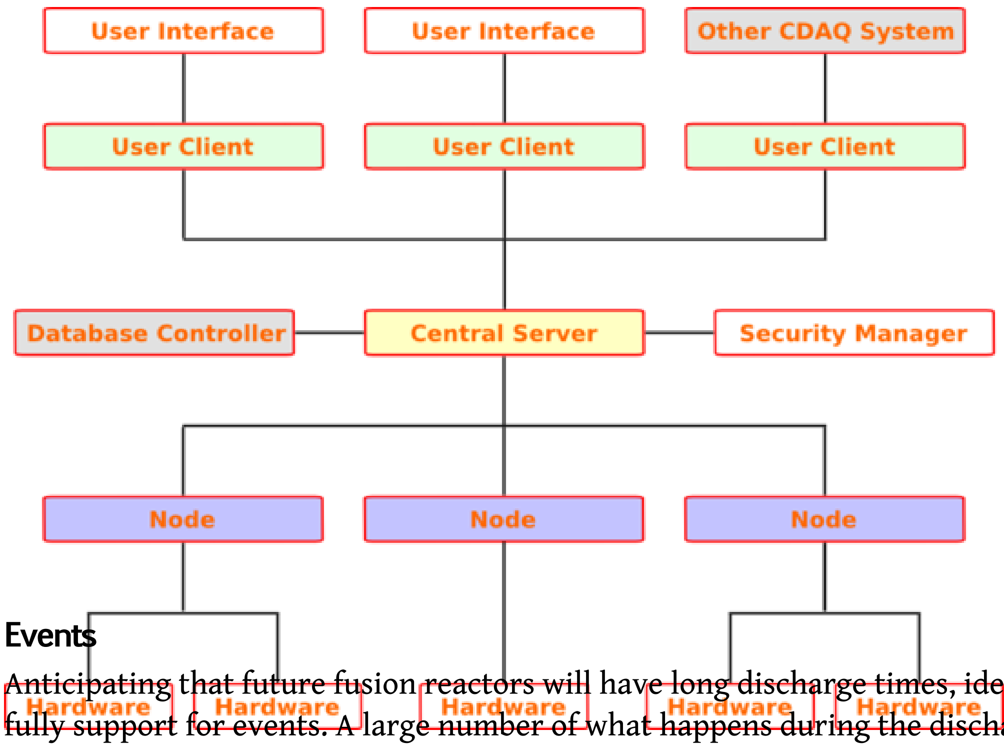
As a full modular system FireSignal avoids dependencies of particular technologies. It is composed of five different modules, which provide
all the functionality to the system:
• Central Server – works as bridge betweenall the other components. It is responsible for managing all the connections, commands
and data broadcast;
• Database connector – controls the installed database, storing and fetching data when ordered by the Central Server. It provides an
abstract layer, independent of the database solution;
• Security Manager – the Central Server queries this component to authenticate users and nodes and to authorize all the operations.
The security schema used (LDAP, SQL, etc.) is abstracted by the security manager;
• Nodes – are responsible for driving the devices and for data readout. Generic nodes, with the communication to the Central Server
already implemented, are provided in different languages and can be extended to integrate any hardware device;
• User Clients – these are the graphical userinterfaces, which allow the interaction with the system. It is the only optional component
as the system can still operate without direct human intervention, e.g. when FireSignal works as a system which receives orders
from other data acquisition system.

All the components are connected through CORBA allowing them to run in different operating systems and to be written in different computer languages. Currently FireSignal is using Java, C++ and Python as main languages.



Events

Anticipating that future fusion reactors will have long discharge times, ideally continuous, FireSignal has fully support for events. A large number of what happens during the discharge is considered to be an event. Examples of events include the discharge start trigger, external triggers and disruptions. Events are defined by a unique name and number and by an absolute time-stamp. In the case of ISTTOK/PT the event shot is defined by the string '0x0000', where the unique number is the shot number and the time-stamp the time of the shot.

Time-stamping

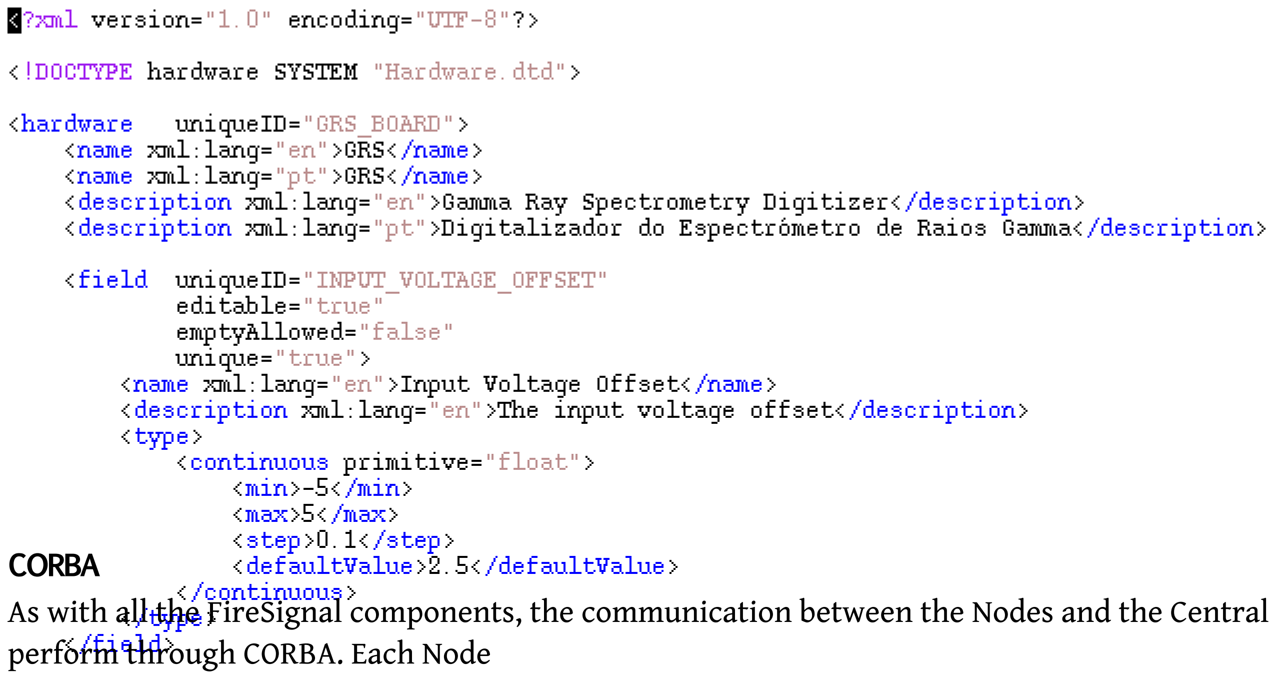
Each cluster of acquired data is tagged with an absolute start and end time-stamps. Knowing the time of the events associated with the signal generation one can easily correlate the time of the events with the times of the acquired data.

Nodes

Hardware clients, also named nodes, provide the interface between the installed devices and the Central Server. Their main responsibilities are the configuration of hardware, tagging data and events with the correct time-stamps and sending them to the server.
Nodes are organized with the following structure:
• Nodes contain hardware devices;
• Devices contain data acquisition channels, also named parameters;
• Each hardware has a set of configuration fields which instructs the node in how parameters can be configured;
• By default all configuration fields are transversal to all parameters. Considering a simple Analogue to Digital Converter as an example, one can think of each acquisition channel as a parameter and the acquisition frequency as a value for the configuration field. The configuration field can impose, for instance, that the frequency must be an integer not superior to 1 kHz. Obviously different parameters may have different values for the same field.

XML Description

The structure presented in the previous paragraph is defined as an XML file for each hardware device. The XML syntax is validated by a Document Type Definition provided to the developers. When creating the XML file one starts by defining an internationalised name and description for the hardware, followed by the configuration fields. There are five different types:
• Continuous values: this field accepts a number (floating point) which is limited by an imposed interval;
• List: the possible values are defined in a set;
• File: this field accepts byte streams representing a valid system file;
• String: any character array is accepted;
• Custom: the value is verified by an external class provided by the developer.
Finally, the parameters are defined and added. A parameter is composed by a suggested name and description and by the type of data it acquires (byte arrays, images, etc.).



```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hardware SYSTEM "Hardware.dtd">

<hardware   uniqueID="GRS_BOARD">
    <name xml:lang="en">GRS</name>
    <name xml:lang="pt">GRS</name>
    <description xml:lang="en">Gamma Ray Spectrometry Digitizer</description>
    <description xml:lang="pt">Digitalizador do Espectrómetro de Raios Gamma</description>

    <field  uniqueID="INPUT_VOLTAGE_OFFSET"
            editable="true"
            emptyAllowed="false"
            unique="true">
        <name xml:lang="en">Input Voltage Offset</name>
        <description xml:lang="en">The input voltage offset</description>
        <type>
            <continuous primitive="float">
                <min>-5</min>
                <max>5</max>
                <step>0.1</step>
                <defaultValue>2.5</defaultValue>
            </continuous>
        </type>
    </field>
```

CORBA

As with all the FireSignal components, the communication between the Nodes and the Central Server is perform through CORBA. Each Node
implements the FireSignal server-node communication standard. The number of functions is quite significant and ranges from hardware
configuration details to event broadcasting and data acquisition.
FireSignal is bundled with server-node communication standard implementations in three different languages: Java, C++ and Python. This allows developers to focus only in the interface between Nodes and hardware drivers without having to learn or concern about the communication protocol with the Central Server.

Plug and Play features

Nodes can be live connected and removed from the system. The Central Server has a dynamic list of all the connected nodes and their locations. These can be disconnected in two ways: the Central Server checks for the Node presence at given intervals and if the Node is unreachable, an error is issued and it is automatically removed; the node warns the Central Server and unregisters in a transparent way. When a new node is connected, the Central Server node list is updated and this information is broadcasted to all modules.
If the registered hardware in the Node supports Plug and Play, the FireSignal communication protocol allows it to inform the Central Server about hardware connections changes. This allows to dynamically add or remove hardware from the system and for these devices to be recognized automatically.

JWS User-interface

The user client is a standalone application, developed in Java and deployed using Java Web Start technology, serving as a front-end of the FireSignal system.

The main window is divided in several areas with information on who is logged in, the nodes connected, the server time, invitations to look at specific data sent by other users, a chat room, a list of events that have occurred and messages sent by the server. In the central area, tabs provide access to information on the current state of the nodes, information on data currently available, a simple data viewer and, if the user has special permissions, configuration of parameters and operation state changing. The size and position of the tabs and areas may be changed by the user, according to his preference, simply by dragging and dropping where desired.

The user-interface is fully internationalised and different languages can be chosen. By default FireSignal will try to adopt the operating system
locale. If this is not available, English will be used. New idioms can be added without the need to reprogram or recompile any of the libraries.
Modern fusion experiments have hundreds of different parameters. In order to quickly find any parameter the client includes a search tool. Usually, a user is only interested in parameters related to a specific diagnostic. Through the creation of specific filters one can toggle the view of parameters, hardware or nodes.

Self-generated GUI

The information provided by the XML files sent by the nodes is used to self generate configuration userinterfaces. Each of the previously defined configuration fields has an associated component which provides automatic validation. For instance: the continuous configuration field is rendered as a text box, which verifies whether the number is valid in the given interval; the list field is shown as drop box with all the available options.

Java Web Start

The application is launched and installed by opening a URL located in a remote server. It works with any browser, Web server and operating system. Java Web Start is a flexible and robust deployment solution for Java technology based applications.
Each time the application is launched a check is performed in order to guarantee that the program and libraries are always up to date, if not, these are automatically downloaded and installed. If needed, it can also download and install the version of the Java Runtime Environment required by the application.

Security

Any open security solution can be easily fitted in FireSignal. The security manager queries the security schema for user and node authentication and operation authorization.

User permissions

Each user has a set of permissions assigned by the security schema. There are four different types of permissions:
• Administrative: add, remove or edit users, add, remove or edit users groups, assign permissions and expels users;
• Operational: send event tables, change hardware status, connect and disconnect hardware and abort experiments;
• Standard: rename and configure hardware;
• Read-Only: experiment following, hardware configuration viewing and data retrieval.

User groups

A hardware group is a defined set of nodes and hardware devices on which users are allowed to perform actions. The concept is very similar to the UNIX groups. Users can be assigned to different groups and, are able to change configurations, connection status and rename nodes, hardware and parameters.