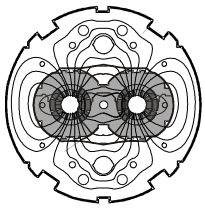
CERN
CH-1211 Geneva 23
Switzerland



the
Large
Hadron
Collider
project

LHC Project Document No.
LHC-AB-CO-xxxx rev 0.2

CERN Div./Group or Supplier/Contractor Document No.
AB-CO

EDMS Document No.
yyyyyy

Date: 31-AUG-07

Functional Requirements

THE DRAG AND DROP DISPLAY AND BUILDER REQUIREMENTS

Abstract

The required elements for the Drag and Drop Display and Builder are listed, described and explained here.

Prepared by :
T. Bolshakov, Fermilab
E. McCrory, Fermilab
A. Petrov, Fermilab
E. Roux, AB/CO/OP
J. Wozniak, AB/CO

Checked by :
D. McGinnis, Fermilab
J. Patrick, Fermilab
S. Gysin, Fermilab
V. Baggiolini, AB/CO
M. Lamont, AB/OP
J. Wenninger, AB/OP
E. Hatziangeli, AB/CO

Approved by:
H. Schmickler, AB/CO
E. Hatziangeli, AB/CO

History of Changes

Rev. No.	*Date*	*Pages*	*Description of Changes*
0.1	13-April-2007	9	First draft
0.2	17-April-2007	8	After comments from E. McCrory
0.3	18-April-2007	8	General changes.
0.4	23-April-2007	9	Additional requirements for components library after discussion with Euginia Hatziangeli.
0.5	7-May-2007	10	Changes after Eric Roux Comments. Added "implementation notes". "Colored comments" should be removed from final version of the documentation.
0.6	24-Aug-07	11	Editing by E. McCrory
0.7	29-Aug-07	10	Editing after comments from J. Patrick

Table of Contents

1. ABOUT THIS DOCUMENT

It is anticipated that the Drag and Drop Display and Builder will be a set of Java applications used by AB/OP and others at the CERN Control Centre and elsewhere during the commissioning and the operation of the LHC at CERN. This tool kit will allow an operator to create quickly a simple, or a complex, visually rich application to monitor and manipulate instruments in the LHC. The components from which the application is made are either general components stored in a component repository or components that the operator has created.

First, we present an overview of this suite; the objectives, the roles and the responsibilities. Then we discuss proposed architecture and present the formal requirements.

The priorities of the requirements are listed as either "Critical", "Expected." The former means that the application absolutely must have this feature. The latter means that the application should have it, but it will not be necessary in the initial version(s) of the application.

In the requirements, we use the abbreviation DnD for "Drag and Drop Display and Builder."

2. INTRODUCTION AND OVERVIEW

2.1 OBJECTIVES

CERN operators will need to monitor and control a multitude of hardware devices during the startup, commissioning and operation of the Large Hadron Collider (LHC). It is very likely that s/he will need to adapt the available control software frequently and to create new accesses to the devices "on the fly." Thus, a general application that allows the operator to view and to manipulate these devices is proposed. In order to follow modern, graphical expectations, the tool proposed here uses visual tools intensely. These visual tools will make the learning period shorter, and the visual nature of the application should help to document the processes being controlled.

Here, we define "Visual Tools" to mean lightweight and user friendly components that can be tied together in logical and meaningful ways on a graphical palette. Most readers will agree that "Drag and Drop" is a big part of saying that a graphical tool is "User Friendly". In this application, we will define a set of pre-designed graphical building blocks, which can be visually placed on the palette. With just a few mouse operations, a complete application can be constructed in minutes. This application is assumed to be monitored and run from the World Wide Web.

Drag and Drop Display and Builder (DnD) is a set of interactive and extendible component-driven tools that provide visual representation of real-time processes. A number of similar systems are in use in Fermilab, CERN and other laboratories. This application is intended to be used in the Large Hadron Collider (LHC).

2.2 ROLES AND RESPONSIBILITIES

There are several roles in the usage of DnD**:**

1. **General public** – anybody in the World. Should be able to watch the Web displays (with some restrictions – the number of possible applications and the traffic may be limited).
2. **Operator** – shall be able to build DnD applications in the DnD Builder (after proper authentication) and to save the application locally and/or in the applications repository (after proper authorization). The operator can make settings (after proper authentication and authorization using standard CERN procedure - RBAC).
3. **Component Developer** – develops new components and places them into repository.
4. **DnD administrator –** grants permissions for **Component Developers**, manages the repositories and the DnD Web site.

2.3 DND TOOLS

It is anticipated that there will be several parts of DnD. We express these pieces as a specific set of tools, listed here:

1. **DnD Builder –** allows for visually building applications. It will be implemented as a Java application.
2. **DnD Runtime –** allows for running an application as Java Program.
3. **DnD Repository** – the component application repository (ies).
4. **DnD Display –** allows viewing the application as the web page.
5. **DnD Components Library –** implements all the required functionality for reading devices and presenting results.

2.4 PROPOSED ARCHITECTURE

The architecture of DnD is based on architecture of similar Fermilab system [1] adapted to use the CERN software system and to incorporate the relevant features of the previous CERN synoptic display system [2].

DnD Builder should allow for building a graphical application (as shown in an example from Fermilab, Figure 1) at CERN, using LSA and JAPC interfaces, in about 0.5 – 3 hours. These graphical applications suppose to be built from blocks, or components, possibly connected to each other via pipes.

Each component should implement a simple, common interface. The set of components is extendible, which means that Component Developer may design new components or component libraries. An extensive standard set of components will be provided from which operators can create their own applications. Each component will have a Java implementation (that is, a Java class that implements the interface) and a formal XML-based description. Component Developers may use standard components as a template.

Builder saves XML Configuration of developed application in the Repository. DnD Runtime reads XML Configuration from the Repository and actually runs the application.

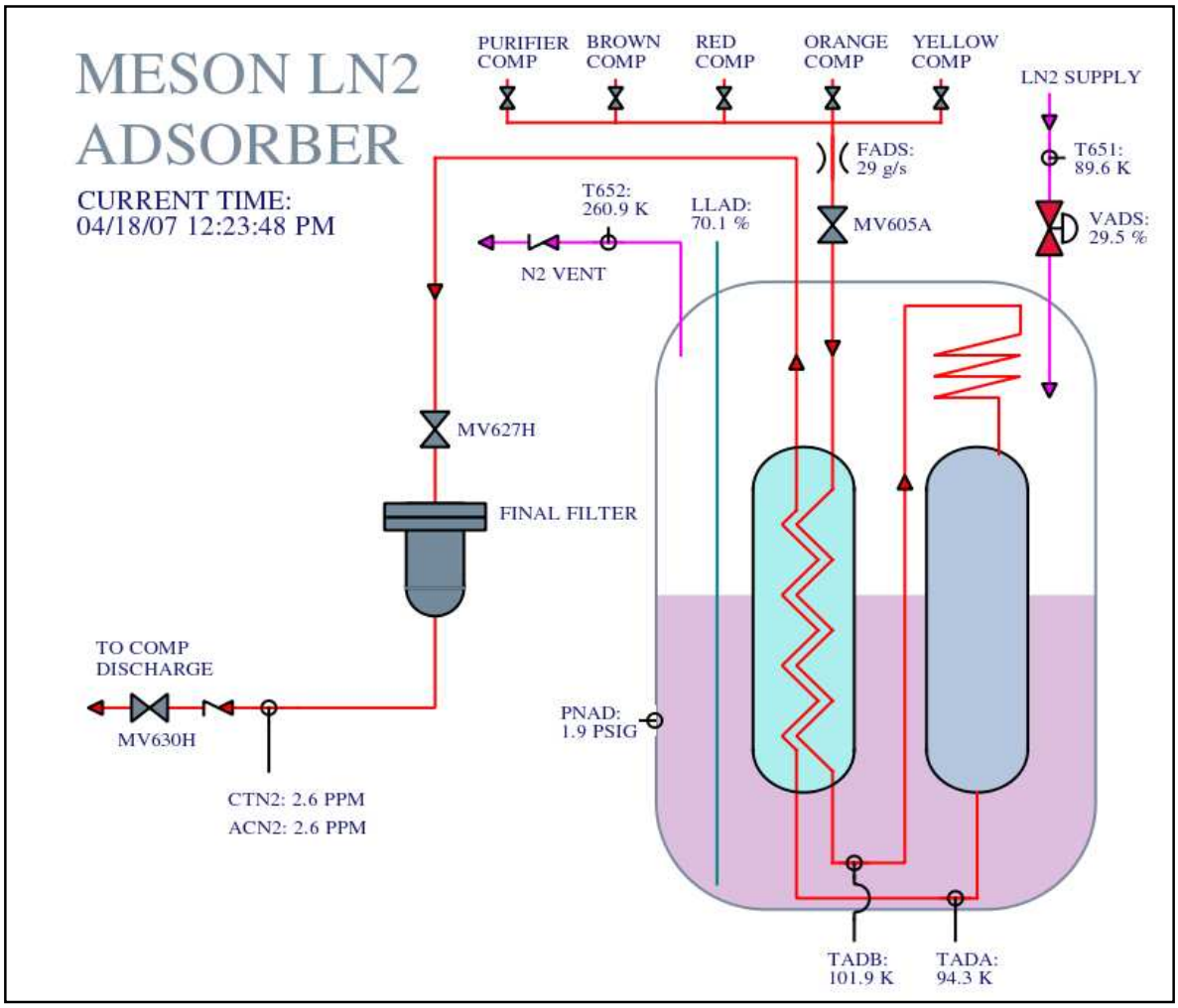


Figure 1. Example of graphical application, built by DnD.

The standard set of components will be stored in centralized repository. Applications, designed with DnD, will be saved in the same (or a similar) repository. The access to this repository will be through the Web. All the tools also can be stored as Java Web Startable Programs.

An application, designed within DnD, should present current device information (reading, setting, status, etc.) to the user, with update rate around 1 to 3 seconds. It is anticipated that the resulting application will be able to be run in a web browser and/or as standalone Java application (in DnD runtime). It will be possible to start such application as a standard CERN Java application with settings enabled or disabled. (Settings may be enabled only if the application is started as a Java program.)

Configuration files produced by the Builder also may be used by other non-Java applications. The settings enabling policy suppose to be in accordance with CERN security model [3].

3. FUNCTIONAL REQUIREMENTS

The functionality and the scope, performance and concurrency of the DnD are discussed and presented in this section.

3.1 DND BUILDER REQUIREMENTS

The builder is a Java application. Here are the basic requirements for the Builder.

Num	Title	Description	Priority	Source
1.1	Common	The DnD Builder shall be written using standard Java.	Critical	
1.2	RBAC	The DnD Builder shall use RBAC to authenticate users and to authorize actions.	Critical	
1.3	Primary function	The DnD Builder shall allow the user to build application.	Critical	
1.4	Components	A user shall use the DnD Builder to construct applications from the components in the DnD Repository.	Critical	
1.5	Web Run	The applications built by the DnD Builder shall be runnable as a web application.	Critical	
1.6	Standalone	The applications built by the DnD Builder shall run as standalone java applications also.	Critical	
1.7	Web Limited	A DnD application started from the web shall only provide read operations	Critical	
1.8	Save	The user shall save DnD-constructed applications either locally or in the DnD Repository.	Critical	
1.9	Start	Start Runtime – if authorized.	Desired	

3.2 DND RUNTIME REQUIREMENTS

Runtime is a Java application. It should be able to:

Num	Title	Description	Priority	Source
2.1	Standard	An application built by DnD Builder shall be a standard Java application.	Critical	
2.2	RBAC 2	An application built by DnD Builder shall use RBAC to authenticate users and to authorize actions.	Critical	
2.3	Run Source	An application built by DnD Builder shall start from a local repository or from the central repository.	Critical	
2.4	Zoom	The user shall be able to zoom in and out on a view built by DND.	Expected	

3.3 DND REPOSITORY REQUIREMENTS

Num	Title	Description	Priority	Source
3.1	Accessibility	The DnD Repository shall store DnD components in a way that allows the DnD Builder to access these components directly	Critical	
3.2	Grouping	The DnD Repository shall provide a way to group similar components and applications	Critical	
3.3	Description	The DnD Repository shall provide a description of each component and application	Critical	
3.4	Saving	The DnD Repository shall allow a user to save components and applications	Critical	
3.5	RBAC 3	The DnD Repository shall use RBAC to authenticate users and to authorize their actions with the repository.	Critical	
3.6	Versions	The DnD Repository shall maintain versions of all components and applications.	Expected	

3.4 DND DISPLAY REQUIREMENTS

The DnD Display shall run within a web browser. Here are the requirements for the DnD Display.

Num	Title	Description	Priority	Source
4.1	Standard Browser	The DnD Display shall display an application created by DnD Builder in a standard web browser (Internet Explorer 7 and Firefox 2)	Critical	
4.2	No Settings	The DnD Display shall not allow any settings to the devices	Critical	
4.2	No Changes	The DnD Display shall not allow any changes to the application	Critical	
4.4	Low Traffic	The peak traffic from the DnD Display shall be limited to 5 kb/sec.	Critical	
4.5	Bandwidth Arbitration	The DnD Display shall arbitrate bandwidth among the users, giving preference to users in the CCC, then to users at CERN.	Critical	
4.6	Zoom	The user shall be able to zoom in and out on a DnD window.	Critical	

3.5 INITIAL SET OF COMPONENTS

Designing the set of components is crucial to the development of the DnD application. Each component that is used by DnD is to be a Java class that implements a simple interface. Each component will also have an XML file, and the properties of this file are described elsewhere.

Individual components may be connected to other components; these connections are called pipes.

Components functionality should be either "data driven" (depends on the data that the component receives) or "presentation driven" (depends on the user interface requirements). This general definition allows ancillary components, like RBAC interfacing and saving, to be implemented as components. These functions can be implemented in DnD Runtime.

The principle we are suggesting here is simple: let make each component *be as simple as possible* and *provide only one function*. This allows us to divide components into several standard groups, as following:

Num	Comp. Type	Description
5.1	JAPC / LSA	Reads data from the hardware. It is expected that these components will send their data by the pipes for processing, display or saving to other components. It is possible that these components will be invisible on the final application display.
5.2	AscBeans	All CERN AscBeans from 'old synoptic' [2] should be supported as components library.
5.3	Visualization	Visualizations of the data: text fields, tables, scopes, histograms, data viewers, thermometers, barrels with floating level, etc. May serve as pipes.
5.4	Processing Pipes	Reads data from input pipes, transforms that data in some defined way, and send transformed data to output pipes. it is likely that these components will also be invisible. Examples: fft, arbitrary expression, averaging, integrating, etc.
5.5	User Input	For example, text fields, knobs, sliders etc.
5.6	Static Graphics	Special symbols of different kinds.
5.7	I/O	For example, with files and databases,

There may be more types of components – for example at Fermilab there are special cryogenic components.

4. ADDITIONAL COMMENTS

4.1 THE SCOPE

The scope of DnD is limited to those LHC control elements that can be represented within DnD. While it is possible to imagine control elements outside of LHC (for example, the electrical infrastructure), it is not covered by DnD at this time.

It is further imagined that every component cannot (or should not) be represented on a world-readable web page. These limits must be established by the author(s) of each component.

4.2 IMPLEMENTATION NOTES

- In order to support data coherency, the Component libraries may be separated into several "component palettes" and be provided by several repositories. That

is needed to provide the user with a clear understanding that all data on the page are provided with in same DnD Context in terms of timing – cycle – supercycle. As a suggestion, datasources from AscBeans should not be mixed with JAPS/LSA datasources or such a mixing should have definite and clear visual distinction.

- The concept of DnD Repository allows us to separate the general-purpose DnD Builder and the DnD Runtime from the implementation of how and where the actual application configuration is stored.
- Zoom is a desired feature; it may be supported only partially – for example only on some of the components, or only in the Web version.
- All the Java applications are to be written according to existing CERN standards. This statement needs some further elaboration.
- The DnD display is based on the Web Fixed Display application.[4]

4.3 PERFORMANCE ISSUES

We discuss several performance related issues here.

- To minimize network traffic, DnD Display should usually transmit only differences in the visual state, not the visual state itself.
- To minimize server load, DnD Display should start one project and share its visual state between web users.
- To minimize server load, DnD Display should stop an application on the server side after several minutes without any requests.
- To minimize network traffic for users with Internet Explorer, Adobe Display should send SVG in zipped format.

Performance of DnD Runtime is defined by the Component Developers as well.

5. REFERENCES

1. "Synoptic display – a client-server system for Graphical Data Representation", proceedings of ICALEPCS'03,
   http://synoptic.fnal.gov
2. "Old Synoptic"
   http://ab-dep-co-ap.web.cern.ch/ab-dep-co-ap/dev
   http://oraweb.cern.ch/pls/abc/w3cons.applics
3. RBAC documentation: Requirements, Specifications, Reports.
4. Web Fixed Display requirements document.