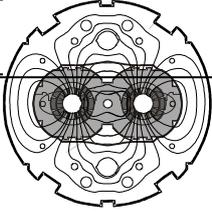


CERN CH-1211 Geneva 23 Switzerland

FNAL, Batavia, IL, USA



LHC Project Document No.

LHC-AB-CO-xxxx rev 0.1

CERN Div./Group or Supplier/Contractor Document No.

AB-CO

EDMS Document No.

YYYYYY

Date: 2007-05-09

Functional Requirements

THE DRAG AND DROP DISPLAY / BUILDER REQUIREMENTS

Abstract

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

Prepared by :

T. Bolshakov, Fermilab
A. Petrov, Fermilab
E. Roux, AB/CO/OP

Checked by :

D. McGinnis, Fermilab
J. Patrick, Fermilab
E. McCrory, Fermilab
S. Gysin, Fermilab

Approved by:

H. Schmickler, AB/CO
[E. Hatziangeli, SL/CO](#)

History of Changes

<i>Rev. No.</i>	<i>Date</i>	<i>Pages</i>	<i>Description of Changes</i>
0.1	13-April-2007	9	First draft
0.2	17-April-2007	8	After comments from Elliott 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.

Table of Contents

ABOUT THIS DOCUMENT.....4

1. INTRODUCTION AND OVERVIEW.....4

OBJECTIVES.....4

ROLES AND RESPONSIBILITIES.....4

PROPOSED ARCHITECTURE.....5

DND TOOLS.....6

2. FUNCTIONAL REQUIREMENTS.....7

BASIC BUILDER CAPABILITIES.....7

BASIC RUNTIME CAPABILITIES.....7

BASIC REPOSITORY CAPABILITIES.....8

BASIC DISPLAY CAPABILITIES.....8

BASIC SET OF COMPONENTS.....8

IMPORTANT IMPLEMENTATION NOTES.....9

SCOPE, PERFORMANCE AND CONCURRENCY.....10

3. REFERENCES.....10

ABOUT THIS DOCUMENT

It is anticipated that the Drag and Drop Display and Builder (**DnD**) will be a set of Java applications used by AB/OP and others in the CERN Control Centre and elsewhere during the commissioning and the operation of the LHC at CERN.

First, we present an overview of the application; 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.”

1. INTRODUCTION AND OVERVIEW

OBJECTIVES

CERN operators will face a need for monitoring and change hardware devices. The general applications that allow to view and to set those parameters should include visual tools. Visual tools will make the learning period shorter, visual information should help to document commissioning process.

Visual tools mentioned above suppose to be lightweight and user friendly. Nothing is more user friendly the Drag and Drop principle. It means that operator has a set of pre-designed blocks, which he can visually place on the project and make an application in minutes. The convenient way to monitor those applications is World Wide Web.

ROLES AND RESPONSIBILITIES

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

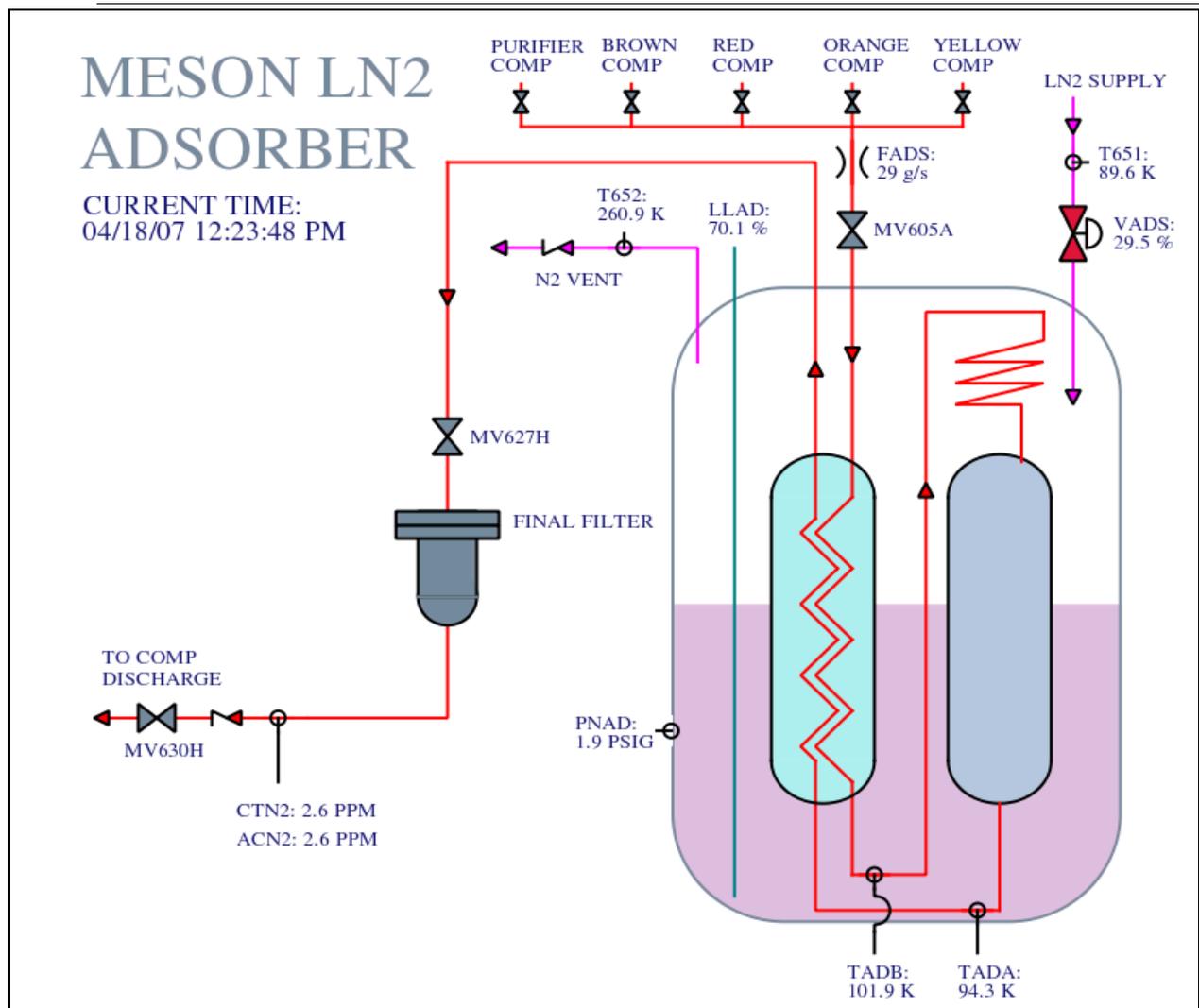
1. **General public** – anybody in the World. Should be able to watch the Web displays (with some restrictions – the number of possible applications and the traffic may be limited).
2. **Operator** – suppose to be able to build DnD applications in Builder (after proper authentication) and save them locally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

PROPOSED ARCHITECTURE

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. Number of similar systems is in use in Fermilab (1-Synoptic, 2-??), CERN (3 old Synoptic, 4 XML Knob) and other High Energy Physics laboratories as well as on other scientific and industrial installations (5 - MEDM, 6 APACS+, 7 PVSS). Discussed set of tools is aimed to be used in the Large Hadron Collider (LHC).

The architecture of proposed DnD is an upgrade / refactoring of similar Fermilab system (1-Synoptic) with consideration of CERN software system and expertise, collected in CERN for the (3 old Synoptic).

This set of tools should allow for building graphical application (pic 1) with reflections of CERN device readings in 0.5 - 3 hours. The application, designed in DnD, supposes to be able to reflect visually current device reading in the web browser and as standalone Java application with update rate ~1-3 sec. It suppose to be possible to start such an application as standard CERN Java program with settings enabled or disabled. Settings may be enabled only if application is started as standalone Java program. Configuration files produced by Builder also may be used by other not Java applications. The settings enabling policy suppose to be in accordance with CERN security model (8 RBAC).



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

The graphical application, mentioned above, suppose to build from blocks, or components, possibly connected between each other. Each component should implement simple common interface, the set of components suppose to be extendible, what means that every CERN operation specialist may develop his own components. We will try to minimize that activity by developing rich standard set of component. Each component supposes to have implementation (Java class implementing the interface) and description – XML file with formal description of its properties.

The standard set of components supposes to be stored in centralized repository. Applications, designed with DnD, suppose to be saved in the same (or similar) repository. The access to this repository supposes to be done by the World Wide Web. All the tools can be also stored as Java Web Startable Programs.

DND TOOLS

The set of tools, required for this project, immediately follows from the proposed architecture:

1. **DnD Builder** allows for visually building applications.
2. **DnD Runtime** allows for running the application as Java Program.
3. **DnD Repository** operates component repository and application repository.
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. FUNCTIONAL REQUIREMENTS

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

BASIC BUILDER CAPABILITIES

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

- | | |
|---|-----------------|
| 1.1 Be standard CERN Java application. | Critical |
| 1.2 Use RBAC to authenticate users / authorize actions (saving in Repository). | Critical |
| 1.3 Be web -startable. | Expected |
| 1.4 Read components from Repository and / or from local disk. | Critical |
| 1.5 Allow the user to build an application from Components. | Critical |
| 1.6 Save applications locally and / or in the Repository. | Critical |
| 1.7 Be user friendly and easy to learn ¹ . | <i>Critical</i> |
| 1.8 Start Runtime – if authorized. | Desired |

BASIC RUNTIME CAPABILITIES

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

- | | |
|---|----------|
| 1.9 Be standard CERN Java application. | Critical |
| 2.1 Use RBAC to authenticate users / authorize user actions (making settings). | Critical |
| 2.2 Be web -startable. | Expected |

- | | |
|---|----------|
| 1.9 Be standard CERN Java application. | Critical |
| 2.3 Read and start applications from local disk or from Repository. | Critical |
| 2.4 Web Presentation should be zoomable . | Desired |

¹ Can be **tested by** group of experts, **watching** DnD class or new operator learning DnD.

BASIC REPOSITORY CAPABILITIES

Repository is a part of DnD's web-tier. It should:

- | | |
|--|----------|
| 3.1 Provide grouping of components and individual component descriptions . | Critical |
| 3.2 Provide applications list and individual application description. | Critical |
| 3.3 Save application if authorized. | Critical |
| 3.4 Keep versions of stored applications. | Desired |

BASIC DISPLAY CAPABILITIES

Display is an AJAX web application, part of DnD web-tier. It should:

- | | |
|---|----------|
| 4.1 Display any application on standard web browser worldwide. | Critical |
| 4.2 Not allow settings . | Critical |
| 4.3 Use low traffic for updates (0.5 – 5 kb/s) . | Expected |
| 4.4 CERN users should have preference over external users. | Desired |
| 4.5 Presentation should be zoomable . | Desired |

BASIC SET OF COMPONENTS

Each **component** is a Java class implementing simple **interface**. Each component should also have an **XML file**, where its **properties** will be described. Designing the set of components is extremely important because set of components will define how useful the whole application is. Individual component may or may not be connected to other components by the pipes.

Components functionality should be “data driven” or “presentation driven” – so, security or “save as image” cannot be implemented as component. Such a functions should be implemented in Runtime.

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

- | | | |
|-----------------------|---|----------|
| 5.1 JAPC / LSA | Reading data from hardware devices and sending this data by the pipes for processing, or writing data from input pipe to device. The proposal is to make these components invisible . | Critical |
| 5.2 AscBeans | All CERN AscBeans from 'old synoptic' should be supported as components library. | Critical |

5.1 JAPC / LSA	Reading data from hardware devices and sending this data by the pipes for processing, or writing data from input pipe to device. The proposal is to make these components invisible .	Critical
5.3 Visualization	Reflecting the data: text fields of different kind, tables, scopes, histograms, data viewers, thermometers and barrels with floatng level etc. May serve as pipes .	Critical
5.4 Processing Pipes	Reads data from input pipes, transforms that data and send transformed data to output pipes. The proposal is to make these components invisible. Examples: fft, arbitrary expression, averaging, integrating, etc.	Critical
5.5 User Input	Text fields, knobs, sliders etc.	Critical
5.6 Static Graphics	Special symbols of different kind.	Expected

There may be more types of components – for example in Fermilab there is a special type for Cryogenic department. The set of components that will write/read to/from some database or file with predefined name. But that set is considered as required minimum.

IMPORTANT IMPLEMENTATION NOTES

1. In order to support **data coherency** Component libraries may be separated into several “component palettes” provided by several repositories. That is needed to provide user with clear understanding that all data on the page are provided with in same **Context** in terms of **timing – cycle – supercycle**. As a suggestion datasources from AscBeans, probably, should not be mixed with JAPS/LSA datasources or such a mixing shoud have definite and clear visual distinction.
 2. Concept of **Repository** allows to separate general–purpose Builder and Runtime from implementation about how and where the actual application configuration is stored. That allows for above item 1.
 3. **Zoom** is a desired feature, it may be supported only partially – for example only on some sets of componets or only in the Web version. Implementation will try to support it generally, but it would not become a first–order priority.
 4. All the Java applications suppose to be written according to CERN standards. This have to be elaborated to describe what we really mean.
-

SCOPE, PERFORMANCE AND CONCURRENCY

The scope of DnD is wide – all of the LHC architecture may potentially be represented as components. Not all of these components will be possible to reflect on the Web page. So, we cannot now define the scope, because it will be defined component developers.

We can define several **performance** related **tricks** here; despite they belong to the “DnD **Implementation**” document, which suppose to follow these Requirements:

- 6.1 To minimize **network traffic** Display should usually transmit only **differenced** of visual state, not the visual state itself.
- 6.2 To minimize **server load** Display suppose to start one project and **share** its visual state between web users.
- 6.3 To minimize **server load** Display suppose to **stop** application on server side after several minutes without requests.

6.4 To minimize **network traffic** for users with IE / Adobe Display should send SVG in **zipped** format.

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

3. REFERENCES

1. "Synoptic display - a client-server system for Graphical Data Representation", proceedings of ICALEPCS'03,
<http://synoptic.fnal.gov>
2. Lexidata application, Brian Hendricks, FNAL
<http://www-bdnew.fnal.gov/operations/fxtar/lexhow2.html>
3. "Old Synoptic"
<http://ab-dep-co-ap.web.cern.ch/ab-dep-co-ap/dev>
<http://oraweb.cern.ch/pls/abc/w3cons.applics>
4. "XML Knob"
5. MEDM:
<http://www.aps.anl.gov/epics/extensions/medm/index.php>
6. APACS+:
<http://www2.sea.siemens.com/Products/Process-Automation/Product/APACS/APACS>
7. PVSS:
<http://itcobe.web.cern.ch/itcobe/Services/Pvss/welcome.html>
8. RBAC documentation: Requirements, Specifications, Reports.