



*Document Title*

**FAIR Control System Development Guideline  
"GUI Development Guideline"**

*Document Name*

**F-DG-C-02e**

*Date yyyy-mm-dd*

**2012-09-07**

## **Abstract**

This document is the GUI Guideline for the FAIR accelerator control system.  
This guideline targets all Java GUI development for applications for the Control Room.

**Table of Contents**

1.	Purpose and Classification of the Document .....	4
1.1.	Responsibilities .....	4
2.	Scope of this Development Guideline .....	4
3.	Introduction.....	5
4.	Application Development .....	6
4.1.	Third party libraries .....	6
4.2.	Reuse .....	6
4.3.	Logging.....	6
4.4.	Configuration .....	6
4.5.	Language.....	7
4.6.	User Input.....	7
4.6.1.	Mouse Interaction.....	7
4.6.2.	Keyboard Interaction .....	8
4.6.3.	View / Edit.....	8
4.6.4.	Entering values .....	8
4.7.	Design Patterns .....	9
4.8.	Code Style .....	9
4.9.	Documentation.....	9
4.9.1.	User Manual.....	9
4.9.2.	Online Help .....	9
5.	Application Layout .....	10
5.1.	Main Window Layout.....	10
5.2.	Secondary Window Layout (Popups, Dialogs) .....	12
5.2.1.	Message Window .....	12
5.2.2.	Progress Window .....	12
5.2.3.	Input Dialogs .....	12
6.	Visual Design .....	12
6.1.	Colors .....	12
6.1.1.	Desktop Background.....	12
6.1.2.	Application Background Colors .....	12
6.1.3.	Application Colors in general.....	12
6.1.4.	Graphical Elements and Lists.....	13
6.1.5.	Graphs and plots.....	13
6.1.6.	Text Fields .....	13
6.1.7.	Device status colors .....	13
6.2.	Window Layout .....	14
6.3.	Text Labels .....	14
6.4.	Fonts.....	14
7.	Icons.....	14
7.1.	Icon sources .....	14
7.2.	Standard icons.....	14
8.	Menus .....	14
8.1.	Types of Menu .....	14
8.1.1.	The Menu Bar .....	14
8.1.2.	Context Menus .....	15
8.2.	Standard Menus .....	15
8.3.	Accelerators.....	15
9.	Toolbars .....	15
10.	Control Elements .....	15
10.1.	Toggle button:.....	15
10.2.	General components.....	15

**Document Title:** GUI Development Guideline

10.3.	Graphs and plots .....	16
10.4.	Lists .....	16
10.5.	Filling Level Indicator .....	16
11.	Feedback.....	17
11.1.	Textual Information .....	17
11.2.	Responsiveness of Applications.....	17
12.	Integration into the Environment.....	18
12.1.	Console Menu Entries.....	18
12.2.	Central Selection Switching .....	18
12.3.	Configuration from the Console Environment.....	19
I.	Related Documentation .....	20
II.	Document Information .....	20
III.1.	Document History .....	20

**List of Tables**

Table 1:	Color Definitions.....	13
----------	------------------------	----

**List of Figures**

Figure 1:	Application Structure.....	10
Figure 2:	Common Accelerator Selector .....	15
Figure 3:	CERNs JDataViewer.....	16
Figure 4:	CERNs ItemListPanel, prefilled with a device list.....	16
Figure 5:	ABeans from Cosylab for value display, e.g. of transformer current....	17

## 1. Purpose and Classification of the Document

The purpose of this document is to specify the development guideline as a directive for designing and developing control room applications for FAIR. Adherence to this guideline will improve product quality and maintainability of the FAIR accelerator control system.

The development guidelines complement the technical guidelines and detailed specifications for the FAIR control system in providing general rules and regulations for control system development.

Whenever regulations and requirements are specified in the General Specifications, Technical Guidelines, Common Specifications or Detailed Specifications of the Control System they are only referenced in this document. The related documents are listed in Appendix II.

No legal or contractual conditions are treated in this document. All related information is given in the General Specifications for FAIR II.

### 1.1. Responsibilities

The responsibilities with respect to changes and modifications of the present document are entirely in the hands of the Accelerator Controls and Electronics Department of the GSI Helmholtz Centre for Heavy Ion Research GmbH (GSI) Darmstadt. For initial information please contact the administration of the Controls Accelerator Controls and Electronics Department.

Further information on the organigram, names of responsible persons and task leaders, as well as the agreed document release and approval procedure is summarized in the organizational note 'Controls Project for FAIR'.

## 2. Scope of this Development Guideline

This guideline targets all Java GUI development done for the FAIR accelerator control system that results in standard applications supposed to be running in the Control Room.

This GUI guideline document cannot cover everything related to GUI development; instead it is always work in progress. New requests for clarification may always lead to additional chapters.

The most recent version of this guideline, including example applications can be found on the Wiki Site of the Accelerator Controls and Electronics Department:

<https://www-acc.gsi.de/wiki/Applications/APJavaDevelopment>

In general, all software development must adhere to the Accelerator "Software Architecture Guideline", see [2].

A general overview about the Java development environment can be found in the document "Java development environment". Please refer to this document for further information about infrastructure related topics.

### 3. Introduction

The following chapters describe GUI Design principles and best practices. In order to achieve a homogenous look and feel of the control applications in the control room, it is necessary that applications are built using a common standard. All application development should adhere to this standard.

Along with this standard there are some principles that should be kept in mind when designing applications.

- If you are designing an application keep the audience in mind you aim the application at. Design your application for the requirements of that audience, it is important to understand their tasks and the goals they try to achieve. For example an application to draw diagrams might look quite different if you aim at software developers, engineers or children.
- Try not to limit your user base. Keep in mind impairments like color-blind people and also design your application so that it can be used by speakers of a language different from yours.
- You must also keep your user informed about what is going on. The user feels safer if he knows what is happening. After a user action provide feedback that the system is operating on the input and (if possible) the status of the operation.
- An application must not block a user, even if working on a task. While e.g. displaying a progress for a task it might be possible for the user to work on another task. The user might even decide to cancel one task in favor of a different one. The user should be put (and feel) in control and not be blocked by modality.
- Since the user is in charge and we all make mistakes, there should be either a possibility to undo the results of an action or if that is not possible warn the user and ask for confirmation. A confirmation should be used only where it is necessary (e.g. dangerous actions) because users tend to ignore them if they appear too often.
- Where it is possible allow the user to interact with the application directly instead of showing dialogs. E.g. if you have a graph of a function allow the user to interact with the function like adding point or dragging them to a different value rather than showing him a separate dialog where he can input function values.

There are a lot of good advices on human interfaces and in general it is worth to have a look on the web before starting your concept.

## 4. Application Development

In general, GUI development is done using Java Swing.

It is not defined yet, whether a dedicated look and feel is provided for the FAIR control system. At the moment it is assumed, that all applications use the standard look and feel and if possible don't hardcode any screen and font sizes (which makes it easier to later introduce a standard into existing applications) – if absolutely necessary, use relative sizes instead of absolute ones.

### 4.1. Third party libraries

In general, standard GUI elements from Swing are being used.

For specific needs, additional libraries on top of Swing may be used. Recommendations are listed here:

- CERN GUI libraries (e.g. ItemListPanel and JDataViewer)
- JGoodies: focuses on nice and usable GUIs using Swing. JGoodies (especially jgoodies-forms and jgoodies-looks) should be considered for nicely looking GUIs or when dealing with rather complicated forms with many input fields. More information can be found here: <http://www.jgoodies.com/>
- ABeans from Cosylab (see appendix)

All mentioned libraries will be available on the release server. The use of additional libraries not yet present on the release server has to be approved by the Accelerator Controls and Electronics Department. Additional libraries can be added, but a thorough check if already a library with similar features exists and about the licensing scheme of the new library needs to be done before.

### 4.2. Reuse

Parts of new development, e.g. specific GUI components or panels, which are already foreseen to be reused by other applications, should go into certain standard packages, e.g. gsi-util. Please contact the Accelerator Controls and Electronics Department, whenever standard components are missing or if you have a component that might be of interest for others.

### 4.3. Logging

All applications should log information with different log levels. For Java applications, the control system logging framework must be used.

It is planned to collect logging information centrally in order to correlate information.

### 4.4. Configuration

A central configuration shall be used for all applications, that ensures using the same directory for files (written files, screenshots), using a central printing configuration etc.

## 4.5. Language

Language of the applications should be English and German, best provided by using resource bundles. Besides the translation feature, using resource bundles makes it much easier to maintain the application, e.g. later on change a GUI text just by editing the resource file. The language of the application must be switchable at runtime!

## 4.6. User Input

### 4.6.1. Mouse Interaction

#### 4.6.1.1. Buttons and their Behavior

In general, the mouse provides the main way of interacting with the graphical applications. The term “mouse” is used here for all devices, that can be used to move the pointer around the screen, this includes also touch panels (where the mouse pointer is simulated in accordance with the finger on the screen) or other pointing devices that might also only represent some parts of the functionality of a real mouse (e.g. some device that simulates clicks).

The left button of a standard mouse is typically used for selections and mouse actions for right-handed users. Even though this might be configured otherwise, this button is hereafter referred to as the “left mouse button”. The right button of a standard mouse is used for context sensitive operations, such as pop-up menus, this button is therefore referred to as “right mouse button”.

A conventional mouse typically offers a third button, typically combined with a scrollwheel, that acts as a button when pushed. This “middle mouse button” is typically used for copy&paste-actions and the scrollwheel for scrolling the window beneath the pointer.

In general, these typical functionalities of the mouse buttons should also be supported by the applications, to provide the user with the same user experience like standard operating systems or commercial applications if possible. Even if not all typical features are supported, it must absolutely be avoided to use mouse buttons in some contrary way (e.g. always use the left mouse button for selection, not one of the other two)!

Every action that is accessible via mouse must also be accessible via keyboard. For more details about keyboard interaction, refer to the next section.

No action must rely on a combination of several mouse buttons, besides being hard to use for a user, it cannot be guaranteed, that every input device provides these buttons.

It must not be assumed, that the middle or right mouse button is always there. Therefore do not depend on input from those buttons, instead make every action that is available through those buttons also available through the left mouse button or the keyboard, e.g. by using appropriate menu entries.

In general, multiple clicking to achieve some action must be avoided, everything should be available by using single-clicks.

#### **4.6.1.2. Selection**

Selection behavior is mainly already standardized by using Swing for creating graphical user applications, i.e. using left mouse clicks as selection action, Ctrl + left click for selection / deselection, Shift + left mouse click for multiple selection, etc. The application developer must ensure that these standard mechanisms are not blocked in any way, i.e. by not overriding this behavior. It must be defined, whether single or multiple selections should be possible, accordingly the application must possibly support multiple selection.

Specific care must be taken in components like tables, trees etc., that selection always works in a seamless way.

Individual types of selection mechanisms might be useful but should be used only where it leads to some form of conform and unified look and feel for the user. (e.g. when the user selects something and then right-clicks on some other object: instead of undefined behavior of the context menu, the application might also switch the selection focus to the element that is present under the mouse cursor).

#### **4.6.1.3. Drag and Drop**

Allow all mouse operations to be cancelled before their completion. Pressing the Esc key should cancel any mouse operation in progress, such as dragging and dropping a file in a file manager, or drawing a shape in a drawing application.

### **4.6.2. Keyboard Interaction**

#### **4.6.2.1. TODO**

#### **4.6.2.2. Tab Behaviour**

TODO Focus

### **4.6.3. View / Edit**

Wherever possible, allow users to act on objects and data directly, i.e. if data can be displayed and edited, the screen should always directly allow editing (no “Edit” button!). Instead, changing the selection or hitting the cancel button should revert any unwanted changes.

### **4.6.4. Entering values**

If possible, avoid lists and tables for displaying and entering data.

As data entry field, use digit wheels, sliders, curves, graphical plots instead of text fields or – if both ways are possible – the graphical data entry should be the default.

All values should have labels and units (including the chart axis)!

For too big or small numbers the scientific number format should be used. In general, numbers should be displayed with and only with the necessary accuracy especially in the fractional part. The same property should have the same accuracy while entering or viewing its data.



## **4.7. Design Patterns**

GUIs design is based on the MVC pattern. The Model-View-Controller pattern is assumed to be well-known, more documentation can be found here:

<http://en.wikipedia.org/wiki/Model-view-controller> (MVC pattern in general)

<http://java.sun.com/blueprints/patterns/MVC-detailed.html> (MVC for Java)

The MVC pattern promotes separation of concerns, different software components concentrate on things they are responsible for, which in general leads to better structured software that is easier to maintain. Using this pattern leads to GUIs, that can be easily adapted or changed without touching the business logic or the data model, which is necessary for long term maintenance. An example GUI application, that adheres to the MVC pattern, will be provided by the application group of the Accelerator Controls and Electronics Department.

## **4.8. Code Style**

A common code style makes code more readable for colleagues and therefore eases maintenance effort. General code style guidelines and best practices can be found at TODO...

## **4.9. Documentation**

Besides program documentation of the written software, for all GUI programs, the developer needs to provide documentation for the end user.

### **4.9.1. User Manual**

The User Manual should describe the application with all its software features. One of the following documentation types is requested:

- Static description: describing possible user activities and program functionalities through applications windows, buttons or menus
- Workflow-oriented description: covering all possible workflow scenarios by going through different "screens" and only describing those GUI components, that are relevant for the given scenario

TODO PDF??

### **4.9.2. Online Help**

The online help is typically based on (excerpts of) the User Manual. The format of the online help is plain html, including pictures. The general online help should be reached from applications by choosing "About -> Help" from the menu, additionally context sensitive help may be provided, that directly positions the help screen on a given topic within the online help.

A standard "help component" for displaying html-content will be provided, that works in the context of applications.

## 5. Application Layout

### 5.1. Main Window Layout

Applications should not use fixed panel or frame sizes. Instead it is mandatory, that the size and screen to position the application can be set from the outside during startup.

Applications are typically configured in such a way, that they always come up in the same screen (a console typically has several screens), which is chosen by the operators and configured in the console environment.

The application structure itself should if possible be the standard structure shown below. It contains:

- header part with program name and version
- menu bar, including on the right side the time in minutes (necessary for error diagnosis and as keep-alive signal) and a button for creating a screenshot (shown as square icon)
- top part, representing the current selection (e.g. virtual accelerator)
- middle part containing the main part of the application, e.g. details for the selection
- bottom console for status information, showing also the actions done by the user

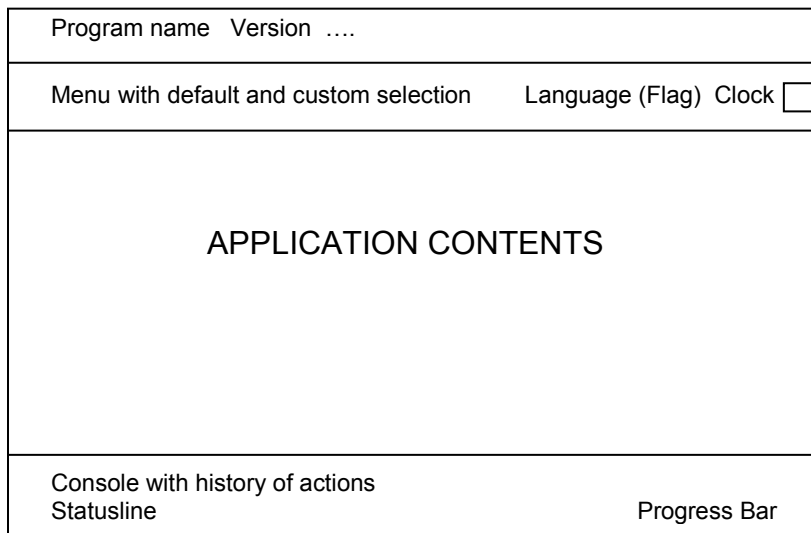


Figure 1: Application Structure

In the menu bar of each application, central selection options should be present in a standard order wherever necessary. Applications, that can be used for several accelerators (e.g. SIS18, SIS100, SIS300) should include here the accelerator selector.

Applications that can display data of several beams should present only those beams as choice that are available at the moment.

**Document Title:** GUI Development Guideline

---

Those standard selectors for accelerators, beams etc. will be provided by the application group of the Accelerator Controls and Electronics Department and should be used instead of reinventing them for each application. This guarantees a standardized look and feel in all applications.

The application itself should in general consist of one window (structure see above), where the most important part of the application should always be visible. More than one main window should be avoided, also the opening of additional windows or too many Popup-Windows. If several different things are to be displayed, using SplitPanels or tabs is preferred (instead of opening other windows). If applications have to open other windows, the behavior of the console environment should be, that if the application is send to the background it can be easily brought to the foreground again together with all its windows. (This point has to be supported by the console manager, rather than by the applications.)

An application should adhere to a clear, structured ordering of its components on the screen. Different data sets on the same screen should be distinguishable by color and / or line style. The reading and editing flow is similar to reading a book: it goes from left to right starting in the left upper corner.

The application should relate to this reading flow:

- components that present an overview or that present selectors should be placed in the upper part of the applications, since they will be noticed first
- lower parts of the application show details
- entering data has to go along the reading flow: no entering of data may affect fields left or above it, but only behind the current field (with respect to the reading flow)

All applications should be scalable. Where possible, use the standard SplitPane, the CERN MultiSplitPane or the like, so that the user is able to enlarge parts of the application, which he is most interested in.

In general, no GUI element should have fixed screen sizes, fixed font sizes, fixed fonts etc. Elements can and even should have a minimum size set, so that e.g. buttons are not too small.

TODO Tabs may be used

TODO As container for the new Application, the CERN Frame should be used. This already provides a frame with menu bar and error / status console.

## 5.2. Secondary Window Layout (Popups, Dialogs)

### 5.2.1. Message Window

TODO Description of Popups used as Information panel, error popup, CERN popup by CERN as error email

### 5.2.2. Progress Window

TODO Example. What to do with non interruptible tasks. Stop button becomes red. Reasonable behavior.

### 5.2.3. Input Dialogs

TODO SSH dialogue as example. OK Cancel. Inputfield validations should not prevent cancel.

**Dialogs** to choose application parameters should have the following buttons (EN / DE):

- Select / Übernahme (position: bottom left of the dialog)
- Cancel / Abbruch (position: bottom right of the dialog)
- and if possible: Default / Standardeinstellung

## 6. Visual Design

### 6.1. Colors

In general, standard color schemes can already be achieved by using the standard look and feel. A class with constants for colors will be provided by the Accelerator Controls and Electronics Department. If however, for some reason, custom colors have to be used, the following rules for colors apply.

#### 6.1.1. Desktop Background

Default Background of any screen without showing any application should be black.

#### 6.1.2. Application Background Colors

Colors in general should be eye-friendly and should not generate too high contrast with very bright or dark elements. Instead a light grey should be used (e.g. the Swing standard grey), so that structures in white, yellow, dark-blue or black are equally easily visible.

#### 6.1.3. Application Colors in general

Colors used in graphical elements should always resemble a certain meaning. E.g. red should be used only for errors and not as a standard color. For just displaying data neutral colors without special meaning are preferred (white, black, gray, dark blue).

**6.1.4. Graphical Elements and Lists**

Lists should slightly differ from the application background, using white or light grey as background. Selections should be clearly colored, but should not be too bright.

**6.1.5. Graphs and plots**

For displaying graphs, plots, bar charts etc. standard axis colors are black axis description on gray background. For the data inside the chart, the following rules apply: While displaying neutral data, neutral colors should be used, colors that imply a certain meaning (e.g. red), are reserved only for this purpose (e.g. error).

**6.1.6. Text Fields**

For text fields, it has to be clearly distinguishable between entry fields (white background) and text fields and labels, that just display data (gray background).

**6.1.7. Device status colors**

Use the standard status GUI component, which can be found on the Wiki page linked above. If it is not possible to integrate those components, the minimum requirements for displaying status are:

- currently, the common device statusbits are:
  - bit 0 power on/off
  - bit 1 remote /local
  - bit 4 emergency
  - bit 5 interlock
  - bit 6 hardware warning
  - bit 7 software error
- common color definitions (some are device dependent):

red	device power off slit/actuator inner end position in beam reached
blinking red	power off but has to be on slit/actuator moving
green	device ready slit/actuator outer end position reached
dark green	device not connected to control system ( vacuum pumps...)
blinking green	device inactive for this virtual accelerator, but has to be active
orange	slit/actuator positioned in beam (not out nor in)
yellow	device/controller offline or unknown
blinking blue	Hardware error

Table 1: Color Definitions

## 6.2. Window Layout

TODO

## 6.3. Text Labels

TODO

## 6.4. Fonts

TODO No fixed size, only +2, i.e. relative.

# 7. Icons

## 7.1. Icon sources

TODO: insert reference to Java Sun Icon list. Icon Loader.

## 7.2. Standard icons

TODO

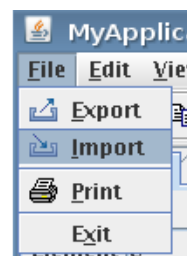
# 8. Menus

## 8.1. Types of Menu

### 8.1.1. The Menu Bar

The menu bar of an application typically combines the user interaction that is possible within a given application. The functionality should be grouped in meaningful groups. Menu entries must contain accelerators (e.g. Strg+S) to be also accessible only with keyboard interaction.

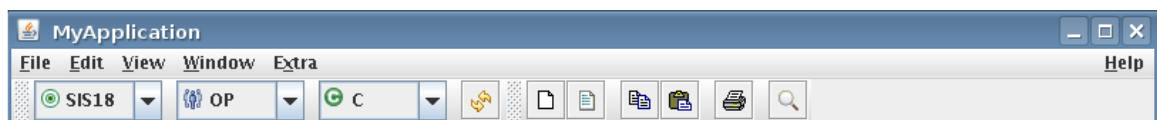
Menus should in general always contain the same, i.e. all entries. If some functionality is not supported in some situations, menu entries should be greyed out, or a warning/error should be displayed when selecting this functionality.



TODO

### 8.3. Accelerators

TODO Appearance and Content, Controlling Display and Appearance, Labels and Tooltips, frequently used functionality can be provided as toolbar icon (max. 10), example



### 10.1. Toggle button:

TODO typical on/off problem, not defined yet

Standard classes and standard GUI components that can and should be used will be provided by BEL-AP.



Figure 2: Common Accelerator Selector

Those components already adhere to the GUI guidelines and already help achieving a common look and feel and behavior. They are maintained centrally and by using them, it is possible to speed up individual development and lower maintenance effort.

A list of provided components will be put on our BEL-AP Wiki.

More detailed information on how to use such a component can be found on the Wiki site (directly or as examples) and within the Javadoc documentation of those components.

### 10.3. Graphs and plots

For displaying graphs, plots, bar charts etc. the CERN jdataviewer is to be used. Doing so ensures a standard look & feel, standard axis colors and background colors.

TODO But check contrasts of displayed plots (line width, background)

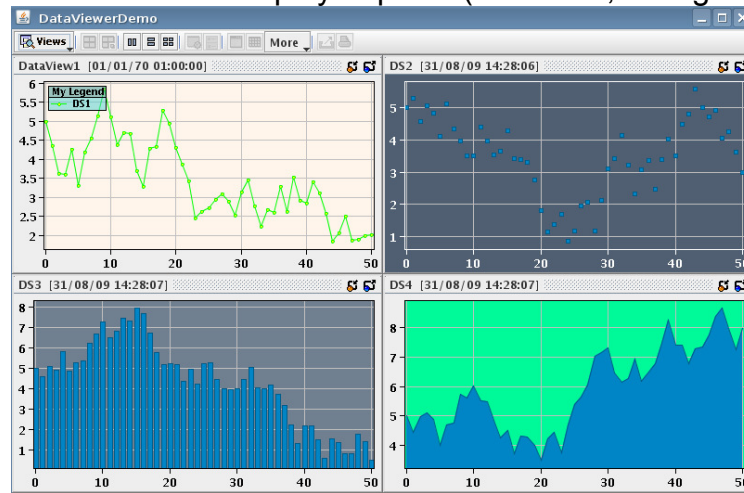


Figure 3: CERNs JDataViewer

### 10.4. Lists

For standard lists, the CERN accsoft.gui.frame.ItemListPanel shall be used, which provides a standard-colored list with features like the “select all” Button and if needed a filtering field, also single or multiple selection can be used.

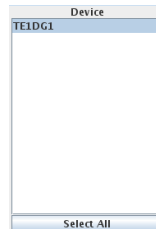


Figure 4: CERNs ItemListPanel, prefilled with a device list

### 10.5. Filling Level Indicator

TODO examples for Cosybeans



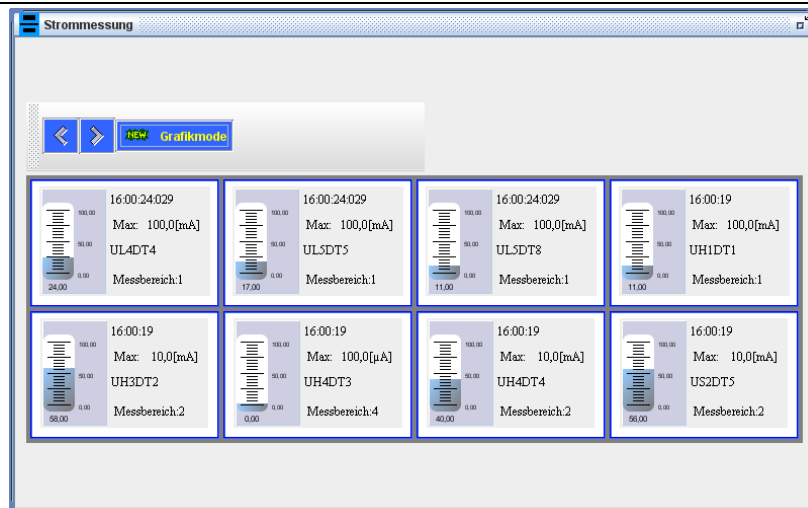


Figure 5: ABeans from Cosylab for value display, e.g. of transformer current

## 11. Feedback

### 11.1. Textual Information

Applications in general as well as specific GUI elements are supposed to have a standard behavior, a standard look-and-feel. Besides a standard color scheme, it is mainly this GUI behavior that leads to “standards” among applications.

Predefined and prefilled GUI elements will be collected on the BEL-AP Wiki Site: <https://www-acc.gsi.de/wiki/Applications/APJavaDevelopment>

- **Device messages:** For displaying messages, please use the jars provided by the condition message project. The corresponding jar can be found on the release server.

### 11.2. Responsiveness of Applications

Highly responsive applications put users in control by quickly acknowledging each user request, by providing continuous feedback about progress toward fulfilling each request, by letting users complete tasks without unacceptable delays, and put the user in charge of tasks that have been started but not finished.

If possible requests are queued and prioritized as the user would expect it, request that are no longer relevant (e.g. because another request makes an older one obsolete) are discarded.

Application responsiveness is not related to the speed of its code or solving tasks. Even an application with long running tasks can be provide responsive feeling to the user.

Some user interface events require shorter response delays than others. For example, an application's response to a user's mouse click or key press needs to

**Document Title:** GUI Development Guideline

be much faster than its response to a request to save a file. The following table shows an example in values to give you an impression of time scales:

<b>Example for UI Event</b>	<b>Maximum Response Time</b>
Events that require hand-eye coordination like mouse click, pointer movement, keypress, drawing.	0.1 second
Short (background) tasks like displaying a progress indicator, closing a window, reformatting a table	1.0 second
Displaying something the user would expect to take long like all voltages for magnet along a beam line, searching for a file or data	10.0 seconds

The `cern.acccsoft.gui.frame.ExternalFrame` provides a status indicator for tasks running in the background. It is displayed on the bottom right of the application and can be clicked to display detailed information about each task and the ability to cancel tasks.

Also `cern.acccsoft.gui.frame.Task` provides the possibility to give tasks a type and to indicate if tasks of the same type should “overwrite” each other.

TODO More information on implementation and pictures.

## 12. Integration into the Environment

The control room consists of a set of operator’s consoles. Each console comprises several screens. The console environment within the control room presents an integrated work environment to the users. All applications that are needed for operating are launched and presented within this environment. Therefore, new applications need to be integrated into this environment. More information on the console environment system can be found in

### 12.1. Console Menu Entries

Integration with the user environment is done by adding a new entry in the menu of the operator’s console. Before doing so it is necessary to define:

- where the application should be used (everywhere or only for a specific accelerator)
- where in a given console the application should open i.e. with which other applications the application is typically used together

After answering those questions, the adding of the application to the menu is done via a central configuration by the application group of the Accelerator Controls and Electronics Department.

### 12.2. Central Selection Switching

The console environment not only provides means for providing and launching applications, it also allows central selections that are then taken over by all applications currently running in that console. That helps the user change the focus

### 12.3. Configuration from the Console Environment

TODO links

<http://www.gsa.gov/graphics/staffoffices/508softwareanddos.doc>  
(flashing frequency)

## I. Related Documentation

- [1] F-CS-LS-01e, FAIR Common Specification "Accelerator Control System"
- [2] F-DG-C-03e, "Software Architecture Guideline"
- [3] F-DS-C-22e, "Console Environment System"

## II. Document Information

### III.1. Document History

Version	Date	Description	Author	Review / Approval
0.1	17. Feb. 2009	First version	Fitzek, Fröhlich, Müller	
0.2	06. Nov. 2009	Draft	Fitzek, Fröhlich, Müller	
0.3	03. Aug. 2011	Draft, splitted into two documents	Fitzek, Fröhlich, Huhmann, Jülicher, Müller	BEL-AP
0.4	31. Jan. 2011	New document structure, included input from other GUI guidelines	Fitzek, Müller	
0.5	21. Feb. 2012	Work in progress	Fitzek, Müller	
1.0	10. Aug. 2012	Draft	CCT	CCT
1.1	07. Sep. 2012	Draft	CCT	CCT