109020500-TD0001

Spallation Neutron Source

EPICS ArchiveViewer Version 1.0.4

May, 2005

A U.S. Department of Energy Multilaboratory Project

SPALLATION NEUTRON SOURCE

Argonne National Laboratory • Brookhaven National Laboratory • Lawrence Berkeley National Laboratory • Los Alamos National Laboratory • Oak Ridge National Laboratory

SNS

Contents

Foreword

Disclaimer

Please note that I have made a considerable effort to ensure that the information in this document is up-to-date and accurate, but I cannot guarantee that it does not contain errors. You must use this document at your own risk or not use it at all.

Homepage

The ArchiveViewer web site can be found at http://ics-web1.sns.ornl.gov/archive/viewer. It has all the latest information about ArchiveViewer, incl. screen shots, download links, JavaDocs, and parts of this document.

Thanks

I would like to thank the whole EPICs community for your support and the following colleagues in particular:

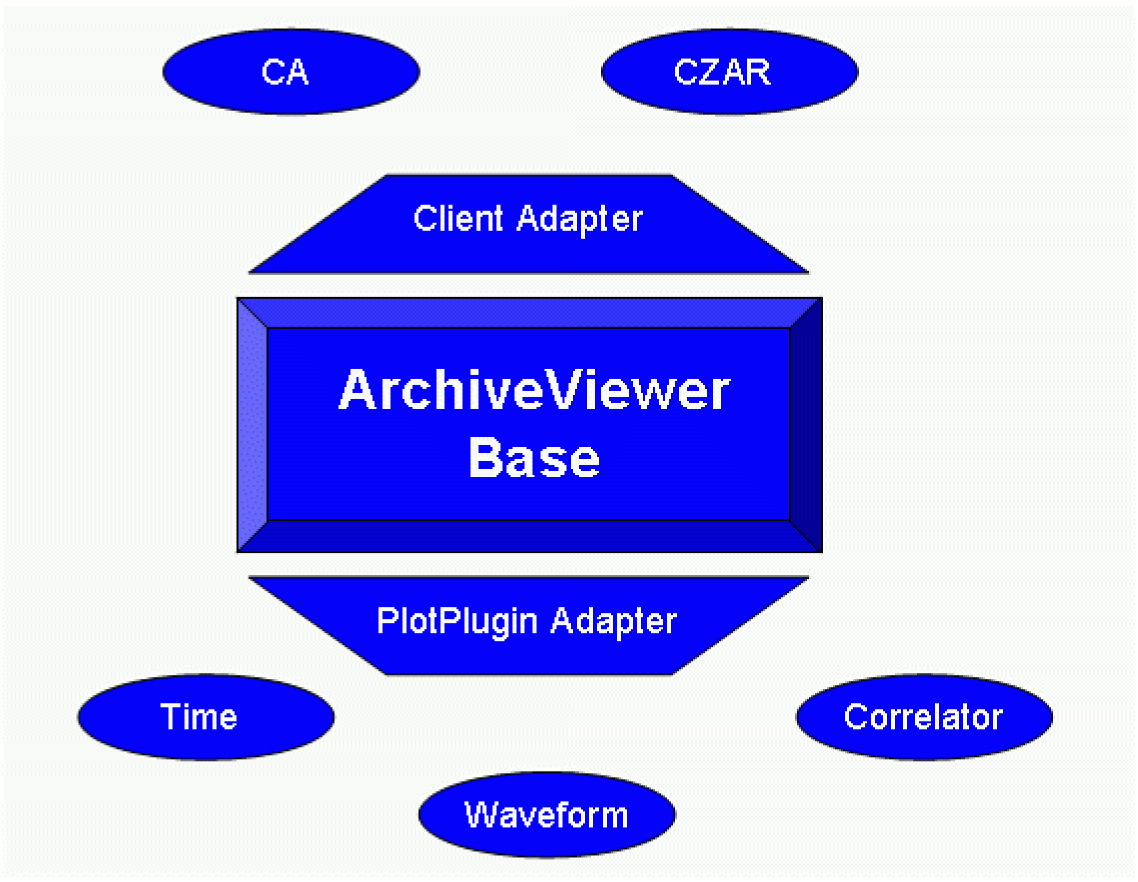
Kay Kasemir, my mentor;
CraigMcChesney, who was part of the ArchiveViewer development team in LANL;
Ike Kritznar from CosyLab, who helped me to cut edges with Java;
Greg Lawson for administrative support;
Hamid Shoaee and Dave Gurd for being such nice managers;
Paul Wright, Herb Strong, Sheng Peng, Johnny Tang, Thomas Birke for testing the hell out of this application;
Bob Dalesio for giving me guidance within and outside the community;
and, of course,
Ernest Williams for being the best colleague and friend one can imagine!

Overview

What is ArchiveViewer?

ArchiveViewer is a free, extensible, multi-threaded client application for processing archived EPICS data. It is written in pure Java and designed to support different servers as well as to produce different kinds of visual output (time plots, correlation plots, oscilloscope simulators, spreadsheets etc.). The user can control the application via command line, a Swing GUI (fully integrated into the XAL framework), or Java Server Pages. ArchiveViewer is distributed with complete source code subject to the terms of the EPICS license.

Features



ArchiveViewer can query supported archive servers for PV data, meta information on PVs, archive settings etc. The retrieved data can take two paths: either be exported to text/binary formats, or plotted by chart widgets. Data from non-waveform, numeric PVs can also be used inside formulas, which the application eventually handles in the same way as genuine PVs.

We abstract the notion of PVs/formulas and use the term ArchiveViewer entry (short **AVE**).

ArchiveViewer also features:
- pluggable clients, plot widgets, and exporters
- different run modes (JSP, command line, XAL GUI)
- caching of retrieved data

- three plot styles (scatter, steps, and lines)
- handler for invalid/non-plottable data
- absolute and relative time queries
- multiple time and range axes
- persistent configurations and preferences
- search for PVs using regular expressions
- progress indicators
- formula creation (basic arithmetical and boolean operations; standard Java as well as min/max/avg functions)
- handlers for vertical and horizontal plot scroll, zoom in/out

Installation

ArchiveViewer was written entirely in Java and should run on JDK 1.4 or later.
Use ANT to build ArchiveViewer and create appropriate jar files (see Appendix B)

If you wish to use ArchiveViewer as Java Server pages, you need to install Apache's Tomcat on a web server (version 5.5 was used for development, but other versions should be supported, too). Execute "ant configure_jsp" in the main directory and start your Tomcat engine.

User's Guide

There are three basic ways to use the ArchiveViewer. One is to start an XAL GUI; another to run it in the command line mode; and the third to go to a server where it is running as a Java service. The actual invocation will differ from site to site. System administrators might want to create an "archiveviewer" command or desktop button that selects the appropriate java runtime environment and invokes the ArchiveViewer with certain default arguments. The included README and build.xml files contain specifics of how to compile sources into a JAR file and how to use it.

The GUI mode

This is the preferred running mode that offers all ArchiveViewer features to the user.
When you start the ArchiveViewer without '-nogui' option (or without any options at all, for that matter), it automatically creates an XAL GUI.

Note in general that there are features available behind every "…"-button.

1. Quick start

- **Main Components**

  ArchiveViewer consists of 4 major components:

  - The control panel contains the PV/formula configurator (on the left) and the axes configurator (on the right)
  - The tabbed pane at the bottom shows currently loaded plot plugins
  - Between the control panel and the plot plugins panel is a (narrow) status panel; it shows some important (dynamic) information. The status panel also contains a progress bar (on the right).
    Tip: Always take a look at the status panel, when you are not sure if a task is still running, or the program unfortunately crashed.

- **Main use case**

  - Go to FILE menu and select NEW CONNECTION (FILE => NEW CONNECTION)
  - Select (or enter) the connection parameter of an archive data server (e.g. a URL) and press OK
  - After server information was retrieved, you can select an archive directory in the main panel (note: the selected archive directory will automatically be assigned to formulas you create)

- Enter a search string into the search text field on the main panel; press the SEARCH button or hit ENTER.
  A search dialog pops up and allows you to configure your search. Its right half is a table that will eventually be filled with meta data from matching PVs (if it doesn't happen too soon, please take a look at the progress status => perhaps too many PVs match your search string)
- Select desired PVs and add them to the main panel by pressing the ADD button or double-clicking the particular PV row). Observe how the left side of the main panel now contains the selected PV names (in default colors).
- Enter time range for the MAIN TIME AXIS (times can also be entered through TIME SELECTORs that appear when you press a "..."- button next to corresponding time entry field)
- Now press the PLOT button and observe the retrieval progress
- After data is retrieved, you can access meta-information about PVs as tooltips on the main panel

2. Axes Configuration

You can configure time and range axes on the right side of the control panel

1. Time Axes

Enter either an absolute or a relative time string (see Time Queries below) into the time fields. Alternatively, you can use time selectors that appear when you press a "..."-button next to the corresponding text fields.
Tooltips are available on the time entry fields

The location box lets you select a location for your time axis (“bottom”, “top”, or “not visible” are the options).

2. Range axes

Enter numeric values into min/max text fields under the range axes selection box. ArchiveViewer understands a huge variety of number formats (e.g. *1e-1* is allowed) and some currently unsupported formats can easily be added in the future, if desired.
If you leave a range limit blank, the actual value is going to be "whatever it actually is".
You can specify the axis scale type and the location of a range axis by selecting the values in the correspondingly labeled combo boxes. Currently supported range axes types are NORMAL and LOG(arithmic). Range axes can be located on the “left”, “right” or be “not visible”.

3. Advanced axes features

You can access advanced axes features by pressing the "..."- buttons in the upper right corner of the axes widgets.
The features are:

- "add new axis" pops up a dialog and asks you to enter the unique label of a new axis.
- "remove selected axis" (note: you can not remove last axis)
- "rename selected axis"
- "update selected axis from active chart" updates the range of the corresponding axis to what is shown in the currently active plot plugin
- "update all axes from active chart" updates all axes ranges to what is shown in the currently active plot plugin
- "reset selected axis" resets the range of the corresponding axis
- "reset all axes" resets all axes ranges

3. Time queries

Time ranges can be entered either manually or through time selectors, which appear when you press one of the "..."- buttons to the right of the time text field. Even if you decide to never enter times manually, you should at least know that there *is* this short reference.

Absolute Times

You can enter absolute times for start and end times of a query. Many of absolute time components are optional. The minimal, mandatory, absolute time string must follow the pattern 'MM/dd' (where MM is the month, dd- the day of the month).
ArchiveViewer will insert following values for missing absolute time components:

- year: current year (as shown by your system clock)
- hours: 00
- minutes: 00
- seconds: 00
- msecs: 000

Keyword "now"

The application accepts the keyword "now" as end time. Milliseconds before the server request is actually sent, "now" is replaced by current system time. If you specified “now” as end time, and press the PLOT button without checking "Keep Plot Ranges", new data request will be sent because "now" is resolved to a later timestamp.

Relative Times

One of ArchiveViewer's most innovative features is its ability to parse relative time strings. However, as with all innovations, correct usage requires some learning time. Below, a short tutorial is presented.

1. Following tokens may be used for each of the relative time components (please, note that they indeed are <u>case sensitive</u>):
   - y - year
   - M - month
   - d - day
   - H - hours
   - m - minutes
   - s - seconds
2. For numeric values, only integers are allowed. For positive integers, you may also use the preceeding "+" sign, which would make sense semantically.
3. You can also use absolute time strings at the end of each relative *date* string.

Some examples of relative time strings:

- Legal

  "+1d", "6m", "-1M 6d", "-8H", "-2d 08:00" (note: no spaces are allowed between numbers and relative time tokens)

- Illegal

  "-1D", "+2 y", "2.5H"

4. Semantics

- The sole notion of "relative" time strings requires a reference point, and indeed there are so-called "base times". A base time is a timestamp relatively to which relative times are resolved. Although in very most cases, base times can be determined intuitively, some examples are provided below for questionable situations. We call a relative time string "positive", if it resolves to a positive number of milliseconds; and we call such string "negative", if the result is a negative number. Note that it makes no sense to use positive, relative time strings for the start time of a query. Thus, there are only 6 cases to consider:
  1. Start time is absolute; end time is absolute
     Example:
     Start: 01/01/2003 8:00
     End: 01/02/2003
  2. Start time is negative, relative; end time is absolute => base time: end time
     Example:
     Start: -10d

End: now
Means: *retrieve data from last 10 days*

3. Start time is absolute; end time is positive, relative =>
base time: start time
Example:
Start: 01/02/2004 10:00
End: +8H
Means: *retrieve data from January 2nd 2004 between 10am and 6pm*
4. Start time is negative, relative; end time is positive relative =>
base time for start time: "now"
base time for end time: start time
Example:
Start: -7d 8:00
End: +8H
Means: *retrieve data from the second daily shift of exactly a week ago*
5. Start time is absolute; end time is negative relative =>
base time: "now"
Example:
Start: 06/01/2004
End: -5d
Means: *retrieve data from the 1st of June up to 5 days ago from now*
6. Start time is negative relative; end time is negative relative => base time: "now" for both time strings
Example:
Start: -10d
End: -5d +8H
Means: *retrieve data from 10 days ago upto 5 days ago, plus 8 more hours of data*

5. Conclusion

Relative time strings are an intuitive and very useful feature, since they allow automatic processing of many practical configurations (just think how easily you can retrieve "data from yesterday's shift" every morning).

4. Formulas

One of the most complex features of the ArchiveViewer is the ability to use PVs from a common archive directory inside a formula. You start the creation of a new formula by pressing the "New Formula"- button on the main controls panel.

Following steps are recommended for creating a formula:

- Select argument AVEs from the supplied AVE list
- Assign a unique variable to each AVE (default variables will be x0, x1, etc.)
- Using these variables, type in your formula, possibly using the supplied calculator buttons.

Variables have two major advantages: resulting formulas will be short and thus easier to read; and complex formulas can be re-used for other AVEs without much hassle.

Supported mathematics

Besides basic arithmetics and boolean operations, all standard Java functions specified by methods of the java.lang.Math class as well as own min, max, and arithmetic mean functions (with indefinite number of arguments) are supported by the internal formula parser.

An example configuration using formulas can be found in the Appendix D.

5. Configuration of ArchiveViewer Entries

Initially, the ArchiveViewer assigns a default color, draw type, time/range axis etc. to each AVE. By pressing the "..."-button in the right column next to each AVE name, you can change any of the following:

- Axes assignment

  Labels in the axes selection boxes correspond to labels of the axes in the main control panel. If you want to plot a graph *normalized*, select the empty entry in the range axes box.

- Draw color

- Draw types

  - scatter
  - lines: connects plot item directly
  - steps: draws a horizontal line from one plot item towards the vertical line that crosses the next item, and then a vertical line to the next item. This draw type is default.

- Draw width

  You can adjust the width of graph lines with the slider

- Visibility

  Uncheck the visibility box, if you don't want to display the AVE graph in the plot. This might be useful if, for example, if plots overlap and you 'd like to see the underlying graphs.

6. Export

- The export dialog is located under FILE => Export...
- **Basic configuration**
    - Generally speaking, you always have to select one archive directory from which you want to export data. However, exporting data from more than one archive at the same time might be supported by some servers, too.
    - You can enter a new time range for the export query. Alternatively, you can load the time range from the selected time axis.
    - ArchiveViewer contains an internal exporter that writes data in CSV format (columns are actually separated by tabs).
    - If you want to omit the export output viewer, you can type the path to a desired file directly. Alternatively, you can display a file dialog by pressing the "..."- button next to the file text field.
    - As soon as you press OK, the export starts. Please take note of the export progress on the status panel.
    - You may interrupt export any time by pressing the red button next to the progress bar. Data that was exported up to the interruption remains available.
- **Advanced options**
    - You can reach advanced settings by pressing the "…"- button in the upper right corner of the export dialog and selecting "More options…"
    - The methods box contains retrieval methods that are supported by current archive server and are suitable for exporting data.
    - You can enter a number of samples per AVE which serves as a limit. For some export methods, it will make more sense to enter a period of time for which you want a single sample (this field and the "max number of values" field are symbiotic)
    - You may enter your own timestamp format (useful, for instance, if you are going to send the file to Europe). Please, take a look at the tooltip to see the supported syntax.
    - Finally, you can export status information in addition to pure data.

7. Menubar items

FILE menu

- New Connection

  Pops up a dialog where you can enter/select a connection parameter for the archive server

- Reconnect

  Sometimes there are (network) errors that may corrupt the connection to the data server. Simply press this menu button to re-establish the connection

- Open, Open Recent..., Save, Save As

  These menu buttons are used for loading and saving the plot configuration

- Export

  See above

- Print... and Page Setup...

  Displays widgets for printing the application window

- Clear All

  Clears the main panel

- Quit

  Exits the application

EDIT menu

- Copy, cut, paste

  standard menu items, have the same effects as corresponding keyboard short cuts (e.g. ctrl + x etc. on Windows)

- Preferences

  Includes a plot title setter, a legend configurator, and a plot plugins loader

VIEW menu

  - Full Screen
  - Search Dialog
  - Show Console, Show Event Log, Show Memory Console

    Some of standard XAL widgets

- **TOOLS menu**
  - Align Range Axes

    Aligns ranges of range axes in the selected plot plugin; may be useful when plots overlap

  - Assign Same Color…

    Assigns same color to AV entries with same names (useful e.g. when desired plot spans over several archive directories)

  - Assign Selected Archive

    Assigns the selected archive to selected graphs (useful for formulas, or if archive names changed etc.)

  - Sort By…

    Sorts AV entries either by their names or by their archive directories

  - Clear Cache

    Clears cached data

- **WINDOW menu**

  - Capture As PNG...

- **HELP menu**

  - Contents

    Opens the user's guide

  - Server Info

    Shows information that current archive server provides about itself

  - About

8. General Plot Features

Some plot plugins may support only subsets of these features, others may have more functions. Please, check the proper references.

- **Plot legend**

  Consists at least of colorful shapes. Plot legend is configured through preferences menu

- **Zoom in**

  You can zoom into the plot by dragging your left mouse. There is a minimum dragging distance to activate the zoom capability. This prevents accidental mouse "hang ups".

- **Plot scroll buttons**

  Located at the very bottom of the window; from left to right:

  - goes back in time the time range(s) you currently see on screen
  - goes forth in time the time range(s) you currently see on screen
  - goes "up" the value range(s) you currently see on screen
  - goes "down" the value range(s) you currently see on screen
  - doubles the time range(s) you currently see on screen
  - doubles the value range(s) you currently see on screen
  - halves the time range(s) you currently see on screen
  - halves the value range(s) you currently see on screen
  - displays a menu for advanced features:
    - uses anti-alias drawing method if checked
    - leaves ignored items if checked (and no new data is retrieved)

- Some typical **"right mouse click" menu** - features

  - Properties: some chart specific properties
  - Save as... : save the plot as a PNG image
  - Print... : prints the plot
  - Show own axis: Shows a colorful range axis for a normalized graph (if applicable)
  - Ignore Item... : if active, ignores the current plot item
  - Next/previous plot: navigates through plot history

- **(Re-)Draw triggers**

  A new plot will be generated in following cases:

  - After pressing the “PLOT”-button
  - After zooming/scrolling

    - When navigating through plot history
    - When a plot is resized

- To **clear** a plot, clear the main panel (FILE=>Clear All…) and press the PLOT button

- **Dock/undock**

  In the top right corner of the plot plugin panel, there is the dock/undock button. Please, note that once a plot plugin is undocked, it can not be controlled from the main controls panel. However, zoom-in and plot manipulation buttons continue to function.

The command line mode

The command line mode is, compared to the recommended GUI mode, limited in functionality and user-friendliness.
To start ArchiveViewer in command line mode, execute archiveviewer.jar with the '-nogui' option. You will then enter a scripting mode where you can interactively execute the ArchiveViewer commands. You can always type '-h' to get help.

Following steps are recommended (some commands can be typed on one line):

1. Retrieve server information
2. Print available archive directories
3. Search for PVs inside archive directories
4. Print an example plot configuration and edit it as necessary
5. Load the new plot configuration file into ArchiveViewer
6. Generate a plot image
7. Quit ArchiveViewer (please, be especially aware that the command line mode won't quit automatically)

In practice, you eventually might want to skip some of the steps.
To export, follow a similar routine (consult help if needed).

The JSP mode

Point your browser to <*tomcat_server_url*>/archiveviewer/
(e.g. at SNS, http://ics-web1.sns.ornl.gov:1982/archiveviewer/).
Follow the instructions that appear on the web site.

Included Plot Plugins

JFreeChart for Time Plots

General description

- Plots data of discrete and numeric non-waveform PVs and formulas against time axes
- Invalid data is plotted scattered underneath the actual (valid) plot area; tooltips display the reason for the invalidity

Plot item tooltips

Each plot item is assigned a tooltip that contains the name and archive directory of the corresponding AVE, the exact value and timestamp of the data sample (in case of discrete values, the actual meaningful discrete value is shown, not the assigned numeric representation).
In cases of lines/steps draw type, the tooltip of a plot item appears over the whole line up to the next value.

Axes

- Multiple time/range axes and all locations for them are supported
- Range axes may be of logarithmic scale
- AVEs without assigned range axes are plotted normalized and their ‘own’ range axes can be displayed using the right-mouse-click menu (they will appear in the same colors as the corresponding graphs)

Zoom/Scroll

- Scrolling supported
- Both, vertical and horizontal zoom in supported

Special features

- The plugin is able to generate a file image directly, thus it can be used when ArchiveViewer runs in the command line mode, or as Java server pages
- Uses double buffering when drawing
- When steps/lines draw type is selected, a horizontal line is plotted from the last data point up to the right edge of the chart panel

Known issues

If at the end of drawing process, the plot appears lost/unfocused, try to resize the plugin slightly and/or press the “PLOT” button again. If the issue is persistent, please contact the developers.

JFreeChart for Correlation Plots

General description

- Plots numeric, non-waveform PVs against a selected domain PV (initially the first valid in the AVEs table)
- Zooming/scrolling bypasses the retrieval of new data
- Discards invalid data

Plot item tooltips

Each data sample is assigned a tooltip that contains the name and archive directory of the corresponding PV; the correlated timestamp, the domain PV, and the range PV values.
In cases of lines/steps draw type, the tooltip belongs to the whole line up to the next value.

Axes

- Only one range axis (for the one domain PV) is supported; it is always located at the bottom of the plot and is of the same color as specified for the domain PV
- Multiple range axes and all locations for them are supported
- Range axes may be of logarithmic scale
- AVEs without assigned range axes are plotted normalized and their 'own' range axes can be displayed using the right-mouse-click menu (they will appear in the same colors as the corresponding graphs).

Zoom/Scroll

- Both, vertical and horizontal zoom in is supported
- Scrolling is supported
- However no new data gets retrieved, and thus only limited navigation through plot history is possible

Special features

- Features a combo box that contains all valid PVs; the selected PV is the domain; the selection of a new domain PV triggers an immediate replot
- Allows data items to be ignored
- Displays the time range of actually displayed data

JFreeChart for Waveforms

General

- Plots waveform PVs
- Consists of two plot panels, one of which is exactly like the panel in the "JFreeChart for Time Plots"- plugin; the other panel is an oscilloscope simulator
- Also features the oscilloscope control panel with a speed spinner; "PLAY" and "STOP" buttons; and a "…"-button that displays a dialog where you can assign the period and the delay (numeric values, or PVs for which data was also retrieved) of the waveform
- The time plot panel acts as visual help only; use the slider underneath to navigate through waveforms in time
- Skips invalid data

Plot item tooltips

The tooltip for data items in the time plot panel consists of the PV name and archive directory, the keyword "waveform", and the timestamp; the oscilloscope tooltip displays PV name and archive directory, the index of the waveform element, and the actual value

Axes

- Accepts only one time axis (the one that is selected in the main ArchiveViewer panel), but multiple range axes are supported
- Range axes can be logarithmic and positioned at any location
- PVs without assigned range axes are plotted normalized and their 'own' range axes can be displayed using the right-mouse-click menu (they will appear in the same colors as the corresponding graphs).

Zoom/Scroll

- Zoom in is only possible in the oscilloscope; supports no zoom out (use the slider under the time plot panel to restore the plot; tip: select the slider and use arrow keys to navigate step-by-step)
- Scroll possible in time plot panel only

Special features

- Uses double-buffer to draw the images
- Allows you to ignore data items (right-click menu)

Developer’s guide

The ArchiveViewer uses many cutting-edge concepts of modern Java programming. Below follows an overview of the ideas, their theory, and a short description of the way they were incorporated into the application.

XML

The ArchiveViewer can save plot configurations as XML files. The advantages are the ease of extensibility (especially, regarding backwards compatibility), portability, and the fact that these files can be edited quickly with any text editor. The application uses Apache’s Xerces library for parsing and serializing the configurations.
The ChannelArchiver client uses Apache’s XML-RPC library to talk to the archive server over the network.

SwingWorker

It often is the case that GUI events initiate very time-consuming algorithms (e.g. pressing “PLOT”-button leads to data being retrieved from the server; transformed into ArchiveViewer’s internal format, cached, and deligated to the plot plugins/exporters). However, it is desired that the GUI always remains responsive (e.g. shows a progress; allows the user to select widgets, press buttons etc.).
This is where SwingWorker comes in. It is an abstract class that has a handler for performing time-consuming operations in a separate thread. Although an implementation is directly available on the Java homepage (see Appendix F), the class is not integrated into any Swing release.
SwingWorker.java was copied into the epics.archiveviewer.base.util package and is used without any alterations.

Double-Buffered Drawing

When utilizing this algorithm, an actual image is drawn as bytes in memory (i.e. off screen) and then made visible by being placed onto a Swing component. The required additional RAM space is juxtaposed to the decrease in draw time.

All aforementioned JFreeChart plot plugins use the free JFreeChart plot library. Unfortunately, the library doesn’t deliver the desired performance. Thus, the class org.jfree.chart.ChartPanel was extended to create a java.awt.image.VolatileImage (instead of a BufferedImage) of the plot. The space that is occupied by the VolatileImage object is drawn onto in a separate thread.
When drawing finishes, the image is placed on screen.

Progress Indicators

Modern GUIs seem to rise and fall with the way the user feels they interact back with him. With this in mind, ArchiveViewer features progress indicators.
Luckily, standard Java Swing library already contains a progress bar widget that needs to be customized only.

Progress support in ArchiveViewer comes in two areas.
First, the SwingWorkers leave GUI responsive when time-consuming tasks are run. Thus an image indicating a progress can be made visible on screen in real-time.
Second, the API gives the developer the ability to pass an epics.archiveviewer.ProgressTask object in which the client can then set a percentage progress value and a descriptive text message.
Often, clients can't provide any reliable progress information, or the actual progress occurs so slowly that no visible feedback is possible. In such cases, ArchiveViewer will switch to "indeterminate" progress bars as known from internet browsers et. al. (a dark rectangle will bounce back and forth, until the task finishes). When actual progress value is updated, the application will display the determinate progress bar with the new value again.

Data caching

PV data lies on the server; retrieving it is a time consuming operation.
Often, the user wants to go back to the previous plot. Thus, the ArchiveViewer caches PV data for each 'request' (a unique combination of start time, end time, retrieval method, and number of values). When cached data exceed a certain limit, cache is cleared automatically. Note that data for PVs that appear in formulas is not cached.

Query processing

Current ArchiveViewer contains a handler for regular expressions and Unix-like glob patterns. It also contains an implementation of the relative time parser (see User's Guide for specification)

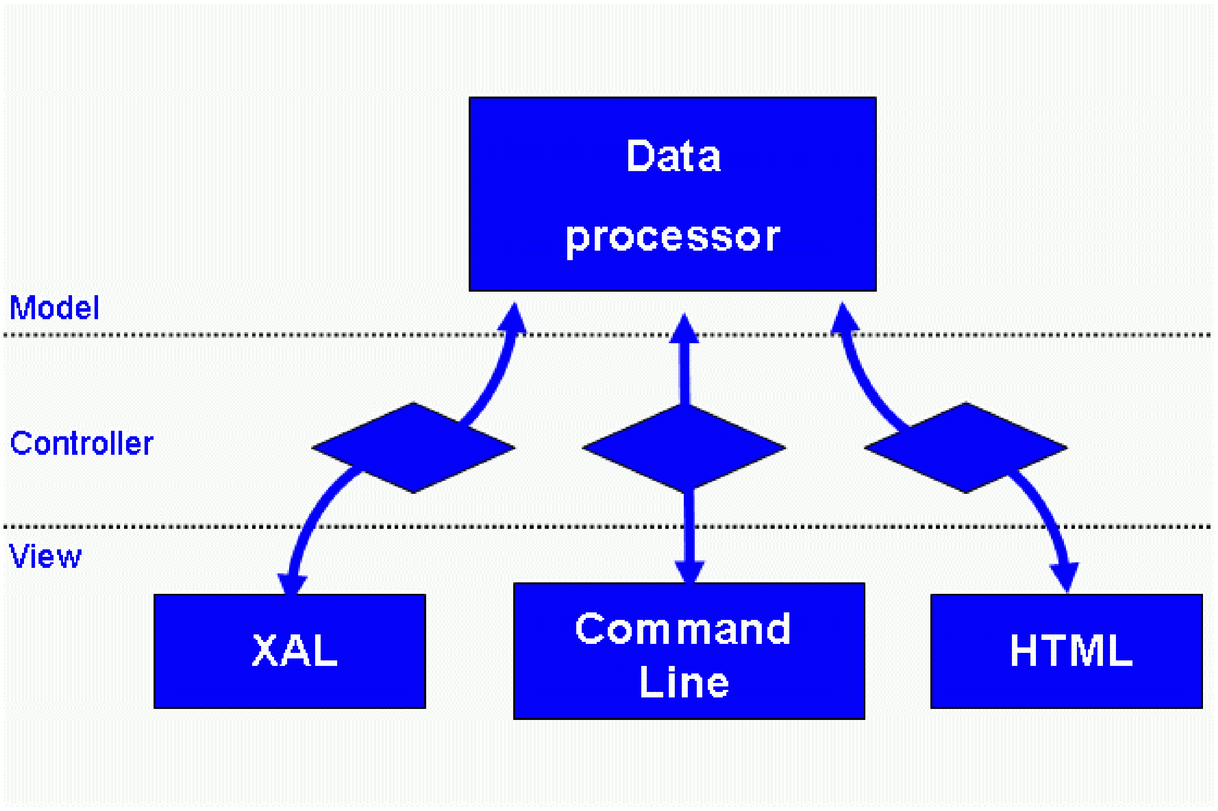
Formulas

PVs (from a common archive directory) can be used as arguments in formulas whose result will be processed by the ArchiveViewer as if it were a regular PV. You can also use formulas inside other formulas.
The formula parser is an extended version of a "3plus4 software" formula library (see Appendix F) that supports all basic arithmetics as well as boolean operations. The supported mathematical functions are the ones in the standard java.lang.Math class, plus own implementations of the max, min, and avg functions (that accept multiple arguments).
Note: The results of the formulas are not cached.

The Model-View-Controller

This is a very famous concept in software engineering. Basic idea is to separate the application code in two: One part takes care of data processing("the model"), the other is responsible for presenting the data to the user ("the view"). That which binds the parts is called "the controller".

ArchiveViewer follows the MVC paradigm. As model, an ArchiveViewer API library exists under epics.archiveviewer.base package. It provides routines for accessing the client, loading plot configurations, processing plot as well as export requests etc.



Three views are available.
The first view is the **XAL GUI**. Classes under epics.archiveviewer.xal.view are pure GUI widgets. They only contain Swing components as attributes and get- methods for them. The epics.archiveviewer.xal.controller classes connect GUI events with ArchiveViewer library by creating customized calls and push results of these calls back into GUI.

The second view is the **command line** (standard in/output). Classes under epics.archiveviewer.commandline are responsible for connecting command line options to the right ArchiveViewer library methods and delivering results of these to standard output. This happens in an infinite loop (until the user enters “-q”, see User’s Guide).

The third view is the **Java Server pages**. These are basically html files that contain special tags that were defined in the ArchiveViewer tag library. The implementation of these tags can be found in epics.archiveviewer.jsp package. This package also contains Servlets that process html form data. Thus, the user basically interacts with the ArchiveViewer library via HTTP methods (i.e. from a web browser).

XAL

The Swing components of the ArchiveViewer GUI are fully integrated into XAL framework (the used version is the one available in CVS on 04/05/2005), developed at SNS.

Integrating JSP

You can use ArchiveViewer JSP mode to plot and export PV data directly from your own web sites.

To plot PVs from one data request against a single range axis, pass following case-sensitive parameters (CGI variables) to

<*tomcat_url*>/archiveviewer/plotDirectly

Name	***Description***	***Notes***
connection_parameter	The server connection parameter	
PV configuration (these parameters can occur several times)		
pv_name	The name of a PV	
pv_directory	The name of an archive directory	
color	The color for corresponding plot	HTML format (e.g. '#FF0000')
draw_type	The type of corresponding plot	"scatter", "lines", or "steps"
Data request configuration		
start_time	The start time of the data request	May be absolute or relative
end_time	The end time of the data request	May be absolute or relative
Plot image configuration		
time_axis_location	The location of the time axis	"top", "bottom", or empty string
min	The lower bound of the range axis	Supports several number formats; may be an empty string
max	The upper bound of the range axis	Supports several number formats; may be an empty string
axis_type	The scale type of the range axis	"normal" or "log"
range_axis_location	The location of the range axis	"left", "right", or an empty string
plot_title	The plot title	May be an empty string
legend	The legend parameters	May be a combination of these: "show_ave_name", "show_archive_name", "show_range", and/or "show_units"
height	The height of the resulting image	May be an empty string
width	The width of the resulting image	May be an empty string

An example HTML form that produces such request can be found in Appendix E

To export PV data from a single archive directory, pass following case-sensitive parameters to <*tomcat_url*>/archiveviewer/exportDirectly

Name	***Description***	***Notes***
connection_parameter	The server connection parameter	
pv_names	A list of PV names	Names separated by standard Java delimiters => `" \t\n\r\f"`
pv_directory	The archive directory to be retrieve data from	
start_time	The start time of the request	May be absolute or relative
end_time	The end time of the request	May be absolute or relative
retrieval_method	Name of a supported retrieval method	
nr_values	The requested number of values	May be an empty string
ts_format	Timestamp format	May be empty; use Java DateFormat tokens
export_status	If present, exports status as well as data	May be empty/missing
exporter_id	The id of a supported exporter	E.g. "spreadsheet"

Plugin development

To develop with ArchiveViewer, the download of source code is absolutely necessary (at least, for now). Also, Apache's ant is highly recommended for any (re-)builds.

Creating a client plugin

To develop a client for your archive server, you must implement the epics.archiveviewer.ClientPlugin interface. The JavaDocs for it can be found in the Appendix C (or on the web site).
After you build your plugin, you have to register it with the ArchiveViewer base library. Currently it works through source code only.
Please, open the file epics.archiveviewer.base.AVBaseConstants.java and change the value of the constant AV_CLIENT_CLASS_NAME to the full classname (i.e. incl. package name) of the new client. Currently, ArchiveViewer supports only one client at a time.
Don't forget to rebuild ArchiveViewer.

Creating an exporter

To develop a new exporter, you must extend the abstract epics.archiveviewer.Exporter class (please, note that your exporter must possess a constructor with no arguments).
Then register your implementation with the ArchiveViewer base library by adding the full class name to the constant AVAILABLE_FOREIGN_EXPORTER_CLASS_NAMES in the epics.archiveviewer.base.AVBaseConstants interface.

Creating a plot plugin

To develop a new plot plugin, you must extend the abstract epics.archiveviewer.PlotPlugin class. Then register your implementation with the ArchiveViewer base library by adding the full class name to the constant AVAILABLE_PLOT_PLUGIN_CLASS_NAMES in the epics.archiveviewer.base.AVBaseConstants interface.

Appendix A
Directory structure

When you download a tar/zip ball or check out from CVS, you will see following important directories/files.

build.xml	The ant build file; see features below
ext_jars/	This directory contains unmodified jar files the application relies on
README	Contains important information for running current release of ArchiveViewer
RELEASE	Contains features of current release
src/	This directory contains all source files
web/	This directory contains jsp and other files that are needed to run ArchiveViewer in Java Server Pages mode (take a special notice of the tag library file web/WEB-INF/AVTag.tld)

Appendix B
ANT targets

The default target is “create_big_jar”. The context is the directory where build.xml lies.

Target	Description
build_all	Compiles all source files and puts them (and other necessary files) into build/ directory
clean_all	Cleans the base directory (in particular, removed the build directory, the archiveviewer.jar and archiviewer files)
configure_jsp	Creates a big ArchiveViewer jar; places it and the actual web files to appropriate Tomcat directories ($CATALINE_HOME must be defined)
create_base_jar	Builds all files and creates a jar of ArchiveViewer base classes only
create_big_jar	Builds all files and creates a jar with all of them
create_ca_client_jar	Builds all files and creates a jar of ChannelArchiver client classes only
create_jfree_plugins_jar	Builds all files and creates a jar of the plot plugins only
create_tar_ball	Cleans the base directory and creates a tar file of all necessary files
create_zip_ball	Cleans the base directory and creates a zip file of all necessary files
install_sns	Creates a big ArchiveViewer jar and installs it as EPICS application according to rules at SNS
javadocs	Creates JavaDocs of relevant classes

Appendix C
Plugin JavaDocs

(for more JavaDocs, go to http://ics-web1.sns.ornl.gov/archive/viewer/javadocs/)

epics.archiveviewer.ClientPlugin

public interface **ClientPlugin**

Implement this interface to develop new ArchiveViewer client plugins

Method Summary

`void`	**`connect`**`(java.lang.String param, ProgressTask progressInfo)` Tries to create a connection to the data server at the specified connection parameter.
`AVEntryInfo`	**`getAVEInfo`**`(AVEntry ave)` Returns an AVEntryInfo object for the specified AV entry; may return NULL
`ArchiveDirectory[]`	**`getAvailableArchiveDirectories`**`()` Returns an array of archive directories known by the server; should not cache them as ArchiveViewer base does it already
`java.lang.String`	**`getConnectionParameter`**`()` Returns the current connection parameter
`int`	**`getMaxNrValuesPerPVPerRequest`**`(int nrPVs)` Returns the maximum number of values the server can retrieve per PV per request; due to desired interactions with users, ArchiveViewer base has an own limit of 1000
`java.lang.String`	**`getName`**`()` Returns the client name
`RetrievalMethod`	**`getRetrievalMethod`**`(java.lang.String methodName)` Returns the retrieval method with the specified name
`RetrievalMethod[]`	**`getRetrievalMethodsForCalculation`**`()` Returns an array of retrieval methods for data that is going to be used to calculate a formula; the first element should be the default method
`RetrievalMethod[]`	**`getRetrievalMethodsForExport`**`()` Returns an array of retrieval methods for data that is going to be exported; the first element should be the default method
`RetrievalMethod[]`	**`getRetrievalMethodsForPlot`**`()` Returns an array of retrieval methods for data that is going to be plotted the first element should be the default method
`java.lang.String`	**`getServerInfoText`**`()` Returns a formatted description of the server
`void`	**`reconnect`**`(ProgressTask progressInfo)`

	Reestablishes current connection
`ValuesContainer[]`	`retrieveData(AVEntry[] archiveEntries, RequestObject requestObject, ProgressTask progressInfo)` Retrieves archived data for specified AV entries and the specified request parameters; returns an array of values containers (elements can be NULL)
`AVEntry[]`	`search(ArchiveDirectory ad, java.lang.String pattern, ProgressTask progressInfo)` Sends a server query for PV names in the specified archive directory that match specified regular expression pattern; returns an array of found AV entries

Method Detail

```
public java.lang.String getName()
```

Returns the client name

Returns:

the client name

```
public void connect(java.lang.String param,
                    ProgressTask progressInfo)
             throws java.lang.Exception
```

Tries to create a connection to the data server at the specified connection parameter. The internal state of the object should not change, if the connection could not be established

Parameters:

`param` - a connection parameter

`progressInfo` - the progress interface (for GUI feedback)

Throws:

`java.lang.Exception`

```
public java.lang.String getConnectionParameter()
```

Returns the current connection parameter

Returns:

the current connection parameter

```
public void reconnect(ProgressTask progressInfo)
               throws java.lang.Exception
```

Reestablishes current connection

Parameters:

`progressInfo` - the progress interface for feedback

Throws:

`java.lang.Exception`

```
public ArchiveDirectory[] getAvailableArchiveDirectories()
                                                   throws
java.lang.Exception
```

Returns an array of archive directories known by the server; should not cache them as ArchiveViewer base does it already

Returns:

an array of archive directories known by the server

Throws:

```
java.lang.Exception
```

```
public RetrievalMethod[] getRetrievalMethodsForExport()
```

Returns an array of retrieval methods for data that is going to be exported; the first element should be the default method

Returns:

an array of retrieval methods for data that is going to be exported

---

```
public RetrievalMethod[] getRetrievalMethodsForPlot()
```

Returns an array of retrieval methods for data that is going to be plotted the first element should be the default method

Returns:

an array of retrieval methods for data to be plotted

```
public RetrievalMethod[] getRetrievalMethodsForCalculation()
```

Returns an array of retrieval methods for data that is going to be used to calculate a formula; the first element should be the default method

Returns:

an array of retrieval methods for data that is going to be used to calculate a formula

---

```
public RetrievalMethod getRetrievalMethod(java.lang.String methodName)
```

Returns the retrieval method with the specified name

Returns:

the retrieval method with the specified name

```
public java.lang.String getServerInfoText()
                                   throws java.lang.Exception
```

Returns a formatted description of the server

Returns:

a formatted description of the server

Throws:

`java.lang.Exception`

```
public ValuesContainer[] retrieveData(AVEntry[] archiveEntries,
                                      RequestObject requestObject,
                                      ProgressTask progressInfo)
                               throws java.lang.Exception
```

Retrieves archived data for specified AV entries and the specified request parameters; returns an array of values containers (elements can be NULL)

Parameters:

`archiveEntries` - the AV entries whose data is to be retrieved

`requestObject` - the request parameters

`progressInfo` - the progress interface for feedback

Returns:

an array of values containers (may contain NULL elements)

Throws:

`java.lang.Exception`

```
public int getMaxNrValuesPerPVPerRequest(int nrPVs)
```

Returns the maximum number of values the server can retrieve per PV per request; due to desired interactions with users, ArchiveViewer base has an own limit of 1000

Parameters:

`nrPVs` - number of PVs whose request for data is going to be sent (may be unnecessary)

Returns:
the maximum number of values the server can retrieve per PV per request

---

```
public AVEntry[] search(ArchiveDirectory ad,
                        java.lang.String pattern,
                        ProgressTask progressInfo)
                 throws java.lang.Exception
```

Sends a server query for PV names in the specified archive directory that match specified regular expression pattern; returns an array of found AV entries

Parameters:
`ad` - the archive directory to search within
`pattern` - a regular expression for PV names to match
`progressInfo` - the progress interface for feedback

Returns:
an array of found AV entries

Throws:
`java.lang.Exception`

```
public AVEntryInfo getAVEInfo(AVEntry ave)
```

Returns an AVEntryInfo object for the specified AV entry; may return NULL

Parameters:
`ave` - the AV entry

Returns:
an AVEntryInfo object for the specified AV entry

epics.archiveviewer.Exporter

---

public abstract class **Exporter**
extends java.lang.Object

Extend this class to create an exporter plugin

Field Summary

`static int`	**EXPORT_DATA_AND_STATUS** a constant for exporting data and status
`static int`	**EXPORT_DATA_ONLY** a constant for exported data only

Constructor Summary

Exporter() The default constructor; must be present in all subclasses!!!

Method Summary

`abstract void`	**export**(`ValuesContainer[] vcs, int firstIndex, int lastIndex, java.io.Writer writer, boolean append, int detailsLevel, java.lang.String tsFormat`)

	Exports data from specified values containers between specified indices to specified writer
`abstract java.lang.String`	**getId**() Returns a short string that identifies this Exporter and is descriptive enough to let the user make the right selection (e.g.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

```
public static final int EXPORT_DATA_ONLY
```

a constant for exported data only

See Also:

Constant Field Values

```
public static final int EXPORT_DATA_AND_STATUS
```

a constant for exporting data and status

See Also:

Constant Field Values

Constructor Detail

```
public Exporter()
```

The default constructor; must be present in all subclasses!!!

Method Detail

```
public abstract java.lang.String getId()
```

Returns a short string that identifies this Exporter and is descriptive enough to let the user make the right selection (e.g. "matlab")

Returns:

a short string that identifies this Exporter and is descriptive enough to let the user make the right selection

---

```
public abstract void export(ValuesContainer[] vcs,
                            int firstIndex,
                            int lastIndex,
                            java.io.Writer writer,
                            boolean append,
                            int detailsLevel,
                            java.lang.String tsFormat)
                     throws java.lang.Exception
```

Exports data from specified values containers between specified indices to specified writer

Parameters:

`vcs` - the values containers to be exported

`firstIndex` - the first index of data to be exported

`lastIndex` - the last index of data to be exported

`writer` - the writer to which data is written

`append` - a flag, indicating if data should be appended to the writer or not
`detailsLevel` - see fields of this class
`tsFormat` - a string containing the Java timestamp format (may be NULL)
Throws:
`java.lang.Exception`

epics.archiveviewer.PlotPlugin

public abstract class **PlotPlugin**
extends java.lang.Object

Extend this class to create new plot plugins

Field Summary	
`static java.lang.String`	**`DESCRIPTION`** This field is accessed when the user tries to load a new plot plugin; please override it in the subclass

Constructor Summary
`PlotPlugin``(AVBaseFacade avbf, ImagePersistenceBean ngpip)` constructor

Method Summary	
`abstract void`	**`clear`**`()` Clears the entire chart
`abstract void`	**`displayGraphs`**`(ValuesContainer[] nonNullVCs)` Displays a plot image for the specified values containers (main method)
`AVBaseFacade`	**`getAVBFacade`**`()` Returns the access object to AV base
`abstract RetrievalMethod`	**`getChosenRetrievalMethod`**`()` Returns the suitable retrieval method for this PlotPlugin
`abstract java.awt.Component`	**`getComponent`**`()` Returns the component of this PlotPlugin that should be added to the ArchiveViewer Swing hierarchy (mouse listeners will be registered to this component)
`abstract double`	**`getCorrespondingDomainValue`**`(java.lang.String xAxisLabel, double xCoordinate)` Returns the value of the specified x axis that corresponds to the specified x coordinate in the space of the component returned by getComponent()
`abstract double`	**`getCorrespondingRangeValue`**`(java.lang.String yAxisLabel, double yCoordinate)`

	Returns the value of the specified range axis that corresponds to the specified y coordinate in the space of the component returned by getComponent()
abstract java.lang.String[]	**getDomainAxesLabels**() Returns an array of domain axes labels (might be different from the time axes specified by the user); needed by AV base to get information on axes bounds
abstract double	**getLowerBoundOfDomainAxis**(java.lang.String domainAxisLabel) Returns the lower bound of the specified domain axis
abstract double	**getLowerBoundOfRangeAxis**(java.lang.String rangeAxisLabel) Returns the lower bound of the specified range axis
abstract java.lang.String	**getName**() Returns a display name of this PlotPlugin
ImagePersistenceBean	**getPersistenceParameters**() Returns the image persistence parameters object
abstract int	**getPlotPanelWidth**() Returns the width of the plot panel
abstract javax.swing.JPopupMenu	**getRightClickMenu**() Returns the right-mouse-click menu
abstract double	**getUpperBoundOfDomainAxis**(java.lang.String domainAxisLabel) Returns the upper bound of the specified domain axis
abstract double	**getUpperBoundOfRangeAxis**(java.lang.String rangeAxisLabel) Returns the upper bound of the specified range axis
abstract java.awt.geom.Rectangle2D	**getZoomablePlotArea**() Returns a Rectangle2D for the part of this PlotPlugin's visual component which the user can zoom into
abstract boolean	**isDomainTime**() Returns true if domain axes are time axes; false otherwise (if true, plot zoom in/scrolling triggers new data retrieval)
abstract void	**setAntiAlias**(boolean flag) Depending on the flag, enables or disables anti alias drawing mechanism
abstract void	**setAvailableRetrievalMethods**(RetrievalMethod[] rms) Sets retrieval methods available from the current server
abstract void	**setDomainAxisBounds**(java.lang.String domainAxisLabel, double min, double max) Sets new bounds for the specified domain axis (not necessarily time axis)
abstract void	**setLeaveIgnoredItems**(boolean flag) Depending on the flag, tells the PlotPlugin to leave (or not) the ignored items

abstract void	`setRangeAxisBounds(java.lang.String rangeAxisLabel, double min, double max)` Sets new bounds for the specified range axis

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

`public static final java.lang.String` **`DESCRIPTION`**

This field is accessed when the user tries to load a new plot plugin; please override it in the subclass

See Also:

Constant Field Values

Constructor Detail

```
public PlotPlugin(AVBaseFacade avbf,
                  ImagePersistenceBean ngpip)
           throws java.lang.Exception
```

constructor

Parameters:

`avbf` - the av base access object

`ngpip` - the image persistence parameters object; may be NULL => draws to screen

Throws:

`Excpetion` - if, among others, the plugin can not be used to create file images directly

`java.lang.Exception`

Method Detail

`public AVBaseFacade` **`getAVBFacade`**`()`

Returns the access object to AV base

Returns:

the access object to AV base

`public ImagePersistenceBean` **`getPersistenceParameters`**`()`

Returns the image persistence parameters object

Returns:

the image persistence parameters object

```
public abstract void displayGraphs(ValuesContainer[] nonNullVCs)
                            throws java.lang.Exception
```

Displays a plot image for the specified values containers (main method)

Parameters:

`nonNullVCs` - an array of values containers to be plotted; elements must not be NULL

Throws:

`java.lang.Exception`

`public abstract java.awt.geom.Rectangle2D` **`getZoomablePlotArea`**`()`

Returns a Rectangle2D for the part of this PlotPlugin's visual component which the user can zoom into

Returns:
a Rectangle2D for the part of this PlotPlugin's visual component which the user can zoom into

---

```
public abstract java.lang.String getName()
```

Returns a display name of this PlotPlugin
Returns:
a display name of this PlotPlugin

```
public abstract java.awt.Component getComponent()
```

Returns the component of this PlotPlugin that should be added to the ArchiveViewer Swing hierarchy (mouse listeners will be registered to this component)
Returns:
the component of this PlotPlugin that should be added to the ArchiveViewer Swing hierarchy

---

```
public abstract void setDomainAxisBounds(java.lang.String domainAxisLabel,
                                         double min,
                                         double max)
```

Sets new bounds for the specified domain axis (not necessarily time axis)
Parameters:
`domainAxisLabel` - the label of the domain axis; may be NULL if only one domain axis present
`min` - the new lower bound
`max` - the new upper bound

```
public abstract void setRangeAxisBounds(java.lang.String rangeAxisLabel,
                                        double min,
                                        double max)
```

Sets new bounds for the specified range axis
Parameters:
`rangeAxisLabel` - the label of the range axis
`min` - the new lower bound
`max` - the new upper bound

```
public abstract boolean isDomainTime()
```

Returns true if domain axes are time axes; false otherwise (if true, plot zoom in/scrolling triggers new data retrieval)
Returns:
true if domain axes are time axes; false otherwise

```
public abstract javax.swing.JPopupMenu getRightClickMenu()
```

Returns the right-mouse-click menu
Returns:
the right-mouse-click menu

```
public abstract double
getUpperBoundOfRangeAxis(java.lang.String rangeAxisLabel)
                                          throws java.lang.Exception
```

Returns the upper bound of the specified range axis
Parameters:
`rangeAxisLabel` - the range axis label
Returns:

the upper bound of the specified range axis
Throws:
`java.lang.Exception`

```
public abstract double
getLowerBoundOfRangeAxis(java.lang.String rangeAxisLabel)
                                          throws java.lang.Exception
```

Returns the lower bound of the specified range axis
Parameters:
`rangeAxisLabel` - the range axis label
Returns:
the lower bound of the specified range axis
Throws:
`java.lang.Exception`

```
public abstract java.lang.String[] getDomainAxesLabels()
```

Returns an array of domain axes labels (might be different from the time axes specified by the user); needed by AV base to get information on axes bounds
Returns:
an array of domain axes labels

```
public abstract double
getUpperBoundOfDomainAxis(java.lang.String domainAxisLabel)
                                           throws java.lang.Exception
```

Returns the upper bound of the specified domain axis
Parameters:
`domainAxisLabel` - the label of a domain axis
Returns:
the upper bound of the specified domain axis
Throws:
`java.lang.Exception`

```
public abstract double
getLowerBoundOfDomainAxis(java.lang.String domainAxisLabel)
                                           throws java.lang.Exception
```

Returns the lower bound of the specified domain axis
Parameters:
`domainAxisLabel` - the label of the domain axis
Returns:
the lower bound of the specified domain axis
Throws:
`java.lang.Exception`

```
public abstract double
getCorrespondingDomainValue(java.lang.String xAxisLabel,
                                                    double xCoordinate)
                                             throws java.lang.Exception
```

Returns the value of the specified x axis that corresponds to the specified x coordinate in the space of the component returned by getComponent()
Parameters:
`xAxisLabel` - the label of the domain axis
`xCoordinate` - the x coordinate in space of the component returned by getComponent()
Returns:

the value of the specified x axis that corresponds to the specified x coordinate

Throws:

`java.lang.Exception`

```
public abstract double
getCorrespondingRangeValue(java.lang.String yAxisLabel,
                                          double yCoordinate)
                                   throws java.lang.Exception
```

Returns the value of the specified range axis that corresponds to the specified y coordinate in the space of the component returned by getComponent()

Parameters:

`yAxisLabel` - the label of the range axis

`yCoordinate` - the y coordinate in space of the component returned by getComponent()

Returns:

the value of the specified range axis that corresponds to the specified y coordinate

Throws:

`java.lang.Exception`

```
public abstract void setAvailableRetrievalMethods(RetrievalMethod[] rms)
```

Sets retrieval methods available from the current server

Parameters:

`rms` - an array of available retrieval methods

```
public abstract RetrievalMethod getChosenRetrievalMethod()
                                          throws
java.lang.Exception
```

Returns the suitable retrieval method for this PlotPlugin

Returns:

the suitable retrieval method for this PlotPlugin

Throws:

`java.lang.Exception`

```
public abstract int getPlotPanelWidth()
```

Returns the width of the plot panel

Returns:

the width of the plot panel

```
public abstract void clear()
```

Clears the entire chart

```
public abstract void setAntiAlias(boolean flag)
```

Depending on the flag, enables or disables anti alias drawing mechanism

Parameters:

`flag` - a flag

```
public abstract void setLeaveIgnoredItems(boolean flag)
```

Depending on the flag, tells the PlotPlugin to leave (or not) the ignored items

Parameters:

`flag` - a flag

Appendix D
Example plot configuration from Bessy
(nice formulas)

```
<?xml version="1.0" encoding="UTF-16"?>
<AVConfiguration>

<connection_parameter>http://www.bessy.de:8080/archive/cgi/ArchiveDataServer.cgi</connection_parameter>
  <time_axis name="Main Time Axis">
    <start>-1d</start>
    <end>now</end>
    <location>bottom</location>
  </time_axis>
  <range_axis name="Main Range Axis">
    <min/>
    <max/>
    <type>normal</type>
    <location>left</location>
  </range_axis>
  <legend_configuration show_ave_name="false"
show_directory_name="false" show_range="true" show_units="true"/>
  <plot_title/>
  <pv directory_name="ctl - machine controls - current week"
name="MDIZ3T5G:lt100">
    <time_axis_name>Main Time Axis</time_axis_name>
    <range_axis_name>Main Range Axis</range_axis_name>
    <color>-65536</color>
    <draw_type>steps</draw_type>
    <draw_width>1.0</draw_width>
    <visibility>false</visibility>
  </pv>
  <pv directory_name="ctl - machine controls - current week"
name="MDIZ3T5G:current">
    <time_axis_name>Main Time Axis</time_axis_name>
    <range_axis_name>Main Range Axis</range_axis_name>
    <color>-16776961</color>
    <draw_type>steps</draw_type>
    <draw_width>1.0</draw_width>
    <visibility>false</visibility>
  </pv>
  <pv directory_name="ctl - machine controls - current week"
name="PKEK1S11B:stat6">
    <time_axis_name>Main Time Axis</time_axis_name>
    <range_axis_name>Main Range Axis</range_axis_name>
    <color>-16711936</color>
    <draw_type>steps</draw_type>
    <draw_width>1.0</draw_width>
    <visibility>false</visibility>
  </pv>
  <formula directory_name="ctl - machine controls - current week"
name="ItauFilter">
    <term>((x0>100)&amp;&amp;(x0&lt;2500))*x0</term>
    <argument_ave name="Itau" variable="x0"/>
    <method_name>linear</method_name>
    <nr_values>800</nr_values>
    <time_axis_name>Main Time Axis</time_axis_name>
    <range_axis_name>Main Range Axis</range_axis_name>
    <color>-16777216</color>
```

```
        <draw_type>steps</draw_type>
        <draw_width>1.0</draw_width>
        <visibility>true</visibility>
    </formula>
    <formula directory_name="ctl - machine controls - current week"
name="Itau">
        <term>(x2&lt;0.1)*x1*x0</term>
        <argument_ave name="MDIZ3T5G:lt100" variable="x0"/>
        <argument_ave name="MDIZ3T5G:current" variable="x1"/>
        <argument_ave name="PKEK1S11B:stat6" variable="x2"/>
        <method_name>linear</method_name>
        <nr_values>800</nr_values>
        <time_axis_name>Main Time Axis</time_axis_name>
        <range_axis_name>Main Range Axis</range_axis_name>
        <color>-65281</color>
        <draw_type>steps</draw_type>
        <draw_width>1.0</draw_width>
        <visibility>false</visibility>
    </formula>
</AVConfiguration>
```

Appendix E
Example form for JSP access

```
<form method="GET" action="your_tomcat_url/archiveviewer/plotDirectly" >

<input
name="connection_parameter"
size="25"
type="text"
value="http://ics-srv-web2.sns.ornl.gov/archive/cgi/ArchiveDataServer.cgi"
/>

<input
name="pv_name"
size="25"
type="text"
value="CCL_Cool:FT101:Seg1_Rtn"
/>

<input
name="pv_directory"
size="25"
type="text"
value="RCCS/Vac (01/01/05 - present)"
/>

<input
name="color"
size="25"
type="text"
value="#ff0000"
/>

<input
name="draw_type"
size="25"
type="text"
value="steps"
/>

<input
name="start_time"
size="25"
type="text"
value="-1d"
/>

<input
name="end_time"
size="25"
type="text"
value="now"
/>

<input
name="time_axis_location"
size="25"
type="text"
value="bottom"
/>
```

```
<input
name="min"
size="25"
type="text"
value=""
/>

<input
name="max"
size="25"
type="text"
value=""
/>

<input
name="axis_type"
size="25"
type="text"
value="normal"
/>

<input
name="range_axis_location"
size="25"
type="text"
value="left"
/>

<input
name="plot_title"
size="25"
type="text"
value=""
/>

<input
type="checkbox"
name="legend"
value="show_ave_name"
checked='checked'
/>

<input
type="checkbox"
name="legend"
value="show_archive_name"
checked='checked'
/>

<input
 type="checkbox"
 name="legend"
 value="show_range"
 checked='checked'
/>

<input
type="checkbox"
name="legend"
value="show_units"
checked='checked'
```

```
/>

<input
name="width"
size="25"
type="text"
value="800"
/>

<input
name="height"
size="25"
type="text"
value="600"
/>

<input type="submit" value="Submit"/>

</form>
```

Appendix F
Important links

http://ant.apache.org
ANT- the Java build tool

http://xml.apache.org/xerces2-j/
Xerces- the Java XML library

http://ws.apache.org/xmlrpc/
Apache's XML-RPC Java library

http://www.jfree.org/jfreechart/
JFreeChart- an extraordinarily versatile Java plot library

http://java.sun.com
All resources on Java itslef

http://www.sns.gov/APGroup/appProg/xal/xal.htm
XAL framework – great for building reliable GUIs

http://jakarta.apache.org/tomcat/
Tomcat – the container for Java Server Pages

http://www.3plus4.de
Provided the initial formula parser (code was checked and modified)

http://java.sun.com/docs/books/tutorial/uiswing/misc/example-1dot4/SwingWorker.java
SwingWorker