



Synoptic User Guide

Revision 1.3 • February 24, 2012

Andrey Petrov <apetrov@fnal.gov>

Project home page: <http://synoptic.fnal.gov>

Last version of this document: <http://synoptic.fnal.gov/doc/guide/index.html>

PDF Version: <http://synoptic.fnal.gov/doc/guide/synoptic-1.3.pdf>

[Revision History](#)

© 2010–2012 Fermi Research Alliance, LLC.

Table Of Contents

1. [Overview of Synoptic](#)
 1. [Introduction](#)
 2. [Getting Started](#)
 1. [Running Web Viewer](#)
 2. [Running the Viewer Application](#)
 3. [Running the Viewer Application from ACNET](#)
 4. [Running the Builder Application](#)
 3. [Architecture](#)
 4. [Open Source Version](#)
 5. [Getting Help](#)
2. [Using Web Viewer](#)
 1. [Launching Procedures](#)
 2. [System Requirements](#)
3. [Using the Viewer Application](#)
 1. [Launching Procedures](#)
 2. [System Requirements](#)
 3. [The User Interface](#)
 4. [Data Connection](#)
 5. [Setting Control](#)
4. [Using the Builder Application](#)
 1. [Launching Procedures](#)
 2. [System Requirements](#)
 3. [The User Interface](#)
 4. [Components](#)
 1. [Active Components](#)
 2. [Static Shapes](#)
 3. [Data Links](#)
 4. [Displays as Components](#)

5. [Loading and Saving Displays](#)
6. [Using the Central Repository](#)
 1. [Editing Existing Displays](#)
 2. [Creating New Displays](#)
 3. [Resolving Conflicts](#)
5. [Credits](#)

1 Overview

1.1 Introduction

Synoptic is a system for graphical representation of real-time data in the Fermilab Accelerator Control System, superseding the legacy ACNET Lex SA application. The Synoptic's functions are similar to those of EPICS EDM, DESY JDDD, and, to some extent, LabVIEW. In a few words, the system offers a way for domain experts to create, in a short time, coherent diagrams representing a machine or a process, along with actual readings from the control system indicating its current state. It also optionally supports setting data back to the control system. The development process does not require the users to be familiar with programming, as the displays are composed in a high-level graphical editor from preexisting building blocks: gauges, alarm indicators, static shapes, and such. Similarly, the runtime environment takes care of all internal "plumbing and wiring" between components and hides the complexity of the actual data acquisition protocols.

The core idea of Synoptic is not new. As a matter of fact, each major control system in the world of high-energy physics has its own, custom implementation of that concept. Recognizing this, the latest version of Synoptic was designed to provide better and rather tight integration with other parts of the Fermilab Accelerator Control System, and it is not readily portable.

Software as a service

Synoptic is not a single program. Instead, it is a set of several components hooked up together and with other parts of the control system. In particular, the infrastructure includes a web server and a central repository with version control. The repository stores all public displays, allowing for sharing them easily among multiple users.

Builder vs. Viewer

Unlike most comparable systems, the Synoptic runtime (Web Viewer, and the Viewer application) can not be used to edit the displays. The number of users who want to see a display running is normally larger than the number of those who should be allowed to change it. The Synoptic runtime is relatively lightweight.

Multiple runtime options

Synoptic displays can be launched in three modes: within a web browser, on a local PC, or from the ACNET console. Different users may prefer different options due to security policies affecting them, software installed on their personal computers, or because their needs to switch quickly between many applications running under certain environment. In a minimum configuration, a conventional web browser with SVG support is sufficient to render displays smoothly, without reloading of the entire document.

1.2 Getting Started

Let's start with a few practical cases of running Synoptic displays from the central repository. A step-by-step instruction on how to create a custom display will follow. As it was said, the reason for multiple running procedures is to make the displays available under as many environments as possible. Detailed requirements for every mode are given in §§ [2.2](#), [3.2](#), [4.2](#).

To try out Synoptic, you may use the following displays:

/Demo/Gauges	Examples of various visual components showing simulated data.
/Cryo/Meson/MESON_CRYO/Adsorbers	Reading actual data from the control system.
/Demo/Controls	Reading and setting a real test device (Z:CACHE) in the control system.

Other displays in the central repository should work too, unless they are in the `/Legacy` or in the `/Work_in_progress` directories.

1.2.1 Running Web Viewer

Web Viewer renders Synoptic displays in conventional web browsers. The use of [Scalable Vector Graphics \(SVG\)](#) with incremental updates eliminates flickering of the image and requires very little bandwidth. All latest versions of Firefox, Safari, and Chrome will work with no additional configuration (client-side JavaScript must be enabled). The Internet Explorer requires [Adobe SVG plug-in](#). This mode requires no user authentication. Setting data to the control system is not supported by design.

To run a display in Web Viewer:

1. Open <http://synoptic.fnal.gov> in a [supported](#) web browser.
2. Locate the display you wish to run in the catalog, and click its name.
3. A new browser window will pop up. The display image will appear in 2–5 seconds.

For more information, see [§ 2 Using Web Viewer](#).

1.2.2 Running the Viewer Application

The Viewer application is a Java program that can run on a local computer. It acquires readings directly from the control system, and is capable of setting data back. The data connection requires a valid MIT Kerberos ticket, and it will work only within the Fermilab network. Authorized users may set data. If the opened display does not use real-time data, the application is not subject to the above restrictions and may work autonomously.

To run a display in the Viewer Application:

1. Open <http://synoptic.fnal.gov> in a web browser.
2. On the left panel, click **Launch Builder**.
3. After the program starts up, click the **Download & Open** button on the toolbar, or use the **File** → **Download & Open...** menu item.
4. In the file dialog, select the display you wish to run and press **Open** (or double-click the display name).

For more information, see [§ 3 Using The Viewer Application](#).

1.2.3 Running the Viewer Application from ACNET

The Viewer application can be used directly from the ACNET Console environment. In this case, the program runs on a central node, while its frames are displayed by the local X server. The system requirements for this mode are similar to those of the ACNET environment. As a rule of thumb, if you can launch ACNET Console, you should be able to run a fully-functional version of Synoptic with settings, as well. No special configuration is necessary either locally or remotely.

To launch a display from ACNET:

1. Start the ACNET environment.

2. In ACNET Console, go to the second D page and click **D99**. (Other choices are **F48**, **N24**, **N25**, **N26**, and **N48**.)
3. After the program starts up, click the **Download & Open** button on the toolbar, or use the **File** → **Download & Open...** menu item.
4. In the file dialog, select the display you wish to run and press **Open** (or double-click the display name).

Note: In the lightweight [ACNET Java Console](#), D99 will start the local Viewer application, as described in § [1.2.2](#), subject to all system requirements and security restrictions.

For more information, see [§ 3 Using The Viewer Application](#).

1.2.4 Running the Builder Application

The Builder application is used to edit Synoptic displays. It is a Java program, which should in most cases run locally. Technically, Builder does not connect to the control system, thus it does not require the Kerberos ticket and the right connectivity. It is possible, however, to launch the Viewer application directly from Builder, in which case all the requirements from § [1.2.2](#) will be in effect.

To launch the Builder application:

1. Open <http://synoptic.fnal.gov> in a web browser.
2. On the left panel, click **Launch Builder**.

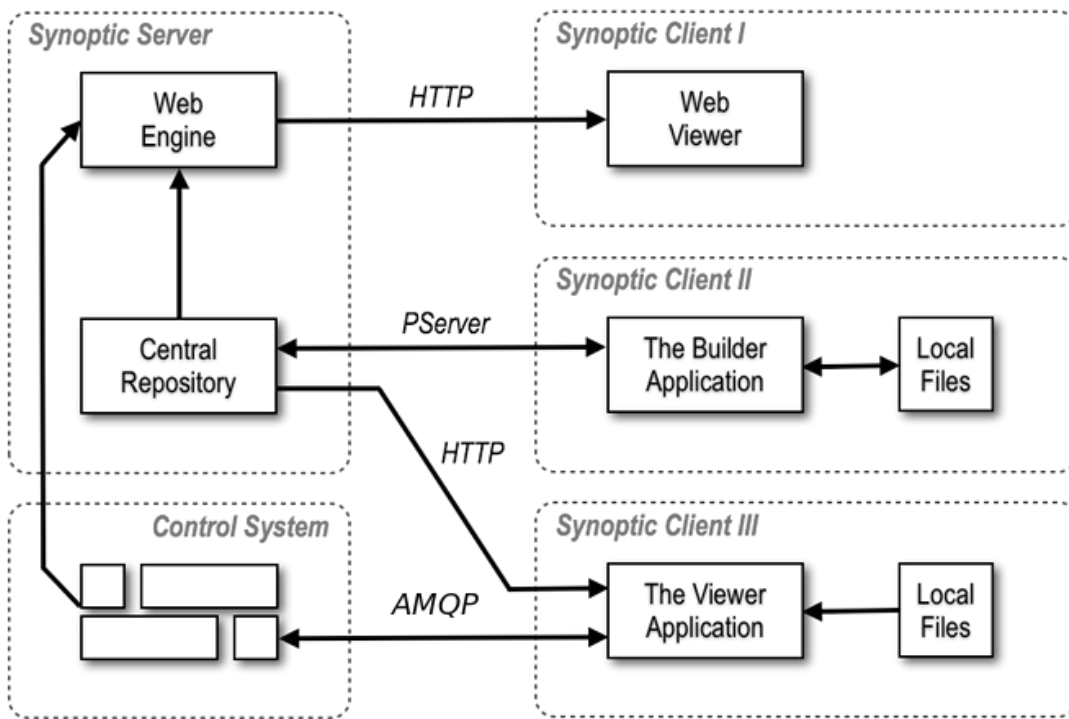
After the application starts up, it will open a new empty project. To create a simple working display using simulated data (thus, not requiring connection to the control system), do the following:

1. In the component palette on the left, click **Simulation** → **Sine**.
2. Move the mouse over the display area, and click once to place the component.
3. Similarly, place **Gauges** → **Text Display** in the display area, next to the first component.
4. Click **Tools** → **Place Link** in the main menu.
5. Move the mouse over the output (green) pin of the Sine component and click once.
6. Move the mouse over the input (cyan) pin of the Text Display component and click once.
7. The display is ready. Click **Tools** → **Launch Display** in the main menu to see it running.
8. You may improve the display by adding and connecting more components, and changing components' properties.

For more information, see [§ 4 Using The Builder Application](#).

1.3 Architecture

The diagram below shows the overall organization of Synoptic.



The system comprises 5 main components:

- **Central Repository** stores all publicly available displays with version control.
- **Web Engine** launches displays by user requests and provides their snapshots images to Web Viewers.
- **Web Viewer** is the combination of a generic SVG renderer and a custom JavaScript executed inside web browsers. It is capable of restoring full display images based on the information received from Web Engine.
- **The Viewer application** is a standalone program rendering displays on client-side computers.
- **The Builder application** is a standalone program used to edit displays.

Note that Web Engine can only launch displays stored in the central repository, whereas the Viewer and the Builder applications are able to work with both the repository and regular files.

1.4 Open Source Version

The open-source [Java Synoptic Toolkit](#) is distributed under a BSD-style license. It includes a core of Synoptic, free of dependencies from the proprietary infrastructure, and can be used with any data acquisition system, providing that an appropriate data access interface is developed. The product can be customized in a variety of ways, for instance, by developing specialized components for rendering data.

The Toolkit consists of the following:

- A basic set of components.
- Display Runtime Environment, including the SVG renderer.
- The Builder application.
- The Viewer application.
- A set of servlets for passing SVG snapshots over HTTP.

1.5 Getting Help

More information on Synoptic, including emergency procedures, technical notes, and the most recent updates not covered by this guide can be found at the project's wiki at <http://www-bd.fnal.gov/issues/wiki/Synoptic> (not available outside Fermilab). To report bugs or make suggestions, write

to synoptic-support@fnal.gov.

2 Using Web Viewer

Web Viewer takes advantage of built-in capabilities of many modern web browsers to display SVG images. A custom JavaScript embedded in the web page allows for smooth updates of the display image without reloading the entire document. The main disadvantage of Web Viewer is that the updates are synchronous: the server has no means to notify the client about new data. The client has to poll the server once every few seconds.

Web Viewer does not allow setting data to the control system, does not require user authentication, and does not impose strict security policies. Most displays can be opened from any location, including those outside Fermilab.

2.1 Launching Procedures

All displays from the central repository can be opened from a catalog at <http://synoptic.fnal.gov> by clicking the display's name.

Each display is associated with a unique URL, which can be treated as a regular web page address: saved in the browser bookmarks, put on the computer's desktop as a shortcut, and so on. The display URL consists of a static prefix, <http://www-bd.fnal.gov/synoptic/display>, followed by the display name (begins with a slash, with no file extension), optionally followed by a comma-separated list of parameters in parentheses.

For example, these are valid display URLs:

- <http://www-bd.fnal.gov/synoptic/display/Demo/Gauges>
- <http://www-bd.fnal.gov/synoptic/display/Cryo/Meson/COMPRESSORS/Compressors>
- [http://www-bd.fnal.gov/synoptic/display/Demo/SmallDaqDisplay\(DEV=M:OUTTMP,BG=khaki\)](http://www-bd.fnal.gov/synoptic/display/Demo/SmallDaqDisplay(DEV=M:OUTTMP,BG=khaki))

2.2 System Requirements

In order to render live Synoptic displays properly, a browser must support [Scalable Vector Graphics \(SVG\)](#) and JavaScript. The following table shows which browsers on which operating system can do this:

	Internet Explorer	Firefox	Chrome	Safari	Opera	Android Browser
Windows	Requires SVG plugin	Yes	Yes		Yes (presumably)	
Mac		Yes	Yes	Yes	Yes	
Linux		Yes				
iPhone & iPad				Yes	No	
Android		Yes			Yes	No

(Assumed reasonably up-to-date versions of software)

3 Using The Viewer Application

The Viewer is a standalone Java application that executes Synoptic displays. It imposes user authentication and supports setting data to the control system. Displays' definitions can be loaded either from the central repository or from plain files. Generally, all production displays are stored in the repository, whereas the files

are used mainly during the development phase.

3.1 Launching Procedures

The Viewer application can run in two modes: locally and on a central node. The latter also includes launching the program off ACNET Console. Both ways offer equal functionality, but there is no single answer as to which mode is to be preferred. Some configuration to the Java environment, firewall, and network connection may be required in order to use Viewer locally, but it will probably run a little faster and will be better integrated with other software on the same machine, local printers, and the file system. Conversely, all central nodes are properly configured and maintained, but it takes some time to log in there, and the application may be less responsive.

The following start-up methods can be used:

- To launch locally, go to <http://synoptic.fnal.gov> in a web browser and click **Launch Viewer** on the left panel.
- To launch from ACNET Console, use **D99**. (There are also specialized displays **F48**, **N24**, **N25**, **N26**, and **N48**)
- To launch from a terminal session on a CLX box, use `synoptic-viewer` command optionally followed by the name of a display.

In all cases, Viewer is being launched by passing a special URL to the Java Web Start utility. That URL can be customized to open a particular default display. It can also be saved as a shortcut on the computer's desktop, placed in a web document, or assigned to an ACNET page.

The URL to open empty Viewer is:

- <http://www-bd.fnal.gov/appix/start?p=60000393&a=-v>

A URL that opens a display from the central repository looks like:

- <http://www-bd.fnal.gov/appix/start?p=60000393&a=/Demo/Gauges>

In this example, a display called `/Demo/Gauges` will be loaded. Note that the display name begins with a forward slash and does not include a file extension. In some cases, the ampersand character inside the URL need to be escaped. For example, in a shell script the entire string has to be enclosed in single quotation marks.

3.2 System Requirements

A host running the Viewer application should satisfy the following requirements:

- Java Runtime Environment:
 - Java SE 6.†
- To read data from the control system:
 - Physical connection to the Fermilab network.†
 - ACNET user account with proper permissions.
 - Valid MIT Kerberos ticket.

The requirements marked (†) are normally met by all centrally administered desktops and laptops at the Accelerator Division. When Viewer is launched on a central node, the local computer should be able to connect to that node, but does not have to satisfy the above requirements.

3.3 The User Interface

The Viewer's graphical user interface is mostly intuitive. A few words should be said about different ways of

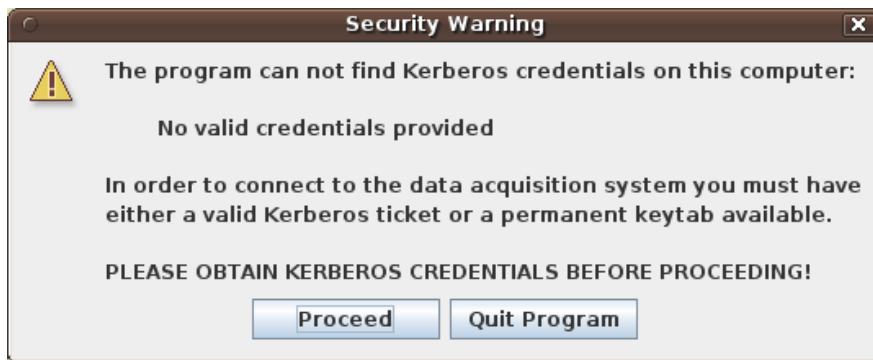
opening displays:



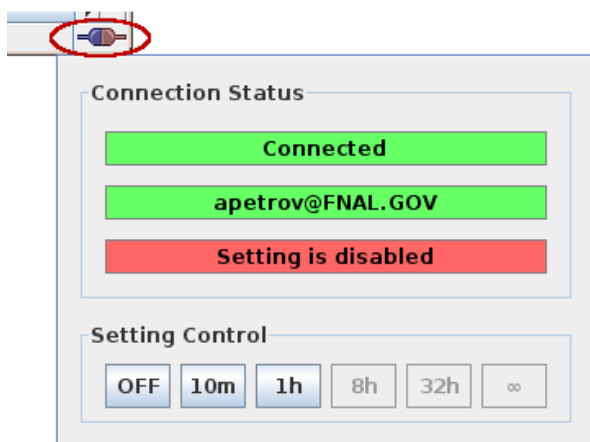
- The **Open** button (corresponds to the **File** → **Open...** menu item) loads the display definition from a plain file. This function is not used very often.
- The **Download & Open** button (corresponds to the **File** → **Download & Open...** menu item) loads the display definition from the central repository. *This is the main way of opening displays.*
- The **Reload Data** button (corresponds to **Tasks** → **Reload Data** menu item or **F5**) reloads the currently opened display from its original location and restarts the data acquisition process. If nested displays are present, the function also refreshes them according to their current definitions.
- The **Back** and the **Forward** buttons (correspond to **Task** → **Back** and **Task** → **Forward**) are used for navigation between displays that contain internal hyperlinks, in a similar way these functions operate in web browsers.

3.4 Data Connection

The Viewer application will decide by itself when to connect to the control system. The connection is established only if the display requests actual data. Displays including only simulated data sources can work autonomously. The users must possess a valid MIT Kerberos ticket before the connection attempt is made. If the program can not find a ticket, it shows the following warning:



Within the program, the data connection is controlled by an icon in the bottom-right corner of its frame. To open a dialog, simply click that button:



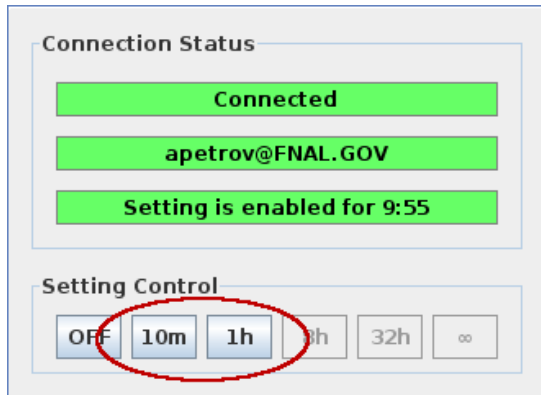
The Viewer application chooses a right data broker automatically. There is no need to connect or disconnect manually.

3.5 Setting Control

After the Viewer application starts up, setting is normally disabled. According to the application security policy, an ordinary user must always unlock setting explicitly and for a limited time.

The display widgets capable of setting data to the control system include input fields, buttons, and sliders. When setting is disabled, a gray cross is shown in front of all setting components connected to real devices. If there is a setting component on the display that is not connected to the control system, the cross does not appear even if setting is disabled.

To unlock setting, open the connection dialog as described above and press one of the button:



Whether and for how long setting may be unlocked is decided by the data broker. After a button is pressed, the program should respond immediately. If it is permitted to enable setting, the third line will turn green, saying "Settings enabled for ...", and all the gray crosses will disappear.

4 Using The Builder Application

The Builder is a standalone Java application that can edit Synoptic displays and save them in the central repository and in local files. It is possible to launch the Viewer application directly from Builder to verify the display under development.

4.1 Launching Procedures

The Builder application can run in two modes: locally and on a central node. This is similar to Viewer, and the trade-offs are, as well, custom configuration vs. speed and usability. In practice, most users should to run Builder in the local mode.

The following start-up method can be used:

- To launch locally, go to <http://synoptic.fnal.gov> in a web browser and click **Launch Builder** on the left panel.
- To launch from a terminal session on a CLX box, use `synoptic-builder` command optionally followed by the name of a display.

The URL that opens empty Builder is:

- <http://www-bd.fnal.gov/appix/start?p=60000393&a=-b>

4.2 System Requirements

A host running the Builder application should satisfy the following requirements:

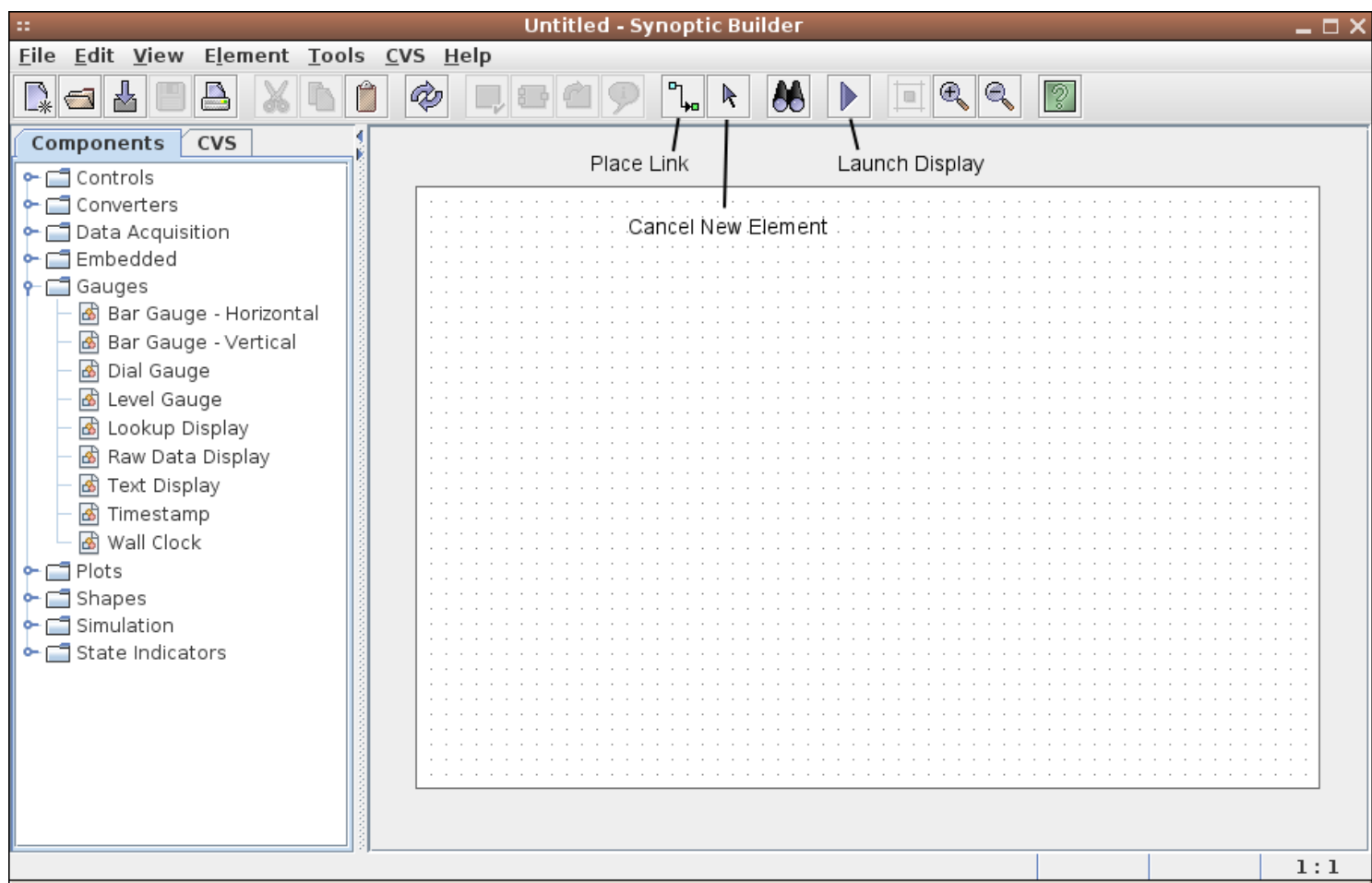
- Java Runtime Environment:
 - Java SE 6.†
- To save displays in the central repository:
 - Physical connection to the Fermilab network.†
 - Controls' CVS account.
 - Authorization to change certain directories in the repository. The displays in `/Work_in_progress` are editable by everybody.

The requirements marked (†) are normally met by all centrally administered desktops and laptops at the Accelerator Division. When Builder is launched on a central node, the local computer should be able to connect to that node, but doesn't have to satisfy the above requirements.

In order to run Viewer from the Builder application, the requirements specified in § 3.2 must also be met.

4.3 The User Interface

Below is a snapshot of the Builder's window, as it looks after the start-up:



There most important control elements are:

- The current display is located in the right part of the screen.
- A tree of available display components is under the **Components** tab.
- The **CVS** tab is used to synchronize local copies of displays with those in the central repository.
- The **Place Link** button is used to create a data link between two components.
- The **Cancel New Element** button is used to cancel the operation of placing a new component on the display.
- The **Launch Display** button can be used to start the execution of the current display in Viewer.

4.4 Components

Each display consists of a list of components. There are four classes of them: active components, static shapes, data links, and displays themselves.

Each component has a list of properties. To call up the property dialog, double-click the component or select **Properties** from the context menu. A set of permitted operation (rotation, adjustment of inputs and outputs, setting a background image) depends on the component's class.

When a component is placed onto a display, the component definition is actually copied into the display definition. After that, any modifications to the original components located under the **Components** tab will not affect the component instances already used by displays.

4.4.1 Active Components

Active components can accept and produce data. Every active component has an associated program code that defines its look and behavior in the runtime. Data acquisition components and data converters are hidden in the runtime. Most others are visible. In Builder, all active components are represented as rectangular shapes that normally (but not always) define the bounds in which the runtime version would fit.

Most of the active component support background images. To set up a background, select **Background Image** → **Load From File** in the context menu. Supported formats are PNG, GIF, and JPEG.

Active components have input and output pins. To change the number of pins, use **I/O Config...** from the context menu.

Active components can not be rotated.

A complete reference on all available active components can be found at <http://www-bd.fnal.gov/issues/wiki/SynopticComponentReference> (not available off-site).

4.4.2 Static Shapes

Static shapes are plain vector images. They have nothing to do with data, thus do not have inputs or outputs. In Builder, shapes can be moved, resized, and rotated.

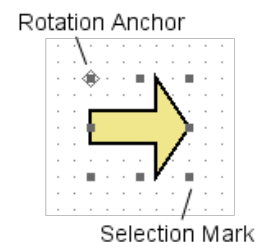
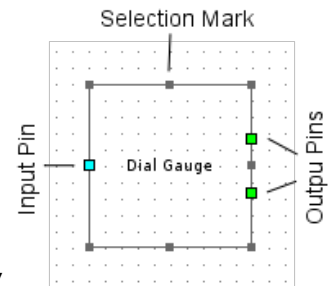
Static shapes look identically in Builder and in Viewer.

4.4.3 Data Links

Data links are virtual "pipes" used to pass data between active components. Each link connects exactly one output with one input. Links are hidden in the runtime. No circular links are permitted.

In order to connect two components, click the **Place Link** button from the main menu, or simply press **C**. Move the mouse over an output (green) pin on the first component and click the left button. Place intermediate points, if needed. Move the mouse over an input (cyan) pin on the second component and click the left button. The link is done.

4.4.4 Displays as Components



Each display itself is an active component. The only difference is that it may not have inputs or outputs. To place one display into another, use **Embedded** → **Embedded Display** on the **Components** tab.

4.5 Loading and Saving Displays

The Builder application has the following options for loading and saving display definitions:

- The **File** → **Open...** menu item opens the display from a regular XML file.
- The **File** → **Download & Open...** menu item opens the display from the central repository via HTTP.
- The **File** → **Save** and **File** → **Save As...** menu items save the current display into a regular XML file.

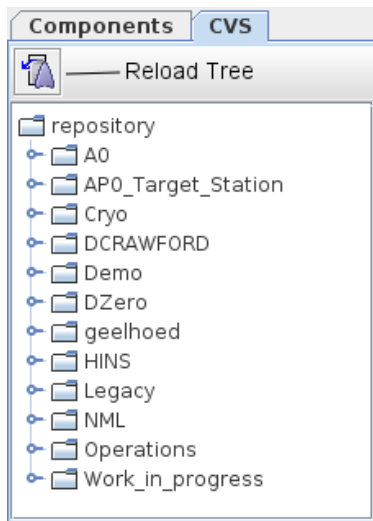
There is no option for saving displays directly to the central repository. This should to be done using the CVS workflow, as described in § 4.6. When a display is opened using **Download & Open...**, the loaded display definition is neither associated with a local file, nor can it be saved back into the repository. As such, this function should be used with caution, only as a quick way to inspect the contents of the repository.

4.6 Using the Central Repository

All displays intended for production use should be stored in the central CVS repository. Builder includes a simple embedded CVS client, which supports three main operations: *status*, *update*, and *commit*. For more complex functions, use generic tools, such as `cv`s command on Unix and Linux, or *jCVS* GUI application.



CVS requires the users to follow proper procedures. Most importantly, before editing a file its local copy should be synchronized with that on the server. Refer to [CVS Manual](#) for more information.






To start working with the repository, click the **Reload Tree** button on the **CVS** tab:



If the local working directory does not exist, the program will ask where to create one. The default location is `<user-home>/synoptic`. You may specify a different path right away, or change it later via **File** → **Preferences...**. After the working directory is configured, the CVS client loads a snapshot of the repository and shows it as a file tree. Normally, this operation completes in a few seconds. The actual files are not downloaded at that time, only their meta information (an empty directory structure is created, though).

Various icons and text colors describe status of each file:

 AP0ventilation.xml	<i>Unknown</i>	The file exists locally, but its status in the repository is unknown. Reload the tree or select Refresh Status .
 AP0ventsys.xml	<i>Up-To-Date</i>	A local version of the file matches its version in the repository.

 2000_Lens.xml	<i>Locally New</i>	The file exists only locally. Select Commit (Check In)... to upload it to the repository.
 BooleanDiscriminator.xml	<i>Locally Updated</i>	A local version of the file is newer than its version in the repository. Select Commit (Check In)... to upload the newer version to the repository.
 ColliderStatusPlots.xml	<i>Remotely New</i>	The file exists only in the repository. Select Update (Check Out) to download it into the working directory.
 AP0ventilation.xml	<i>Remotely Updated</i>	A local version of the file is older than its version in the repository. Select Update (Check Out) to download the newer version into the working directory.
 Button.xml	<i>Conflict</i>	Both a local and a remote version of the file have changed.

To refresh status, commit, and update files use the context menu. These operations work on both files and directories. To open a display, double-click its name in the tree or select **Open** in the context menu. Before committing a file, the system will ask to enter a note describing the changes that have been made.

4.6.1 Editing Existing Displays

1. Reload the tree.
2. Locate the file to edit.
3. If a version of this file in the repository is newer than the local one, select **Update** in the context menu. For *Remotely New* files this step is performed automatically and may be omitted.
4. Double-click the file in the tree or select **Open** in the context menu.
5. Edit the display.
6. Save the display.
7. The file status in the tree should change to *Locally Modified*; otherwise reload the tree.
8. **Commit** the new version back to the repository using the context menu.

4.6.2 Creating New Displays

1. Select **File** → **New**.
2. Edit the display.
3. Save the display inside the local working directory (you may want to put it in a new subdirectory).
4. The file status in the tree should change to *Locally New*; otherwise reload the tree.
5. **Commit** the new version back to the repository using the context menu. (The CVS client performs necessary *add* operations automatically.)

4.6.3 Resolving Conflicts

It is hard and tricky to resolve conflicts on CVS. Try to avoid them by making sure that the file you are going to change is in sync with the repository and it is not changed simultaneously by somebody else. If you detect a conflict, do the following:

1. Copy the conflicting file with a different name.
2. Remove the original conflicting file from the working directory.
3. Open the `CVS/Entries` file and remove a line containing the name of the conflicting file.
4. **Refresh** the repository tree in Builder. The file in question should become *Remotely New*.
5. **Update** the file. In other words, download the server version into the working directory.
6. Resolve the conflict by editing the version you have just downloaded. Use the stored conflicting copy as a reference.
7. **Commit** your changes back to the repository.

Credits

The Synoptic project was started back in 2001 by [Tim Bolshakov](#) and [Andrey Petrov](#) of Accelerator Controls Department at Fermilab. The initial idea came from the ACNET Lex SA application by Brian Hendricks. Tim designed the Synoptic runtime. Andrey implemented the Builder application. Both contributed to the component library. In 2002, after Tim has invented a new mechanism of partial XML updates, the project got its current shape. Many updates and improvements to Synoptic were inspired by Brian DeGraff of Cryogenics.

Since Tim's departure from Fermilab in early 2008, Synoptic is maintained by Andrey. By December 2008 the system has been completely refactored, according to the new vision on its use in Fermilab Accelerator Control System.

