CATS User Guide (Version 2.43)

7/14/2021

1 Introduction

The Computer Automated Traffic System (**CATS**) is a versatile program for controlling a model railroad. It supports CTC, DTC, ABS, or APB disciplines over blocks of railroad.

Because it is built on top of the JMRI model railroad interface, it can be used with different computers, DCC, and serial control systems.

CATS was originally written for Pat Lana’s N scale Cedar River and Iowa Central (Crandic) model railroad layout, but was generalized so that it can control turnouts and signals on most model railroads. Without the Crandic, this program would not exist. Without JMRI and the volunteers who have worked on it, this program would exist only for the Crandic.

CATS is the second program in a suite. It requires an XML description of the railroad, generated by the **designer** program, the first program in the suite. The **designer** is a graphics based program which allows the user to visualize the dispatcher panel presented by **CATS** and to associate DCC information with items on the layout (signals, turnouts, etc.). So, before **CATS** can do anything useful, it needs the XML description from **designer**. To get you started, I have included two sample layouts. Both are based on a layout created by John Armstrong (Figure 11-4, “Minimum sized loop to loop” in the second edition of Track Planning for Realistic Operations). ArmstrongMagnet.xml is the track plan with no layout connections and ArmstrongFull.xml is the same plan, with connections to internal JMRI devices; thus, both will run with no layout. I also included some Operations files so you can watch JMRI Operations and CATS interact.

The price paid for the versatility of **CATS** is that the user must also have the Java runtime environment (Oracle’s jre), JMRI library, and jar files. So, you will need to download them before beginning. But this is a small price because the JMRI programs are useful in their own right for programming decoders, monitoring DCC, discovering sensor addresses, testing out things, and so on.

1.1 *What is New and Different*

Version	Date	Changes	Section
0.24	7/29/05	• Added Jobs screen • Added checks on version of XML file • Added metered controls	3 (item 8) 3 (item 4)
0.25	9/8/05	• Reworked logging • Refresh screen changes cursor back to arrow • File selections start in CATS directory • Added file name filter for .xml files • Added DTC • Added fast clock selection • Fixed bug where sensors had to trip twice	3 (item 3) 3 (item 3) 3 3 4.1
0.26	11/11/05	• Added an indeterminate state for turnouts with feedback • Reworked jobs, crew, and train menus • Changed Loconet governor default to 0 • DTC signals are control points • Track reservations are cleared when a turnout	4.2 3 3 (item 3) 4.1

		moves • Changed train tracking to be more robust	
0.27	11/20/05	• Added a time on duty field to Crew screen • Added routine for computing relative time • Added Crew hours support	4.7 4.7 4.7
0.28	2/5/06	• Minor bug fixes	
0.29	4/6/06	• Minor bug fixes • Added turnout tests • Rewrote tracking algorithm	
0.30	4/26/06	• Minor bug fixes	
0.31	8/30/06	• Made compatible with designer for N Scale convention	
0.32	9/21/06	• Implemented approach lighting • Implemented software light flashing • Reworked decoder lists so that all outputs can be a list • Added drop downs so new JMRI device types can be added without rewriting this • Added train label selection	
0.33	9/25/06	• Fixed bug with reservations • Fixed outstanding bug with protected speeds	
0.34	10/22/06	• Bug fixes – no operational changes	
0.35	2/10/07	• Bug fixes • Added support for Loconet push button switches • Changed the “refresh” and Loconet “governor” values – 1 millisecond is higher resolution than supported by most computers • Adjustments can be created in **designer** and imported into **CATS** when the layout description is loaded • The screen size and location can be specified in **designer** and imported into **CATS** when the layout description file is loaded • This version of **CATS** should be able to run older layout description files	
0.36	3/29/07	• Added OOS and track authority messages to the session log • Added the ability to replay one or more session logs • Fixed a bug where the JMRI Loconet turnout decoder prefix was defined as “LN”, rather than “LT”. • Changed the priority on mouse clicks so that a	6 3 (item 3) 4.4

		train label is looked for first. Labels were being stuck behind signals and hard to move. • Enhanced APB so that a vacated block looks at the neighbor of the entrance point to see if it should delete the route or keep it. This eliminates the need for control point signals between OS sections.	
0.37	4/20/07	• Fixed a bug when reacting to Turnout (“T”) decoders. The listeners reacted to the previous state and not the current, so were “one step behind”. • Fixed a bug when sending to a Sensor (“S”) decoder. Throw and Close were reverse between sending and listening. Since most Sensors are used for listening, the sending changed. • Added an option to change the base of a signal icon to an inverted “tee”. • Added an option to remove the arrow head from track routes. • Added an option to disable the algorithm that attempts to minimize the number of tracks that span from a row to the next under it. • Added an option to consider decoder addresses when checking if it is safe to throw a turnout. • APB routes can only be created on a block protected by a panel signal. This eliminates some false routes when starting **CATS**. • Fixed a sequence problem on turnouts with feedback. If the feedback decoder address was the same as the command, the turnout would appear indeterminate. • Fixed a bug when reading in a layout file where text values were mangled. • Fixed a bug in replaying a session where a crew member was reassigned. This resulted in a Null Pointer Exception. • Fixed a bug in replaying a session where tying down a train was not recognized. • Modified “Refresh Layout” so that the commands to request the layout’s state are queued, so they go out after the commands that update the turnouts and signals.	
0.38	6/3/07	• Added a menu item to set all controlled turnouts to their straight route	

		• Modified how a layout is loaded. There was a race condition involving Swing when the XML file specified the panel size. • Corrected signals so that a Control Point is any signal on the panel and an Intermediate is a signal on the layout, but not the panel. • Fixed a bug seen on Macintoshes that did not provide a way for entering the name for a recording file. • User names are now added to Named Beans so that scripts can refer to them. • Reset the default refresh delay back to 0 msec. • Reworked session logs so that only tabs are accepted as field separators. • Fixed a bug in loading a layout which set all decoders to their default state. • Fixed a bug where a turnout with feedback, moved under local control might require the dispatcher to move it twice to regain control.	
0.39	8/2/07	• Fixed a bug that prevented a route from being re-established if a block in it cleared.	
0.40	8/18/07	• Implemented "Advance Approach", "Advance Limited Approach", "Advance Medium Approach", and "Advance Slow Approach". • Added an option for tracing decoder locks	
2.0	1/13/08	• Replaced the jdk-jdom11.jar library with jdom.jar, which resulted in changes to some XML method names. This makes the runtime compatible with JMRI 2, but no previous versions.	
2.1	3/3/2008	• Fixed some bugs with advance indications. • Fixed a bug with an intermediate signal protecting a dispatcher controlled turnout. • Separated the tracks in a crossing (diamond) into separate detection blocks. • Fixed a screen painting problem in which the screen area painted was too small. • Added some shortcut keys.	4.8
2.11	3/23/08	• Corrected a bug when shared decoder addresses might lock up. • Reworked right mouse button click on a train label so that all the information on the train can be edited. • Fixed a bug when tying down a train did not change the train label's color or release the	9.6 4.4

		crew. • Fixed a bug where the Enter key had to be pressed in a menu for editing on the field to be complete. • Restored the cursor to the default whenever a window closed. • Added a test in APB to resolve an ambiguous situation when deciding which train entered a block.	
2.12	7/15/08	• Fixed a bug in crossings where both legs used the same decoder address. Occupancy would color only one leg. • Fixed a bug in which it was hard to grab a train label with the mouse. Larger labels were harder to grab than smaller ones. • Added a keyboard shortcut (control+e) for quickly changing the train labels between train symbol and lead engine (just like the prototype!). • Removed a Null Pointer Exception bug when moving a train label.	
2.13	12/28/08	• Added some preventative code for a null pointer exception • Fixed a bug where trains that have completed their work could be assigned a crew. • Added signals to spurs so the dispatcher cannot move points on spurs. • Reworked how signals pass information to fix a bug. • Changed window title on crew edit window. • Changed "Save" to "Start Recording" • "Start Recording" now begins by saving the state of the CTC panel. • Fixed a bug when changing an extra job to a non-extra job. • Added support for **Train Status Client** applications.	3 6 3 (item 5)
2.14	2/2/09	• Fixed a bug when a turnout moves in a reservation. • Fixed a bug introduced in 2.13 involving spurs in ABS. • Reworked how CATS decided when to create a JMRI device handler to remove duplicate Loconet handlers. • Granting Track Authority can light/extinguish a	

		lock light on a spur. • Refreshed the JMRI name list.	
2.15	7/5/09	• Fixed a bug in the logic for locking decoders where a lock would not clear if a locked decoder moved. • Added "alignment" attributes for the fields in Train, Crew, and Job tables so the cell text can be left justified, centered, or right justified. • Added enforcement of dispatcher control of turnouts so that if a local crew moves a turnout without track authority, CATS restores the turnout. • Fixed some problems when using hidden track. • Added a trace capability to Java log files so that each Section's coordinates are recorded in the log, when the Section is read from the XML file. • Added crossings on Section edges, permitting natural looking "scissors".	Section 3 (item 4)
2.16	7/21/09	• Updated for JMRI 2.5.5+ compatibility.	
2.17	9/25/09	• Added the ability to change the color of things independent from other things • Changed the internal handling of tables – should be invisible • Removed some compiler warnings – should be invisible • Fixed a bug in software controlled flashing of signals.	
2.18	11/20/2009	• Added a user definable color to panel signal icons for signals not involved in a route. • Fixed some bugs (see the release notes)	
2.19	3/28/2010	• Updated version number only • Added build instructions to these notes	
2.20	11/23/2010	• Updated for Java 5 generics • Compiled under Java 1.5_22	
2.30	11/26/2010	• Updated for JMRI 2.10 compatibility	
2.21	2/28/2011	• Maintained JMRI 2.8 support • Reworked CATS for supporting TrainStat edit requests • Fixed some bugs in CATS (see release notes)	
2.31	2/28/2011	• 2.21, but support for JMRI 2.10	
2.32	1/2/2012	• Added support for releases later than JMRI 2.14 • Added interworking with Operations • Completed lunar aspects	

		• Fixed a bug where arrows were dropped • Compiled to JMRI compliance levels	
2.33	7/29/2012	• Added more interworking with Operations – CATS gets # of cars, length, and weight after each move • Updated check on version that layout was created to accept 2.13	
2.34	2/14/2013	• No functionality changes. Fixed an incompatibility introduced in JMRI 3.3.1	
2.35		• Test release	
2.36	2/14/2016	• No functional changes. Fixed an incompatibility introduced in JMRI 3.10. Built with Java 1.8.	
2.37	3/20/2016	• 2.36 was built on a development release • Signals protecting unbonded and end of track were wrong • Removed telling Operations when trains are terminated (the dispatcher could terminate a train on an A/D track which would free its cars before they are classified)	
2.38	7/16/2016	• JMRI 4.3.5 had a change that prevented CATS from completely terminating when the Exit button was selected or window closer clicked • JMRI 4.4 changed something in how Logix starts up so that JMRI panels imported into CATS via designer do not start Logix. A consequence of fixing this change is that the CATS window closer is ignored.	
2.39	1/15/2018	• Changes to JMRI after JMRI 4.8 broke things in CATS. These changes make CATS compatible, again. • Modified the designer scripts so that they don't specify the libraries needed, but the location of the libraries.	
2.40	1/6/19	• Some JMRI libraries marked as **deprecated** were finally removed, so CATS had to pick up the new interfaces • A Linux user reported that COLORDEFINITIONs were not being found. CATS explicitly initializes them. • Mac users found that the Mac designer.csh fails because it uses an unsupported Mac Java option. I put back the old designer .csh from 2.38 and named it MacDesigner.csh. • Restored approach lighting.	

		• Updated the cats.csh launcher to be more like the JMRI launcher.	
2.41	8/25/2020	• Fixed some incompatibilities between CATS and JMRI • A note is written on the console when CATS loads a JMRI panel • CATS can use a default folder for file storage, other than the JMRI folder • Reworked identifying turnouts with shared decoders, field movement of turnouts, and reversing field operations	
2.42	1/2/2021	• Fixed an incompatibility between CATS and JMRI 4.22 • CATS can be a TrainStat client • Train labels can be positioned vis transponding, RFID, RailCommand, etc. • Approach lighting of signals is flexible and can be triggered by a JMRI device	
2.43	7/1/2021	• Added the ability for train labels to have backgrounds • Fixed a bug in identifying network connections in logs • Fixed bugs in chains to work as diode matrixes • Advance aspects were not picked up from XM files • Added warnings when detecting missing signal templates • Changes for compatibility with JMRI 4.24 • Updated cats.csh to be more like JMRI launcher	

2 Setting Up

Most files are placed in the same directory as the JMRI files

- cats.bat is a command script to simplify running **CATS** under Windows**.** It uses the JMRI launcher. It has a command line for Windows 32 and one for Windows 64 and the default, which should work in either. Uncomment (remove the ::) for the version of Windows you are using.
- cats.csh is a command script for running **CATS** under Linux or MacOS.
- cats.doc is this document (should not be in the JMRI folder).
- cats.jar is the dispatcher panel program.
- catsManual.pdf is this document in PDF format (should not be in the JMRI folder).
- COPYING is the license (should not be in the JMRI folder).

- crandic.gif is a logo for the Crandic that appears on the main JMRI window. It should be placed in either your JMRI folder or the resources folder in the JMRI folder.
- ArmstrongFull.xml and ArmstrongMagnet.xml are sample layout files (should not be in the JMRI folder).
- designer.bat is a command script to simplify running the **designer** program under Windows. It has multiple versions of a command to launch **designer**, depending upon how Windows is configured.
- designer.csh is a command script for running **designer** under Linux.
- Designer.doc is the user manual for **designer** (should not be in the JMRI folder).
- designer.jar is the program used to describe the layout.
- DesignerManual.pdf is the designer user manual in PDF format (should not be in the JMRI folder).
- Operations is a folder of Operations data that shows how **CATS** and JMRI Operations exchange information. It should be "restored" through Operations and works with the Armstrong files.
- Lib is a folder for the libraries needed by **designer**
- .ICO are icon files to attach to launchers

The minimum files to develop and run **CATS** are:

1. cats.bat (or cats.csh)
2. cats.jar
3. designer.bat (or designer.csh)
4. designer.jar
5. crandic.gif

The cats and crandic.gif file need to be in the JMRI folder. The designer files can be in a separate folder, if it contains a lib sub-folder with a log4j and a jdom jar file.

Under Windows, we made shortcuts to the .bat files and placed the shortcuts on the desktop.

CATS was created under Java 1.8.0_261. It needs Java 1.8 or later, as does JMRI.

2.1 *Starting CATS – Windows*

The easiest way to launch **CATS** under Windows is to create a short cut to cats.bat and place the shortcut on your desktop. As explained in the troubleshooting section (8), it is also possible to bring up a command window, "cd" to the JMRI directory and just enter "cats".

2.2 *Starting CATS – Macintosh*

One way to start **CATS** is to open a terminal window. In that window, navigate to where cats.csh is installed and execute it. There may be other ways, but I do not have access to a Macintosh to discover them.

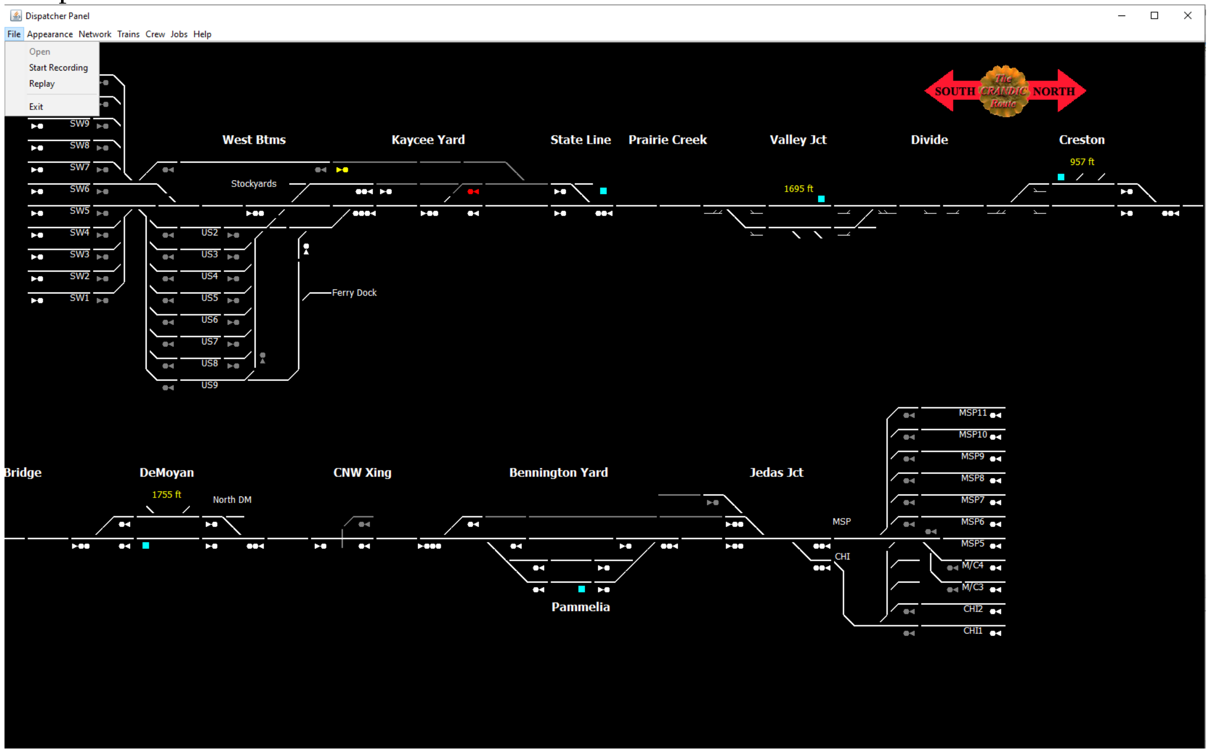
2.3 *Starting CATS – Linux*

Since **CATS** requires graphics, you must install a windowing package (e.g. gnome or kde) on your computer. With the window manager running, you can open a terminal window. In that window, navigate to the JMRI directory, where you installed **CATS**. You can execute **CATS** from there (e,g, ". cats.csh", "sh cats.csh", etc). Your window manager may also allow you to create a shortcut that you can place on your desktop.

3 Getting Started

Assuming a layout description exists and that it contains all three of the layout, typical trains and jobs, **CATS** is most easily started by running an operating system dependent command script (see previous section). The command script has all the magic incantations to Java.

1. Simply launch cats.bat (Windows) or cats.csh (Macintosh OS X or Linux) to start up **CATS**[1].
2. Several windows will pop up. All but one are part of JMRI, so the JMRI tool set is available while running the dispatcher panel.
3. The last window created is a blank dispatcher panel. Use the **File** -> **Open** menu item to navigate to the XML description file and open it.
   If you look at File again, you will see that **Open** is greyed out and **Start Recording** is an option.

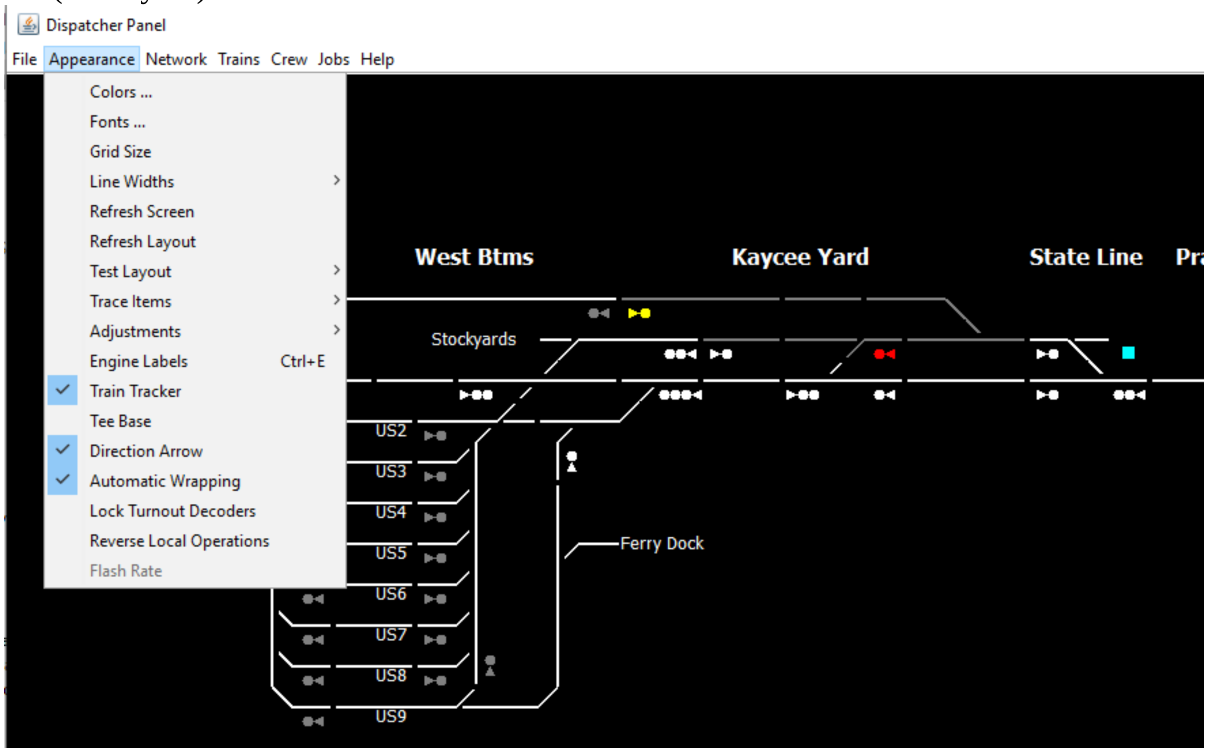


If you select **Start Recording**, then the current train locations and crew assignments

[1] If a file name is provided to CATS when it is invoked, it will attempt to open that file as the layout description. For example, if "crandic.xml" is added to the end of the cats.bat (cats.csh) file, then when cats.bat(cats.csh) is run, it automatically loads crandic.xml.

(as well as subsequent train movements and crew assignments) are stored in a file you select. These actions are time stamped to allow the session to be recreated. Thus, the file provides a record of the operating session. It is not intended that the record be used for grading the operators, but for being used in conjunction with multiple records for adjusting train schedules. See section 6 for details of the record log. See item 12 for other ways to use the session log.

4. Under the **Appearance** menu are controls for changing the appearance of the screen (and layout).



With these items, you can change colors, line widths, character sizes, and other things. So, if you don't like the color of an item, select it and you will be greeted with a color selector through which you can choose a different color. You will have to resize the screen to make changes in the grid size and automatic wrapping take affect. Note that these changes apply only to this run of **CATS**. Permanent changes are made with **designer** and recorded in the XML file.

For lack of a better place to put them, **Appearance** contains several other menu items. There is a menu item (*Test Layout*) for running tests on the layout. It has an item for testing all signals on the layout, either have them set to the appropriate aspect for a common indication or a common aspect. There is another test item for throwing all turnouts (useful for initializing a layout to a known state) and for closing all turnouts. This is useful for synchronizing the positions of feedback turnouts with what **CATS** thinks they are. There is a test item for setting all the turnout lock lights. Another test exists for setting all "automatic" switches to their straight route. The last should be done before beginning operations. Though it checks for block occupancy before throwing a turnout, some cars may not trip the detector and this would move the turnout under them.

After running tests, you can select *Refresh Layout* and CATS will attempt to set all signals, lock lights, and turnouts to the condition shown on the screen.

If the screen looks funny (for example, the mouse cursor is not a pointer), you can select *Refresh Screen* and have the screen redrawn. With a change in version 2.11, the cursor should always be a pointer, but if it is not, click on *Refresh Screen*.

The **Appearance** menu also contains some items for debugging and tuning the layout control. Under *Trace Items* a checkbox exists for printing on the Java console signal changes when sent to the layout and for printing when the number of locks on a decoder changes. It also has an option for tracing the communications with JMRI Operations.

Under **Appearance**->**Adjustments**, *Occupancy Debounce* controls a "debounce" value that can be selected for occupancy reports. CATS will hold the occupancy report until it has been stable for the number of seconds desired; thus, filtering out "phantom" reports. Another item (*Refresh Delay*) selects a delay introduced after every message sent to the layout when doing a layout refresh. This delay is intended to prevent **CATS** from overwhelming the layout with messages.

The *Engine Labels* checkbox is used for selecting what the train labels on the panel show. If checked, then they are the lead engine number. If not checked, it is the train's symbol.

Train Tracker is a menu item for enabling or disabling automatic train tracking. When train tracking is enabled, **CATS** will attempt to move a label when an occupancy detection is received. Thus, train labels move around the layout without dispatcher intervention. Train labels do not move automatically from detected tracks into undetected tracks because undetected tracks do not report occupancy. *Train Tracker* is an option because it works well when the detectors behave themselves and do not generate false reports. If the detectors are not well behaved, then it is best to disable automatic tracking. Label movement due to transponding is not affected by this setting.

Tee Base changes the base of signal light icons. When checked, the bases are drawn as an inverted "tee". When unchecked (the default), the bases are drawn as triangles. Changing this option takes affect immediately, but you will need to refresh the screen.

Direction Arrow is another option for changing how the screen looks. If checked, arrow heads are placed on the exit ends of routes. When unchecked, arrow heads are not drawn. Changing the option takes affect immediately, but only for routes created after the change. Existing routes are not changed.

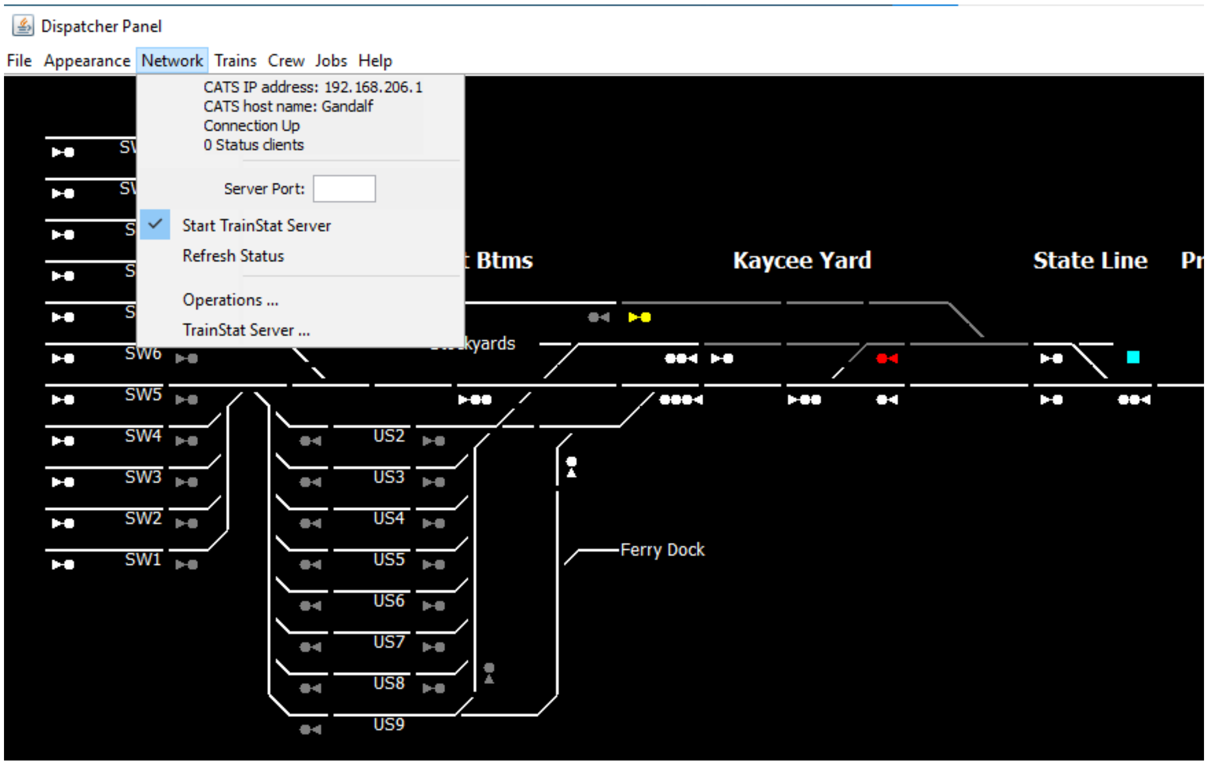
Automatic Wrapping is an option for controlling how the tracks are distributed on the panel. If the box is not checked, **CATS** tries to put as much on a "row" of tracks as it can, from the left edge of the panel to the right. If the box is checked, **CATS** computes which columns will fit on a row, then works from the right edge back to the middle of the row and breaks the row at the minimum number of tracks farthest to the right that it can find. It tries to minimize the number of tracks that "wrap" from one row to the next. After changing this option, the screen size must be readjusted (i.e. resize the panel) to trigger a relayout of the tracks.

Lock Turnout Decoders is an option for those layouts that might have multiple turnouts using the same decoder addresses (e.g. on a cross over). When checked, **CATS** will query all turnouts sharing the decoder address of a turnout being requested to move. If any would disallow it, then the request will be denied. This could be considered extended interlocking between turnouts. If your turnouts do not share decoder commands, it is best to leave this box unchecked. Should you encounter a turnout that was locked (i.e. was occupied, was in a route, was OOS, or was granted track authority), appears like it should be unlocked, but does not move, unchecking this box will turn off decoder locking. If you then check the box, locking will be turned on again, but the locks prior to unchecking the box are lost. So, to clear all decoder locks, uncheck, then check the box.

Reverse Local Operations is a checkbox for enforcing dispatcher control of turnouts. When it is checked and **CATS** sees that a turnout without track authority moves, **CATS** will send the commands to the layout to move the turnout back. If turnouts are not behaving properly with the box checked, uncheck it and the change will take effect immediately. This can have unintended interactions with *Lock Turnout Decoders*. If the latter is unchecked and one side of a cross over is moved, the other side will object. The other side will attempt to restore the alignment and move the points back, causing the first side to move them, etc. What this means is, if you have points sharing decoder addresses so that they move in tandem, you should either check *Lock Turnout Decoders* or uncheck *Reverse Local Operations*. You can check both.

If **CATS** provides the timing for flashing signals (software flashing), then the last item (*Flash Rate*) allows you to fine tune the flash rate.

5. The **Network** pull down is used for setting **CATS** into a server mode, where it will report changes in trains (primarily crew and location) to a **Train Status Client** and for connecting to JMRI Operations.



CATS IP address: 192.168.206.1
CATS host name: Gandalf
Connection Up
0 Status clients

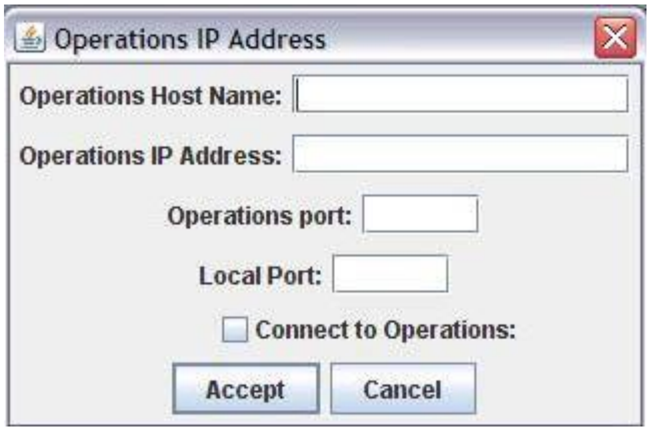
The top third of the pull down describes the network:

- IP address being used by the **CATS** computer
- Network name of the **CATS** computer
- If the **CATS** computer has an active network connection or not
- How many **Train Status Clients** are listening for changes to trains.

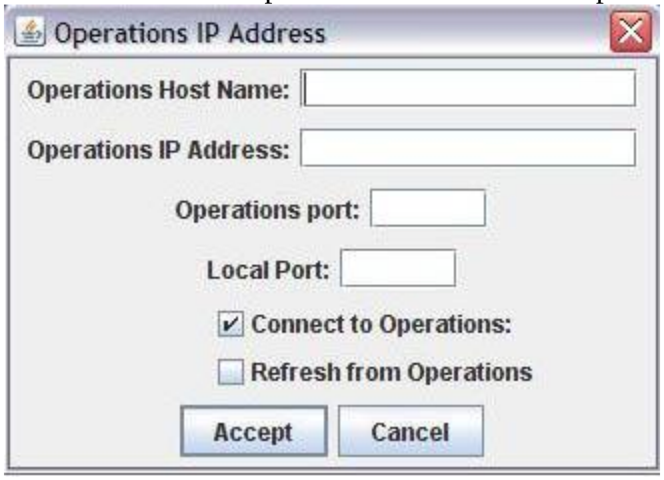
The middle third of the pull down is used to control the train status server function.

- The *Server Port* is used to change what network port the train status server is using. This is useful if there is another application using the same port, a firewall problem, or possibly some other network problem. The default port is 54321. If you feel that you need to change the port number, first uncheck “Start TrainStat Server” so that **CATS** releases any connections that it has to the network. Enter the new port number and touch Enter (to tell **CATS** that the entry is complete). Finally, check “Start TrainStat Server”. If the new port is also not available, you will see an error message (on the console).
- A *Start TrainStat Server* checkbox. When checked, **CATS** is requested to run the train status server function.
- *Refresh Status* forces **CATS** to update all **Train Status Clients** with the current state of all trains.

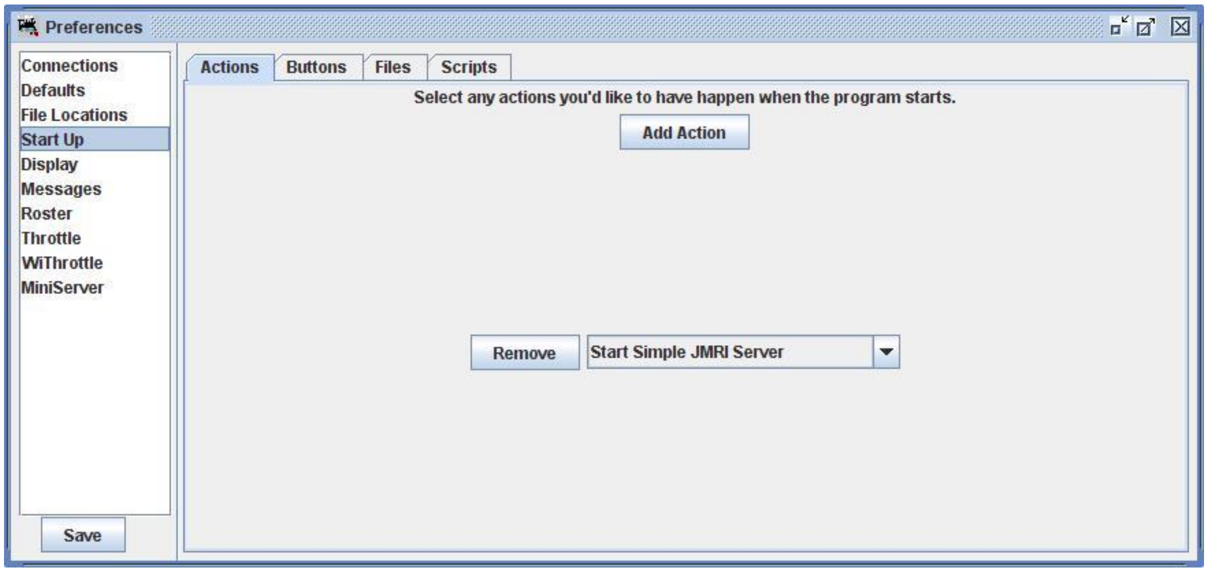
The Operations … line of the pull down creates a pop up window for configuring the network connection to JMRI Operations. JMRI Operations can be running on another computer (in which case, you will need to fill in the network information for how **CATS** can find the other computer) or on the same JMRI instance as **CATS** is using (in which case, most of the information can be left blank). The pop up looks like:



If known, you enter the network name of the JMRI computer into the top field. If the name is not known, but the IP address is, then the IP address is entered into the second field. The port fields can be left blank unless there is a conflict with another application using the same ports. By default, Simple JMRI Server uses port 2048 and **CATS** uses port 51431. Finally, the check box requests a connection to the Simple JMRI Server (checked) or drops the connection (unchecked). Just because the box is checked, a connection may not exist (e.g. a firewall prevents **CATS** from talking to Simple JMRI Server). The way you can tell is that when a connection is made, the above window has an additional option – to Refresh from Operations:



In addition, Simple JMRI Server must be started on the JMRI instance running Operations (regardless of if it is the same JMRI as **CATS** or not). From the splash screen, Edit->Preferences,



Select any actions you'd like to have happen when the program starts.

Start Simple JMRI Server

Changing this window will require the JMRI running it to restart.

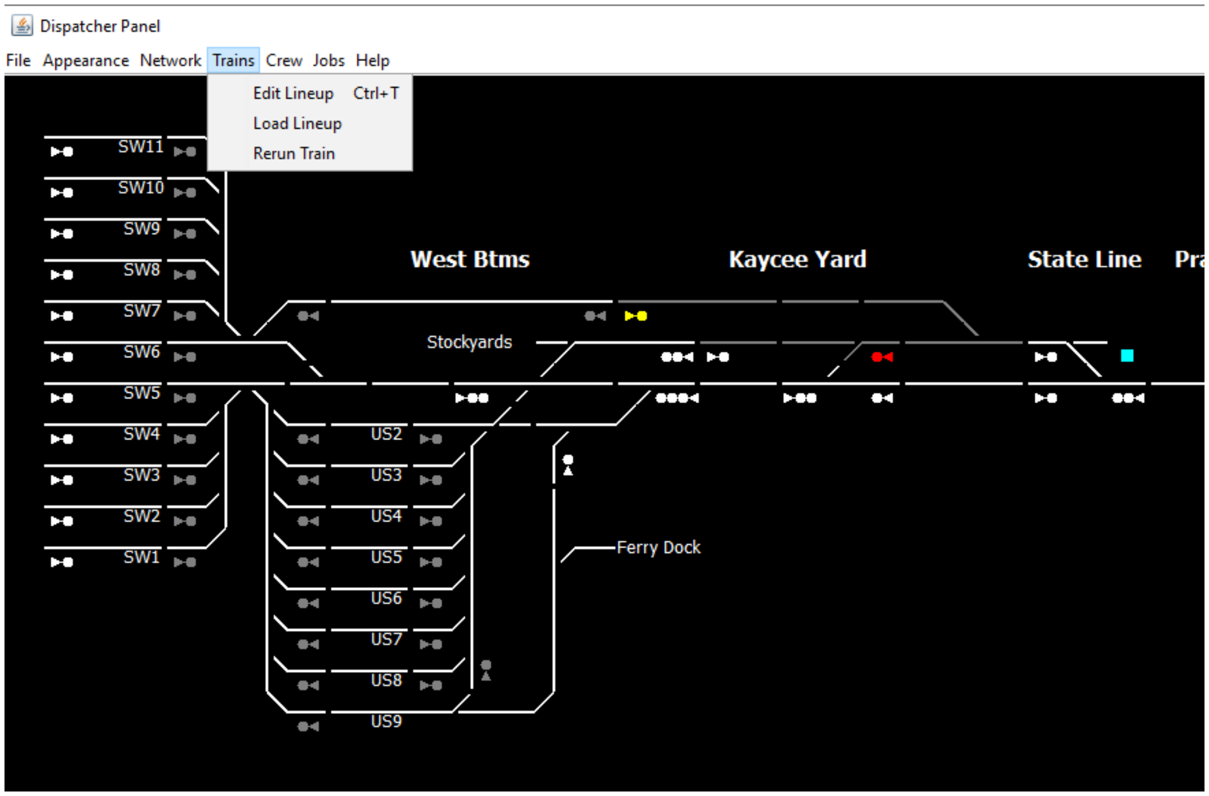
After **CATS** is configured to communicate with a Simple JMRI Server, **CATS** will receive all train length, cars, and weight information from Operations and update its internal train tables. In addition, **CATS** will tell **Train Status** whenever a change in length, cars, or weight is detected (after moving a label). **CATS** will tell Operations whenever a train arrives in a Station (Operations Location). Finally, **CATS** will tell Operations whenever a train terminates or ties down[2].

The last line, TrainStat Server …, works similarly to establishing a connection to Operations, but establishes a connection to a **CATS** operating as a **Train Status** server. Doing so make that **CATS** a **Train Status** client. This allows the train, crew, and job information to be shared among multiple **CATS**, in a multi-desk (multi-dispatcher) system. Furthermore, when a train enters a station, the label is automatically moved on all **CATS** displays that show that station. This facilitates hand offs between dispatcher desks.

CATS should not be set up as both a **Train Status** server and a client.

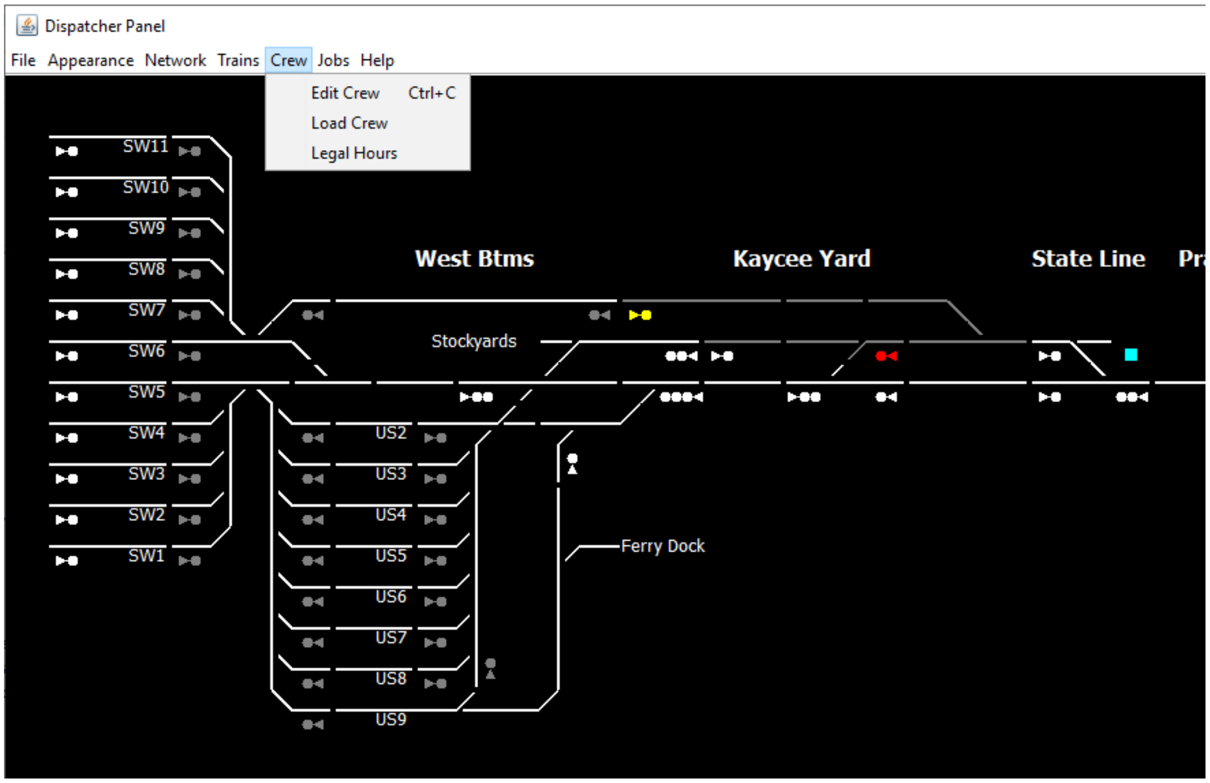
6. The next menu pull down is **Trains** (also via the control+t key sequence).

[2] To rerun a train, you will need to rerun it in **CATS** and in Operations.



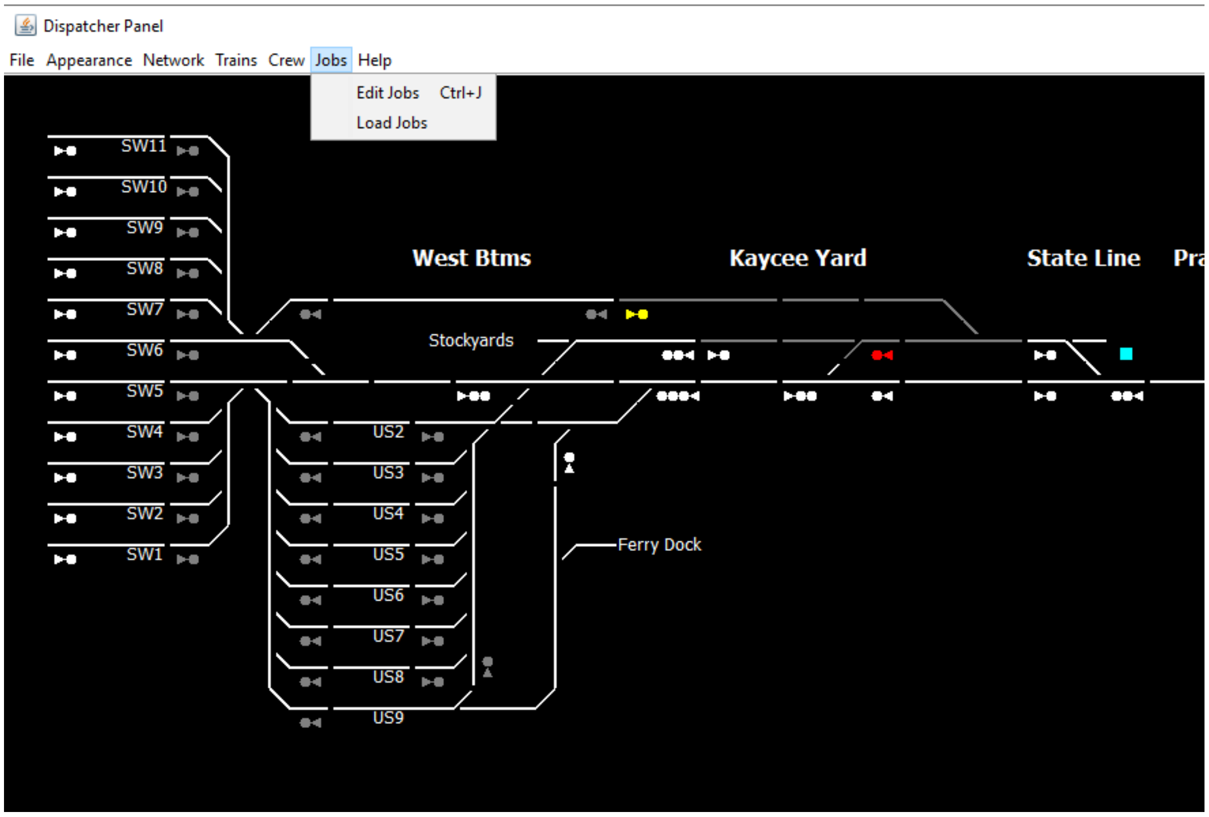
It is used for creating trains and changing their state. It has one menu item (**Load Lineup**) for reading in a lineup. The lineup is an XML file, created through **designer**. By allowing the lineup to exist in a separate file, you can have multiple lineups (e.g. even day/odd day or morning/afternoon) and select the one you want to use on the fly. It will be described in detail later. It has an **Edit Lineup** item for viewing, adding, and changing the information about a train. The last item, **Rerun Train**, is a way of running a train again, that has completed its work.

7. Next to **Trains** is **Crew** (also via the control+c key sequence).

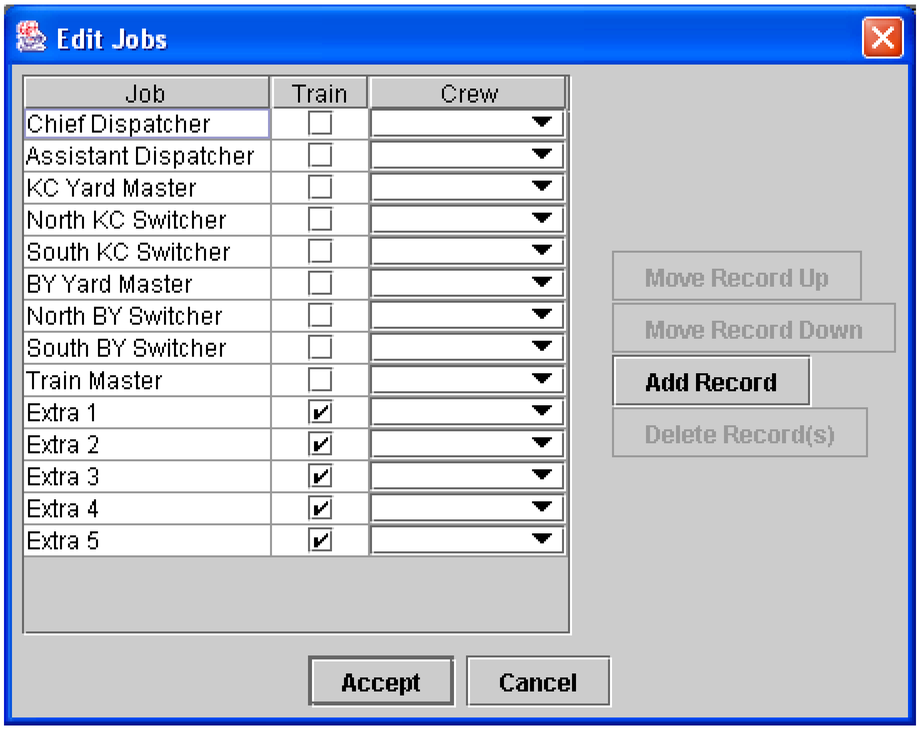


Use the **Edit Crew** selection to identify the crew assigned to each train. Though you can add and delete crews at any time, it is usually easiest to add them before starting the operating session. It has an option (**Load Crew**) for reading in a file which contains crew names, one name per line. It also has an option (**Legal Hours**) for setting the "hog law" – hours a crew can work before they must be relieved.

8. The next menu is the **Jobs** menu (also via the control+j key sequence).



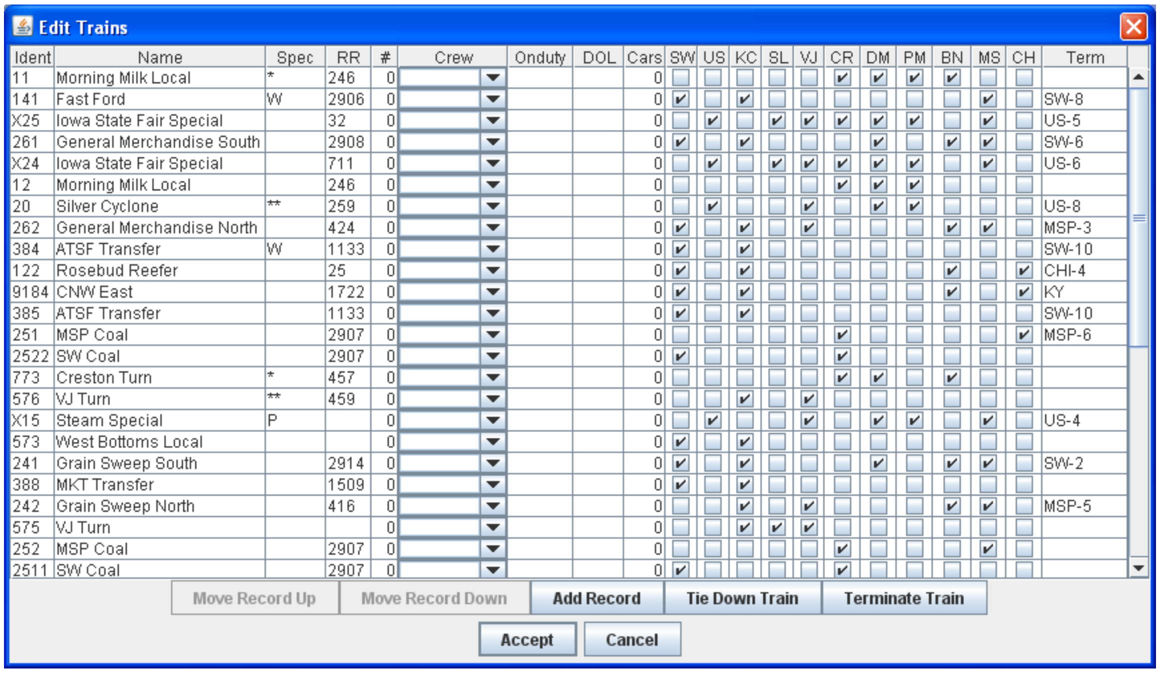
Like the Crews menu, it has an option to edit the list of jobs and job assignments and another option to read in a list of jobs.


Edit Jobs

Job	Train	Crew
Chief Dispatcher	☐	
Assistant Dispatcher	☐	
KC Yard Master	☐	
North KC Switcher	☐	
South KC Switcher	☐	
BY Yard Master	☐	
North BY Switcher	☐	
South BY Switcher	☐	
Train Master	☐	
Extra 1	✓	
Extra 2	✓	
Extra 3	✓	
Extra 4	✓	
Extra 5	✓	

9. After the trains, crew, and jobs are read in, you should assign crew to jobs. This step is strictly optional. Any crew not assigned to a job or assigned to a job with the "Train" field checked will be put on the "Extra Board" and appear in selection lists for assigning crew to trains.
   Notice that you can add multiple jobs at any time. You can select a block of jobs and reorder the list. You can adjust the column widths and if you select the "Accept" button, the adjustments will be remembered for the duration of the operating session. Finally, you can remove superfluous jobs for the rest of the operating session. However, you can change the titles and hide or expose other fields only in **designer**.
10. This next step prunes the train list. Often a schedule will have trains that will not run in a particular operating session. At the beginning of the operating session use **Trains** -> **Edit Lineup**[3].

[3] Alternatively, you can record the preparation of an operating session and terminate unused trains during preparation. When the preparation log is replayed, they will be removed from the active lineup.



Edit Trains

Ident	Name	Spec	RR	#	Crew	Onduty	DOL	Cars	SW	US	KC	SL	VJ	CR	DM	PM	BN	MS	CH	Term
11	Morning Milk Local	*	246	0				0						✓	✓	✓	✓			
141	Fast Ford	W	2906	0				0	✓		✓							✓		SW-8
X25	Iowa State Fair Special		32	0				0		✓		✓	✓	✓	✓	✓		✓		US-5
261	General Merchandise South		2908	0				0	✓		✓				✓		✓	✓		SW-6
X24	Iowa State Fair Special		711	0				0		✓		✓	✓	✓	✓	✓		✓		US-6
12	Morning Milk Local		246	0				0						✓	✓	✓				
20	Silver Cyclone	**	259	0				0		✓			✓		✓	✓				US-8
262	General Merchandise North		424	0				0	✓		✓		✓				✓	✓		MSP-3
384	ATSF Transfer	W	1133	0				0	✓		✓									SW-10
122	Rosebud Reefer		25	0				0	✓		✓						✓		✓	CHI-4
9184	CNW East		1722	0				0	✓		✓						✓		✓	KY
385	ATSF Transfer		1133	0				0	✓		✓									SW-10
251	MSP Coal		2907	0				0						✓					✓	MSP-6
2522	SW Coal		2907	0				0	✓					✓						
773	Creston Turn	*	457	0				0						✓	✓		✓			
576	VJ Turn	**	459	0				0			✓		✓							
X15	Steam Special	P		0				0		✓			✓		✓	✓		✓		US-4
573	West Bottoms Local			0				0	✓		✓									
241	Grain Sweep South		2914	0				0	✓		✓				✓		✓	✓		SW-2
388	MKT Transfer		1509	0				0	✓		✓									
242	Grain Sweep North		416	0				0			✓		✓				✓	✓		MSP-5
575	VJ Turn			0				0			✓	✓	✓							
252	MSP Coal		2907	0				0						✓				✓		
2511	SW Coal		2907	0				0	✓					✓						

That will bring up the list of all trains known by the dispatcher panel. Select one (or a block) and click on **Terminate Train** and that train will be hidden from lists and not appear on the layout. If you removed the wrong train, do not worry, it can be put back on the known list by using the **Trains** -> **Rerun Train** selection. You should cut back the list early, so you do not have to look at unwanted trains. Like the job screen, you can reorder the trains, add new trains, change the column widths, and change any information for a train that was set as "Editable" in **designer**. The **Tie Down Train** button works a little like the **Terminate Train** button. It is used to tell **CATS** that the train has completed its work, but the train's label remains on the panel.

Often, a visitor will bring a guest train. It is added to the list of trains with the **Trains** -> **Edit Train -> Add Record** menu item. Selecting it pops up a blank row under the currently selected row. Trains can be added at any time, but like crew, it is best to add them before the operating session begins, when the dispatcher is under less pressure.

11. Now that the train list is tailored to the operating session, you can position the trains on the layout screen[4]. To position a train, move the mouse cursor to the section of track corresponding to where the train should sit and touch the right mouse button. You will see a pop up menu. Select the **Position train** menu and click on **Accept**. You will see a list of trains which are not removed and are not on the screen. Select the one that is sitting on that track. This is another operation that can be done at any time, but it is best to set all the starting conditions before beginning so that the screen does not show an occupied track as empty. Some trains will not be on the layout when the operating session begins (for example, a train goes out with one symbol and

[4] See also the discussion on recording sessions and replaying logs for a way to automate this step.

comes back with another). That is fine. When the train appears on the layout, position it at that time.

12. Under the **File** menu is the **Replay** button. It is used to read in a session log file (created by **Start Recording** from above). It "replays" the log and moves trains, changes assignments, etc. It serves two useful purposes. It was intended to restore an operating session. On the Crandic, one of our dispatchers is quite talented at crashing **CATS** late in an operating session. Because of this creativity, it was easier to replay the activities that happened before the crash than "user proof" **CATS**. A fall out of adding this capability is that you can record the initial positioning of trains and other things (such as taking tracks out of service) in preparation for an operating session. At the beginning of the actual operating session (after loading the layout, trains, and crew), you can replay the preparation log. It much simplifies starting **CATS**.

    Replay will ask if you want to preserve timestamps from the log. The guideline is answer "no" when starting an operating session. Then, the timestamps will reflect the time at which the log is replayed, so all the movements will be recorded as though they just happened, which makes later analysis easier. If you are resuming an operating session, you will probably want to answer "yes" so that the durations of the previous segment are preserved.

13. We are almost ready to begin, but first the crew must be given their initial assignments. This step is also optional. There are several ways of doing this. At the beginning of an operating session, the easiest way is **Train** -> **Edit Lineup** or **Crew** -> **Edit Crew**. They do similar things, but from different perspectives. **Train** -> **Edit Lineup** brings up a list of trains that have not been removed and has a pull down list containing the crew list for each train. By selecting an item from the crew list, you associate a crew with a train. Conversely, **Crew** -> **Edit Crew** creates a list of crew (those not assigned to a job or assigned to a job with the Train box checked) with a pull down menu of trains that have not been removed. To associate a train with a crew, select the train. While you are making the assignments, nothing prevents you from assigning one crew to multiple trains or multiple trains to one crew. However, when you click on **Accept**, the assignments will be checked and a pop up error will tell you that there is a problem. The assignment window will remain until the conflict is resolved or the **Cancel** button is clicked.

<u>In general, the human interface follows the model of Windows programs. A single click of the left mouse button performs an action immediately. A single click of the right mouse button brings up a menu if the mouse is positioned over a train label, a section of track, or a signal symbol. The menu is tailored for the object under the mouse.</u>

Now we are ready to dispatch, or as Fred Carlson used to say, "Let the killing begin."

4 The Operating Session (CTC and DTC)

This section discusses running the program as a CTC machine. This is the way we like to hold an operating session on the Crandic. We like to run with ABS or APB during open houses because trains tend to simply run without doing much switching. This gets pretty boring and tedious for the dispatcher. Plus guests often break the dispatcher's concentration. The next section (5) will discuss what makes ABS and APB different from CTC[5].

4.1 *Setting Signals*

Under CTC, signals are in their most restrictive aspect (Stop), until the dispatcher reserves a route from the signal to its successor, in the direction of travel. When the dispatcher reserves a route, all signals in one direction of travel may show "movement allowed" indications. All signals facing the opposing direction of travel remain in their most restrictive aspect. Thus, a train is given permission to go from only point A to point B. Movement from point B to point A is prohibited by signals.

Reserving a route also locks the route. This means that until the existing reservation is cleared, a reservation cannot be made in the opposing direction or a turnout on the route cannot be changed by the dispatcher. The computer will not allow the dispatcher to set an unsafe route. An unsafe route is one which conflicts with an existing route, one in which the dispatcher has granted local switching to a block, one in which the dispatcher has taken a block out of service, one which has one or more turnouts aligned to a different route, or one in which a block is shown as occupied.

Depending upon how the layout is configured, local crew may still move turnouts. The **Appearance** -> **Reverse Local Operations** checkbox will cause **CATS** to move turnouts back, if the turnout has not been covered by track authority. The local crew will learn that moving turnouts without permission does them no good.

The signals for the reserved route will obey the "Signal Aspects and Indications" of the employee handbook (you supply the employee handbook). The symbols representing the signals on the dispatcher panel will be "empty" (white or grey) if not involved in a reserved route; red, if in the opposing direction; yellow, if the next signal is red; or green, if the next signal is not red. Thus, the symbols mirror the signals the engineer sees, to the extent that can be done with five colors. The colors of the signal icons are only loosely connected to the actual layout aspects. It is possible to define an aspect to show yellow (e.g. normal approach medium) and the icon to be green (because the next signal is non red).

The way the dispatcher reserves a route is to click the left mouse button when the mouse is positioned over an "empty" or "off" signal symbol. If the reservation is accepted, then

[5] Actually, all four disciplines can be mixed on the same layout. CTC or DTC is used where the dispatcher has direct control over the signals. ABS or APB is used where the dispatcher has no control over the signals.

the signal symbol changes color and the tracks composing the reserved route turn green with an arrow head pointing to the exit[6] of each block. A subtle distinction exists between white "empty" icons and "grey" empty icons. White ones have a physical signal associated with them on the layout. Grey ones do not; thus, the color difference is a reminder to the dispatcher that the train engineer does not see a signal that the dispatcher does.

Tip: you will get more reliable response by clicking on the signal icon "head" (not the mast) because of the way **CATS** looks for "hot zones".

There are two ways to clear a reservation. The dispatcher can cancel a reservation by clicking the left mouse button when positioned over a signal icon that is green or yellow. Alternatively, when a block within the reserved route is occupied, the reservation is cleared, but the block still shows occupied. This means that the signals again show their most restrictive aspect.

A Control Point (CP) is a signal (icon) on the dispatcher panel. An Intermediate Signal (IS) is a signal on the layout without an icon on the dispatcher panel. A route request propagates from the CP where the request is made, down the tracks, to the next CP. If there is an opposing reservation anywhere on any block, the request will be rejected. If there is at least one IS between the request origination signal and an obstruction, then the request will be allowed, but the reservation will stop at the IS protecting the obstruction. When the obstruction clears, the reservation will continue to propagate. If there is an obstruction between the request origination signal and any signal, the request will be denied. Similar rules hold for clearing a route.

Routes can be "fleeted" by using the right mouse button when positioned over a signal symbol. Fleeting is best used when one or more trains are taking the same route, in the same direction without opposing or crossing traffic. Fleeting means that after a block in a reserved route is occupied, then emptied, the reservation is renewed automatically. Fleeting is cleared by using the right mouse button and selecting the menu item or clicking the left mouse button when positioned over a signal symbol.

The logic behind fleeting forms the basis of DTC. Direct Train Control can be used when there are no signals protecting a block on the layout. In that situation, the dispatcher verbally tells the crew the limits of their train movement. The dispatcher clicks on the signal (similar to CTC) and the route is reserved. After the train passes over each block, the block does not return to idle, but shows a different color, indicating that the track is not in use, but has had its reservation fulfilled; thus, it belongs to the train that went over it. The dispatcher regains ownership by clicking on the signal.

If the dispatcher places the mouse cursor over a signal symbol that is red and clicks the left button, then the signal on the layout shows "Stop and Proceed". This behavior was requested (because the flashing red looks cool) so that the dispatcher could indicate that a train had permission to enter a block that contained a potential hazard (such as another

[6] The arrow head will be painted only if **Appearance->Direction Arrow** is checked.

train). (this feature is not yet implemented, but granting track authority yields a similar affect). Some railroads name this feature “call on”.

4.2 *Throwing Turnouts*

If a section of track contains a turnout, that turnout is under dispatcher control, and the block is not occupied, reserved, or given to local control, then the dispatcher can move the turnout by clicking the left mouse button when not over any signal symbols or train labels while the mouse cursor is near the switch points (preferably in the “vee” between the routes). If the layout supports turnout feedback, the track symbol will show no points aligned until told by the layout. If feedback is not supported, then the track symbol will change immediately. Note that in the latter case, the symbol may not accurately reflect the true state of the layout. If the points simply do not move on the panel, then be sure that the *spur* box is not checked on the points in **designer**.

If **Appearance->Lock Turnout Decoders** is checked, and the command to move a turnout would also move a turnout that is locked, **CATS** will not move the selected turnout. Thus, turnouts can be locked at the decoder level as well as the panel view.

The “hot zone” for detecting a mouse click is the rectangular tile enclosing the turnout. So, the mouse cursor does not have to be on the track line to register as a request to throw a turnout. A tile could have switch points on multiple edges. In that case, **CATS** cycles through the edges, flipping each, every time a click is detected.

4.3 *Train Detection*

If the layout supports occupancy, the tracks on the dispatcher panel will change to “occupied” (red) in response to detection messages from the layout. An occupied block in a reserved route will turn red, but the exit arrow will remain, showing the expected direction of travel of the train. When the detection clears, the reservation will be removed (unless fleeting is in affect for the block).

Blocks can manually be marked as occupied or cleared by using the right mouse button when positioned over the desired track.

Tracks which do not have detectors associated with them are painted in a grey color, to distinguish them from detected tracks. This is a reminder to the dispatcher that reservations on those tracks will not clear automatically. However, positioning a train label on undetected track will tell **CATS** that the track is occupied and **CATS** will color the track accordingly.

4.4 *Tracking Trains*

As noted in “Getting Started”, train labels can be placed on sections of tracks to record where trains are. Trains move, so the labels need to move. If train tracking is enabled

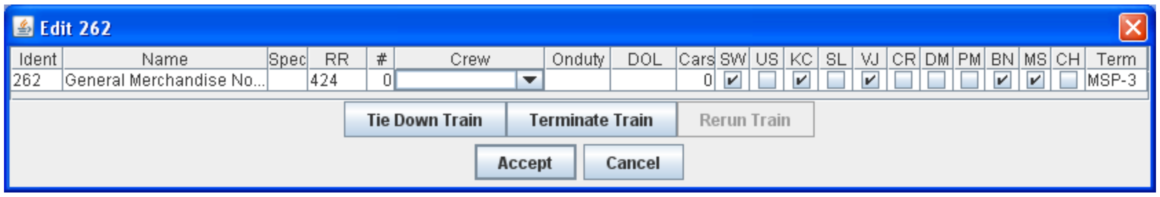
(**Appearance** -> **Train Tracking**), then the labels will follow detection reports automatically. Otherwise, the dispatcher will have to move them manually.

The simplest way to move a train label is to place the mouse cursor over the label and "drag" the label to another block by moving the mouse while holding down the left mouse button. The cursor changes from its default symbol to a cross inside a circle when the program recognizes the left button has been pushed when over a train label.

The other way of moving a train is to use the four arrow keys on the keyboard. The problem is knowing which train will move. The normal life cycle of a train is something like the following: it is created; it is positioned on the layout; crew is assigned to it; it does its work; it is tied down; it may be removed. The color of the train's label indicates which state it is in. Trains in the first and last state have no labels on the dispatcher panel; thus, have no color. A train that is positioned without a crew is "empty" (default light grey). A train with a crew is almond or blue. A train that has completed its chores is a rose color. Only one train will be colored blue – the one that has the focus and will be moved by the arrow keys. This train is one of the ones that is positioned on the layout and has a crew assigned to it. The "Page Up", "Page Down", and tab keys are used to cycle through this set of trains (many computers do not tell **CATS** when the tab key is pressed, so do not be surprised if it doesn't work). So, to move a train on the dispatcher panel it must have a crew assigned to it (coloring it almond or blue). If it is almond, it is selected by repeatedly pushing the "Page Up" or "Page Down" or "Tab" buttons until it turns blue. Then the arrow keys move it. The train will move in the direction of the arrow key, if the track goes in that direction. So, pushing the up arrow when a train label is located on horizontal track does nothing.

If recording is turned on, every time a train is moved, the movement is given a timestamp and recorded for further analysis.

If the right mouse button is clicked while the cursor is positioned over a train label, the following screen pops up:



Ident	Name	Spec	RR	#	Crew	Onduty	DOL	Cars	SW	US	KC	SL	VJ	CR	DM	PM	BN	MS	CH	Term
262	General Merchandise No...		424	0				0	✓		✓		✓				✓	✓		MSP-3

This can be used to edit the information about the train under the cursor and is very similar to the train edit screen, except only the information on one train is shown. Any changes take affect when the Accept button is pushed. Note the 3 buttons in the middle:

- "Tie Down Train" releases the crew and changes the color of the train's label, leaving it on the panel.
- "Terminate Train" releases the crew and removes the train's label from the panel.
- "Rerun Train" is an option on "tied down" trains, initializing one so that it can work some more.

4.5 *Track Authority*

Track Authority is granted to a train to perform local switching. This means the turnouts in the block are unlocked and the signals protecting the block are set to their most restrictive aspect, protecting the block from other trains.

Track Authority is placed on a block by positioning the mouse cursor over a block, clicking the right mouse button, selecting "Track Authority", and pushing the "Accept" button. Track Authority is removed by a similar operation. When Track Authority is placed on a block, the block is painted blue.

On the Crandic, giving Track Authority on a block is reflected on the layout by presenting a "Stop and Proceed" (flashing red) aspect on signals protecting the block.

Tip: It requires fine motor skills and a responsive mouse to position the cursor over a single track in an area of congested tracks. So, for using track authority, forcing occupancy, and using out of service, select a track in the desired detection block that does not have other nearby tracks.

4.6 *Out of Service*

If a block is having maintenance performed on it, then the dispatcher should place Out Of Service on the block. This is accomplished like "Track Authority" – placing the mouse cursor over the track, clicking the right mouse button, and selecting "Out of Service". OOS is removed by the same process.

On the Crandic, no special signal aspects are used on the layout to designate OOS, but the protecting signal drops to "Stop".

4.7 *The Callboard*

The crew callboard (created during setup) lists the crew and which trains they are assigned to. Whenever the dispatcher ties down a train or removes a train from the panel, the crew becomes unassigned and the callboard reflects that status. So, the program helps the dispatcher keep track of available crew.

If your operating session tracks crew time, you can set up **CATS** to assist in monitoring when crew was assigned to a train and when they have to be relieved. The "time on duty" algorithm is a little complex, but flexible. If all of the "ONDUTY" "EXPIRES AT", or "TIME LEFT" columns are hidden, then time is not monitored.

When determining "time on duty", CATS looks at the train's information. In the following order:

- If "ON DUTY" is blank, then the clock starts when the assignment is made.
- If "ON DUTY" is an absolute value (i.e. "HH:MM" with no leading '+' or '-'), then that is the time the first crew was assigned to the train. If the crew is relieved (another crew is assigned to the train), the relief crew is on duty when the relief

assignment is made. This option simulates the crew jumping on a train at the scheduled time, driving the train off-layout for awhile, and reaching the layout. It accounts for the fact that the crew had worked the train before it appeared on the layout from "somewhere else".

- If "ON DUTY" is a relative value (starts with '+' or '-'), the value is added to or subtracted from train's "DEPARTURE" time. This also simulates a crew working a train for some time before the train appears on the layout. In this case, though, the time worked is tied to the train's scheduled departure time, so if it changes, the crew "on duty" time changes.
- If the "DEPARTURE" time is blank, then the "ON DUTY" time is relative to when the train assignment is made.
- If the "DEPARTURE" time is absolute, then the "ON DUTY" time is relative to that time. For example, if "DEPARTURE" is 11:45, then "ON DUTY" times of "11:00" and "-00:45" are equivalent.
- If the "DEPARTURE TIME" is relative, then the departure time is computed relative to the time the assignment is made and the "ON DUTY" time is computed relative to that time.

CATS tries to make relative times absolute when they are first used. For "DEPARTURE", this is when a crew assignment is made. For "ON DUTY", this is when the train screen or crew screen is pulled up, if "DEPARTURE" is absolute.

The "dead on the law" time is computed from the "ON DUTY" time. The "HOURS" value is added to the "ON DUTY" time to arrive at when the crew should be relieved (EXPIRES). Thus, the "TIME LEFT" value is simply the amount of time between the current time and the "EXPIRES" time.

The clock used for computing these can be either the computer clock or a fast clock. So, HOURS should be chosen appropriately for the clock.

A future feature is to have **CATS** alert the dispatcher when a train is scheduled to depart or the crew should be relieved.

4.8 *Keyboard Shortcuts*

Moving a mouse around to the menu bar, pulling down a menu, and selecting a pop up window can be tedious and time consuming, particularly in the heat of dispatching. **CATS** supports a few keyboard shortcuts for accessing the frequently used pop up windows:

Key sequence	**Pop up menu**
Control+c	Edit Crew menu
Control+e	Switch between Engine labels and train symbols
Control+j	Edit Jobs menu
Control+t	Edit Trains menu

"Control+c" means touch the "c" key while holding down the control (or "ctrl") key.

5 The Open House

Layouts tend to be run differently during open houses (or work sessions) than during operating sessions. First, the dispatcher does not exert as much control over train movement because the emphasis of an open house is usually to keep trains moving as opposed to simulating prototypical operation. Second, the dispatcher often has distractions from visitors, so cannot concentrate as well during an open house. Thus, a signaling discipline different from CTC, that does not require as much care and feeding, might work best at an open house. ABS or APB fits that bill.

CTC could be called a restrictive discipline. Train movement is prohibited except on explicit action of a dispatcher. ABS and APB could be called permissive disciplines. Because no dispatcher is directly involved with the signals, train movement is allowed unless it is unsafe (the role of the dispatcher in ABS and APB is to prohibit movement, through some other means, such as track warrants, when it would be safe in the short term, but could cause larger scale problems). These opposing concepts are reflected in the “idle” signal indications. With CTC, an idle signal is the most restrictive aspect (stop). With ABS or APB, an idle signal is the least restrictive for the conditions (clear, approach, etc.).

What this means at an open house, is that the dispatcher only needs to tell operating trains when to take sidings or mains. The signals do the rest automatically. It requires less concentration for the dispatcher and is less button pushing. It is somewhat self-regulating at the expense of ignoring train priorities.

There is a significant difference between ABS and APB. ABS works best to maintain separation between trains going the same direction. APB works better than ABS with opposing traffic. APB incorporates ABS for traffic in the same direction.

The details are that ABS provides a two signal buffer on either side (in front of and behind) a train. If there is nothing in that buffer, then the engineer sees a “clear” aspect. If there is an occupied block one signal away, then the engineer sees an “approach” aspect. Finally, if there is a train in the block protected by the next signal, the engineer sees a “stop” aspect. Note that direction of travel is not a factor in determining signal aspect; thus, it is possible for two trains to end up nose to nose at an intermediate signal – a standoff because neither can go forward due to the other. APB enhances ABS by considering travel direction. If a train enters a clear block, then the computer does the same thing as when the dispatcher reserves a route – sets all signals up to the next control point, in the direction of travel to the least restrictive indication under the conditions, and sets all opposing signals to the most restrictive.

5.1 *Setting Signals*

Signal indications are determined by the computer based on how turnouts are aligned and where trains are. One difference between CTC and ABS or APB is that the signal symbols on the dispatcher panel reflect the aspect of the signal on the layout. With CTC, only the signal symbols involved in a reserved route are non-white (or non-grey). With

ABS or APB, no signals are white. Though not strictly prototypical, the dispatcher can set signals with ABS and APB and throw turnouts. This provides a hybrid form of dispatching.

5.2 *Throwing Turnouts*

Turnouts are thrown the same as with CTC – click the left mouse button while over track. If the track is colored empty (white grey), the turnout will change. If it is occupied or reserved through APB, it will not move.

5.3 *Train Detection*

Train detection is the same as with CTC. A block reporting train occupancy is colored red. Turnouts cannot be moved in an occupied block. The signals protecting an occupied block show their most restrictive aspect.

Occupancy always clears a reserved route because the computer does not know from which end of a block the next train will enter (there is no fleeting). ABS ignores direction of travel, so blocks are never reserved. APB tries to prevent head on meets, so it monitors the sequence in which blocks are occupied and reserves blocks between control points based on the order blocks are occupied.

5.4 *Tracking Trains*

Train labels are added to the dispatcher panel exactly as with CTC. Train labels are moved across the panel exactly as with CTC.

5.5 *Track Authority*

Track Authority is applied to a block exactly as with CTC.

5.6 *Out of Service*

A block is placed out of service exactly as with CTC.

5.7 *The Callboard*

The callboard works the same as under CTC.

6 Record/Playback

When I started writing **CATS**, I wanted it to record all train movements because I tend to forget to do so when I have dispatched model railroads. There are some dispatching systems (e.g. Time Table and Track Orders), where keeping paper records is “half the fun”, but since **CATS** simulates a modern, computer controlled dispatcher panel, the computer can record consistently and more accurately than the dispatcher.

I wanted the train movement records to reconstruct an operating session, as an assist into creating and fine tuning schedules. They have proven useful in other ways:

- Recording who worked which jobs (e.g. for the NMRA dispatcher AP)
- For automating setting up for an operating session
- For restarting an operating session that was interrupted

You may find other uses for the recordings, as well. Consequently, the following table describes what is recorded. The recording log is a set of ASCII text lines, with tab characters separating fields. This means you should not use tab characters in crew names, train symbols, block names, and other text entries into **CATS**. In the following, fields in **bold** are literal words and phrases in the records.

Event	Tag	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
First Record	**Created:**	Date	**Version:**	#	**CATS**	Version	
Last Record	**Ended:**	Date					
Add Record	**Added**	Time	Database Name	Field 1	Field 2	Field 3	Field 4
Change Record	**Changed**	Time	Database Name	Field 1	Field 2	Field 3	Field 4
Delete Record	**Deleted**	Time	Database Name	Field 1	Field 2	Field 3	Field 4
Train Assignment	**Assign:**	Time	Crew	**running**	Train		
Crew Relieved	**Assign:**	Time	Crew	**reassigned**			
Crew Reassigned	**Assign:**	Time	Crew	**reassigned**	Train		
Job Assignment	**Assign:**	Time	Crew	**assigned to**	Job		
Train Movement	**Move:**	Time	Train	From	**to**	To	Coordinates
Train Removed	**Terminated:**	Time	Train	Coordinates	Location	State	
Train Tied Down	**TiedDown:**	Time	Train	Coordinates	Location	State	
Train Rerun	**Rerun:**	Time	Train	Coordinates	Location	State	
Set OOS	**OOS:**	Time	**add**	Block			
Remove OOS	**OOS:**	Time	**remove**	Block			
Set Track Authority	**T&T:**	Time	**add**	Block			
Remove Track Authority	**T&T:**	Time	**remove**	Block			

Definitions:

- Date is the computer time and date
- Time is the computer time and date (if no fast clock) or the fast clock time and date
- # is the version number of the format of the record log. Currently, there are only two versions of the records (though the version number increments) Version 2 is the oldest and doesn't handle as many cases as later versions.
- Version is the CATS version identifier
- Database Name is the name of one of the collection of records (TRAINDATA, TRAINEDIT, JOBDATA, JOBEDIT, CREWDATA, CREWEDIT). The xEDIT collections describe the format and presentation of the xDATA collections. TRAINx, JOBx, and CREWx refer to the respective train, job, and crew information.
- Field n is the contents of a field in a collection. It is formatted as "field name"="field value".
- Crew is the name of a crew member
- Train is the unique symbol for a train
- Job is the name of a job
- From identifies a place on the layout, depending upon what has been defined. It is the "Station" entry from a detection block definition or "unknown" if the block defines no Station.
- To also identifies a place on the layout
- Coordinates are the coordinates on the layout, X, Y, and edge (if present). Edges are numbered 1-4, in the order right, bottom, left, top
- State contains the internal state of the train, for debugging
- Location is a place on the layout.
- Block is the name of a block

There can be a lot of tangled record changes in a single operation. Suppose two crew, assigned to two trains are interchanged. Four records will be altered in two phases. In the first phase, one crew will go from assigned to unassigned; one train will go from having a crew to having no crew; one crew will change trains, and one rain will change crew. In the second phase, the free crew will be assigned to the free train.

7 Compatibility

CATS depends a lot on some other programs. It needs a JRE (Java Runtime Environment, available for free from Sun Microsystems). This version of **CATS** was written with Java 1.8. It might run under versions earlier than 1.5, but that cannot be guaranteed. JMRI will not, so that is a moot observation.

CATS needs a layout description file created by **designer**. The XML structure also changes over time as features are added and changed and introduces the possibility of an incompatibility. The plan is that more attention will be paid to making **designer** backwards compatibility and providing the translation mechanism from older to newer versions.

Check the web site (or release notes for compatibility between CATS, **designer**, and JMRI.

8 Troubleshooting

I wish that I could say that **CATS** operated perfectly and has no bugs. I cannot. It is over 30,000 lines of Java code and not everything that can be done with it has been tested. Nonetheless, here are a few tips on troubleshooting problems.

9 The Disappearing Window

Many users create a shortcut and icon to the .bat or .csh file that launches **CATS** and puts it on the desktop, providing "one click launching". The problem occurs when the "one click" creates a console window, **CATS** has trouble starting up, dumps some messages to the Java console, dies, and the console window disappears. All of this happens so quickly that you cannot see what the trouble is. The solution is to run the startup script (.bat or .csh file) by hand. Depending on your operating system, launch a command window. Under Windows, this is performed by <u>Start</u>-><u>Run</u>. Then "change directory" to where the startup script is located. Under Windows, the command is usually "cd C:\"Program Files"\JMRI". Then run **CATS**. Again, under Windows, type "cats".

9.1 *Warnings and Error Messages*

Assuming the Java console window does not disappear, it could contain pleas for help from **CATS**. So, if things seem to quit working, look at the Java console.

Do not be alarmed if a message pops up when loading an older layout description telling you under what version of **designer** it was created with. As features are added to and changed in **designer** (and **CATS**), efforts are made to allow the new versions to read in descriptions created by older versions. This is called backwards compatibility. However, sometimes it is not possible for a particular version of **CATS** to read in any description created by any earlier version of **designer**. Because the amount of testing grows arithmetically with every new release, backwards compatibility is often not tested with descriptions. So, that message can provide some idea of how old the description is that is being updated. As noted above, you should try reading the old description into **designer**, saving it, and trying the transformed file.

9.2 *"Found a track in section (x,y) that is not in a Block"*

This message is shown during loading a CTC panel when **CATS** tries to find which detection block a track is in and none is found. Though it may happen at other times, it will occur when a panel created prior to version 2.1 containing a crossing is loaded into **CATS** 2.1. With that release, the two tracks composing a crossing were separated into distinct detection blocks (so that each could be controlled by a different signal discipline). The solution is to load the panel into **designer** 2.01 (or later) and check that the two tracks are in distinct detection blocks.

9.3 *Java Logging*

The debug philosophy for **CATS** has been evolving over time. The Java console is used for most warnings and error reports, but I am increasing the use of the log4j facilities. The file named “default.lcf” contains a filter on the kinds of messages that will be written and where they will be written. The location of this file depends on the operating system. It is usually in the same directory as the JMRI roster and configuration files.

9.4 *JMRI Preferences*

A very common problem is that you install Java, JMRI, **designer**, and **CATS** on your layout computer. You have worked through all of its quirks and have them running. You then copy JMRI and **CATS** to another computer to show it off (either in a positive or negative sense). You fire up **CATS**, get the disappearing window, and follow the troubleshooting tips above. You see a short message about “missing connection”. What is happening is that JMRI requires a configuration file that describes the connection to the layout. The first time JMRI is run, it prompts you for the connection and saves it in the JMRI configuration file. That file is usually not copied to a different computer because it does not live in the usual JMRI directory. So, when **CATS** runs, JMRI cannot find the configuration file and quits. The solution is to run one of the JMRI tools (DecoderPro, PanelPro, etc.) and set the connection.

9.5 *CATS Cannot Talk to the Layout*

From reading the JMRI user group discussions (http://groups.io/jmriusers/), it seems that one of the most frequent problems is getting the JMRI to talk to the layout. I would suggest that you get DecoderPro or PanelPro working before trying **CATS**. A lot more people work on JMRI than **CATS** and a lot more people use JMRI than **CATS**, so the experience base is considerably larger. **CATS** has been used primarily on Loconet systems with Locobuffer interfaces, so I am not familiar with other systems and stationary decoders. **CATS** relies on the expertise of JMRI for the layout connection.

9.6 *Turnouts Do Not Move*

Sometimes you will click on a turnout and nothing happens. The three frequent reasons for this are:

- The turnout is not under dispatcher control. It does not have an “automatic switch machine”. This means it is defined in **designer** as a “spur” or it may even be a crossing and not a turnout, at all.
- Java did not pick up a mouse event in the “hot spot”.
- The decoder is locked.

The **CATS** “hot spot” is the grid tile containing the turnout. For some reason, swing does not pass along the mouse click. So, you can try to click it again, or use the right mouse button. See also Section 4.2.

Another reason is that the turnout is locked. If decoder sharing is enabled, all turnouts sharing the same address will be locked when one is locked.

9.7 *JMRI Version*

CATS requires JMRI version 4.22 or later. Earlier releases of JMRI used an old library (jdk-jdom11.jar) for reading and creating XML files. Version 2 uses jdom2.jar, which has similar functions, but under different names.

9.8 *The Signals are not Right*

CATS takes multiple steps to generate the presentation on the signals from track condition (occupancy, turnout alignment, route reservations, etc.). So, if the presentation is not what you expect, there are several places to look. Here is how CATS works:

- It looks at the track conditions and generates a "rule".
- It passes the "rule" to the Signal.
- The Signal retrieves its aspect template (defined in **designer**).
- The Signal finds the aspect associated with the "rule" in the template.
- The signal retrieves the SignalHeads (JMRI abstraction) for each head that it drives.
- It picks the appearance out of the aspect for the appropriate head (assuming more than one), and tells the corresponding SignalHead to present that appearance.
- The SignalHead figures out the commands that implement that appearance and sends them to the layout.

So, what could go wrong?

- **CATS** is not generating the right "rule" for the track conditions (in which case it is a "bug", possible enhancement, or **CATS** cannot do it).
- The Signal is using the wrong aspect template (in which case, it can be corrected in **designer** on the screen where the physical signal is specified).
- The template has an incorrect aspect for the "rule" (which can be corrected in the aspect table in **designer**).
- The Signal is sending the wrong commands to the layout (which can be corrected in the decoder definition page for the signal in **designer**).

The JMRI Signal table is useful in determining what appearance **CATS** is trying to drive to the SignalHead. If it is as expected, then the problem is in the decoder definitions. If it is as not expected, then the problem is in the first three items.

9.9 *Command Line Options*

CATS looks one the command line for a few options when it starts up:

-WIDTH=nnn sets the initial width of the **CATS** window to be nnn pixels wide. The default is 640 pixels.

-HEIGHT=nnn sets the initial height of the **CATS** window to be nnn pixels high. The default is 400 pixels.

XML file name If the last thing on the command line is neither of the above two, **CATS** will assume that it is a layout file to be loaded when **CATS** starts up. **CATS** will do so. This makes launching **CATS** a one click operations.

Here is an example of a modified Windows .bat file command line:

```
"C:\Program Files (x86)\JMRI\LaunchJMRI.exe" cats.apps.Crandic -HEIGHT=900 -WIDTH=1200 Crandic.xml
```

And, here is an example for Linux or Mac:

```
. cats.csh Crandic.xml
```

For Linux and Mac, you can also set these in jmri.conf with the default_options line.

9.10 *Default Files Folder*

CATS needs files (e.g. the layout description). You can tailor where **CATS** looks (what folder it looks in) for its files. It uses two Java properties for guidance. Here is the order that it uses:

1. If it finds *cats.folder* defined on the command line, it will start with that folder.
2. If *cats.folder* is not defined or its definition cannot be found, it will look for *user.home*, which it should find because the Java jre defines that one. In Windows, that typically is C:\users\username\. Under Mac and Linux, it is the login folder (e.g. /home/me).
3. If that folder cannot be found, it uses the folder that **CATS** is launched from.

Java jre properties are defined on the command line. For example, to define the *cats.folde*r property, the command line would contain something like *–Dcats.folder=/home/me/cats*. This also can be placed in the jmri.conf file for Mac and Linux.

Here is a sample Windows .bat command:

```
"C:\Program Files (x86)\JMRI\LaunchJMRI.exe" -J-Dcats.folder="C:\Users\Rodney\Documents\tests"
cats.apps.Crandic
```

Under Mac and Linux, the command line might look like:

```
. cats.csh –Dcats.folder=/home/myuser/layout_files
```

There are also ways to turn these into .bat or .csh scripts or to set options in jmri.conf, making launching even easier. See the authoritative source at https://www.jmri.org/help/en/html/doc/Technical/StartUpScripts.shtml

10 Installing and Building from the Source Code, Under eclipse

This section is not for the faint of heart. It is for anyone interested in building the **CATS** application from the source code. I built it using *eclipse* (from www.eclipse.org, an open source development toolset). It should build under other development environments, but I had one that works, so have not experimented.

When I release a new version of **CATS**, I also post the source code. I package the source and *eclipse* settings in a zip file for separate download. Here is how I set up a new build environment:

1. I create a new *eclipse* project.
2. I download the *cats.zip* file from the web site.
3. I unzip the *cats.zip* file in a separate directory (folder).

4. I select (or create) the cats project from the *eclipse* project navigation list.
5. I import the *cats* “file system” where the unzipped files reside

CATS needs a few supporting files from the JMRI project:

- jmri.jar
- jdom2.jar
- log4j.jar

jmri.jar requires many other libraries, so I usually create another project in my eclipse workspace, “git” the jmri project, and use the jmri project as a required *eclipse* project. To run **CATS** under *eclipse*, I have to copy the jython, help, lib, resources, and xml folders from the jmri folder (either where the jmri executable lives or the jmri project).

It also needs a Java runtime, which comes with the Java Software Development Kit (SDK).

The launchers (*cats.bat* and *cats.csh*) are constructed by hand. They are very simple and rely on the JMRI launchers.

11 References

- “How to OPERATE Your Model Railroad”, by Bruce Chubb
- “Realistic Model Railroad Operations” by Tony Koester
- “All About Signals” by John Armstrong
- “Railroad Signaling” by Brian Solomon
- NMRA Operations Special Interest Group (OP_SIG) - http://www.opsig.org/resources.shtml
- “Absolute-Permissive Block Signals”, parts 1-5, by Jay Boggess, Model Railroader, Nov. 1991 – Mar 1992
- “Signal Basics”, parts 1-3, by Doug Geiger, NMRA Bulletin, Aug.-Nov., 1996
- http://www.lundsten .dk/us_signaling /abs_st_sp/ p_index.html
- http://broadway.pennsyrr.com/Rail/Signal/
- jmri.sourceforge.net
- http://deltareum.com/signal_progressions.htm