CATS User Guide (Version 3.0)
9/20/2023

1 Introduction .................................................................................................................. 2
  1.1 What is New and Different .................................................................................. 3

2 Setting Up .................................................................................................................... 11
  2.1 Starting CATS – Windows .................................................................................. 12
  2.2 Starting CATS – Macintosh .................................................................................. 12
  2.3 Starting CATS – Linux ......................................................................................... 13

3 Getting Started an Overview of the Pull Down Menus .............................................. 13
  3.1 Window Sizing ...................................................................................................... 13
  3.2 File Menu ............................................................................................................... 14
  3.3 Appearance Menu ............................................................................................... 14
    3.3.1 Colors .......................................................................................................... 15
    3.3.2 Fonts ........................................................................................................... 18
    3.3.3 Grid Size ...................................................................................................... 20
    3.3.4 Line Widths ................................................................................................. 20
    3.3.5 Refresh Screen .............................................................................................. 20
    3.3.6 Refresh Layout ............................................................................................. 20
    3.3.7 Test Layout .................................................................................................. 21
    3.3.8 Trace Items .................................................................................................. 21
    3.3.9 Adjustments .................................................................................................. 23
    3.3.10 Engine Labels ............................................................................................. 24
    3.3.11 Train Tracker ............................................................................................... 24
    3.3.12 Tee Base ..................................................................................................... 25
    3.3.13 Direction Arrow ......................................................................................... 26
    3.3.14 Automatic Wrapping .................................................................................. 26
    3.3.15 Lock Turnout Decoders ............................................................................. 27
    3.3.16 Reverse Local Operations ......................................................................... 28
    3.3.17 Flash Rate .................................................................................................... 28
  3.4 Network Menu ........................................................................................................ 28
  3.5 Trains Menu .......................................................................................................... 31
  3.6 Crew Menu ............................................................................................................ 32
  3.7 Jobs Menu ............................................................................................................. 33
  3.8 Assigning Crew to Jobs ....................................................................................... 35
  3.9 Positioning Train Labels ....................................................................................... 35
  3.10 Recording and Playing Back a Session .............................................................. 35

4 The Operating Session (CTC and DTC) .................................................................... 36
  4.1 What is a Route? .................................................................................................... 37
  4.2 Setting Signals (Signal Menu Items) .................................................................... 38
    4.2.1 Quick Route ................................................................................................. 39
    4.2.2 Extended (N/X) Route ............................................................................... 40
    4.2.3 Stacked Route ............................................................................................. 43
    4.2.4 Viewing and Changing Stacked Routes ...................................................... 44
    4.2.5 Call On (Return to Train) ............................................................................ 45
    4.2.6 Fleet ............................................................................................................. 46
    4.2.7 Grant Track Authority .................................................................................. 47
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 in the distribution package. 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 or OpenJDK), 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, car routing, and so on.

### 1.1 What is New and Different

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
<th>Section</th>
</tr>
</thead>
</table>
| 0.24    | 7/29/05  | • Added Jobs screen  
          |          | • Added checks on version of XML file  
          |          | • Added metered controls             | 3 (item 8) |
|         |          | • Added checks on version of XML file  
          |          | • Added metered controls             | 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) |
|         |          | • 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) |
|         |          | • 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 | 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 moves  
          |          | • Changed train tracking to be more robust | 4.3.1 |
|         |          | • 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 moves  
          |          | • Changed train tracking to be more robust | 3 |
|         |          | • 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 moves  
          |          | • Changed train tracking to be more robust | 3 (item 3) |
| 0.27    | 11/20/05 | • Added a time on duty field to Crew screen  
          |          | • Added routine for computing relative time  
          |          | • Added Crew hours support | 4.4 |
| 0.28    | 2/5/06   | • Minor bug fixes  
          |          | • Minor bug fixes | 4.4 |
| 0.29    | 4/6/06   | • Minor bug fixes  
<pre><code>      |          | • Minor bug fixes | 4.4 |
</code></pre>
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 0.30    | 4/26/06  | - Added turnout tests  
                  - Rewrote tracking algorithm                                             |
| 0.31    | 8/30/06  | - Minor bug fixes                                                      |
| 0.32    | 9/21/06  | - Made compatible with designer for N Scale convention                  |
|         |          | - 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 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”)                           |

Additional notes:
- **4/26/06**: Added turnout tests, rewrote tracking algorithm.
- **8/30/06**: Minor bug fixes.
- **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.
- **9/25/06**: Fixed bug with reservations, fixed outstanding bug with protected speeds.
- **10/22/06**: Bug fixes – no operational changes.
- **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.
- **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 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.
- **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
- 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.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.39</td>
<td>8/2/07</td>
<td>Fixed a bug that prevented a route from being re-established if a block in it cleared.</td>
</tr>
<tr>
<td>0.40</td>
<td>8/18/07</td>
<td>Implemented “Advance Approach”, “Advance Limited Approach”, “Advance Medium Approach”, and “Advance Slow Approach”. Added an option for tracing decoder locks</td>
</tr>
<tr>
<td>2.0</td>
<td>1/13/08</td>
<td>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.</td>
</tr>
<tr>
<td>2.1</td>
<td>3/3/2008</td>
<td>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.</td>
</tr>
<tr>
<td>2.11</td>
<td>3/23/08</td>
<td>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 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 recording file.</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Changes</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 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. |
| 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”.

### 2.16 7/21/09
- Updated for JMRI 2.5.5+ compatibility.

### 2.17 9/23/09
- 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
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.35</td>
<td></td>
<td>* Test release</td>
</tr>
<tr>
<td>2.37</td>
<td>3/20/2016</td>
<td>* 2.36 was built on a development release&lt;br&gt; * Signals protecting unbonded and end of track were wrong&lt;br&gt; * 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)</td>
</tr>
<tr>
<td>2.38</td>
<td>7/16/2016</td>
<td>* JMRI 4.3.5 had a change that prevented CATS from completely terminating when the Exit button was selected or window closer clicked&lt;br&gt; * 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.</td>
</tr>
<tr>
<td>2.39</td>
<td>1/15/2018</td>
<td>* Changes to JMRI after JMRI 4.8 broke things in CATS. These changes make CATS compatible, again.&lt;br&gt; * Modified the designer scripts so that they don’t specify the libraries needed, but the location of the libraries.</td>
</tr>
<tr>
<td>2.40</td>
<td>1/6/2019</td>
<td>* Some JMRI libraries marked as deprecated were finally removed, so CATS had to pick up the new interfaces&lt;br&gt; * A Linux user reported that COLORDEFINITIONs were not being found. CATS explicitly initializes them.&lt;br&gt; * 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.&lt;br&gt; * Restored approach lighting.&lt;br&gt; * Updated the cats.csh launcher to be more like the JMRI launcher.</td>
</tr>
<tr>
<td>2.41</td>
<td>8/25/2020</td>
<td>* Fixed some incompatibilities between CATS and JMRI&lt;br&gt; * A note is written on the console when CATS loads a JMRI panel&lt;br&gt; * CATS can use a default folder for file storage, other than the JMRI folder</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Changes</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.42</td>
<td>1/2/2021</td>
<td>- Reworked identifying turnouts with shared decoders, field movement of turnouts, and reversing field operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fixed an incompatibility between CATS and JMRI 4.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CATS can be a TrainStat client</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Train labels can be positioned via transponding, RFID, RailCommand, etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Approach lighting of signals is flexible and can be triggered by a JMRI device</td>
</tr>
<tr>
<td>2.43</td>
<td>7/1/2021</td>
<td>- Added the ability for train labels to have backgrounds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fixed a bug in identifying network connections in logs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fixed bugs in chains to work as diode matrixes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Advance aspects were not picked up from XML files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added warnings when detecting missing signal templates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Changes for compatibility with JMRI 4.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Updated cats.csh to be more like JMRI launcher</td>
</tr>
<tr>
<td>2.99.1</td>
<td>7/14/2022</td>
<td>- Many changes.</td>
</tr>
<tr>
<td>2.99.7</td>
<td>8/10/2022</td>
<td>- Reading JMRI devices on startup used to lockup</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fixed problem in crossings (diamonds)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fixed problem when setting following routes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added optional debouncing on loss of occupancy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fixed problems with fleeting and N/X routes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Changed priority on finding N/X routes from moving fewest points to most normal routes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Renamed signal icon menu item “Change Stack Routes” to “View/Change Stacked Routes”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Log short duration occupancy or unoccupancy reports</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Illustrated in this document occupancy analysis</td>
</tr>
<tr>
<td>2.99.8</td>
<td>10/27/2022</td>
<td>- Try to keep train labels on the leading detected occupancy block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a trace flag to assist in finding a rare keyboard lockup problem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Extended CATS so that only one occupancy detector needs to be defined in designer; unless 2 are defined, CATS will assume the other is the opposite polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added more information to identify incomplete</td>
</tr>
</tbody>
</table>
### SignalHead definitions for heads without software flashing
- Fixed a bug with fleeting where the controlling signal mast would cycle through several aspects
- Revised what happens when the first block in a route becomes occupied

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
<th>Section</th>
</tr>
</thead>
</table>
| 2.99.9  | 5/22/2023  | - Reworked Transponding  
- Added some changes to logging  
- Updated designer version accepted without warning  
- Pop-up windows are positioned close to cursor | 3.9     |
| 3.0     | 9/20/2023  | - Moved route processing from Swing thread to worker thread  
- Fixed a bug when deleting a stacked route which would delete it twice, resulting in a lockup  
- Fixed a bug that allowed a route to be created in the immediate block when occupied if there was already a route in the block |         |

## 2 Setting Up
Most files should not be placed in the same directory as the JMRI program 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 and add it for each you are not using. It must be in the JMRI program files folder.
- cats.csh is a command script for running CATS under Linux or MacOS. It must be in the JMRI program files folder.
- cats.doc is this document (should not be in the JMRI programs folder).
- cats.jar is the dispatcher panel program. It must be in the JMRI program files folder.
- 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 (should not be in the JMRI folder).
- designer.csh is a command script for running designer under Linux or MacOS (should not be in the JMRI folder).
- 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 (should not be in the JMRI folder).
- 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 (should not be in the JMRI folder).
- Lib is a folder for the libraries needed by designer (should not be in the JMRI folder, but must be in the same folder as designer.jar).
- .ICO are icon files to attach to launchers (should not be in the JMRI folder).

The minimum files to develop a panel 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 files 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 1.11, with 1.8 compliance. It needs Java 1.8 or later and JMRI 4.24-4.26 or 5.0-5.2. JMRI 5 requires Java 11.

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”. If using a shortcut, be sure to start in your JMRI program folder.

cats.bat invokes the JMRI launcher, telling it to start up with CATS (rather than PanelPro or DecoderPro, yet, they will get launched). The launcher will use the first cats jar file that it finds, regardless of name. That means to be sure that the version of CATS you want to execute is selected, remove all other versions of cats jar files from the JMRI program folder.

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 an Overview of the Pull Done Menus**

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 (MacOS or Linux) to start up CATS (See also Section 8.10 for startup customizations).
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.

**Warning**: if you have initial refresh enabled, be sure to wait 20-30 seconds after loading your panel for CATS to read back the state of your layout. If you try setting routes during the read back interval, signals may not show the correct colors.

3.1 **Window Sizing**

CATS borrows heavily from the idea of a word processor. With a word processor, the author just types. When the typing reaches the right hand margin, a hidden line feed is inserted and the typing picks up on the left hand margin of the next line down. Assuming the track plan was laid out in designer as a linear flow, CATS will do the same task of determining “line feeds”. As it reads in the layout description, it starts drawing track from the upper left, until it reaches the right edge. Then it jumps down past the drawn tracks and again begins drawing from the left edge.

With a word processor, adjusting the margins adjusts where he hidden “line feeds” occur. Similarly with CATS, resizing the enclosing window moves tracks between horizontal bands on the screen. This permits the user to adjust the screen to the “best” looking track presentation. It can be an iterative process, where track is added and removed in designer to shift where the track “breaks” to the next band in CATS.

**CATS** borrows another feature from word processing. Some systems have an option to “run the letters from margin to margin” or to put only complete words on a line, resulting in variable blank space along the right margin. **CATS** can do something similar, but rather than look for breaks between words, CATS uses the minimum number of tracks that have to be broken, beginning at the right edge. See Section 3.3.14 for more details.
3.2 **File Menu**

Use the **File -> Open** menu item to navigate to the XML description file and open it.

![Dispatcher Panel](image)

**Figure 1 File Menu Pull Down**

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 (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 analyzing train schedules. See section 6 for details of the record log. See item 3.10 for other ways to use the session log.

3.3 **Appearance Menu**

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 effect. Note that these changes apply only to this run of CATS. Permanent changes are made with designer.

3.3.1 Colors
The Color dialogue looks like:
These are the things that CATS paints in various colors and their current values. To change one, mouse click on the Color column for the row for the desired item. For example clicking on the Block Occupied row, Color column generates a pop up window that looks like (on Windows)

![Color Dialogue](image)

**Figure 3 Color Dialogue**
This is a standard Java Swing color selector. Note that other colors can be revealed through the HSV, HSL, RGB, and CMYK menus.

Mouse click on the desired color, then mouse click on the OK button. Most color changes will take effect immediately; if not, refresh the screen. New colors can be created through the Add button on the dialogue. Colors that you created can be deleted through the Delete button.

<table>
<thead>
<tr>
<th>Name</th>
<th>Object</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Occupied</td>
<td>Track</td>
<td>Occupancy is forced; or block detector active; or train label on undetected track</td>
</tr>
<tr>
<td>Block Empty</td>
<td>Track</td>
<td>not occupied, reserved, out of service, or involved in track authority</td>
</tr>
<tr>
<td>Block Reserved</td>
<td>Track</td>
<td>Track is included in a route reservation</td>
</tr>
<tr>
<td>Block Out of Service</td>
<td>Track</td>
<td>Dispatcher has taken the track out of service for maintenance</td>
</tr>
<tr>
<td>Block Held</td>
<td>Track</td>
<td>Unused</td>
</tr>
<tr>
<td>Track Authority</td>
<td>Track</td>
<td>Dispatcher has granted track authority on the track for local work</td>
</tr>
<tr>
<td>DTC Trail</td>
<td>Track</td>
<td>A DTC route that has been occupied and unoccupied. It designates the track as idle, but within the limits of some track permit.</td>
</tr>
<tr>
<td>Dark</td>
<td>Track</td>
<td>Undetected track, without a train.</td>
</tr>
<tr>
<td>Set Route</td>
<td>Track</td>
<td>Icon is half way through creating an extended route; the entry icon has been selected and is flashing; the extended route will be complete when the exit icon is selected</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Fleeting</td>
<td>Signal</td>
<td>For future use</td>
</tr>
<tr>
<td>Signal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entry/Exit</td>
<td>Signal</td>
<td>Unused</td>
</tr>
<tr>
<td>Stack Route</td>
<td>Signal</td>
<td>Unused</td>
</tr>
<tr>
<td>Authority</td>
<td>Signal</td>
<td>For future use</td>
</tr>
<tr>
<td>Mast</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Signal</td>
<td>Unused</td>
</tr>
<tr>
<td>Occupied</td>
<td>Track</td>
<td>Track has Track Authority or Out of Service and is occupied by a train</td>
</tr>
<tr>
<td>Warrant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Depot</td>
<td>Station</td>
<td>The Station icon</td>
</tr>
<tr>
<td>Idle Aspect</td>
<td>Signal</td>
<td>For CTC and DTC, no route reservation begins at the signal icon</td>
</tr>
<tr>
<td>Stop Aspect</td>
<td>Signal</td>
<td>the color used to paint signal icons which are not under the dispatcher’s direct control (i.e. ABS and APB) or are involved in an authorized train movement showing a Stop indication</td>
</tr>
<tr>
<td>Restricting</td>
<td>Signal</td>
<td>Unused</td>
</tr>
<tr>
<td>Aspect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call-on Aspect</td>
<td>Signal</td>
<td>Unused</td>
</tr>
<tr>
<td>In transition</td>
<td>Track</td>
<td>Unused</td>
</tr>
<tr>
<td>Approach</td>
<td>Signal</td>
<td>The physical signal is showing an Approach indication and it has a route (CTC and DTC) or is ABS or APB</td>
</tr>
<tr>
<td>Aspect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear Aspect</td>
<td>Signal</td>
<td>The physical signal is showing a Clear indication and it has a route (CTC and DTC) or is ABS or APB</td>
</tr>
<tr>
<td>No Physical</td>
<td>Signal</td>
<td>There is no physical signal and no route reservation begins at the signal icon. It is a reminder to the dispatcher that a command to “proceed on signal indication” cannot be obeyed.</td>
</tr>
<tr>
<td>Signal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Background</td>
<td>Panel</td>
<td>The background color of the panel</td>
</tr>
<tr>
<td>Label</td>
<td>Train</td>
<td>When backgrounds are enabled on train labels, this is the color of the background.</td>
</tr>
<tr>
<td>Background</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label Border</td>
<td>Train</td>
<td>When backgrounds are enabled on train labels, this is the color of the background’s border</td>
</tr>
</tbody>
</table>

### Table 1 Color Usages

#### 3.3.2 Fonts

Fonts are composed of:
- Color
- Size
- Emphasis (Plain, Bold, Italic, Bold and Italic)

Fonts are used when text strings are displayed. They are created and edited like colors.
A font color can be changed by mouse clicking on the Color column for an item. That will generate a color selector window, like above. Each entry in the Size and Emphasis columns are pull down selections. Changes generally take effect immediately; if not, refresh the Screen. Fonts can be created through the Add button. Fonts that you created can be deleted through the Delete button.

The items are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Object</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid Labels</td>
<td>Background</td>
<td>The default font to use for text strings on the panel</td>
</tr>
<tr>
<td>Train Identities</td>
<td>Train Labels</td>
<td>The default font to use for a train if CATS cannot find the requested font.</td>
</tr>
<tr>
<td>On Call Train</td>
<td>Train Labels</td>
<td>CATS uses this font on the panel with trains that have no crew assigned to them and have not been Tied Down or Terminated</td>
</tr>
<tr>
<td>Active Train</td>
<td>Train Labels</td>
<td>A train with a crew assigned and is not selected</td>
</tr>
<tr>
<td>Selected Train</td>
<td>Train Labels</td>
<td>A single train (selected from among the Active Trains through PgUp and PgDn keys). The arrow keys move the Selected</td>
</tr>
</tbody>
</table>
Table 2 Font Usages

<table>
<thead>
<tr>
<th>Font Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tied Down Train</td>
<td>A Train that the dispatcher has Tied Down (Terminated, but not removed from the panel).</td>
</tr>
<tr>
<td>Train Labels</td>
<td>The single character icon used to indicate that the crew has unlocked a turnout in the field.</td>
</tr>
</tbody>
</table>

Note that most of the fonts are used for painting train labels on the dispatcher panel. More fonts can be created and added to Trains for tailoring Train information on TrainStat clients. Those fonts do not affect the panel presentation. The font names get sent to the clients and the clients then map them to a local presentation. Thus, a train could stand out on one client and be subdued on another.

Furthermore, color selection is critical when being used with Train Label Backgrounds. CATS uses the fonts independent of the Label Background setting; thus, the label colors should be chosen to stand out against the background.

3.3.3 Grid Size

*Grid Size* is a placeholder for a future enhancement. Currently, the grid size of the underlying panel is fixed at 20 pixels. Someday, it will be adjustable.

3.3.4 Line Widths

The *Line Widths* sub-menu is used for changing the widths of the track. It has two sub-pieces. One is for changing the width of horizontal and vertical track. The other is for changing the width of diagonal track. The values are real numbers of the form x.y. Larger values result in wider track and smaller result in thinner track.

3.3.5 Refresh Screen

*Refresh Screen* reads back the JMRI devices (decoders) and repaints the dispatcher panel based on the values read back; thus, it synchronizes the view CATS has of the layout with the view JMRI has. As a little background, CATS listens to JMRI for the values of JMRI devices. JMRI gives CATS a value only when JMRI sees a change. So, if CATS misses the change notification, then CATS and JMRI are out of synchronization on that value. CATS reads back all JMRI devices when *Refresh Screen* is requested. It then executes the same program code as though JMRI told it that a value changed. If CATS has the same value, then nothing happens. If CATS had a different value, then CATS handles it as a change, which can result in changes to the panel.

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*.

3.3.6 Refresh Layout

*Refresh Layout* is the reverse of *Refresh Screen*. CATS tells JMRI what it thinks should be the value of each JMRI device that CATS controls. Typically, this operation should be done after running a device test in CATS, to restore the layout to the state CATS saw it before the testing.
3.3.7 Test Layout
This section contains several tests designed to exercise the layout hardware, providing a way to quickly check that the hardware is functioning properly. It provides signal and turnout tests. No safety checking is performed, so be sure that the turnouts are clear before running a turnout test. If something does not do what is expected, there are two important places to look:

- The wiring
- The XML file

After completing testing, run Refresh Layout to change all the devices back to the last known operating states.

3.3.8 Trace Items
Trace items branches out into multiple things that can provide information as to what CATS is doing. I use it for on demand debugging, but most people will not find most traces helpful. These are all checkboxes (i.e. they are on or off). A few that might be useful are:

- Occupancy Duration attempts to record how long each occupancy detector showed occupied and vacant. It does so by timing when the occupancy report was received to when it was removed. The timings are collected into buckets and the number of counts in each bucket is displayed in JMRI Memory devices. The intent is to identify any abnormally short occupancies, which may indicate a hardware problem or any abnormally short unoccupancies, which may indicate a faulty detection resistor, warped truck (all four wheels are not touching the rails), dirty wheels, etc.
- XML Reader is useful if there are problems loading an XML file (i.e. CATS crashes on reading a file). Check mark it, load the file, and see the line number in the file where CATS crashed.
- Signal Changes is useful for seeing what indications and aspects CATS computes for various layout events. Naming each signal in designer helps identify what signal is changing.
- Operations is useful for seeing the message exchanges between CATS and JMRI Operations.
- Transponding Messages is useful for discovering what values are placed in JMRI Reporters when a transponder is read or an RFID chip is read.

In all cases (except Occupancy Duration), the trace results appear on the JMRI System Console. In addition, for Occupancy Duration tracing, messages will appear in the System Log, identifying any durations of less than one second, including the Block name attached to the detector with the report. For example
The occupancy duration “spectrum” is recorded in JMRI main screen->Tools->Tables->Memory Variables, like

**Figure 6 Short Occupancy Messages**

Note two groups of lines. The ones with System Names like IMDECODER_ACTIVE_n are counters for the duration of block occupies. Those with System Names like
IMDECODER_INACTIVE_n are for counters for the duration of block unoccupies. The “_n” denotes the length of the duration. “_1” counts the number between 0 and 1 second, _2 counts the number between 1 and 2 seconds, etc. up to _5 for 4 to 5 seconds. All durations over 5 seconds are accumulated in _max. By themselves, these counts should not be alarming. Rather, if the CTC panel is affected (e.g. routes are clearing unexpectedly or train labels do not track well), then you might change the Adjustments->Debounce values to filter out the short duration events. The counts should provide guidance as to how much debounce to apply, typically 1 to 2 seconds. Setting a debounce timer will not affect these counters. It will affect which reports are acted on.

### 3.3.9 Adjustments

Adjustments contain items that are timed events in CATS. Mouse clicking on anyone pulls up a selection list of reasonable values.

- **Occupancy Debounce** is a way of filtering out short occupancy reports. If non-zero, CATS starts a timer when an occupancy report is received. If the report does not drop before the timer expires, then CATS accepts the report and processes it. If it does drop, then CATS ignores the report. This does mean that occupancy reports will be delayed for the amount of time selected.

- **Unoccupancy Debounce** is a way of filtering out short unoccupied reports. If non-zero, CATS starts a timer when an unoccupancy report is received. If an occupancy report is not received before the timer expires, then CATS accepts the unoccupancy report and processes it. If one is received, then CATS ignores the unoccupied report. This does mean that unoccupancy reports will be delayed for the amount of time selected.

- **Refresh Delay** slows down the messages sent to the layout in response to Refresh Layout. The problem is that CATS and JMRI possibly can generate messages for the layout faster than they can be delivered. When the delay is set to a non-zero value, CATS will send 10 messages, wait for that amount of time (in milliseconds), send the next batch of 10, etc.

- **Code Line Delays** are used for removing some of the video game feel from CATS and providing a more realistic dispatching experience. Typically, prototypical signal systems have a two level architecture. The office equipment (dispatching computer) sits in a central location distant from the track and signals. The office equipment sends commands to the field equipment (trackside) and receives indications (status reports) back. When all of this was done with relays, it took seconds to transmit a message (code) from the office equipment to the field equipment and seconds to receive the indication. Furthermore, some commands (e.g. move the points) take even more time to accomplish before returning an indication. To simulate a code line, CATS will delay commands and indications, slowing things down. There are separate delays for commands destined for the layout and indications coming back from the layout. Note that when changing from some delay to no delay, it can take some time for the queued up messages and indications to transmit before the no delay takes effect.

- **Time Lock** specifies how long CATS should “run time” on turnouts. “Running time” occurs when a route is cancelled (signal drops to red) in front of a train. The engineer would see the signal go red, but may not have time to come to a complete stop and
will overrun the signal. To minimize derailments, the turnouts protected by the signal are locked for a period of time and the dispatcher is prevented from moving them. The amount of time that they are locked is from the selection list. CATS will “run time” only when a signal has been cleared for a route, the route is cancelled (for any reason), and the block immediately in front of the signal is occupied.

- Turnout Safety Delay is used in conjunction with CATS deciding to throw turnouts. The impetus behind this delay is that occupancy detection on a model railroad is often not reliable. Specifically, block occupancy may not be solidly active for the duration of a train passing through the block (typically over turnouts). The occupancy will blink “off” for a second or two, then back on. This can be seen by watching tracks blink on and off as a train passes through them. Furthermore, CATS features involving extended routes, can move points. To minimize throwing points under trains, CATS wants the turnouts to be stable – unoccupied – before moving them. In those situations, CATS will wait for the block containing the turnout to report unoccupied and wait for the safety delay timeout to expire. If there were no occupancy reports during that time, CATS will continue with the operation; otherwise, CATS will start the whole process over, waiting for the unoccupied report. If your occupancy detectors are reliable, then select a small value. If they are mostly reliable, then select a larger value. If they are not reliable, either be prepared for derailments due to CATS throwing points under trains or don’t use extended routes.

3.3.10 Engine Labels
This checkbox selects the presentation of the train label on the panel. When unchecked, the train symbol is used as the label. When checked, the lead engine identity is used as the label.

![Figure 8 Train Label Selection](image)

3.3.11 Train Tracker
The Train Tracker checkbox controls the display of train labels. When checked, train labels are displayed to show the position of trains on the layout. When unchecked, labels are not shown.

A label is placed on a track by right mouse click over track and selecting Position Train.
That action will pop up a list of trains, from which one can be selected.

The label will follow block occupancy reports, if detection is reliable. The dispatcher can also “drag and drop” the label, if it goes to the wrong track. The label remains on the layout until the train is terminated. The general rule is to position the label on the lead occupied block. The label may be lost if the train reverses direction.

If a block has a Reporter attached to it, then when block detection hardware (e.g. RFID reader or transponding) picks up the train identity, CATS will move the label to the reported block.

3.3.12 Tee Base
The Tee Base checkbox selects the how the base of signal icons are drawn. When unchecked, the bases are drawn as triangles.
When check marked, bases are drawn as inverted tees.

It may be necessary to Refresh Screen when switching between the two.

### 3.3.13 Direction Arrow

The Direction Arrow checkbox controls the presentation of arrow heads on routes, showing the direction the train will proceed across the route. When checked, routes have arrow heads. When unchecked, they do not.

### 3.3.14 Automatic Wrapping

The Automatic Wrapping checkbox is used in drawing tracks on the screen. When unchecked and if the track plan is wider than the screen, **CATS** will draw tracks up to the right screen edge before continuing drawing from the left edge, below the drawn track. When checked, **CATS** will determine where the right edge would fall on the track plan, then work to the left, looking for where the edge would cross the fewest number of tracks (in the right ¼ of the plan). It will then “break” at that point to draw the next band of tracks. To fit
the most track on the screen, uncheck this box and adjust the right window edge for best appearance.

3.3.15 Lock Turnout Decoders
In some track configurations (e.g. crossover), multiple turnouts share a JMRI device address, so they change in unison. When one set of points move, the others also move. When this checkbox is check marked, then CATS will look at the turnouts sharing a device before issuing the command to move any of them. They all must be safe for CATS to move any of them. Thus, CATS looks for occupancy, Track Authority, Out of Service, etc. on all shared turnouts before sending any layout commands. This is to prevent derailments. For example, consider a crossing from one track to a parallel track with both turnouts synchronized to be lined for the parallel route or for the crossing route. Thus, when CATS sees the feedback from a requested turnout movement and the feedback from the other (non-requested) turnout movement, it will allow the latter, since that is a result of sharing the JMRI address. When
not check marked, CATS treats the turnouts as uncoupled; thus, is surprised by the turnouts moving.

The general rule is if there is even one set of turnouts sharing a JMRI device, then check mark this box. If there are no shared JMRI devices, then uncheck mark it.

### 3.3.16 Reverse Local Operations

As noted in the designer document, there are multiple ways to wire up turnouts for multiple control schemes. In some of these, the layout owner may desire that the dispatcher have absolute control over moving points. The field can move points only when the turnout is under Track Authority or Out of Service. However, the turnout may be wired so that the field crew can move it at any time. This checkbox tells CATS to prevent unauthorized turnout movement by the field (to the extent allowable by the wiring). CATS does this by watching for unexpected movements of points and when discovered, sending the commands to move the points back (command override or reverse local operations). When the box is checked, CATS will send the override commands. When the box is not checked, CATS will ignore the unauthorized turnout movement.

Note that there are undesirable interactions between this setting and Lock Turnout Decoders. If this box is checked and the latter is not, then a situation of infinite do/undo can result. Suppose the dispatcher moves the turnout on one side of a cross over. Since the turnouts may share the JMRI address, the other side may move, as well. But, since the dispatcher did not ask it to move, CATS sees the movement as unauthorized and sends the commands to restore its alignment. This results in the authorized turnout moving back, which is itself unauthorized, so CATS tries to move it back to the desired alignment. This sequence will happen forever or until the hardware breaks, CATS is shutdown, or one of these check boxes is changed.

### 3.3.17 Flash Rate

If CATS provides the timing for flashing signals (software flashing), then the last item (Flash Rate) allows you to fine tune the flash rate. This is the amount of time the signal head is on or off, in milliseconds. Larger amounts mean they flash slower and smaller amounts mean they flash faster.

### 3.4 Network Menu

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.
Figure 15 Network Menu

The top third of the pull down describes the network:

- IP address being used by the CATS computer (this is not accurate, depending on Java security settings)
- Network name of the CATS computer
- If the CATS computer has an active network connection or not
- A count of 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 JMRI main screen, Edit->Preferences->Start Up->Add->Perform Action …,
Figure 18 Starting JMRI Simple 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 down¹.

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 makes 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 must not be set up as both a Train Status server and a client.

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

¹ 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.

The trains are not needed to dispatch a layout; however, they are needed for tracking the labels on the screen (See 3.3.11).

### 3.6 Crew Menu

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.

The crew table is not needed for dispatching; however, it is needed for moving train labels with arrow keys.

### 3.7 Jobs Menu

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.

The jobs table is optional for dispatching. Ignoring it does not affect dispatching activities.
3.8 Assigning Crew to Jobs

After the trains, crew, and jobs are read in, you can 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 (columns) only in designer.

3.9 Positioning Train Labels

You can position the train labels on the layout screen. 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 the session so that the screen does not show an occupied track without a train. Some trains will not be on the layout when the operating session begins (for example, a train goes out with one symbol and comes back with another). That is fine. When the train appears on the layout, position it at that time. See also Section 3.3.11.

3.10 Recording and Playing Back a Session

Under the File menu is the Replay button. It is used to read in a session log file (created by Start Recording from Section 3.1). 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 completely “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.

---

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

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.

<table>
<thead>
<tr>
<th>Object under mouse</th>
<th>Left mouse</th>
<th>Right mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal symbol</td>
<td>Set/Clear route</td>
<td>Menu – see Section 4.2</td>
</tr>
<tr>
<td>Train label</td>
<td>Drag and drop</td>
<td>Menu – see below</td>
</tr>
<tr>
<td>Track</td>
<td>Move Turnout points</td>
<td>Menu – see below</td>
</tr>
</tbody>
</table>

The menus are context sensitive (i.e. some options may be disabled by appearing to be greyed out, if they cannot be done, based on current conditions).

The signal menu is the most complex.

Reset Route is always an option. It is intended to reset the contents of the control point and all tracks in all directions, up to the next control points. This is to clear things out if routes are denied for no apparent reason.

See Section 4.2 for details on the other selections.

A train label menu looks like:

---

3 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.
It is the row from the Trains table. It can be edited, just like rows in the Trains table can.

A track menu looks like:

The selections apply to the block holding the track:

- **Occupy Block or Unoccupy Block** – force the block to be occupied or unoccupied
- **Grant Track Authority or Remove Track Authority** – unlocks dispatcher control of any turnouts in the block; colors the track to show track authority; and prevents routes being set through the block
- **Grant Out of Service or Remove Out of Service** – similar to Track Authority, but is intended to mark track as non-functional rather than allow local work
- **Position Train** – generates a list of trains which have not been terminated or tied down; the dispatcher selects one and its label is placed on the track

It is possible to have two tracks running through the grid tile. So, ambiguity exists as to which the track menu will apply to. It is best not to pull up the track menu in such a situation.

### 4.1 What is a Route?

The primary job of the dispatcher is to manage the flow of trains across the layout. The way this is done is by mouse clicking on icons on the **CATS** panel, which generate commands to the layout to effect changes (particularly move turnouts and change the colors or blade positions on signals). With CTC and DTC dispatching, signals show the train crew two things

- The safety of the tracks past the signal, up to the next signal (in the direction of travel)
- Permission to occupy those tracks past the signal, up to the next control point signal (in the direction of travel)

The principle mechanism the dispatcher uses for encapsulating that information is a *route*. The *route* begins at a dispatcher controlled signal and traverses all tracks from that signal to
the next dispatcher controlled signal. A track touched by a *route* becomes locked from most kinds of changes:

- An opposing *route* cannot be created
- A conflicting *route* cannot be created
- Turnouts cannot be moved
- Turnouts cannot be unlocked for field movement (though the dispatcher can grant Track Authority on a block involved in a route)

Signal icons designate dispatcher controlled signals. A left mouse click on a signal icon requests that a *route* be created. It will be denied if any of the following unsafe conditions are discovered on any track that would be involved in the *route*:

- An opposing *route* exists or is being created
- A conflicting *route* exits or is being created
- A turnout has been unlocked (either by the dispatcher or in the field)
- A block shows that it is occupied by a train
- A turnout is lined so that the train would derail going over it
- A block is under Track Authority
- A block is marked as Out of Service
- An external hazard is active (e.g. tunnel curtain closed, draw bridge up, derail active, etc.)

The *route* exists until one of the following happens:

- The dispatcher removes it
- When no blocks are occupied after one has become occupied
- Some other action nullifies it (such as the field crew moves a turnout)

A *route* may span multiple blocks, multiple intermediate signals, and multiple turnouts.

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* propagates from the CP where the request is made, down the tracks, to the next CP. If there is an opposing route anywhere on any block, the request will be rejected. If there is any obstruction (except for occupancy) between the origination CP and ending CP, the request will be denied. If the blocks between the origination CP and first IS (or ending CP) are occupied, the request will be denied. The request will be allowed if blocks protected by IS are occupied (the IS will provide protection for following movements). The color of the origination icon will show when another route request can be made.

Clearing a route will clear up to any occupied block. Occupied blocks will clear as they become unoccupied.

### 4.2 Setting Signals (Signal Menu Items)

Under CTC, signals are in their most restrictive aspect (Stop), until the dispatcher sets a route from the signal to its successor, in the direction of travel. When the dispatcher sets 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 (Stop) aspect. Thus, a

---

4 **CATS** supports sectional release of a *route*. As a block in the route is occupied then vacated, the block is freed for another *route* or operation.
train is given permission to go from only point A (first signal icon) to point B (last signal icon). Movement from point B to point A is prohibited by signals. This prohibition applies to even “backing up” a train without dispatcher permission.

Setting a route also locks the route. This means that until the existing route is cleared, a route 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 fouling alignment, or one in which a block is shown as occupied.

Depending upon how the layout is wired, local crew may move turnouts, even without dispatcher permission. The Appearance -> Reverse Local Operations checkbox (see Section 3.3.16) will cause CATS to move turnouts back, if the turnout has not been covered by track authority. The local crew quickly will learn that moving turnouts without permission does them no good.

The signals for the route will obey the “Signal Aspects and Indications” of the employee handbook (you supply the employee handbook). The icons representing the signals on the dispatcher panel will be “idle” (white or grey) (see Table 1 Color Usages) if not involved in a route; red, if some track is blocked; yellow, if the next signal is red; or green, if the next signal is not red. Thus, the icons 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).

### 4.2.1 Quick Route

This is the easiest and fastest way to set a route. The dispatcher sets a route by clicking the left mouse button when the mouse is positioned over an “idle” or “off” signal icon. If the request is accepted, then the icon changes color and the tracks composing the route turn green with an arrow head pointing to the exit\(^5\) of each block. A subtle distinction exists between white “idle” icons and “grey” idle 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.

The icon reflects the condition of the tracks in a route up to the next signal – either an IS or CP – and not the condition of the complete route. Due to sectional releasing, a reverse route can be lined as soon as a route through a block is cleared. Plus, the controlling signal icon will go to “idle” when the first block is occupied and unoccupied.

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”.

---

\(^5\) The arrow head will be painted only if Appearance->Direction Arrow is checked.
There are several ways to clear a route. The dispatcher can cancel a route by clicking the left mouse button when positioned over a signal icon that is green or yellow. Alternatively, when no blocks within the route are occupied (after one is), the route will automatically clear. Finally, if the crew move a turnout in the field, the route will clear. It must be noted that no turnouts are moved in creating a Quick Route. The route follows the path of the tracks from the originating signal until discovering:

1. A CP in the direction of travel
2. End of track (end of track is defined as no more track or a block whose discipline is ABS or Unknown. The latter is so that routes do not traverse yards or unbonded track).
3. A blocking condition

In other words, the path for the route is fixed when the route is created and the task of CATS is to find the end of the route by following the tracks.

4.2.2 Extended (N/X) Route

An Extended Route is a chain of Quick Routes. The dispatcher selects the signal icon where the route originates, selects the icon where the route ends, and CATS determines if an open path exits between the two. If such a path exists, CATS will move any turnouts needed to create that path, then create a Quick Route on all CPs along the path.

4.2.2.1 Creating an Extended Route

In more detail, the process of creating an extended route begins with the dispatcher right mouse clicking on a signal icon. That brings up the signal menu (Figure 19 Signal Menu). The dispatcher highlights “Set N/X Route” (if it is not greyed out) and clicks on Accept. The icon then blinks in the Entry/Exit color (Figure 3 Color Dialogue). The blinking indicates that CATS will use that signal as the starting signal and is waiting for the dispatcher to select an ending signal, with a left mouse click. Strictly speaking, N/X (eNtry/eXit) suggests that the second signal should be in the opposing direction of travel (the exit signal). That seems a little counter intuitive to any one not steeped in dispatching because the Extended Route will go through that signal up to the next CP in the direction of travel. It seems more natural to select the end of the route by clicking on a CP in the same direction of travel. There is value in allowing selecting the exit signal because there may not be a “next” CP in the direction of travel (e.g. going off layout, entering a siding, and other cases). Consequently, CATS will accept either direction of travel for the ending signal. If it is an opposing direction, then CATS will continue the Extended Route to the next signal in the direction of travel.

In the process of searching for the end of route signal, CATS will remember all possible paths (track alignments) to get from start to end. Metrics are generated for each segment of each path. The metrics count:

1. Blocking conditions (occupancy, Track Authority, Out of Service, external events)
2. Number of turnouts that must be moved to generate that path
3. Number of normal (as opposed to diverging) turnouts traversed

The metrics are determined on a segment basis and summed on a path basis. If a path has no blocking segments (item 1 above is 0), then an Extended Route will be created. If there are multiple possible paths, then the one with the most number of normal turnout alignments will
be chosen. The final tie breaker is to use the path with the least number of turnouts to move and the largest number of normal turnout alignments.

Several quirks should be noted about creating Extended Routes:

- **CATS** does not consider possible paths after an exit signal. It uses the existing points alignment. If there is a blockage after an exit signal and before the path to the next CP, then **CATS** will reject the Extended Route, even if moving a turnout could yield a clear path.
- **CATS** will run a route across the normal alignment of a spur, but not the non-normal alignment, even if the spur is set for the non-normal position. This is because the dispatcher has no control over how spurs are lined.
- The route construction algorithm prefers normal paths through turnouts over current alignment. The intent is to route through main tracks over sidings. The way to override this preference is to select an exit signal icon in the siding.
- There may be two paths (one from each direction) to an ending signal if there is a reversing loop. To bias the selection, set a turnout or two on the reversing loop.
- Though **CATS** does a pretty good job of remembering what tracks it has visited when discovering paths, loops can foul it up. So, use Extended Route creation with caution (or not at all) within a loop, if the track plan contains a loop.
- To avoid getting stuck searching around a loop indefinitely, **CATS** has a maximum depth of route segments that it searches. This means it may not find the ending signal on a large track plan – just don’t ask to create an Extended Route between signals that are too far apart.
- **CATS** issues commands and waits for turnouts to line to the desired path before creating a Quick Route through a CP (the Quick Route will fail if the turnouts are not lined correctly).
- Because **CATS** will try to move turnouts and moving turnouts under a train can cause derailments, **CATS** will wait for the occupancy on a turnout to remain stable at unoccupied for a few seconds before issuing the command to move the points. The amount of time it waits is the Turnout Safety Delay Adjustment (See Section 3.3.9).
- While waiting for the dispatcher to mouse click on the second icon, **CATS** will ignore all other keyboard and mouse input and the display will freeze.

### 4.2.2.2 Extended Route Creation Failures

There are three reasons an Extended Route may not be created.

The first is that the second mouse click (for the ending signal) is not over an icon. The dispatcher will see the following pop up.
To continue, the dispatcher clicks on OK and tries again to click on an icon. This also provides a handy way to cancel the creation process. Click on blank screen, see the above pop up, and click on Cancel.

The second failure mechanism is that no path can be found from the starting signal to the desired second signal. In that case CATS will show

The third mechanism is that the second signal has been identified and can be reached from the first signal, but there is no path without a blockage. In that case, the below pop up will be shown.

4.2.2.3 Ending an Extended Route
An Extended Route is a chain of CP, with Quick Routes. There are several ways of removing Extended Routes. The simplest is to let a train run the chain of CP. As each Quick Route is cleared, the CP is removed from the Extended Route. Thus, in a typical scenario, the Extended Route will “roll up” from the starting signal as the train progresses. When the last Quick Route has cleared, the chain is destroyed.
The links (CP) in the chain do not have to clear in a particular order. When a Quick Route clears, it is removed from the chain. Thus, it is possible to left mouse click on a signal icon with a Quick Route in a chain. That will both clear the Quick Route and remove the CP from the chain. Once removed, the CP cannot be added back into the chain. If there is a question if a CP is in a chain, right mouse click on the signal icon. If the top menu selection says “Cancel N/X Route” then the CP is in an Extended Route. If it says “Set N/X Route”, then it is not in an Extended Route.

A final way to cut down an Extended Route is to right mouse click on an icon in the Extended Route and select “Cancel N/X Route”. If accepted, the CP associated with the icon and all CP after it in the chain will have their Quick Routes cancelled, clearing the Quick Route and removing the CP from the Extended Route.

4.2.3 Stacked Route

A Stacked Route is an Extended Route which cannot be created immediately, but is deferred until a clear path can be found. A Deferred Extended Route contains the track traversal information (all possible paths) needed to create an Extended Route; however, there is no clear path from start signal to end signal, so no Extended Route can be created. CATS maintains a stack of Deferred Extended Routes. A stack is an ordered list of Deferred Extended Routes. Oldest ones are pulled from the “top” of the stack. Newest ones are added to the bottom of the stack, so in an algorithmic sense, technically CATS maintains a “queue” (first in, first out) of Deferred Extended Routes. The term “stack” will be used here because that term is more commonly used in railroad dispatching.

Route stacking allows the dispatcher to schedule Extended Routes to be created later. This is useful for planning moves, scheduling them, and focusing attention on something else. The stacked routes will happen “automatically”.

The procedure for a dispatcher to stack a Deferred Extended Route is similar to creating an Extended Route. It is a two-step process of right mouse click on a signal icon, selecting “Stack Route”, and Accept. The icon will begin blinking (signifying that CATS will use the identified signal as the start of the Extended Route and is waiting for the dispatcher to identify the second signal to end the Extended Route). If a second icon is identified, then CATS pops up a confirmation dialog and when accepted, will add the Deferred Extended Route information to the stack of Deferred Extended Routes. The confirmation dialog looks like

![Figure 25 Route Stacked Confirmation](image)

Notice that the names of the signals which constitute the ends are listed on the pop-up.
Eventually, a Deferred Extended Route will reach the “top” of the stack (there is only one). When it does, all nodes (CP, Intermediate Signal, and turnouts) in the information will be asked to trigger the Extended Route creation mechanism when something (occupancy, Track Authority, etc.) changes. When something does change, the creation mechanism regenerates the set of metrics over the Deferred Extended Route information. If a clear path can be found, then the information is used to create an Extended Route (as above, after the second signal is confirmed). The nodes in the Deferred Extended Route information will be asked to stop telling the creation mechanism about changes and the Deferred Extended Route will be replaced on the “top” of stack by the next in line.

All of the behavior of an Extended Route will be in effect after a Deferred Extended Route evolves to an Extended Route. Here are a few additional things to note:

- Only the “top” of stack Deferred Extended Route is eligible for conversion to an Extended Route. This means that something on one end of the railroad can prevent a Deferred Extended Route involving the other end of the railroad from becoming an Extended Route. This behavior will be changed in a later release of CATS.
- The path is chosen for the Extended Route not at the time the Deferred Extended Route is stacked, but when it reaches “top” of stack. Thus, a lot of things could happen on the layout between the two times, resulting in potentially wildly different Extended Routes.
- After a clear path has been discovered, CATS will wait for Turnout Safety Delay Adjustment (See Section 3.3.9) seconds before deciding on a path, to be sure that the path is still clear.
- In general, route stacking works best with opposing traffic and fleet (Section 4.2.6) works best with following traffic.

4.2.4 Viewing and Changing Stacked Routes

The previous section discussed how to create and stack Deferred Extended Routes. They are sitting in the background, waiting for their turns. Often the “best laid plans of mice and men go astray”, so some facility is needed to adjust the contents of the route stack. That facility is invoked by the “View/Change Stacked Routes” selection on any signal icon menu. To use it, right mouse click on a signal icon (it does not have to be involved in a stacked route) and select “View/Change Stacked Routes”. If there are none, then the selection will be greyed out and nothing will happen. If there is at least one then a pop up will appear with a list of stacked routes. For example
Each line is the label for a Deferred Extended Route in the stack (same as in the confirmation pop-up). The label is formed from the name of the start signal, the word “to”, and the name of the ending signal. This encourages you to name all your signals (at least the CPs) with meaningful names. The diagram also illustrates that Deferred Extended Routes can be duplicates. The Deferred Extended Route at the top of the list is the one waiting for a clear path (the “top”).

There are three buttons on the right. They are enabled when a Deferred Extended Route is selected. “Delete Route” will remove the selected Deferred Extended Route. “Move Route UP” and “Move Route Down” are used to reorder the sequence of when Extended Routes are created. The changes will take place when the Accept button is pushed.

4.2.5 Call On (Return to Train)
A common dispatching situation is that a train will leave some cars on a track (usually the main), and move over to an adjacent track for picking up or setting out cars. When its work is done, the crew will want to leave the adjacent track, go out to the main, and back onto the
cars that it left on the main. A similar situation happens when adding helper locomotives to a train on the main.

There are several ways to handle this situation. One is for the dispatcher to grant Track Authority on the turnout(s) separating the two pieces of the train. That permits the train crew to move points and perform back and forth moves as needed. Another solution is for the dispatcher to keep control of the turnout(s) and “flag” the local crew past signals displaying Stop. Another solution is to set a Quick Route from the adjacent track out to the main and when the train has cleared the turnout(s), line them back to the main, where the rest of the train sits. The dispatcher will not be able to set a Quick Route back to the train because the cars would show the main as occupied. Thus, the dispatcher would have to “flag” the power back or use the CATS Call On (or Return to Train) feature.

The dispatcher uses Call On to temporarily override the signal safety system for one “move”. To activate it, the dispatcher lines the points, right clicks the mouse over the signal icon being overridden, selects the Call-on item, and Accepts the screen. The icon will then blink, indicating Call On is active. Simultaneously, the signal in the field will show a Restricting indication, to let the train crew know that they have permission to pass the signal, without talking to the dispatcher.

Call On is a dispatcher option only when the immediate block protected by the signal doesn’t have Track Authority or Out of Service and is not occupied. Furthermore, the next block must be occupied.

Call On is ended when the dispatcher cancels it via the signal menu or the immediate block protected by the signal is occupied.

4.2.6 Fleet

Another common dispatching situation is when two or more trains are following each other with no opposing train to break up the caravan. In CTC/DTC the dispatcher would set up a route for the first train. As it traverses the route, the route clears behind it, dropping the signals to Stop. The dispatcher then would set up new route for the second train and the process would repeat. Fleeting is a CATS function that immediately recreates a route as the signals drop. This allows trains to closely follow each other (the signal system provides the safety cushion between following trains) without more dispatcher intervention.

Routes can be “fleeted” by using the right mouse button when positioned over a signal symbol involved in a route. It is not an option if the CP does not have a route. To add fleeting to a route, the dispatcher selects the “Turn on Fleeting” option on the signal’s menu. If the CP is in an Extended Route (“Cancel N/X Route” is also an option), then fleeting will be set in the CP and all following CPs. If the CP is in only a Quick Route, then just that CP will be fleeted.

Fleeted signals are identified by the icon coloring. This is to be implemented.
Though train occupancy does not remove fleeting, manually cancelling the route will. A left mouse click on the icon cancels the fleeting for that CP (and will remove it from an Extended Route). Fleeting can also be removed through the signal menu, “Cancel Fleeting” selection. Note that “Cancel Fleeting” will also cancel an Extended Route from the icon to the end; thus, perhaps the simplest way to cancel fleeting on an Extended Route with a train is to cancel the Quick Route on each CP as the train passes it.

In general, fleeting works best for following train movements and route stacking (Section 4.2.3) works best for opposing movements.

The logic behind fleeting forms the basis of DTC. Direct Train Control can be used when there are no physical 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 created. 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.

4.2.7 Grant Track Authority
Nothing has been designed or implemented for this.

4.2.8 Take out Of Service
Nothing has been designed or implemented for this.

4.3 Track Actions (Track Menu Items)
This section explains in detail how to do things to tracks.

4.3.1 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 Lock Turnout Decoders (see Section 3.3.15) 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 one, every time a click is detected.
Code line delay (See Section 3.3.9) can affect how quickly the panel responds to the request to move the points. A larger delay (either send or receive or both) increases the “lag” time between mouse click and presentation change. The delay may be frustrating, but more accurately simulates the prototypical dispatching experience.

Running time (see Section 3.3.9) also affects how responsive CATS is to requests to move points. If a signal is running time, then the turnouts that it protects are prevented from being moved.

4.3.2 Train Detection

If the layout supports occupancy, the tracks on the dispatcher panel will change to “occupied” (default red) in response to detection messages from the layout. An occupied block in a route will turn red, but the exit arrow will remain, showing the expected direction of travel of the train. When the detection clears, the route 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 (See Section 4).

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 routes 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.

4.3.3 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 (See Section 3.3.9), 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:

![Train Manager Screen](image)

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 effect 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.

4.3.4 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 Stop and Proceed, 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.

If a block with Track Authority is also occupied, then the track will be painted with the Occupied Warrant color (See Section 3.3.1).

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,
using out of service, select a track in the desired detection block that does not have other nearby tracks.

### 4.3.5 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.

If an Out of Service block is also occupied, then the track will be painted with the Occupied Warrant color (See Section 3.3.1).

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

### 4.4 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 a while, 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.

4.5 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:

<table>
<thead>
<tr>
<th>Key sequence</th>
<th>Pop up menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control+c</td>
<td>Edit Crew menu</td>
</tr>
<tr>
<td>Control+e</td>
<td>Switch between Engine labels and train symbols</td>
</tr>
<tr>
<td>Control+j</td>
<td>Edit Jobs menu</td>
</tr>
<tr>
<td>Control+t</td>
<td>Edit Trains menu</td>
</tr>
</tbody>
</table>

“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.

CATS 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 lined and where trains are. One difference between CTC and ABS or APB is that the signal icons on the dispatcher panel reflect the aspect of the signal on the layout. With CTC, only the signal icons involved in a reserved route are non-white (or non-grey). With ABS or APB, no signals are white. The dispatcher cannot set signals with ABS and APB, but can 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.
ABS ignores direction of travel, so signals use the two buffer protection scheme both in front and behind a train. APB tries to prevent head on meets, so it monitors the sequence in which blocks are occupied. Signals between control points in the opposing direction of travel “tumble down” to Stop.

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 be useful in reconstructing 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.

<table>
<thead>
<tr>
<th>Event</th>
<th>Tag</th>
<th>Field 1</th>
<th>Field 2</th>
<th>Field 3</th>
<th>Field 4</th>
<th>Field 5</th>
<th>Field 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Record</td>
<td>Created:</td>
<td>Date</td>
<td>Version: #</td>
<td>CATS</td>
<td>Version</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Record</td>
<td>Ended:</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Record</td>
<td>Added</td>
<td>Time</td>
<td>Database Name</td>
<td>Field 1</td>
<td>Field 2</td>
<td>Field 3</td>
<td>Field 4</td>
</tr>
<tr>
<td>Action</td>
<td>Time</td>
<td>Database Name</td>
<td>Field 1</td>
<td>Field 2</td>
<td>Field 3</td>
<td>Field 4</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
<td>---------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>Change Record</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deleted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train Assign</td>
<td></td>
<td>Crew</td>
<td>running</td>
<td></td>
<td></td>
<td>Train</td>
<td></td>
</tr>
<tr>
<td>Crew Relieved</td>
<td></td>
<td>Crew</td>
<td>reassigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crew Reassigned</td>
<td></td>
<td>Crew</td>
<td>reassigned</td>
<td></td>
<td></td>
<td>Train</td>
<td></td>
</tr>
<tr>
<td>Job Assign</td>
<td></td>
<td>Crew</td>
<td>assigned to</td>
<td>Job</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train Move</td>
<td></td>
<td>Train</td>
<td>From</td>
<td>to</td>
<td>To</td>
<td>Coordinates</td>
<td></td>
</tr>
<tr>
<td>Train Removed</td>
<td></td>
<td>Train</td>
<td>Coordinates</td>
<td>Location</td>
<td>State</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train Tied Down</td>
<td></td>
<td>Train</td>
<td>Coordinates</td>
<td>Location</td>
<td>State</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train Rerun</td>
<td></td>
<td>Train</td>
<td>Coordinates</td>
<td>Location</td>
<td>State</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set OOS</td>
<td></td>
<td>add</td>
<td>Block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove OOS</td>
<td></td>
<td>remove</td>
<td>Block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set T&amp;T</td>
<td></td>
<td>add</td>
<td>Block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove T&amp;T</td>
<td></td>
<td>remove</td>
<td>Block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 train 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 Oracle and other places). This version of CATS was written with Java 1.8 compatibility. It might run under versions earlier than 1.8, but that cannot be guaranteed; however, it will run under later versions (e.g. Java 11). CATS uses components of JMRI for controlling the layout and communicating with Operations Pro; therefore, JMRI dictates what version of Java is needed to run CATS. The CATS web site lists which version of CATS to use for which version of JMRI.

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. That means layout files created with newer versions of designer will not work with older versions of designer or CATS. There could be problems with using layout files created with older versions of designer. 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.

8.1 The Disappearing Window

Many users create a shortcut and icon to the .bat or .csh file that launches CATS and put 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 Start->Run. 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”.

8.2 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.

8.3 “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.

8.4 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.

8.5 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 (profile) 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 folder. 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.

### 8.6 **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 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.

### 8.7 **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 4.3.1.

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.

### 8.8 **JMRI Version**

**CATS** requires JMRI version 4.24 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.

If **CATS** starts up far enough, the main JMRI window will pop up. It contains the JMRI version.
The Signals are not Right

CATS takes multiple steps to generate the presentation on the signals from track conditions (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 signal indication (rule).
- It passes the indication to the Signal and tells JMRI to set the signal\(^6\).
- If signal control is defined in CATS, the Signal retrieves its aspect template (defined in designer).
- The Signal finds the aspect associated with the indication in the template.
- The Signal retrieves the SignalHeads (JMRI abstraction) for each head that it drives.

\(^6\) Unless Hold is set in the designer signal template. In that case, CATS only monitors changes in the signal.
- 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 indication 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 indication (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 Mast and Signal Head tables are useful in determining what appearance **CATS** is trying to drive to the Signal. If it is as expected, then the problem is in the decoder definitions. If it is not as expected, then the problem is in the first three items.

### 8.10 Customizing Command Line Options

**CATS** looks on 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.

### 8.11 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.folder` 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

Note: CATS does not use cats.folder or user.home in searching for an initial XML file specified on the command line. Consequently, if specifying an initial layout description file on the command line, the specification must be either relative to the JMRI program folder or an absolute path.

9 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.

I have been using bitbucket for source control. The git repository is at https://bitbucket.org/Kb0oys/cats.

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.

**10 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](http://www.opsig.org/resources.shtml)
- [http://www.lundsten.dk/us_signaling/abs_st_sp/p_index.html](http://www.lundsten.dk/us_signaling/abs_st_sp/p_index.html)
- [jmri.sourceforge.net](http://jmri.sourceforge.net)
- [http://deltareum.com/signal_progressions.htm](http://deltareum.com/signal_progressions.htm)