TRS2006 Sessions & Rules Guide

Contents

Part 1 - Introduction

This document provides a comprehensive overview of how sessions and rules are used in TRS2006 to provide an interactive environment for gameplay. This includes the theory of what sessions and rules are as well as going into the actual details of how a session works.

It is assumed that the reader already has some proficiency in using Trainz. This would include knowing how to save/load maps as well as a basic understanding of Trainz items like assets and train consists. Specific experience with TRS2006 is not essential and someone who is familiar with a previous version of Trainz should also be able to follow this document.

Although no programming knowledge is required to work with sessions and rules and understand this document’s contents, a basic understanding of program design and flowcharts would be beneficial.

Part 2 - Basic Session and Rule Theory

This section of the document gives a brief theoretical overview of the components and elements that make up a Trainz session and the rules that run the session. It is strongly recommended the reader should familiarize themselves with the basics of sessions and rules presented here before getting into the following chapters.

Anatomy of a Session

A session is a Surveyor-created asset that groups together a variety of elements to create a playable environment for Driver. This includes the route, rules, consists and custom configuration properties.

Every session is attached to a specific route that it is executed on to provide the gameplay. All route objects are part of the session so scripted assets like industries can be become an interactive part of the session.

Consists placed on the route in Surveyor are part of the session, so different sessions can use the same route, each with their own consists and property settings.

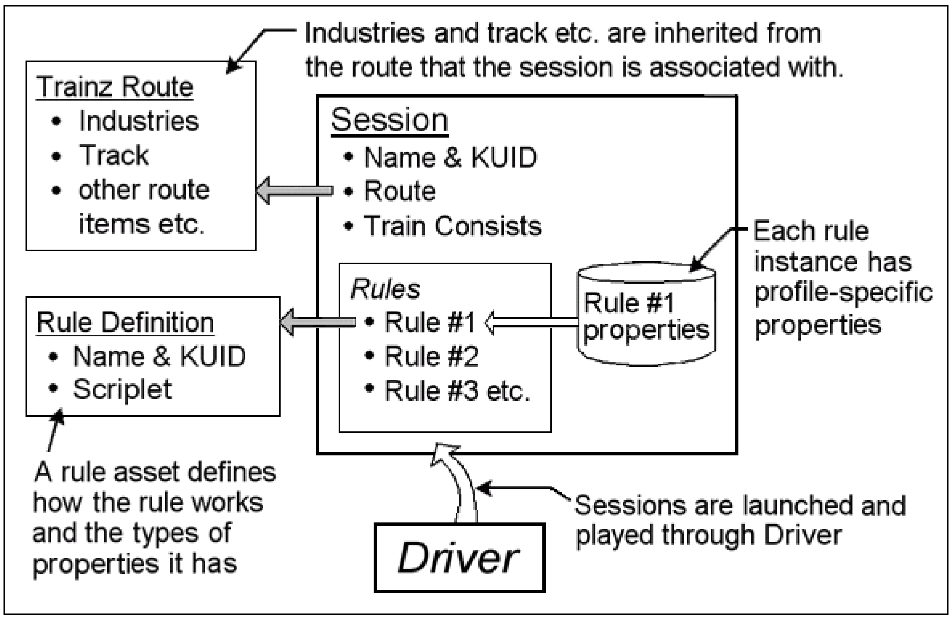

Industries and track etc. are inherited from the route that the session is associated with.
Trainz Route
• Industries
• Track
• other route items etc.
Session
• Name & KUID
• Route
• Train Consists
Rules
• Rule #1
• Rule #2
• Rule #3 etc.
Rule #1 properties
Each rule instance has profile-specific properties
Rule Definition
• Name & KUID
• Scriplet
A rule asset defines how the rule works and the types of properties it has
Driver
Sessions are launched and played through Driver

***Diagram 2.1*:** Conceptual view of a session and its main components.

Of all the types of asset items that can be used to make a session, the most important type is the rule. Rules are scripted Trainz assets used to control and dictate the behavior of the session. Trainz executes the rules to run the session.

The session has a properties database for each rule instance it uses so that rule's particular configuration for the session can be saved/loaded. Multiple instances of the same type of rule can exist in a session, each with its own unique configuration. This works in the same way that multiple instances of a specific type of locomotive can exist in the session where each locomotive can have its own name and running number.

A session is really a container module that bundles together other assets and provides a way for these asset's to have unique properties specific to that session. The session itself does not really do much – it is the rules that do the work. Hence most of this document is actually devoted to rules, not sessions.

Rule Overview

Rules are scripted assets that operate in the scope of the session to do things in the Trainz world. They can be added and removed from the session as well as configured through the **Edit Session** window in Surveyor.

What a rule does and how it can be configured is dependent on what the rule author has decided to allow. Generally rules will perform tasks like monitoring and reacting to events allowing the session's gameplay to run.

Although what a rule can do is a rather open-ended definition, there are general patterns of behavior that Trainz expects a rule to follow. At their most basic level, rules are independent modules that perform a specific task. Trainz will start/stop the execution of the rules at certain times to run the session.

As mentioned in the previous section, a rule's configuration is saved as part of the session. The configuration of a rule is (commonly referred to as the rule's properties) is set through the rule's properties window that is accessed by the **Edit** button in the **Edit Session** window.

Why Sessions and Rules?

Trainz activities were traditionally created with scenario scripts but this required programming knowledge and locked many people out of making there own playable Trainz activities.

Even though programming skills are required to create a rule asset, TRS2006 comes with more than 50 rules and from these it is quite possible to construct complex interactive sessions. The TRS2006 Tutorials were constructed entirely using the set of rules available.

Part 3 - Sessions and Rules in Action

This section of the document goes over how rules are run and behave in a session being executed by Driver. Familiarity with the previous theory section is essential before reading ahead.

Session and Rule Execution

When the session is launched into Driver, Trainz will execute the session's rules. This involves the rules being set from a paused to an un-paused state. From there, it is left to the rules to do their work.

Note: Not all rules (such as child rules for example) are necessarily started as the session launches, but for the sake of simplicity, you can safely assume for now that all rules are un-paused and executed when the session starts.

Generally speaking, once a rule's job is complete, it will set itself to a 'Complete' state. Usually once a rule is complete, it will be sitting there idle as it has done its job with nothing further to do.

Before the rules are un-paused and started however, they need their settings to be initialized, so Trainz will initialize all of the rules with their respective session-specific properties.

Note: The initialization process of rules from the session's database also applies to all child rules, even though they are not executed as the session starts.

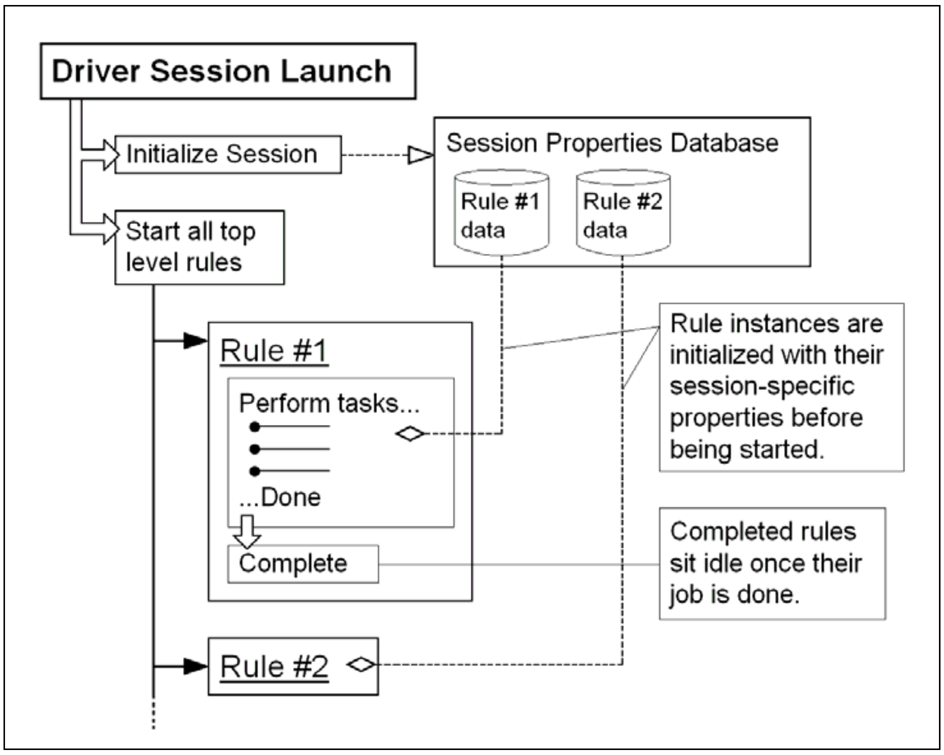

Driver Session Launch
Initialize Session
Session Properties Database
Rule #1 data
Rule #2 data
Start all top level rules
Rule #1
Perform tasks...
...Done
Complete
Rule instances are initialized with their session-specific properties before being started.
Completed rules sit idle once their job is done.
Rule #2

***Diagram 3.1*:** A Driver session being launched with a rule completing itself after being started.

As *Diagram 3.1* shows, a rule will perform its tasks once started and when finished, complete itself.

When the session is launched with multiple rules to start, it won't execute them in a sequential order one after the other. Instead, all rules will be started almost simultaneously. This means that in the diagram above, Rule #1 and Rule #2 will both be running and doing their respective tasks when the session starts.

A simultaneous start of rules can cause problems if for example Rule #2 is dependent on Rule #1 having done a certain task. It is most likely that Rule #2 will be running before Rule #1 has a chance to finish and the possibility for Rule #2 to do something before Trainz is ready or be stuck in a perpetual waiting cycle exists.

There is also the possibility that Rule #1 completes almost instantaneously such that by the time Rule #2 is started, everything it depends on Rule #1 for is done, so Rule #2 will work as planned. However relying on Trainz to consistently behave like that for a session to function properly is not recommended.

There are ways of dealing and working around such a potential problems that will be examined later on in this document.

So far, the use of top-level rules at startup has been looked at and it introduces the reader to how a session runs in simple terms. However executing all rules on startup does have limitations and to create stable and interactive sessions, more sophisticated rule arrangements are required which is what the next section examines with the use of child rules.

Rule Hierarchy and Child Rules

As mentioned in the previous section, just having a collection of rules all being executed as the session launches makes for a limited session. It may even seem that sessions are not really capable of much. It is only when the child rule features are used that the full capabilities of sessions can be achieved.

A child rule is still a regular normal rule. Any existing rule asset that is well behaved can be used as a child rule. What makes a child rule different from the top-level rules executed on startup is that the child rule is not started until its parent rule explicitly starts it.

Note: To make a rule into a child rule, it is maneuvered into a position indented to the right and below the desired parent rule. This is examined in the next section that demonstrates how to make an actual session.

Diagram 3.2 shows how the child rule relates to its parent rule and the session as well.

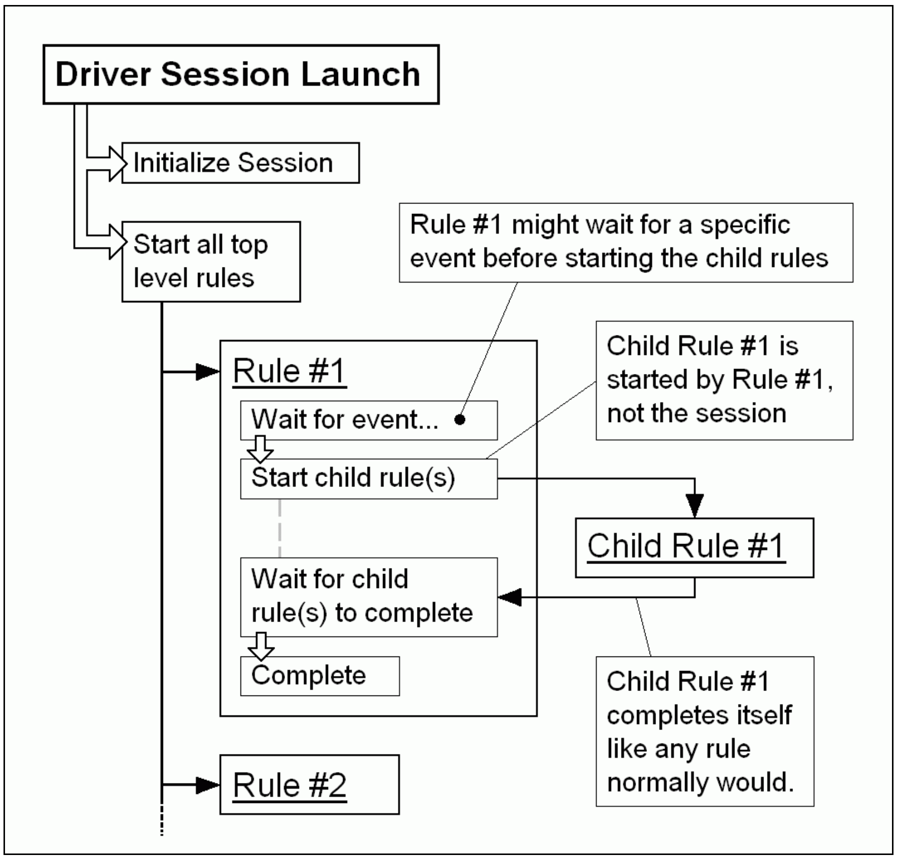

Driver Session Launch
Initialize Session
Start all top level rules
Rule #1 might wait for a specific event before starting the child rules
Rule #1
Wait for event...
Start child rule(s)
Wait for child rule(s) to complete
Complete
Child Rule #1 is started by Rule #1, not the session
Child Rule #1
Child Rule #1 completes itself like any rule normally would.
Rule #2

Diagram 3.2**:** A session running in Driver with a rule waiting for its child rule to complete before completing itself.

The child rule mechanism is a way of controlling when a particular rule gets executed, as it is often the case that a rule doesn't need to be executed the moment the session starts.

Although any rule can be used as a child rule, only some rules execute child rules and are suitable for being used as parent rules. The Rule Reference section in Part 7 of this document explains the capabilities of each TRS2006 rule in detail.

More Rule Behavior

There is a lot of variety and diversity in how the rules work and behave in the session environment. Some rules are meant to perform a specific task and be completed while others are more sophisticated as they can react and respond to events in the Trainz world.

Note: Although this section is intended to explain the main types of rule encountered in TRS2006 and how they work in general, there are always exceptions. These types are not mutually exclusive rule definitions and can overlap. If in doubt or curious to know how a particular rule behaves, look it up in the Rule Reference section at the end of this document.

There are four main types of rules provided in TRS2006:

- Task Rules
- Event Rules
- Check Rules
- List Rules

The behavior and properties of these different types of rules is discussed in the subsequent sections below.

Task Rules

Task rules are the simplest type of rules to understand. They usually perform their job the moment they are started and once finished, they don't wait around and will complete themselves.

Once a task rule has been complete, it will sit idle and not do anything further in the session. Unless a parent rule resets it, the task rule won't run again for the lifespan of the session.

It is most likely that a task rule is run during the session startup or executed by a parent rule in response to an event the parent rule is set to monitor.

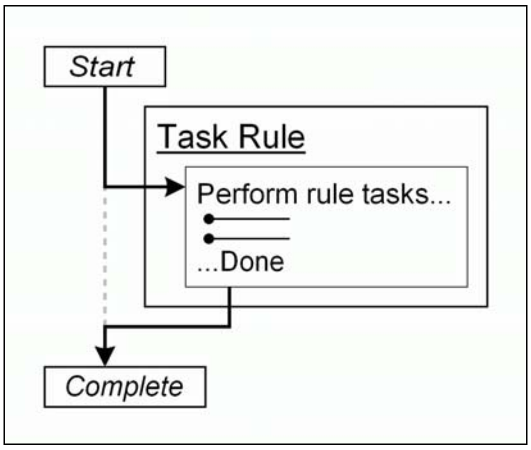


***Diagram 3.3*:** Flowchart view of how a task rule would generally behave.

Event Rules

Event rules are used to wait for a certain event to occur in the Trainz world and react to that event in some way. The event could happen quite soon after the event rule is started or the event rule could be waiting for many minutes.

Note: Regardless of how long it has been waiting, if the actual event happened before the event rule was even started, the event rule may never complete and do its job.

Usually the event rule will not do any direct action itself once the event has occurred, instead any child rules nested below it will be executed. Because any rule can be nested as a child rule, the event rule can be used to start off whatever rule(s) the session author desires. Even new rules written by 3rd party content creators after the release of TRS2006 can be easily linked into a session like this.

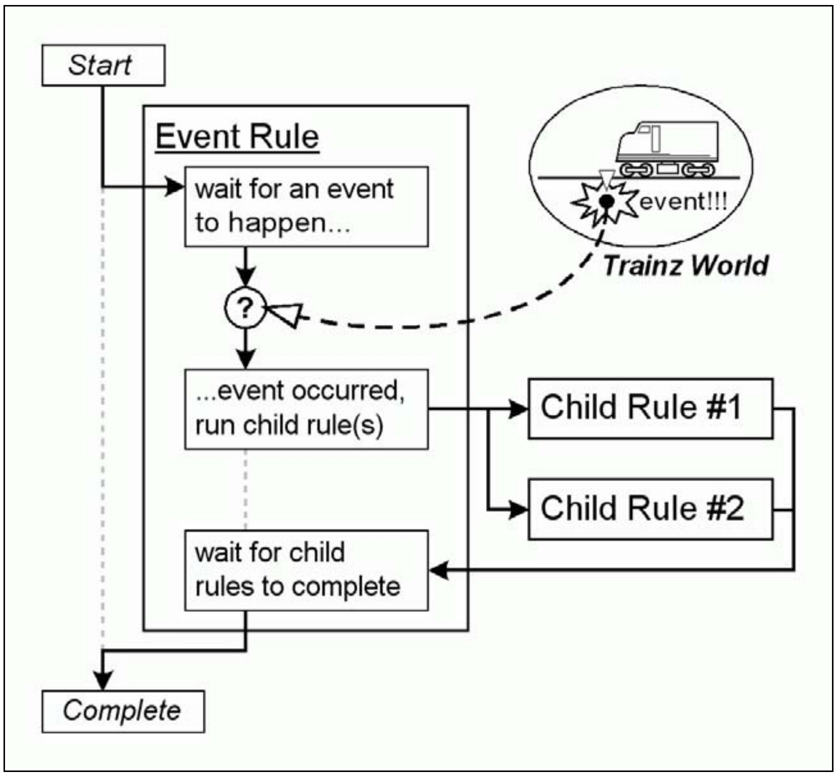

Start
Event Rule
wait for an event to happen...
event!!!
Trainz World
?
...event occurred, run child rule(s)
Child Rule #1
Child Rule #2
wait for child rules to complete
Complete

***Diagram 3.4*:** Flowchart view of how an event rule would generally behave.

Creating an event rule to wait for a certain event and than react to that event with a particular reaction is quite feasible, but there are some very good design reasons why this is not how the TRS2006 rules are done. Mainly because a generic event rule that executes whatever child rules there are provides much more flexibility for everyone concerned than a 'hard coded' event rule could.

An example of some event rules in TRS2006 would be either *Trigger Check* or *Wait for Driver On/Off Train*.

Note: There are some rules that although they are not event-type rules, they do support the running of child rules. These are the check and list rules that are examined in the next sections.

Check Rules

Check rules are used to verify something in the Trainz world and if the rule's checking is satisfied, it will execute the child rules, otherwise it completes straight away without having done anything.

Although the check rule performs a similar role to that of the event rule, the key difference is that a check rule doesn't wait for anything. If the condition of the check rule is not satisfied, then the rule completes straight away. This behavior is similar to the task rule because if the check is to be done again, then the check rule needs to be reset and started again.

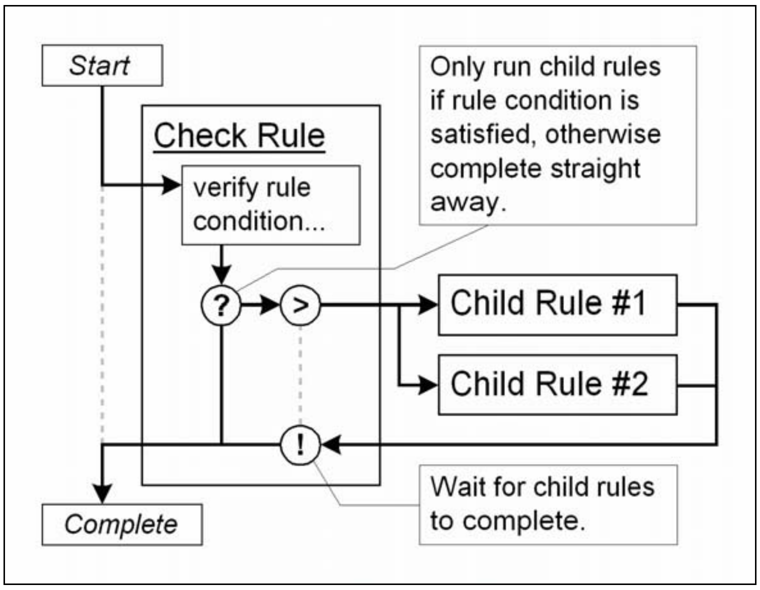

Start
Check Rule
verify rule condition...
Only run child rules if rule condition is satisfied, otherwise complete straight away.
?
>
Child Rule #1
Child Rule #2
!
Wait for child rules to complete.
Complete

Diagram 3.5: Flowchart showing how a check rule can be used.

An example of a check rule is *Variable Check*.

List Rules
As noted previously, not all rules necessarily support child rules and those rules that do support child rules may not execute the child rules as in some cases, just executing all of the child rules simultaneously is not suitable.

Sometimes executing all child rules simultaneously is not appropriate and a more controlled running of the child rules is needed. This is where the list rules come in. List rules are rules designed to specifically to execute and manage the running of child rules in a certain way. For example, a list rule might run the child rules in order or only execute one particular child rule, not all of them.

Like other rules, list rules can be used as child rules so in the case of an event rule simply executing all of its child rules, the list rule is a child rule of the event rule and it manages the execution of its child rules once the event rule has started it. With this arrangement, there are now two levels of child rules and the hierarchy of rules begins to emerge.

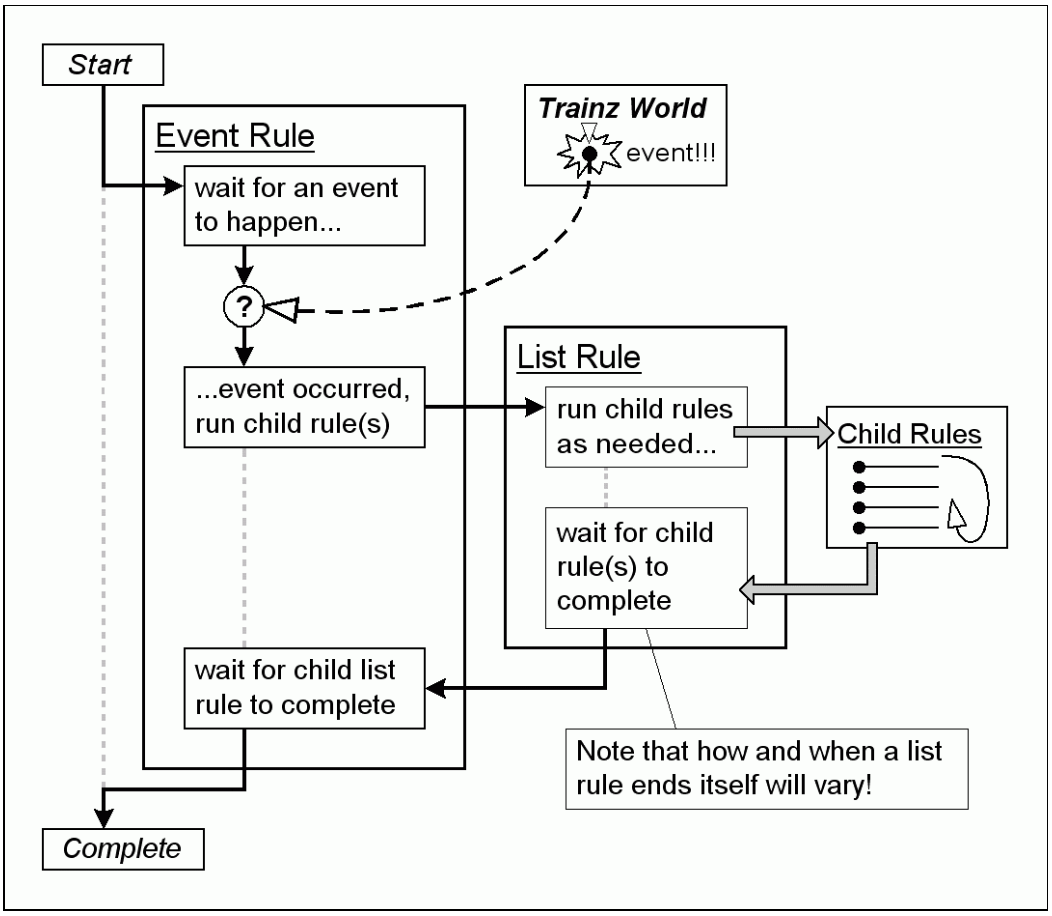

Start
Event Rule
wait for an event to happen...
Trainz World
event!!!
?
...event occurred, run child rule(s)
List Rule
run child rules as needed...
Child Rules
wait for child rule(s) to complete
wait for child list rule to complete
Note that how and when a list rule ends itself will vary!
Complete

***Diagram 3.6*: ** Flowchart showing how a list rule can be used.

List rules provided with TRS2006 are:

- Ordered List
- Progressive List
- Random List
- Reset List
- Simultaneous List

Details on each of these rules can be found in the **Rule Reference** section at the end of this document.

Part 4 – Example 1: A Very Simple Session

This first tutorial is a very simple session that introduces the use of nested child rules and the use of a rule to respond to an event. Although it is a very simple session that may not be very interesting to play, all of the lessons learnt creating it should be helpful to the reader wishing to get into session creation. Further tutorials will examine how to make more complex and playable sessions.

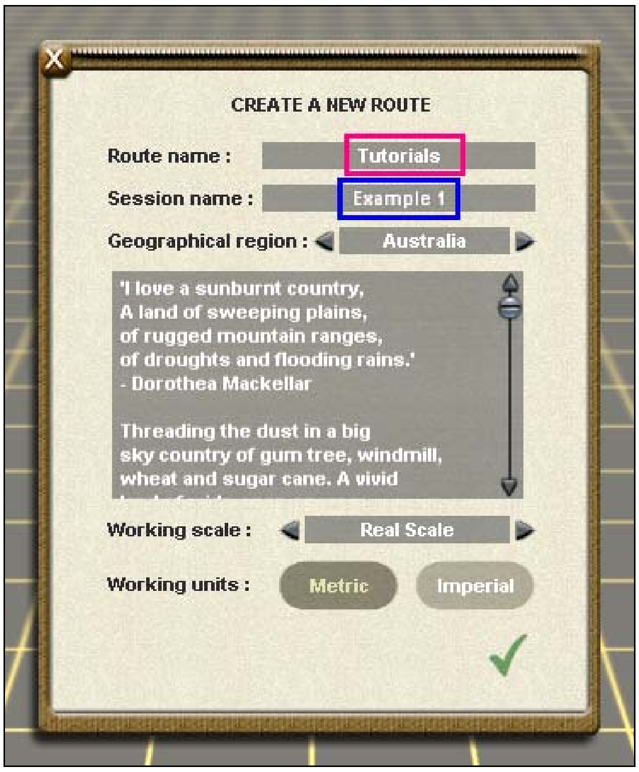
The actual session being created will display a HTML page and play an audio file when a locomotive enters a trigger.

Creating the Route

As a session always needs a route to run on, getting the route ready is always the first part of creating any session. In this case, a route is being created from scratch for the session.

This example session is based around a route that has a single piece of track that is approximately half a baseboard in length and has one trigger and one locomotive placed on it.

To get started, go to the Surveyor map selection menu by clicking on the "**Surveyor**" option on the Trainz main menu entry screen. Once the Surveyor menu screen has opened, click on the "**Create New**" option to get started on a new route.


CREATE A NEW ROUTE

Route name : Tutorials

Session name : Example 1

Geographical region : Australia

'I love a sunburnt country,
A land of sweeping plains,
of rugged mountain ranges,
of droughts and flooding rains.'
- Dorothea Mackellar

Threading the dust in a big
sky country of gum tree, windmill,
wheat and sugar cane. A vivid

Working scale : Real Scale

Working units : Metric Imperial

Screenshot 4.1: The Create a New Route window in Surveyor.

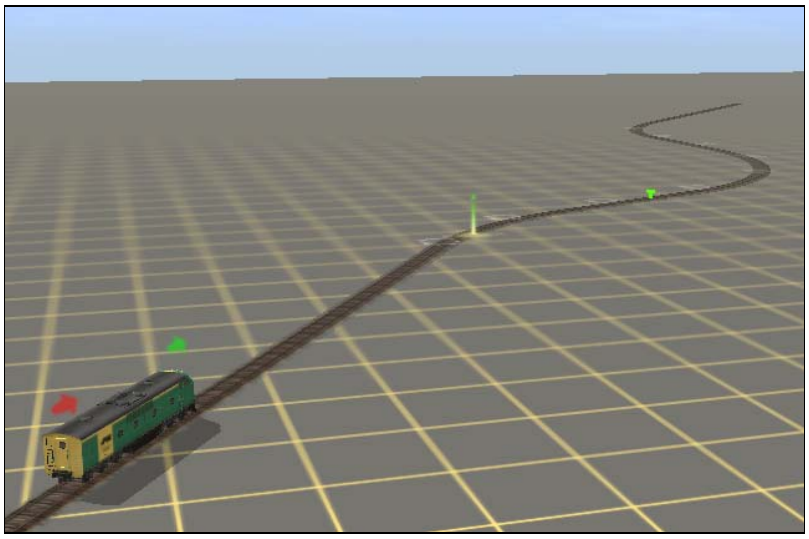
When Surveyor is launched to create a new empty map, the **Create a New Route** window as shown in *Screenshot 4.1* will appear. As the default session and route names offered in this window are not very descriptive, set the route name to "**Tutorials**" with a session name of "**Example 1**".

Once the names for the route and session have been entered, click on the check icon in the lower right corner of the window or press enter to close the window. This will result in two new assets being created, the route and the session.

Remember that a route and a session are separate assets with their own names and unique KUID, although there is no rule against a route and session sharing the same name.

Once in Surveyor on the empty route, create a length of track with a locomotive at one end and a trigger somewhere in the middle. Make sure that the locomotive and trigger are not within 3-4 grid squares of each other. The name of the trigger does not really matter so leave it at whatever default name Trainz gave it.

As this document is about sessions, not route creation it is assumed the reader knows how to do a basic route. *Screenshot 4.2* shows the type of basic route this tutorial session requires.

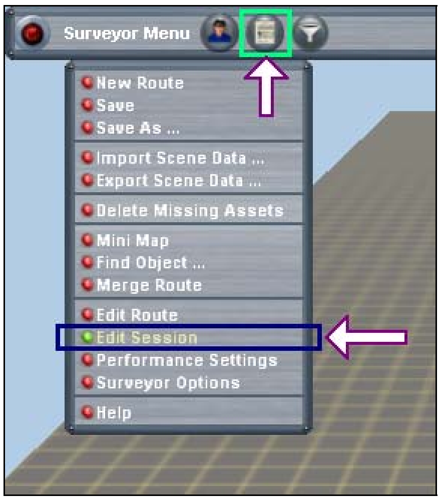


***Screenshot 4.2*:** A simple route with track, trigger and a locomotive as needed for this session.

Now that the route is created and ready, it is time to move on to what this tutorial is really about and that is sessions.

Creating the Session

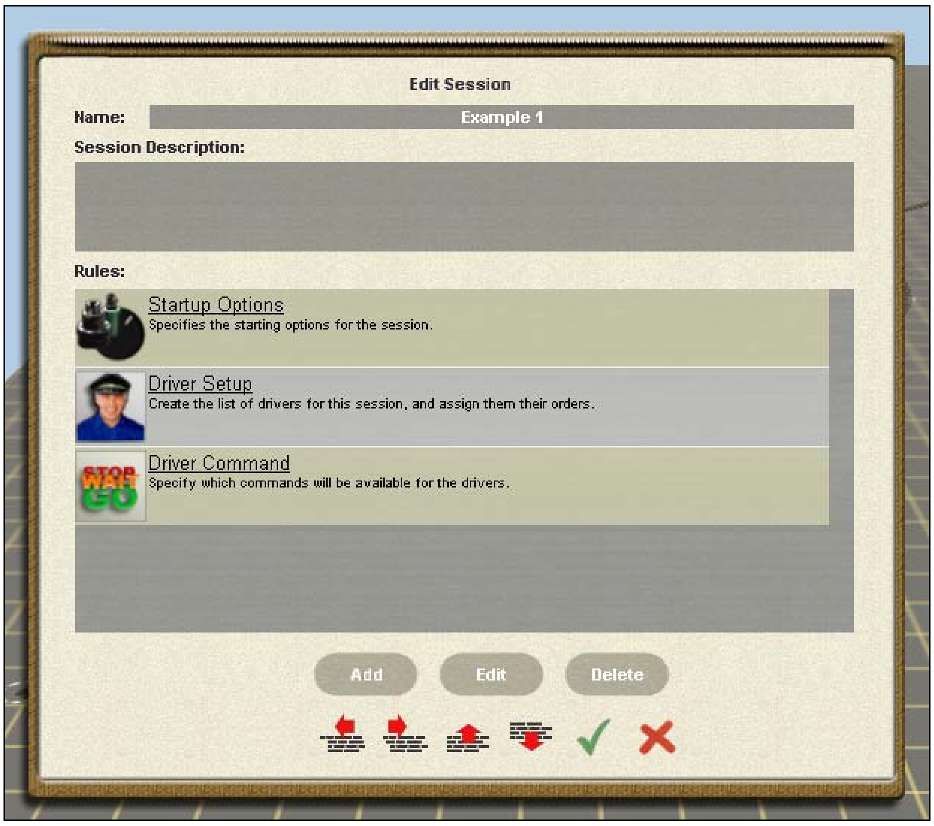
Sessions in Surveyor are created and edited in the "**Edit Session**" window. To open this window either select "**Edit Session**" on the **Surveyor Menu** or click on the clipboard icon that can be found on the left side of the Surveyor menu bar as shown in *Screenshot 4.3*.



Screenshot 4.3: Two different ways to open the "Edit Session" window.

The "**Edit Session**" window will now appear. As this is a new session, the only rules present are the basic default rules place in all new sessions. The session name entered previously in the create route window will appear at the top in the "**Name**" field.

Screenshot 4.4 shows what the "**Edit Session**" window should look like when opened for this new session.


Edit Session

Name: Example 1

Session Description:

Rules:

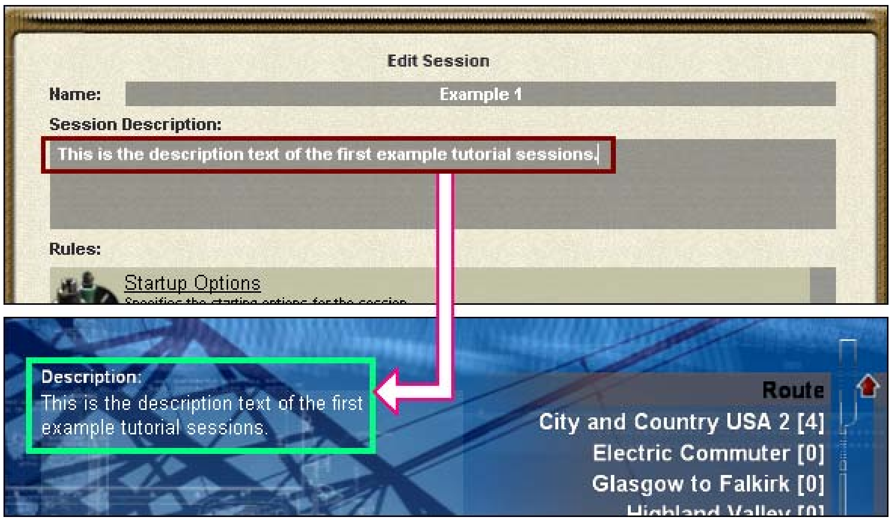
Startup Options
Specifies the starting options for the session.

Driver Setup
Create the list of drivers for this session, and assign them their orders.

Driver Command
Specify which commands will be available for the drivers.

Screenshot 4.4: The **Edit Session** window with a fresh new default session.

Before getting into crafting the session, click on the gray box below the text "Session Description" and enter a simple description for the session. This description will appear on the main menu screens for both Surveyor and Driver when the session is highlighted. *Screenshot 4.5* illustrates this.

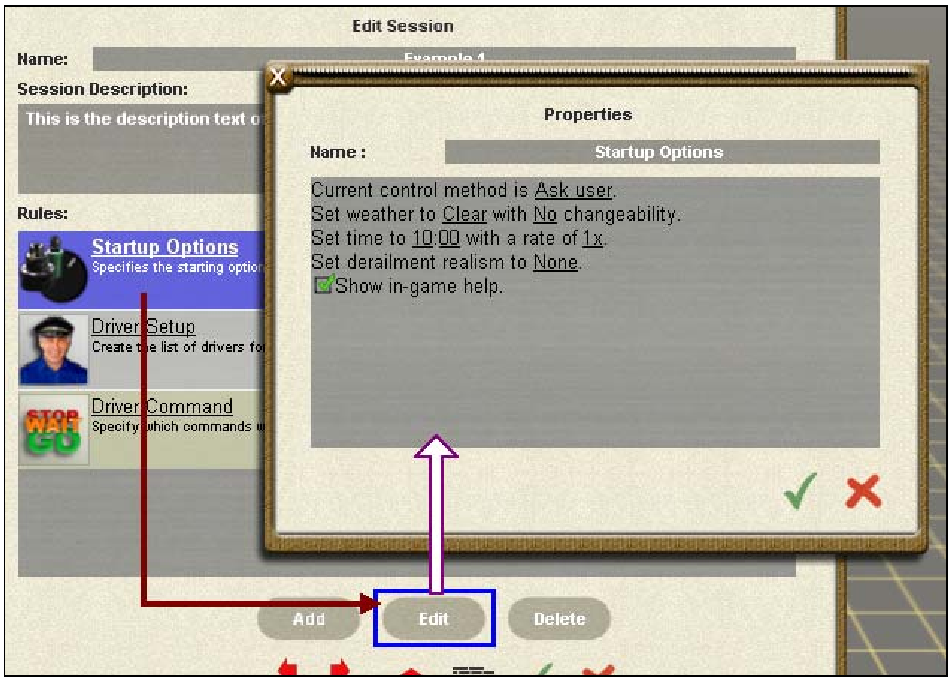

Edit Session
Name:
Example 1
Session Description:
This is the description text of the first example tutorial sessions.
Rules:
Startup Options
Description:
This is the description text of the first example tutorial sessions.
Route
City and Country USA 2 [4]
Electric Commuter [0]
Glasgow to Falkirk [0]

Screenshot 4.5: How the session's description text is used in Trainz.

Editing the Startup Options Rule

Before getting into adding extra rules, a minor change is going to be made to the *Startup Options* rule. This particular rule is used to set a few various parameters of the Trainz environment as the session is started in Driver. It is a simple task rule that sets the parameters and completes once done.

To edit, click on the text/icon of the Startup Options rule at the top of the rules list. Once selected, it should be highlighted in blue. Then click on the "**Edit**" button as show in *Screenshot 4.6*.



Current control method is Ask user.
Set weather to Clear with No changeability.
Set time to 10:00 with a rate of 1x.
Set derailment realism to None.
✓Show in-game help.

Screenshot 4.6: The *Startup Options* rule properties window opened by clicking on the **Edit** button.

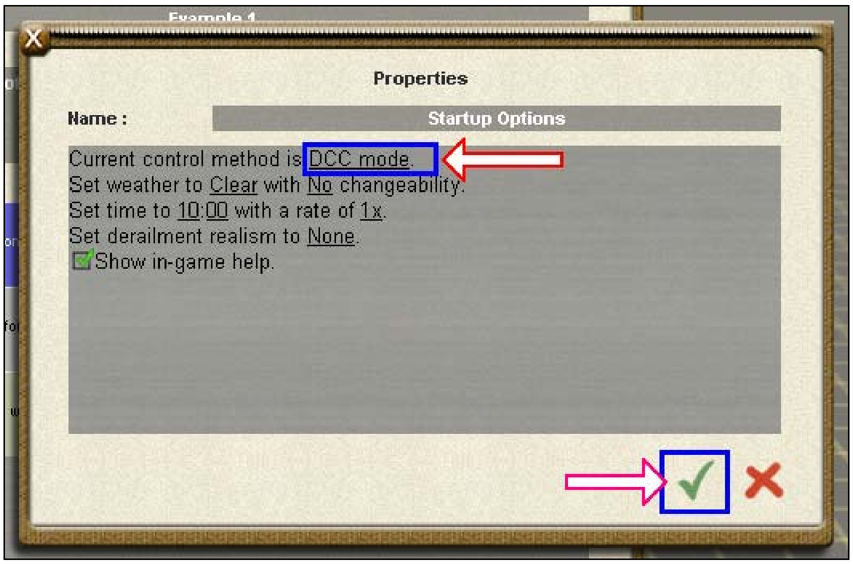
Clicking on the **Edit** button will open the properties window for the currently selected rule. The contents of a properties window and what it offers depends on what the rule provides when Trainz asks it to provide a properties window.

The settings of the rule made in the properties window are specific to that rule instance in the session and are used by Trainz to configure the rule as the session is loaded either in Surveyor or Driver.

Note: Some rules may have default properties already set while others might require further configuration to be useful.

In the properties window for *Startup Options*, click on the "Ask user" link at the end of the top line. It will than change to "Cabin mode". Click again on this link to get it to change to "DCC mode".

This property setting will be used by the rule to tell *Trainz* what control mode is going to be used for the session. As *Startup Options* is aligned to the left of the rules list and not a child rule, it will be executed the moment the session is started in Driver and in this case set the control mode to DCC before the user gets a chance to do anything.


Properties

Name : Startup Options

Current control method is DCC mode.
Set weather to Clear with No changeability.
Set time to 10:00 with a rate of 1x.
Set derailment realism to None.
✓ Show in-game help.

***Screenshot 4.7*:** Setting the *Startup Options* rule to set the session to DCC mode.

Once the *Startup Options* rule has been set to use DCC mode, click on the green check-mark in the bottom right corner of the properties window to close it and have the rule's properties updated. *Screenshot 4.7* shows how the properties window should look once set.

Clicking on the '**X**' button next to the check-mark will cancel any changes made and go back to the **Edit Session** window with the rule being left in the same state it was when the **Edit** button was initially clicked to open the properties window.

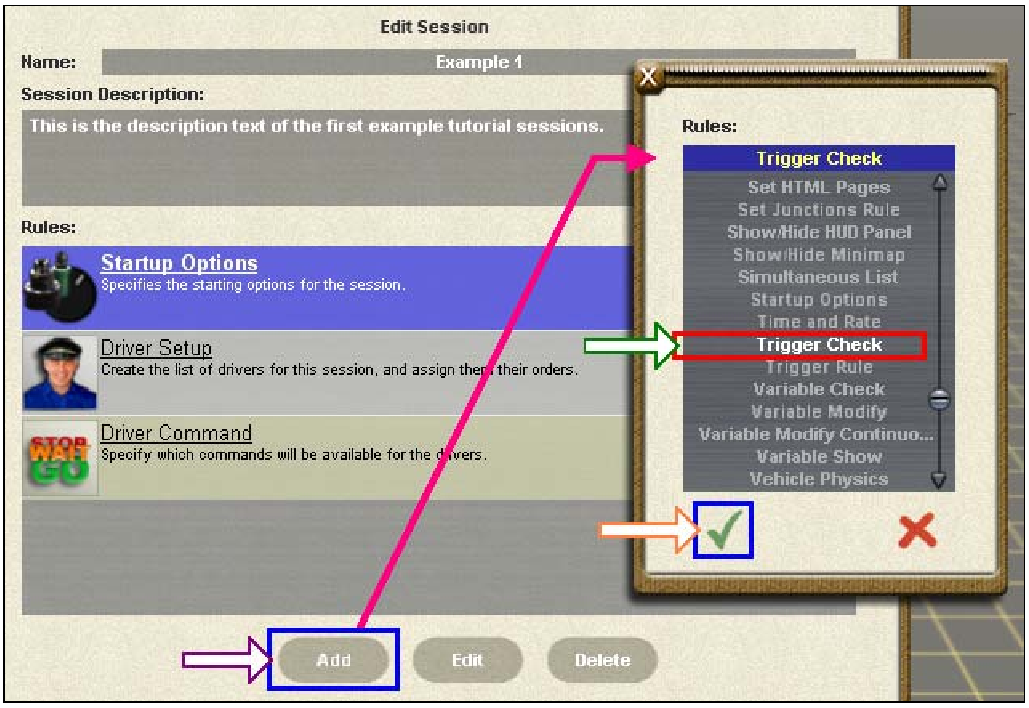
Note: Even though closing the properties window of a rule does mean the rule's properties are updated as set, these settings have not been saved to disk. To save changes made, the Edit Session window needs to be closed (again with the check-mark) so the session can be saved from the Surveyor Main menu.

Adding and Configuring the Trigger Check Rule

Now that the simple setup of the session has been taken care, some actual session creation can begin.

As this session's goal is to do something when the locomotive enters a trigger, the *Trigger Check* rule needs to be added to watch out for that event. *Trigger Check* is used as it has the ability to watch triggers and execute child rules in response to a trigger being entered.

To add the *Trigger Check* rule to this session, click on the "Add" button at the bottom of the *Edit Session* window. A pop-up list of all available rule assets will appear. Scroll down the list and select "**Trigger Check**" as highlighted in *Screenshot 4.8*.


Edit Session

Name: Example 1

Session Description:

This is the description text of the first example tutorial sessions.

Rules:

Startup Options
Specifies the starting options for the session.

Driver Setup
Create the list of drivers for this session, and assign them their orders.

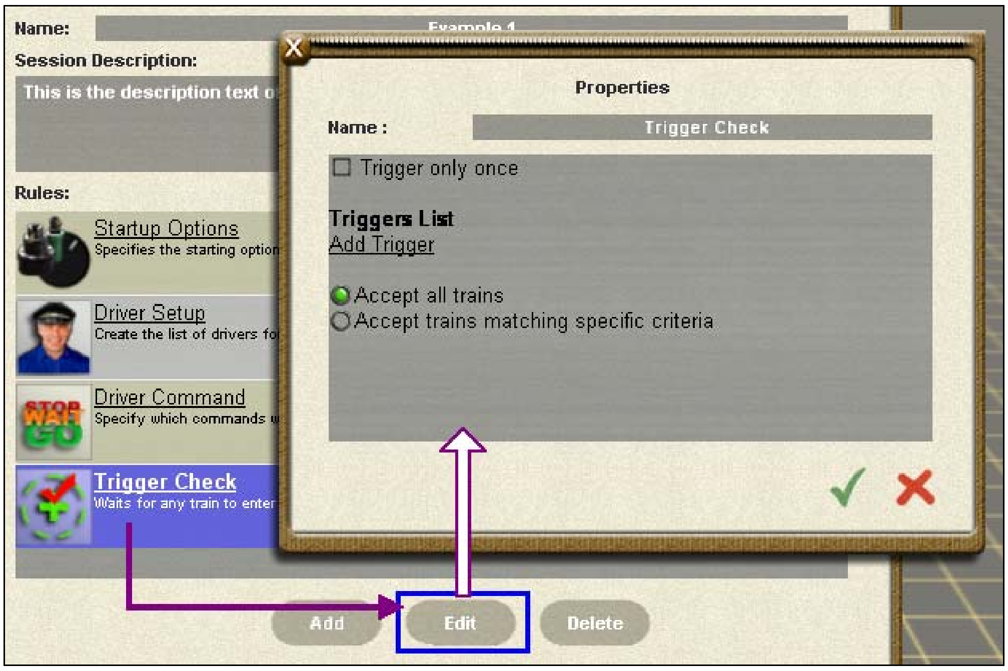
Driver Command
Specify which commands will be available for the drivers.

Screenshot 4.8: Adding the *Trigger Check* rule to the session.

Once "**Trigger Check**" has been selected on the list, click on the green checkmark at the bottom left corner of the window to close it and add a instance of *Trigger Check* to this session.

The newly added *Trigger Check* rule will appear at the bottom of the rules list aligned to the left of the window. It will also be highlighted in blue because the newly added rule ends up being the currently selected rule.

As the *Trigger Check* rule is already selected, just click on the **Edit** button to open its properties window as shown in *Screenshot 4.9*.



Name: Example 1

Session Description: This is the description text o

Rules:

Startup Options — Specifies the starting optio

Driver Setup — Create the list of drivers fo

Driver Command — Specify which commands u

Trigger Check — Waits for any train to enter

Properties

Name : Trigger Check

☐ Trigger only once

Triggers List

Add Trigger

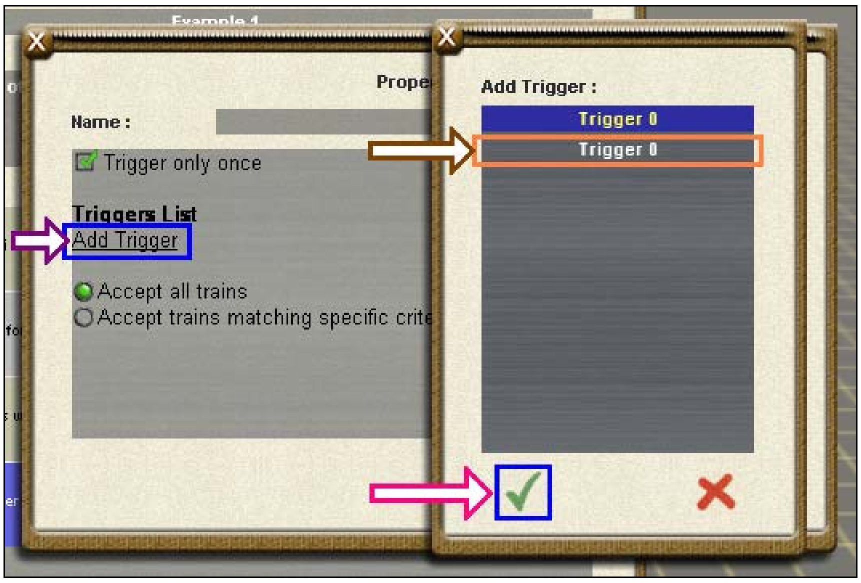
● Accept all trains

○ Accept trains matching specific criteria

Screenshot 4.9: Editing the properties of the newly added *Trigger Check* rule.

To add a trigger to the rule’s observation list, click on the "Add Trigger" links. A pop-up list of all available triggers on the route will appear. As this simple layout only has one trigger, there will only be one choice so the trigger will already be selected.

Click on the green checkmark at the bottom left corner of the pop-up window to add the trigger to the rule’s observation list as shown in *Screenshot 4.10*. Alternatively, just press the "Enter" key that will have exactly the same effect as clicking on the checkmark does.



Name :

☑ Trigger only once

Triggers List

Add Trigger

● Accept all trains

○ Accept trains matching specific crite

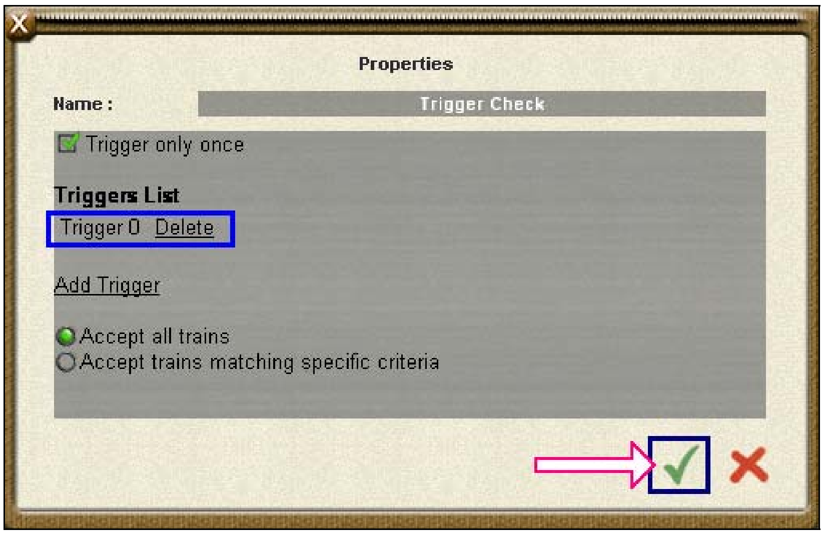
Add Trigger :

Trigger 0

Trigger 0

Screenshot 4.10: Adding a route trigger to the rule’s observation list.

Once the trigger selection window is closed, the *Trigger Check* properties window will now have the selected trigger in its list as *Screenshot 4.11* shows.



Properties

Name : Trigger Check

✓ Trigger only once

Triggers List

Trigger 0 Delete

Add Trigger

● Accept all trains

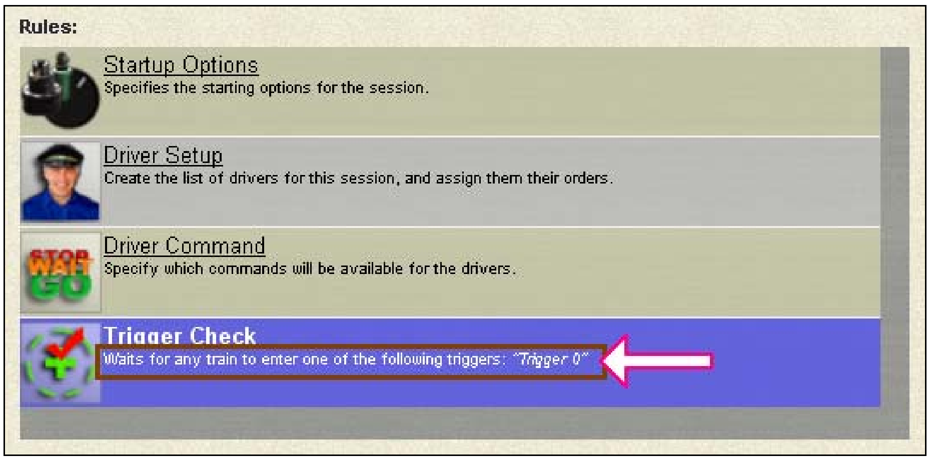
○ Accept trains matching specific criteria

***Screenshot 4.11*:** How the *Trigger Check* rule properties window should look once configured for this session.

As well as having the trigger named added, click on the checkbox next to the "**Trigger only once**" text in the tope left corner of the properties window. By selecting this option, *Trigger Check* will only wait for a trigger event to happen once. After that has happened, it will have completed itself and no longer be active.

Leave the "**Accept all trains**" setting alone for now. As there is only one locomotive in this session, there is no need for filtering train consists.

Once finished configuring *Trigger Check*, close it with the property settings saved by either clicking on the green checkmark or pressing Enter.



Rules:

Startup Options
Specifies the starting options for the session.

Driver Setup
Create the list of drivers for this session, and assign them their orders.

Driver Command
Specify which commands will be available for the drivers.

Trigger Check
Waits for any train to enter one of the following triggers: "Trigger 0"

***Screenshot 4.12*:** The session's rules so far just after *Trigger Check* has been added and configured.

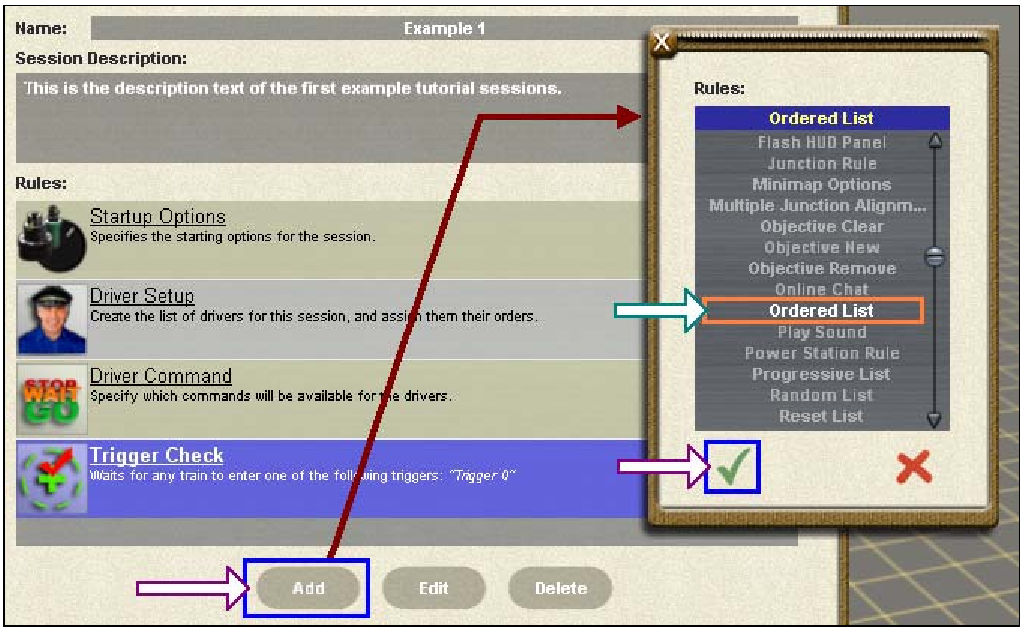
Back in the Edit Session window, note how the descriptive text for *Trigger Check* provides some minor details on its configuration as highlighted in *Screenshot 4.12*.

Adding and Indenting the Ordered List Rule

The next stage in creating this session is to add an *Ordered List* rule and indent just below the *Trigger Check* rule such that it will be a child rule.

The reason for using this rule is that is can execute child rules in order one after the other – something *Trigger Check* can't as it starts all child rules simultaneously. By using *Ordered List* as a child rule, *Trigger Check* can still be used to have other rules executed in a controlled manner.

To get started, find the Ordered List rule in the Add pop-up window and add it as demonstrated in *Screenshot 4.13*.


Name: Example 1

Session Description:

This is the description text of the first example tutorial sessions.

Rules:

Startup Options
Specifies the starting options for the session.

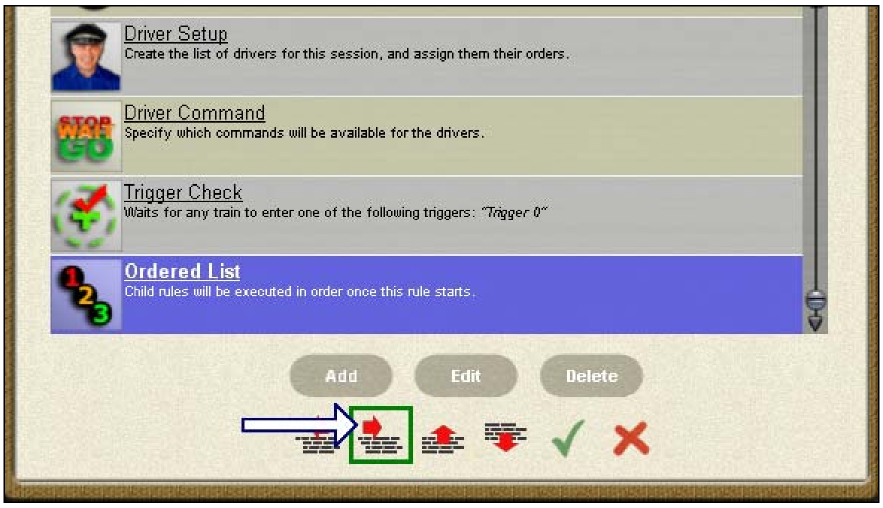
Driver Setup
Create the list of drivers for this session, and assign them their orders.

Driver Command
Specify which commands will be available for the drivers.

Trigger Check
Waits for any train to enter one of the following triggers: "Trigger 0"

Screenshot 4.13: Adding the *Ordered List* rule to the session.

Once *Ordered List* has been added and is sitting highlighted below *Trigger Check*, click on the indent left button once as highlighted in *Screenshot 4.14*. This will result in *Ordered List* moving across slightly so it is offset one position to the right from the edge of the **Edit Session** window.


Driver Setup
Create the list of drivers for this session, and assign them their orders.

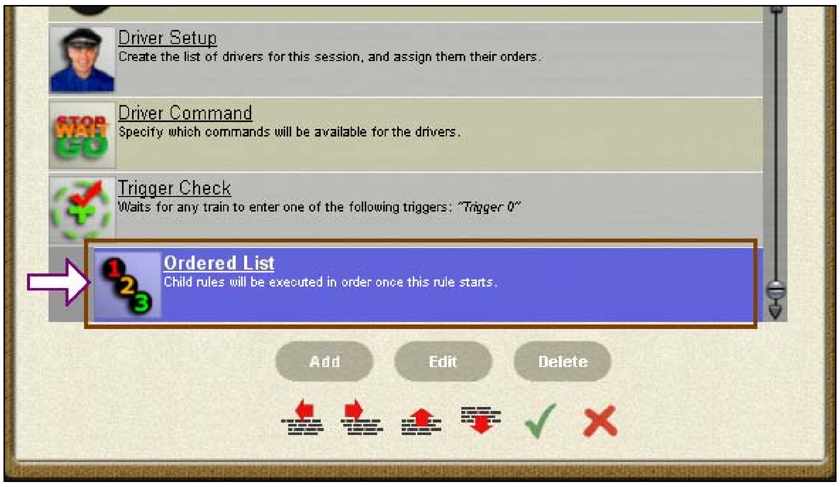
Driver Command
Specify which commands will be available for the drivers.

Trigger Check
Waits for any train to enter one of the following triggers: "Trigger 0"

Ordered List
Child rules will be executed in order once this rule starts.

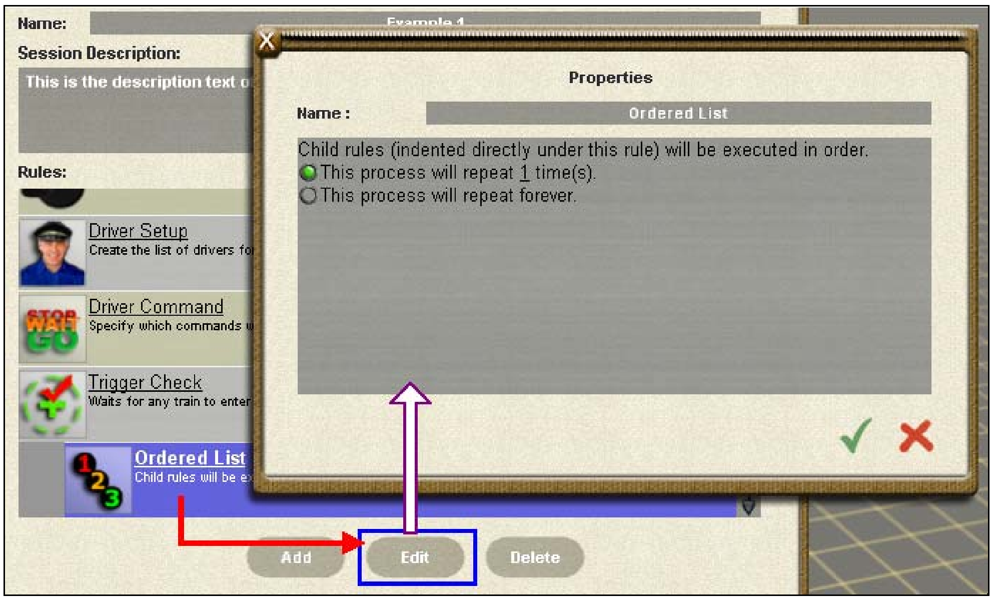
Screenshot 4.14: Session rules just after *Ordered List* has been added.

After indenting *Ordered List*, the rules window should have the same arrangement depicted in *Screenshot 4.15*.


Driver Setup
Create the list of drivers for this session, and assign them their orders.
Driver Command
Specify which commands will be available for the drivers.
Trigger Check
Waits for any train to enter one of the following triggers: "Trigger 0"
Ordered List
Child rules will be executed in order once this rule starts.

Screenshot 4.15: Session rules with *Ordered List* now a child rule of *Trigger Check*.

Although the default configuration for *Ordered List* suits the requirements of this basic session exactly, open the properties window anyway (as shown in *Screenshot 4.16*) to see what it has to offer.


Name: Example 1
Session Description:
This is the description text o
Rules:
Driver Setup
Create the list of drivers fo
Driver Command
Specify which commands w
Trigger Check
Waits for any train to enter
Ordered List
Child rules will be ex

Properties
Name : Ordered List
Child rules (indented directly under this rule) will be executed in order.
This process will repeat 1 time(s).
This process will repeat forever.

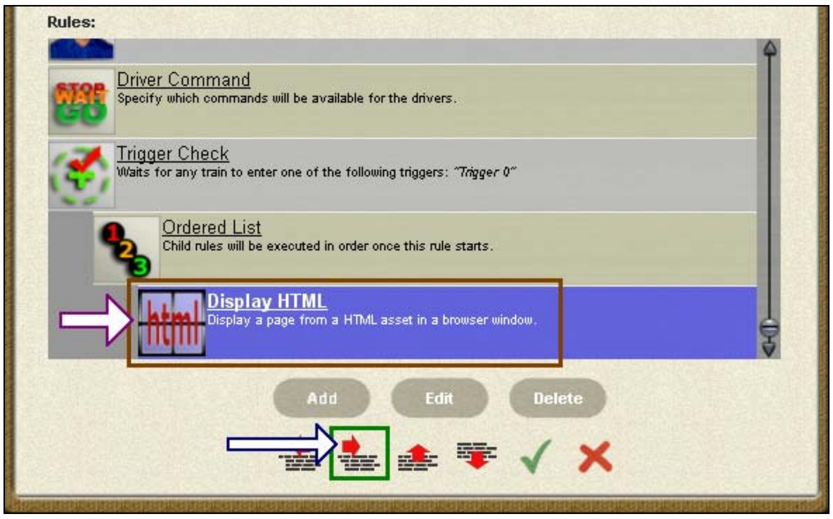
Screenshot 4.16: Default properties of the *Ordered List* rule.

As shown above in *Screenshot 4.16*, the *Ordered List* rule by default is set to repeat its process of executing child rules in order once. It can be set to run its child rules in order multiple times or indefinitely.

Adding and Configuring Display HTML

The first task that needs to be done once the trigger event occurs is to display a page on the screen. For this, the Display HTML rule will be used. It is a simple rule that displays a HTML page in a new browser window and completes once the user has closed that browser window.

So that *Display HTML* is run once the trigger event happens, add the rule to the session and indent it twice so it appear as a child rule of Ordered List as seen in *Screenshot 4.17*.



Rules:

Driver Command
Specify which commands will be available for the drivers.

Trigger Check
Waits for any train to enter one of the following triggers: "Trigger 0"

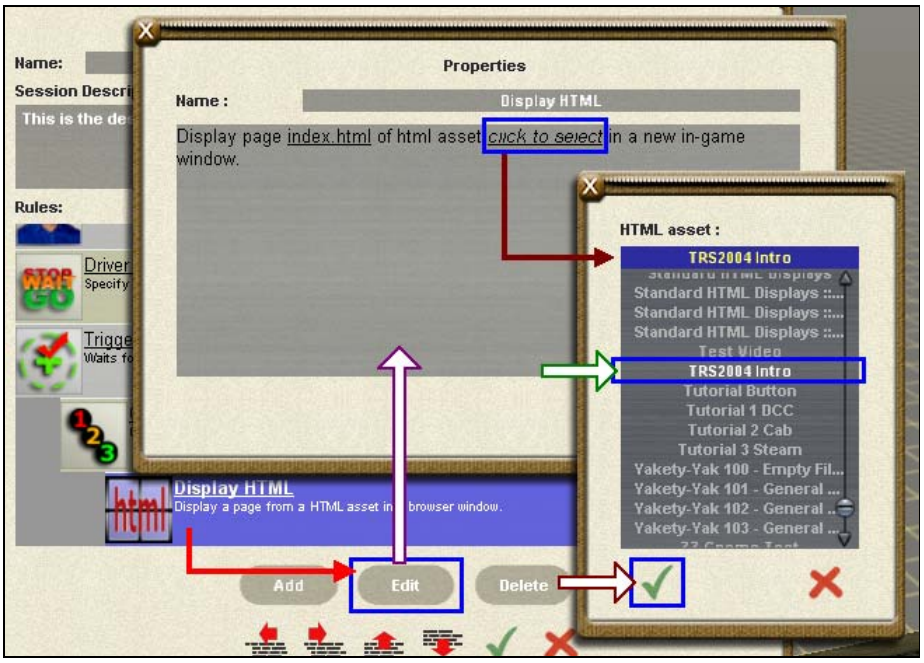
Ordered List
Child rules will be executed in order once this rule starts.

Display HTML
Display a page from a HTML asset in a browser window.

Screenshot 4.17: The *Display HTML* rule positioned to be a child rule of *Ordered List*.

Little is required to configure the *Display HTML* rule. All it needs is the name of a HTML file and the HTML asset where it can find that file. For this session **index.html** from the HTML asset **TRS2004 Intro** will be used.

Open the properties window for *Display HTML* and click on the "click to select" link. This link opens up a list of all available HTML assets. Select the "**TRS2004 Intro**" item as shown in *Screenshot 4.18*.



Properties

Name : Display HTML

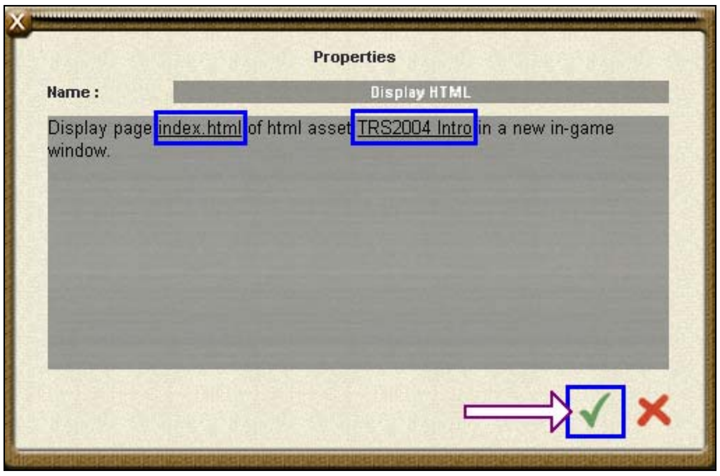
Display page index.html of html asset click to select in a new in-game window.

HTML asset : TRS2004 Intro

Standard HTML Displays
Standard HTML Displays ::...
Standard HTML Displays ::...
Standard HTML Displays ::...
Test Video
TRS2004 Intro
Tutorial Button
Tutorial 1 DCC
Tutorial 2 Cab
Tutorial 3 Steam
Yakety-Yak 100 - Empty Fil...
Yakety-Yak 101 - General ...
Yakety-Yak 102 - General ...
Yakety-Yak 103 - General ...

Screenshot 4.18: The *Display HTML* properties window with the source asset being chosen.

The default file name setting for *Display HTML* is **index.html** so there is no need to change the page in this case. Once the rule is configured as shown below in *Screenshot 4.19*, close it either by clicking on the green checkmark or pressing enter.



Name : Display HTML

Display page index.html of html asset TRS2004 Intro in a new in-game window.

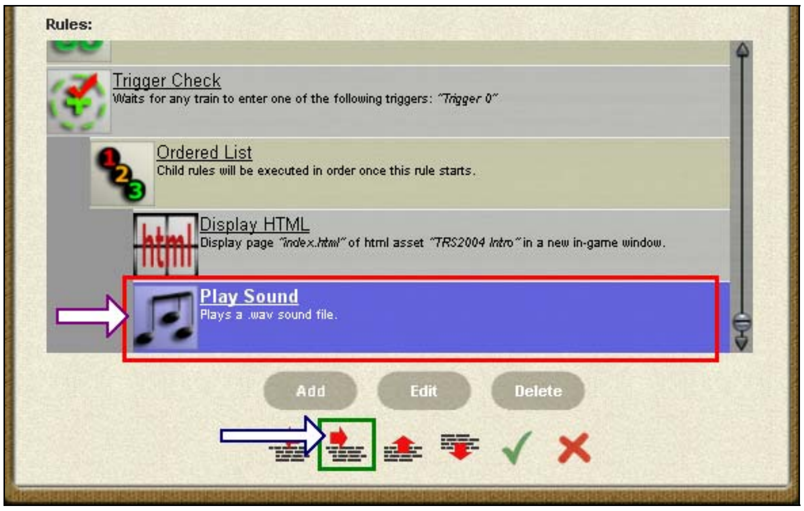
Screenshot 4.19: Fully configured *Display HTML* rule with asset and filename set.

Adding and Configuring the Play Sound Rule

After Display HTML has been setup and its properties window closed, it is time to add the final rule in this tutorial and that is the *Play Sound* rule.

Play Sound is a basic task rule that plays a .wav sound file from the directory of a host HTML asset in the same way Display HTML also uses a HTML asset to find a file. Depending on its settings, it will either complete the moment the sound has started to play or once the sound has completed playing.

As this tutorial session requires *Play Sound* to be played after the user has closed the browser window opened by *Display HTML*, it needs to be placed below *Display HTML* as the second child rule of *Ordered List* as show below in *Screenshot 4.20*.



Trigger Check
Waits for any train to enter one of the following triggers: "Trigger 0"

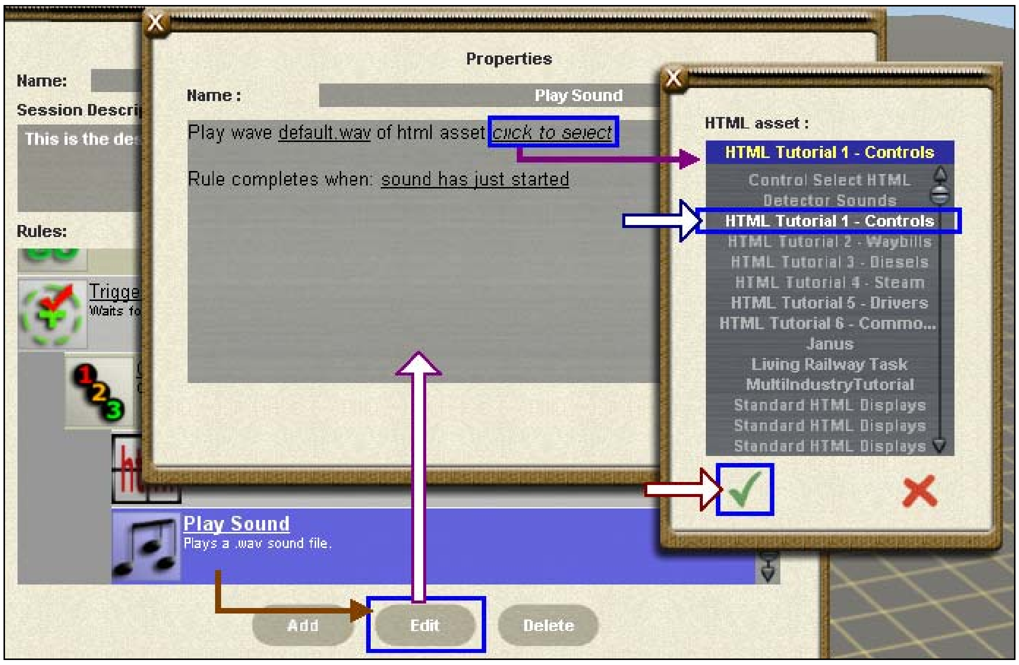
Ordered List
Child rules will be executed in order once this rule starts.

Display HTML
Display page "index.html" of html asset "TRS2004 Intro" in a new in-game window.

Play Sound
Plays a .wav sound file.

Screenshot 4.20: The *Play Sound* rule indented below *Display HTML* as the second child rule of *Ordered List*.

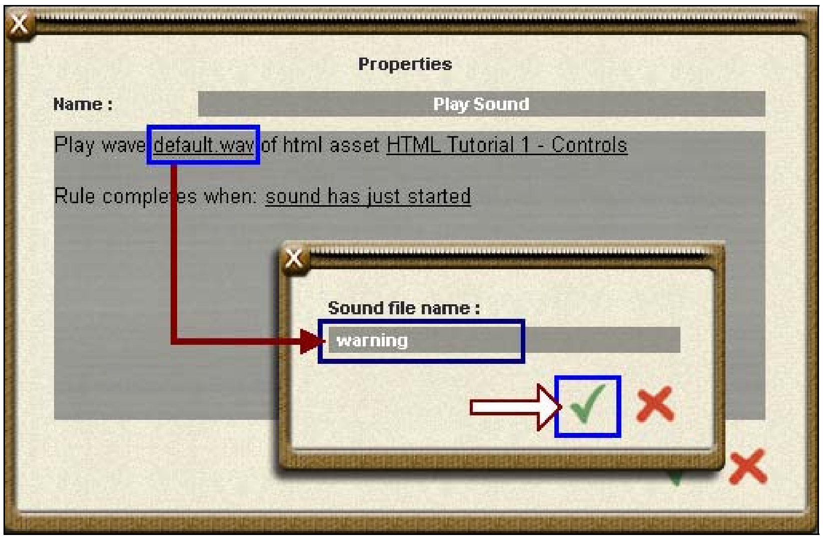
Once positioned, open the properties window of *Play Sound*. Just like *Display HTML*, click on the "click to select" link to choose the HTML asset for the rule to find the file in. When the list of HTML assets appears, select "**HTML Tutorial 1 – Controls**" as shown in *Screenshot 4.21*.



Screenshot 4.21: Setting the asset where *Play Sound* can find a sound.

The file "**default.wav**" is a default placeholder name and this needs to be changed to "**warning.wav**", a sound file located in the directory of the "**HTML Tutorial 1 – Controls**" asset.

To set the sound file, click on the "default.wav" link. This will open the "**Sound file name**" text entry box where the name of the sound file can be entered. As **warning.wav** is the desired file name, enter in the text "**warning**" which is the sound file name stripped of its **.wav** extension as shown below in *Screenshot 4.22*.



Properties
Name :
Play Sound
Play wave default.wav of html asset HTML Tutorial 1 - Controls
Rule completes when: sound has just started
Sound file name :
warning

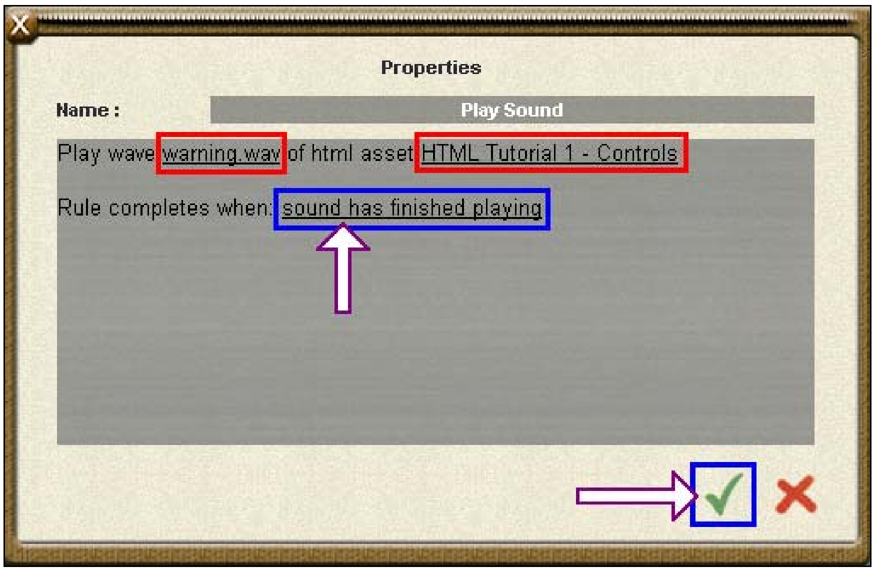
Screenshot 4.22: Setting the file name property of the *Play Sound* rule.

When entering a file name in a text entry box in Trainz, full stop characters cannot be used, hence file extensions are not supported by any rule that deals with files such as *Play Sound* or *Display HTML*.

In the case of audio files, it is essential that the file name is specified correctly and refers to a valid file for the chosen asset. If this is not done correctly, the session will be interrupted with a script exception. Care should be taken such that script failures are never imposed on someone playing the session.

Note: Further details on using filenames within this rule are available in the description of the *Play Sound* that can be found in the Rule Reference section at the end of this document.

Before closing *Play Sound*, click on the "sound has just started" link so it changes to "sound has finished playing" as shown below in *Screenshot 4.23*. This tells the rule to complete itself once the sound file has finished playing (effectively it waits for the length of the sound file in seconds before completing itself). As a result of this setting, the parent *Ordered List* rule won't complete until the sound has finished playing.


Properties

Name : Play Sound

Play wave warning.wav of html asset HTML Tutorial 1 - Controls

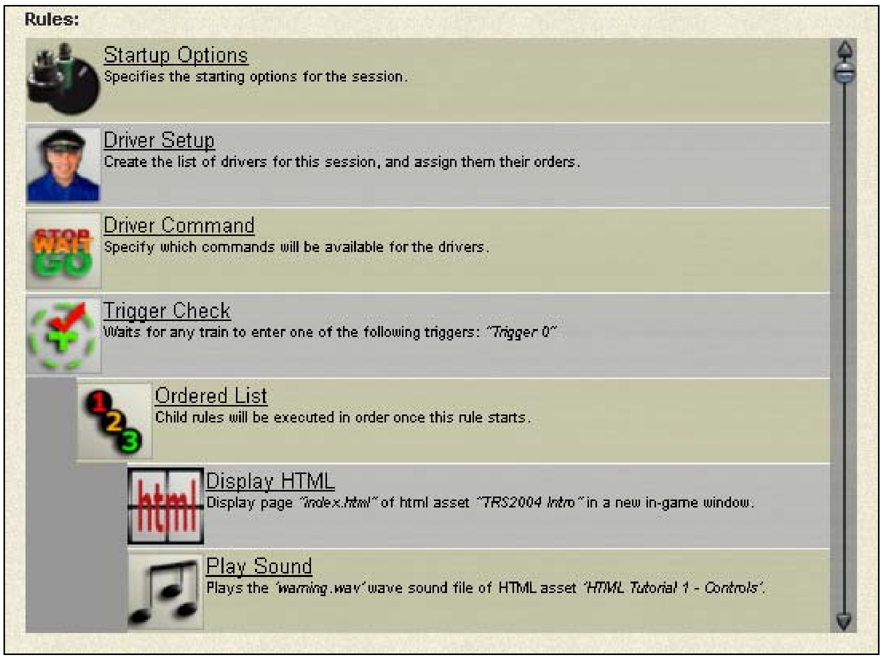
Rule completes when: sound has finished playing

Screenshot 4.23: Setting the *Play Sound* rule to complete when the sound has finished playing.

Close the configured *Play Sound* properties window and return to the **Edit Session** window.

Saving and Running the Session

After the Play Sound rule has been closed, this example session is complete and ready to done. The rules should be arranged in a hierarchy that matches up with the arrangement shown in *Screenshot 4.24*.


Rules:

Startup Options
Specifies the starting options for the session.

Driver Setup
Create the list of drivers for this session, and assign them their orders.

Driver Command
Specify which commands will be available for the drivers.

Trigger Check
Waits for any train to enter one of the following triggers: "Trigger 0"

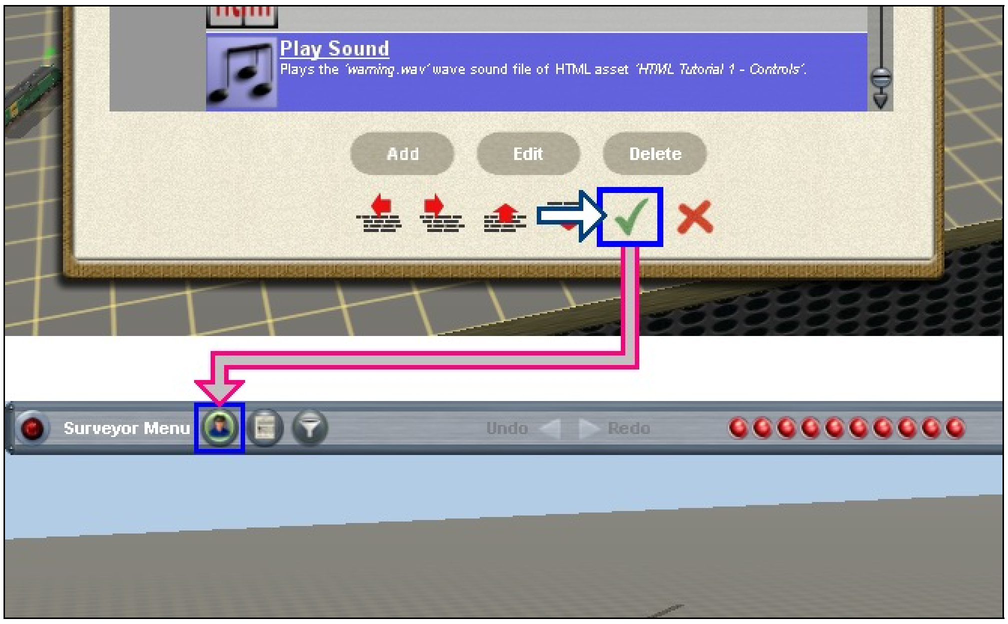
Ordered List
Child rules will be executed in order once this rule starts.

Display HTML
Display page "index.html" of html asset "TRS2004 Intro" in a new in-game window.

Play Sound
Plays the 'warning.wav' wave sound file of HTML asset 'HTML Tutorial 1 - Controls'.

Screenshot 4.24**:** How the complete fully configured session should look like in the **Edit Session** window.

Now that the session has been fully setup, it is time to save it and run it in Driver. To save the session, close the **Edit Session** window by clicking on the green checkmark near the bottom right corner. Once back in the main Surveyor screen, click on the "**Quick Driver**" button that is the first of the three icons adjacent to the **Surveyor Menu** as shown in *Screenshot 4.25*.


Play Sound
Plays the 'warning.wav' wave sound file of HTML asset 'HTML Tutorial 1 - Controls'.

Screenshot 4.25**:** Saving and launching the session in Surveyor.

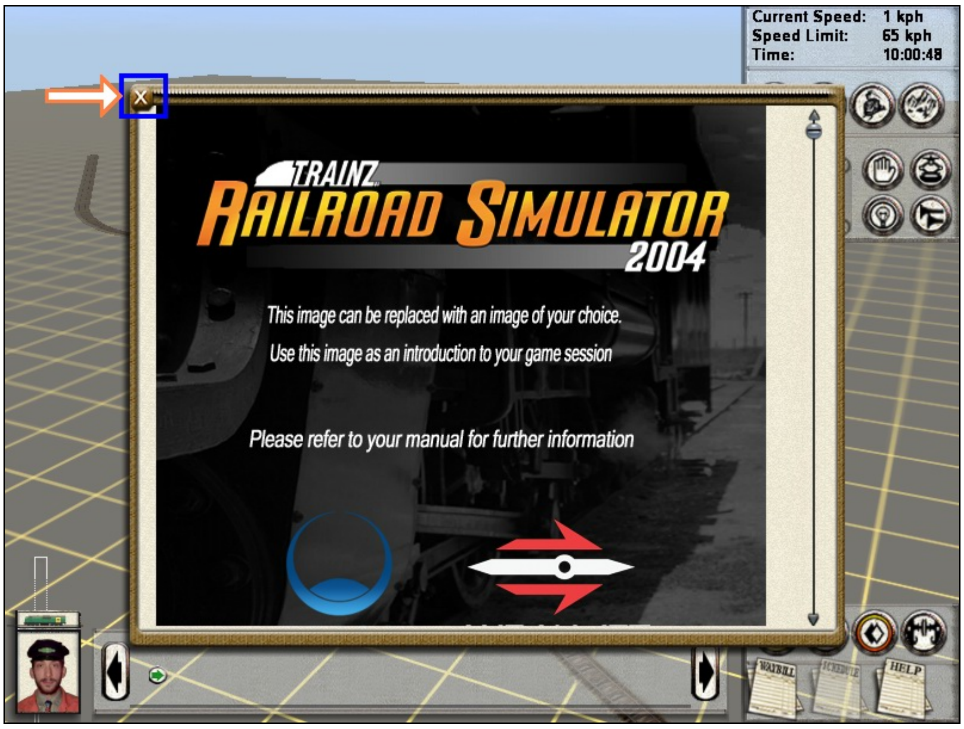
This quick drive feature allows a session to be started from within Surveyor without having to exit back to the Trainz main menu and then go into the Driver session selection window. When developing a session gradually, this will save a lot of time.

Note: As an alternative to the Quick Drive button on the Surveyor menu bar, the "Ctrl + F2" shortcut key is also available.

When in Driver with the locomotive on the track ready to go, drive it forward slowly so that it will reach the trigger.

Note: Triggers are not visible from Driver but they can be seen be on the Minimap that is accessed by using the "Ctrl + M" shortcut keystroke.

When the locomotive reaches the trigger, the *Trigger Check* rule will notice this event and start its *Ordered List* child rule. *Ordered List* will then in turn start its first child rule of Display HTML which will result in a browser window being opened as shown in *Screenshot 4.26*.


Current Speed: 1 kph
Speed Limit: 65 kph
Time: 10:00:48
TRAINZ
RAILROAD SIMULATOR
2004
This image can be replaced with an image of your choice.
Use this image as an introduction to your game session
Please refer to your manual for further information
WAYBILL
SCHEDULE
HELP

Screenshot 4.26: Browser window created by the **Display HTML** rule.

Once the browser window appears, click on the '**X**' at the top left hand corner as shown above in *Screenshot 4.26*. When this window is closed, *Display HTML* will complete itself and *Ordered List* will then start its next child, which is the Play Sound rule.

Play Sound will play the **warning.wav** file, which is a short beeping sound of several seconds in length. Once this sound has finished playing, *Play Sound* completes itself and as it's the last child rule of *Ordered List*, the *Ordered List* rule will also be complete as it has been set to only cycle through its child rules once.

One potential trap with *Trigger Check* is that it is only active while the train is within the scope of one of the observed triggers. If the train leaves the trigger's scope (20 meters in either direction along the track), the *Trigger Check* rule will stop its child rules. In this case when the locomotive is going fast, the user may not get a chance to close the browser window in time as the locomotive has left the trigger and *Trigger Check* has suspended child rules which will result in the *Display HTML* rule being terminated. The tutorial instructions did specify to drive slowly for this reason.

For the sake of learning and getting into the habit of making a session based on a required situation, here are some suggestions of things to do with this example session:

- Play the sound first and display the page after the sound has completed playing.
- Have the sound played and the HTML page displayed simultaneously when the trigger is activated while still using the *Ordered List* rule and not making the *Display HTML* or *Play Sound* rules into child rules of *Trigger Check*.
- Have the page and sound played every time the locomotive enters the trigger, not just once.

All of these tasks are possible with a few simple alternations to this example session's rules, and that's all there is to it!

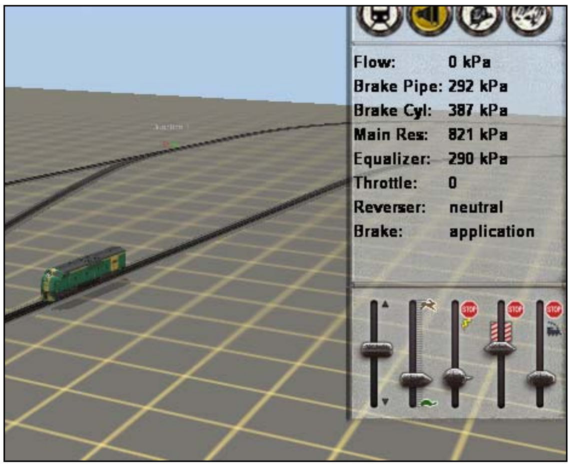
Part 5 – Rule Reference

This section provides a description of all of the rules provided with TRS2006. All rules are listed by name in alphabetical order.

Note: Not all the rules included with TRS2006 are documented here. This section only covers rules written by Auran. 3rd Party provided rules included with TRS2006 or provided via the Download Station are not covered.

Cab Controls HUD

This rule shows/hides the *Cab Controls HUD*, which is a HUD panel that allows locomotive controls such as throttle and brakes to be manipulated via levers on the Driver interface.



```
Flow:        0 kPa
Brake Pipe:  292 kPa
Brake Cyl:   387 kPa
Main Res:    821 kPa
Equalizer:   290 kPa
Throttle:    0
Reverser:    neutral
Brake:       application
```

***Screenshot 5.1*:** The Driver interface with the Cab Controls HUD panel present.

From left to right, the levers are:

- Reverser
- Throttle
- Train brakes
- Dynamic brakes
- Loco brakes

The Cab Controls panel will vary slightly if the current vehicle in camera focus is a steam locomotive.

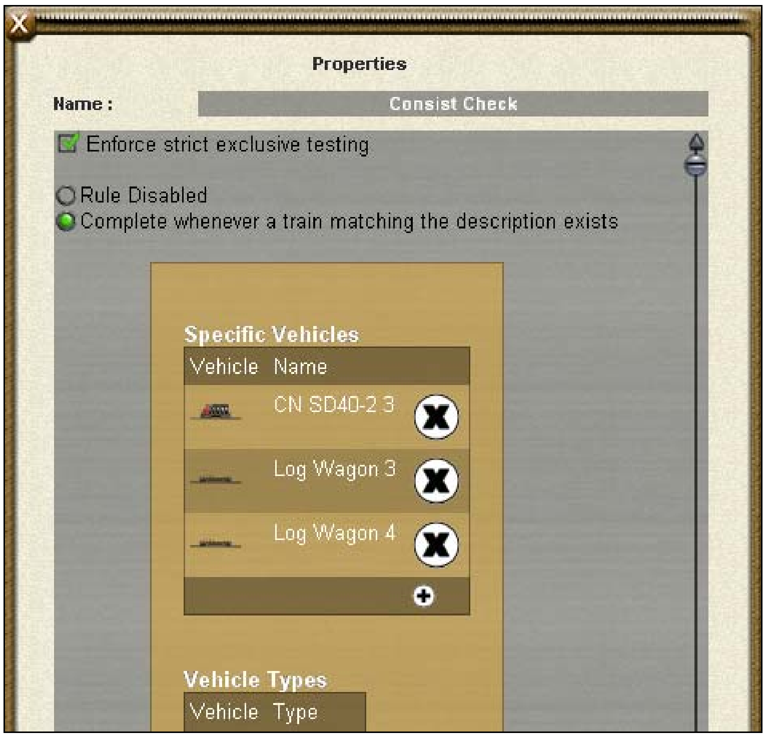
Once started, the *Cab Controls HUD* rule shows/hides the Cab Controls HUD panel and then completes itself. It does not wait for anything or execute any child rules. It is a 'use once' Task rule that needs to be reset if more than one run of it is required.

Note: If Driver is in DCC mode when this rule is run, it will have no effect.

Consist Check

This rule waits for a train consist matching specific criteria to exist on the route. Once a train matching the criteria exists, the child rules will be started. When all the child rules have completed, the *Consist Check* rule will the complete itself.

The criteria used in *Consist Check's* properties allows a consist definition to be specified in terms of specific vehicles, vehicle types and driver characters. There are two ways the criteria can be applied: normal or strict. This criteria can be changed by clicking on the "Enforce strict exclusive testing" property checkbox as seen in *Screenshot 5.2*.



Properties

Name : Consist Check

☑ Enforce strict exclusive testing

○ Rule Disabled
● Complete whenever a train matching the description exists

Specific Vehicles

Vehicle	Name
	CN SD40-2 3
	Log Wagon 3
	Log Wagon 4

Vehicle Types

Vehicle	Type

Screenshot 5.2: Properties window of the *Consist Check* rule.

In normal mode, a train consist will match the criteria filter if it has all of the items in it that the filter has. The consist will also pass the filter check if it has additional vehicles/drivers that are not specified in the filter as long as it still has all the items that the filter does require.

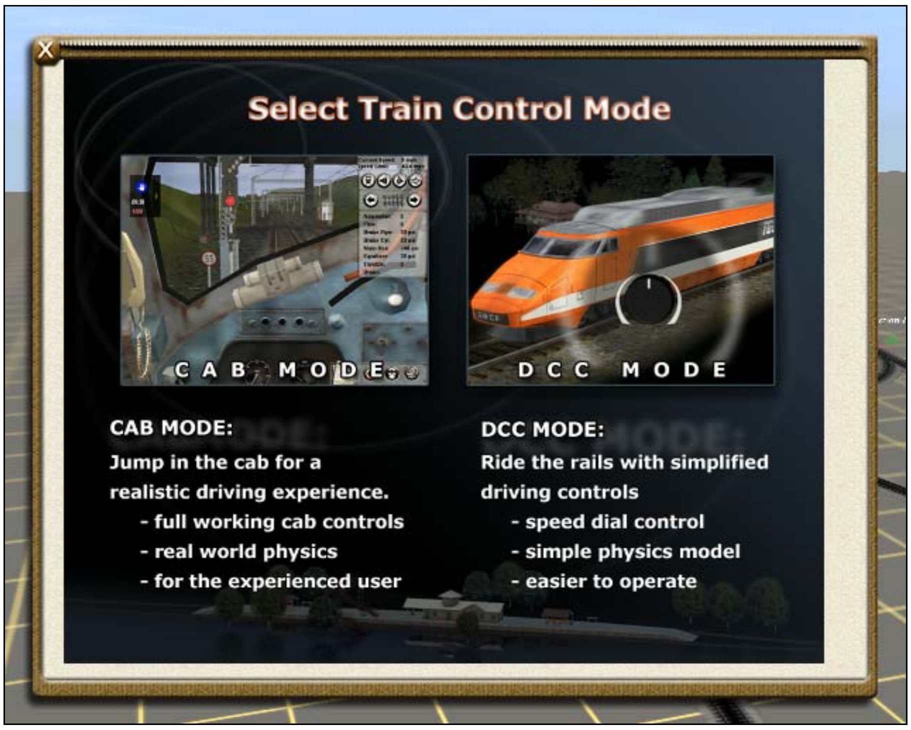
When running in strict mode, an exclusive test to a train consist is applied such that even if the train consist has all the items required, the test will fail if there are specific vehicles in the train consist that are not in the filter's vehicle list.

Consist Check is a 'use-once' rule *Event* rule that once activated, runs the child rules and waits for them to complete. After the child rules are complete, *Consist Check* will complete itself and stay idle. It will need to be reset if another run is required.

Control Type

This rule provides a way to set the driving mode in Driver. It allows either DCC or Cab Control mode to be specified. Alternatively it can also be configured to let the user decide what mode they want to use through a browser window.

If *Control Type* has been configured to directly set Driver to DCC or Cabin mode, it rule will set the control mode and complete straight away. Otherwise, it will wait for the user to choose their preferred mode from the pop-up window and then complete once a mode is chosen. If the user quits the window without choosing, DCC will be used as the default.


Select Train Control Mode

CAB MODE

DCC MODE

CAB MODE:
Jump in the cab for a realistic driving experience.
- full working cab controls
- real world physics
- for the experienced user

DCC MODE:
Ride the rails with simplified driving controls
- speed dial control
- simple physics model
- easier to operate

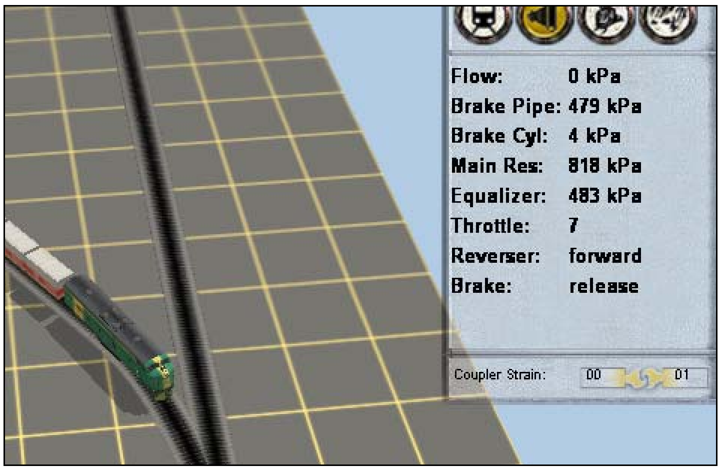
***Screenshot 5.3*:** The selection window that that "Ask user" option will cause this rule to display when run.

Regardless of whether it is a pre-set mode or user-selected, this rule completes without executing any child rules. It is a 'run once' *Task* rule that needs to be reset before it can be executed again.

Tip: The *Startup Options* rule can also do the control mode setting/selection this rule does as well as set a few other things in the Trainz world.

Coupler Breakage HUD

This rule shows/hides the *Coupler Breakage* HUD, which is a HUD panel that indicates the level of coupler strain on the train currently targeted by the camera. *Screenshot 5.4* below shows the *Coupler Breakage* HUD when displayed.



```
Flow:        0 kPa
Brake Pipe:  479 kPa
Brake Cyl:   4 kPa
Main Res:    818 kPa
Equalizer:   483 kPa
Throttle:    7
Reverser:    forward
Brake:       release

Coupler Strain:   00   01
```

***Screenshot 5.4*:** The Coupler Breakage HUD panel in Driver.

Once started, the *Coupler Breakage* rule shows/hides the Coupler Breakage HUD panel and then completes itself. It does not wait for anything or execute any child rules. This is a 'use once' *Task* rule that needs to be reset if more than one run of it is required.

Note: The Coupler Breakage panel will work in both DCC and Cabin mode. However you can't break couplers in DCC mode and the HUD won't be as useful due to limited physics DCC mode has.

Tip: The Coupler Breakage HUD can also be shown/hidden through the *Vehicle Physics* rule.

Custom Script

Allows a custom script to be defined specific to the rule instance. Not fully documented yet.

Derailment Realism

This rule sets the derailment realism that applies to all trains running in Driver. There are three levels of derailment realism provided:

- None
- Arcade
- Realistic

Even if the derailment mode is set to 'None', it is still possible for a derailment to occur. Although this mode allows speeding around curves safely, driving a train over the end of a siding will cause a derailment, as will a collision between two different trains at a junction.

Once started, the *Derailment Realism* rule sets the desired derailment settings for the Trainz world and completes itself straight away. It does not wait for anything or execute any child rules. It is a 'use once' *Task* rule that needs to be reset if more than one run of it is required.

Tip: The *Startup Options* rule can also set the derailment realism as well as several other parameters in the Trainz world.

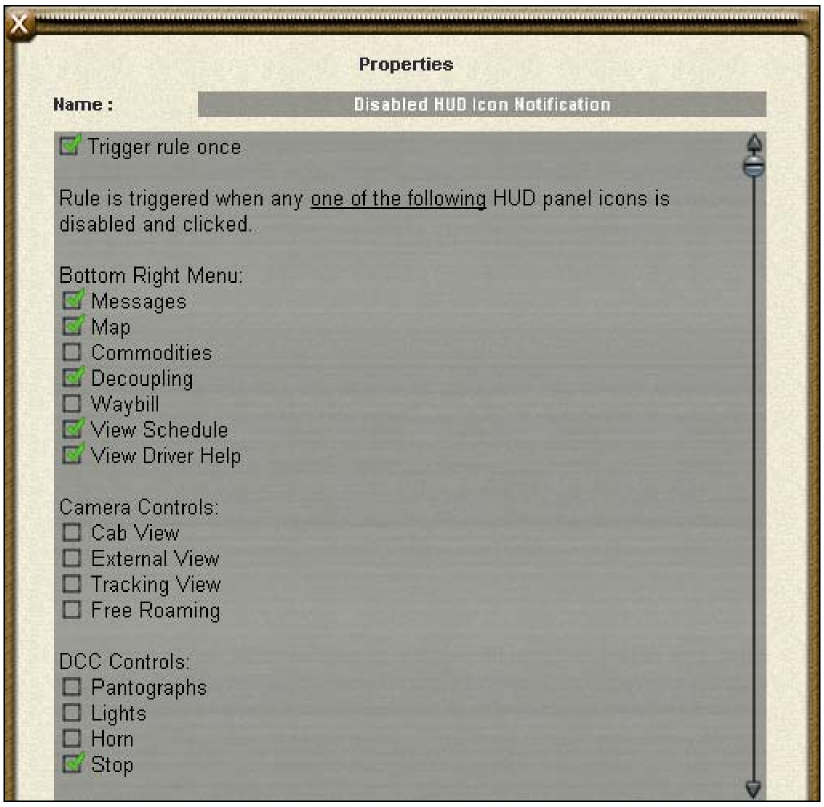
Tip: The *Wait for Derailment* rule allows a derailment to be detected and acted on.

Disabled HUD Icon Notification

The *Disabled HUD Icon Notification* rule waits for a click to occur on a disabled HUD panel icon. The HUD panel icons that this rule can monitor for a click on (when disabled) grouped by panel are:

- *Bottom Right Menu:* Messages, Map, Commodities, Decoupling, Waybill, View Schedule & View Driver Help
- *Camera Controls:* Cab View, Chase View, Tracking View & Free Roaming
- *DCC Controls:* Pantographs, Lights, Horn & Stop

This rule can be set to either respond to any disabled icon or only specific ones that have been checked in its properties. In either case, the response to a click on a disabled HUD icon will be the same.



Name : Disabled HUD Icon Notification

- [x] Trigger rule once

Rule is triggered when any one of the following HUD panel icons is disabled and clicked.

Bottom Right Menu:
- [x] Messages
- [x] Map
- [ ] Commodities
- [x] Decoupling
- [ ] Waybill
- [x] View Schedule
- [x] View Driver Help

Camera Controls:
- [ ] Cab View
- [ ] External View
- [ ] Tracking View
- [ ] Free Roaming

DCC Controls:
- [ ] Pantographs
- [ ] Lights
- [ ] Horn
- [x] Stop

Screenshot 5.5: Properties window of the *Disabled HUD Icon Notification* rule.

If the rule is to listen for specific disabled icons, the icons are to be chosen by selecting the appropriate checkboxes as shown in *Screenshot 5.5*.

Note: If an icon is not disabled, this rule won't be able to detect an attempt to click on it. The *Wait for Click on HUD* Icon rule provides a way to detect a click on an enabled HUD icon while *Enable/Disable HUD Icons* can enable/disable HUD icons.

When started, this rule waits until a disabled HUD icon is clicked on. Once this has happened, all child rules will be started. When all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the "Trigger rule once" property setting.

Display Custom HUD

This rule shows or hides the Custom HUD panel in Driver. This HUD panel is used to display extra information about the current train as well as room for custom variables.

The initial values that are on the Custom HUD panel are:

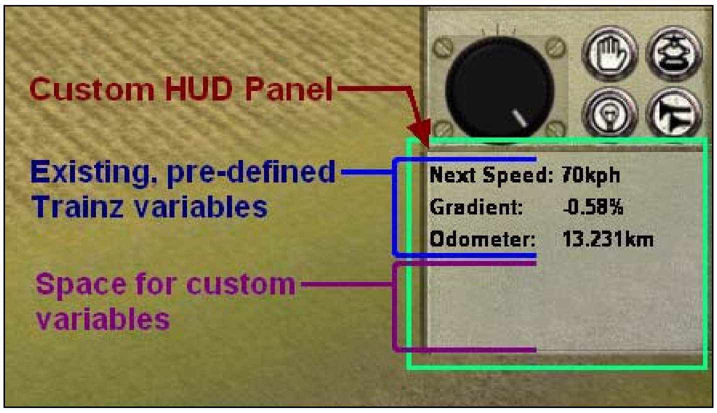
- Next speed limit sign (if any)
- Current gradient
- Locomotive odometer

The gradient is based on the current locomotive that the camera is focused on. It is expressed as a percentage and refers to the level of rise relative to track distance. For a downward slope, a negative value is used. The gradient value is not calculated for vehicles that are not locomotives.

The locomotive odometer tracks the mileage for each locomotive on an individual basis and will be based on the locomotive that is currently targeted by the camera.

These three values are already part of the Custom HUD and are not removable and will always be present on this HUD.

In addition to the existing *Custom* HUD variables, there is also space for 2 custom variables. See *Screenshot 5.6* for an idea of where they can fit.


Custom HUD Panel
Existing, pre-defined Trainz variables
Space for custom variables

Next Speed: 70kph
Gradient: -0.58%
Odometer: 13.231km

Screenshot 5.6: The Custom HUD panel in Driver.

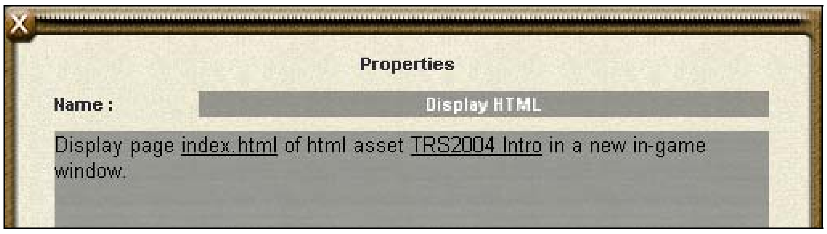
Custom variables are virtual integer variable values that are created, deleted, modified or checked using the various variable rules provided. The Custom HUD can be used to display such values if needed, although they can also be used as tracking values that the user need not see. See *Variable Show* for more details on how to create session variables.

The *Display Custom HUD* rule itself will either show or hide the Custom HUD Panel when started. Once done, this rule completes itself straight away without running any child rules. It is a 'use once' *Task* rule that will need to be reset before being run again.

Display HTML

This rule displays a HTML page in a browser window and waits for that browser window to be closed before completing itself.

The HTML page is sourced from the directory of a HTML asset. The page name is entered in through the properties window and the HMTL asset can be selected from a pop-up list of available assets.



Display HTML

Display page index.html of html asset TRS2004 Intro in a new in-game window.

Screenshot 5.7: Properties window of the *Display HTML* rule set to show the page "index.html" from the HTML asset "TRS2004 Intro".

When executed *Display HTML* creates the browser window and loads the page. It will then complete once the user has closed the page. It will not execute any child rules and is a 'use once' *Task* rule that needs to be reset if the page is to be displayed again.

To function properly, *Display HTML* is dependent on the user entering a valid HTML file from the chosen HTML asset. If not, an empty browser window will be displayed as a HTML file couldn't be found. The same will also occur if a HTML isn't selected.

Note: When entering the file name do not include the .html extension, as Trainz does not allow full stop characters ('.') to be entered. Enter the file name without the .html extension. For this rule to work, the HTML file in the asset directory must have a .html extension, *NOT* a .htm extension.

Display HTML Pages

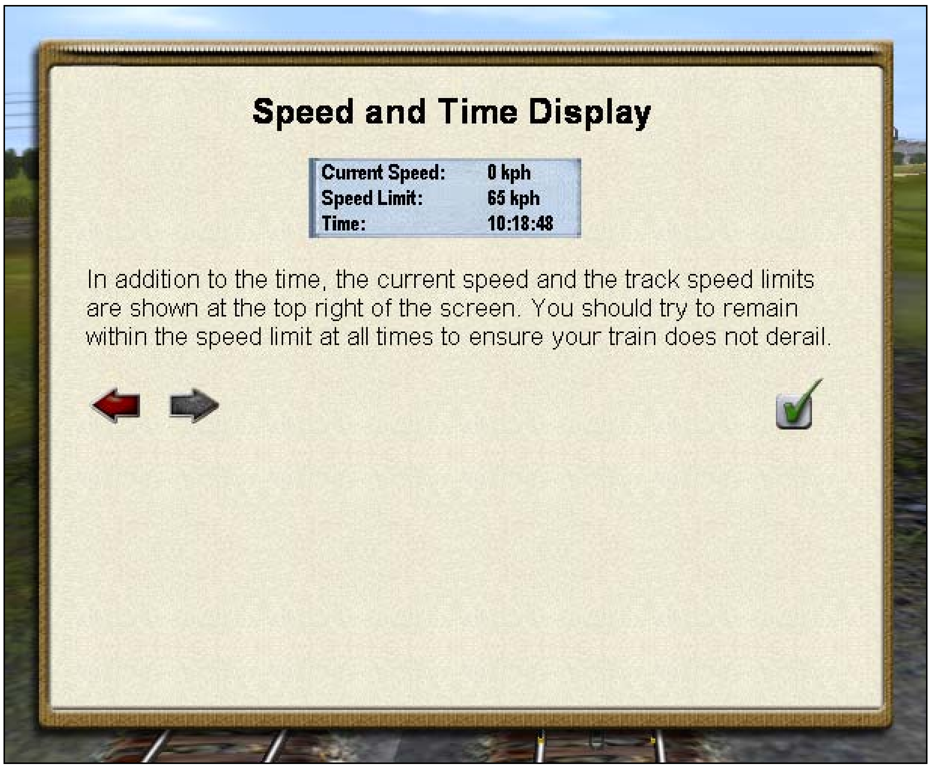
This rule displays a linear collection of HTML pages in a browser window that can be navigated through in both forward and backward directions. This provides the advantage of being able to display a sequence of instructions across multiple pages that the user can go refer back to and call back on as needed.

Display HTML Pages was primarily written for use in the TRS2006 tutorial sessions and was never intended to be a general-purpose page displaying solution for everything. However it might still be of interest to some. Be warned that this rule is very strict on its requirements for HTML assets and is not forgiving or guaranteed to work with assets that don't match the specifications provided here exactly.

Once started, this rule forwards on the pages and property settings to the sequence browser part of the Trainz interface and will complete itself straight away. It does not have an on-going role in managing the sequence as the Trainz interface now takes care of things.

Navigation of the pages is done through three links: 'previous', 'next' and 'done'. How these links appear is dependent on the HTML pages provided are defined and covered later on.

In the case of the TRS2006 tutorial sessions, there are two red arrow buttons for the 'previous' and 'next' in the bottom left corner of the page and the 'done' button is a green checkmark in the bottom right corner. *Screenshot 5.8* shows an example of one of these pages.



Speed and Time Display

Current Speed: 0 kph
Speed Limit: 65 kph
Time: 10:18:48

In addition to the time, the current speed and the track speed limits are shown at the top right of the screen. You should try to remain within the speed limit at all times to ensure your train does not derail.

***Screenshot 5.8*:** The sequence browser window with navigation buttons.

Sometimes one of the navigation functions may not be available for that particular page. If that is the case, an alternative 'dead' version of that navigation function will be inserted. As with the actual navigation links, the HTML asset creator must also provide dead versions of each of the links. In *Screenshot 5.8*, the 'previous' and 'done' navigation buttons are available but that 'next' button isn’t as the grayed out disabled version of it has been used.

Generally speaking, the next link goes along to the next page and the previous link goes to the previous page. The default behavior of the ‘done’ is to close and terminate the sequence browser when clicked. Once closed, the browser is gone and can’t be brought back for the session. There are ways to configure the browser to either go to the next page or just become invisible when the 'done' link is clicked.

Variations of this basic behavior and more complex arrangements are possible with the sequence browser through the properties of this rule as well as that of the *Set HTML Pages* rule.

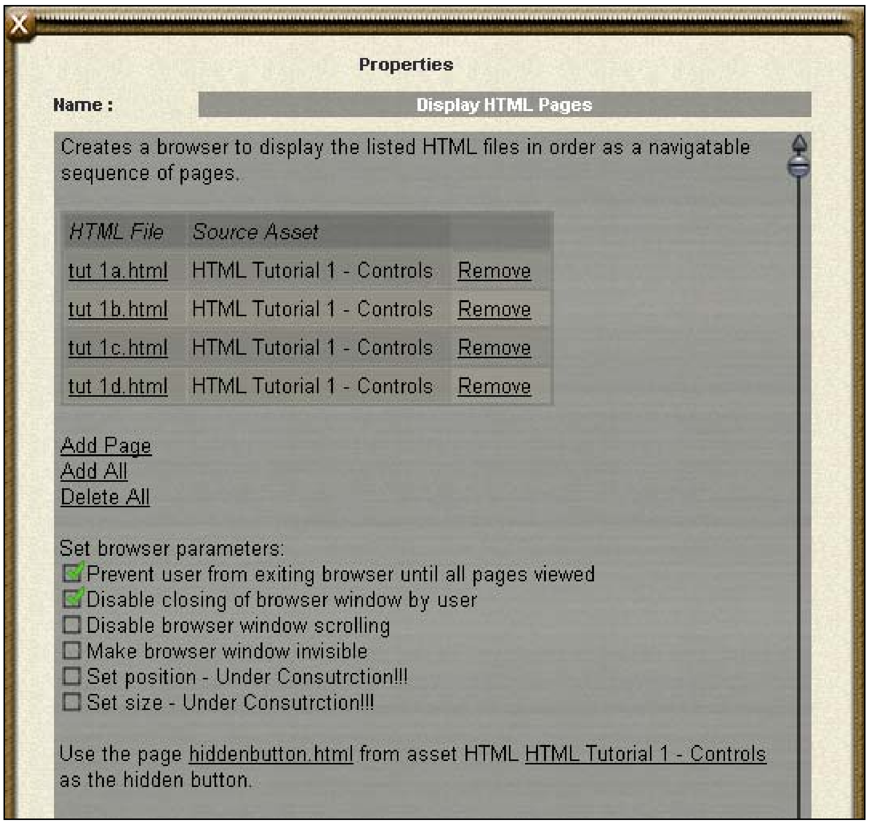
Tip: The opening/closing of a page by the sequence browser can be waited for on by the *Wait for HTML Pages* rule.

The configuration options for the sequence browser are quite complex and the best way to learn how to use it with this rule (along with *Set HTML Pages* and *Wait for HTML Pages*) is by example. Checkout the attached HTML assets (which were used for the tutorials), the session configuration for the tutorials and Part 5 of this manual for actual demonstrations of the sequence browser being used to run an interactive session.

HTML Pages List

The pages this rule displays are defined in a list at the top of the properties window as shown in *Screenshot 5.9*. Each page in the list item is specified as page name and the HTML asset that it belongs to.

The HTML asset is selected from the list provided by clicking the 'Add Page' link. Once added to the list, the default page of "index.html" can be changed to the desired page by clicking on it. As with all HTML files in properties windows, the page name is specified without the .html extension. The reasoning for this is discussed in the *Display HTML* rule description. The 'Remove' link adjacent to the page item will remove that item from the list.



Name : Display HTML Pages

Creates a browser to display the listed HTML files in order as a navigatable sequence of pages.

HTML File	Source Asset	
tut 1a.html	HTML Tutorial 1 - Controls	Remove
tut 1b.html	HTML Tutorial 1 - Controls	Remove
tut 1c.html	HTML Tutorial 1 - Controls	Remove
tut 1d.html	HTML Tutorial 1 - Controls	Remove

Add Page
Add All
Delete All

Set browser parameters:
- [✓] Prevent user from exiting browser until all pages viewed
- [✓] Disable closing of browser window by user
- [ ] Disable browser window scrolling
- [ ] Make browser window invisible
- [ ] Set position - Under Consutrction!!!
- [ ] Set size - Under Consutrction!!!

Use the page hiddenbutton.html from asset HTML HTML Tutorial 1 - Controls as the hidden button.

Screenshot 5.9: Properties window of the *Display HTML Pages* rule.

At the bottom of the list below the "Add Page" link, there is the "Add All" and the "Delete All" links. "Delete All" will empty out the entire pages list while "Add All" can be used to add all the pages of some HTML assets at once.

Add All Requirements

The "Add All" link is a shortcut that allows all pages in the directory of a selected HTML asset to be added to the list at once, however this only works on HTML assets that have a file list specified in the string table of their configuration.

The string table entries must be numbered from 0 through to however many files there are with a prefix of "**html-page-**". For example, here are the string table entries for the "HTML Tutorial 1 - Controls" asset:

String Table Entry	*String Text*
html-page-0	"tut_1a"
html-page-1	"tut_1b"
html-page-2	"tut_1c"
html-page-3	"tut_1d"
html-page-4	"tut_1e"
html-page-5	"tut_1f"
html-page-6	"tut_1g"
html-page-7	"tut_1h"
html-page-8	"tut_1i"

Note: When added through CCP, the string table text is not wrapped in quotation marks as they are in the above example.

The numbered ordering must be linear starting from 0. Any break of this linear order will result in pages missing from the list generated by *Display HTML Pages* when it examines the string table.

As with the names of HTML pages entered through the properties window in Surveyor, HTML file names in the string table must be specified ***WITHOUT*** the .HTML extension.

Note: This page gathering is not a HTML asset feature. It is simply a case of the *Display HTML Pages* rule looking for certain things in an asset's configuration. Such string table entries are not used anywhere else and should ***NOT*** be relied on for any other use.

Browser Parameters

Below the HTML page list, there is the browser parameters section. These properties define how the sequence browser is to behave once created. The parameters that currently work are:

- *Prevent user from exiting browser until all pages viewed*: Enabling this property prevents the user from closing the browser by user.
- *Disable closing of browser window by user*: Enabling this property results in the sequence browser window not having a 'X' button in the top left corner, preventing the user from closing it.
- *Disable browser window scrolling*: Enabling this property means the sequence browser window won't have a scrollbar, even when a page is loaded that is longer than the window.

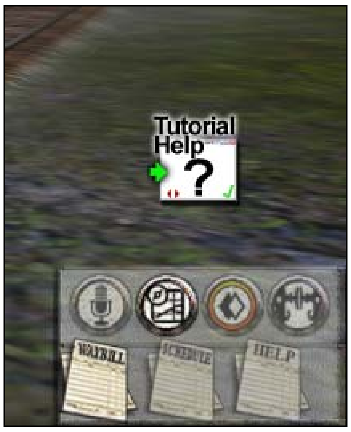
- *Make browser window invisible*: Enabling this property will cause the sequence browser window to be invisible when created.

The "Set position" and "Set size" parameters are not currently implemented or supported.

Hidden Button

The last property at the bottom of the properties window is the hidden button. This optional setting allows a page to be specified that is used as a button to show the main sequence browser to be shown again if it has become hidden or invisible.

The hidden button is specified as a HTML page in a host HTML asset. When displayed, the Trainz interface only provides a borderless browser window in the lower right corner of the screen as shown in *Screenshot 5.10*.



***Screenshot 5.10*:** Button to show/hide a hidden sequence browser window.

The HTML page used as the hidden button will only be given an area 64x64 pixels in size and must have a URL of "**live://pages/showhide-button**".

TRS2006 already comes with two HTML pages ready to use as a hidden button:

- button.html from HTML asset "Tutorial Button"
- button.html from HTML asset "Info Button"

The hidden button this rule creates is not suitable for being used to activate or trigger anything else and should not be relied on for anything else apart from its intended purpose.

Note: The hidden button shows the sequence browser window if it has been hidden. It won't even be available if the sequence browser has been closed.

HTML Page Requirements

For a HTML page to work with the navigation buttons this rule allows for, it needs to fulfill several strict requirements. The first requirement is that the HTML file itself needs to have 3 string parameters. These can be placed anywhere, as long as there is $0, $1 and $2 for the 'previous', 'next' and 'done' links respectively.

As an example, this extract from a TRS2006 tutorial page places the parameters in a table at the bottom of a page such that the previous and next links are at the left side of the page with the done link off to the right:

```
<table width="512" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td width="10"></td>
    <td>$0</td>
    <td>$1</td>
    <td width="349"></td>
    <td>$2</td>
  </tr>
</table>
```

The string parameters are is all that is needed for the HTML file. Most of the tricky parts go into string table of the HTML asset that the file is based in.

There are six strings are needed in the HTML asset's config.txt file. There are two needed for each type of button: an active link and a disabled link. The six string table entries required are:

- html-pages-button-prev
- html-pages-button-prev-disabled
- html-pages-button-next
- html-pages-button-next-disabled
- html-pages-button-done
- html-pages-button-done-disabled

It is essential that the "**html-pages-button-prev**", "**html-pages-button-next**" and "**html-pages-button-done**" strings contain a URL link that corresponds to their respective use. For example, "**html-pages-button-prev**" must have a URL of "**live://pages/prev**". What this URL is wrapped in does not matter – it can be an image or text. The other essential URLs are "**live://pages/next**" and "**live://pages/done**".

The disabled strings will be used to place in the HTML page when that particular navigation button is not available. There is no need for a link or URL in the disabled string.

The string table entries below show a complete set of these strings. These particular strings were used for the TS2006 tutorial sessions.

html-pages-button-prev:

```
"<a href='live://pages/prev'>
  <img src='images/button-prev.tga'
       mouseover='images/button-prev-on.tga' width=40 height=40>
</a>"
```

html-pages-button-prev-disabled:

```
"<img src='images/button-prev-off.tga' width=40 height=40>"
```

html-pages-button-next:

```
"<a href='live://pages/next'>
  <img src='images/button-next.tga'
       mouseover='images/button-next-on.tga' width=40 height=40>
</a>"
```

html-pages-button-next-disabled:

```
"<img src='images/button-next-off.tga' width=40 height=40>"
```

html-pages-button-done:

```
"<a href='live://pages/done'>
  <img src='images/button-done.tga'
       mouseover='images/button-done-on.tga' width=40 height=40>
</a>"
```

html-pages-button-done-disabled:

```
"<img src='images/button-done-off.tga' width=40 height=40>"
```

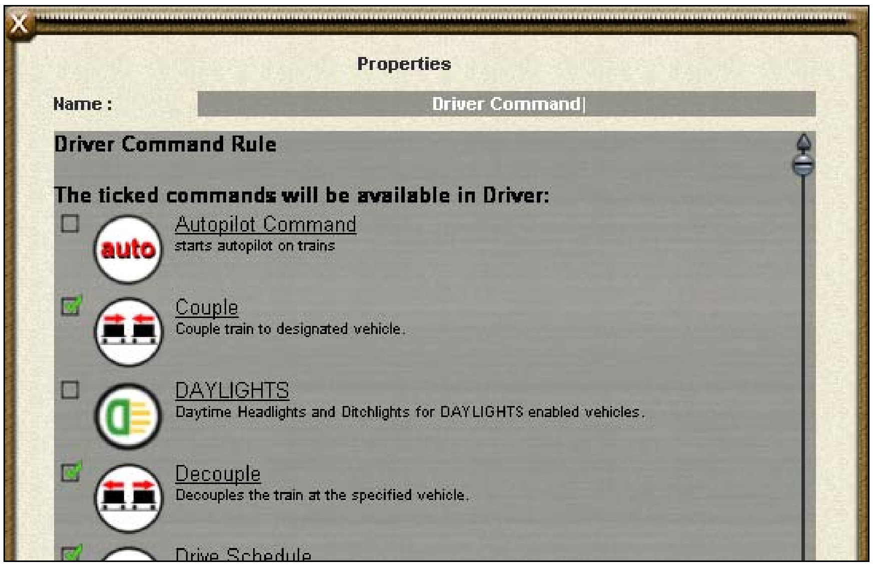
It is generally assumed the author of the HTML file assets being used with this rule knows what they are doing and strictly adheres to these requirements.

Display Video

Plays a video file embedded in a HTML page. Not documented yet.

Driver Command

This rule enables/disables the driver commands such that these settings apply to all drivers in the session. A checkbox list of all available commands is provided to select (enable) or deselect (disable) each command individually as shown in *Screenshot 5.11*.


Properties

Name : Driver Command|

Driver Command Rule

The ticked commands will be available in Driver:

☐ Autopilot Command
starts autopilot on trains

✓ Couple
Couple train to designated vehicle.

☐ DAYLIGHTS
Daytime Headlights and Ditchlights for DAYLIGHTS enabled vehicles.

✓ Decouple
Decouples the train at the specified vehicle.

✓ Drive Schedule

***Screenshot 5.11*:** Properties window of the *Driver Command* rule.

The *Driver Command* rule does not wait for anything or execute any child rules. Once started it will enable/disable the driver commands according to its settings and complete straight away. It is a 'use once' *Task* rule that needs to be reset if more than one execution is required.

Note: Even though the *Enable/Disable Driver Commands* rule is also capable of enabling/disabling access to driver commands, it is a different rule with slightly different features. The main difference is that *Enable/Disable Driver Commands* has selective filtering so commands can be enabled/disabled on certain trains/drivers only. The other major point to note is that *Enable/Disable Driver Commands* is meant to be used when the session is running while *Driver Command* is intended to be an initialization rule run when the session starts.

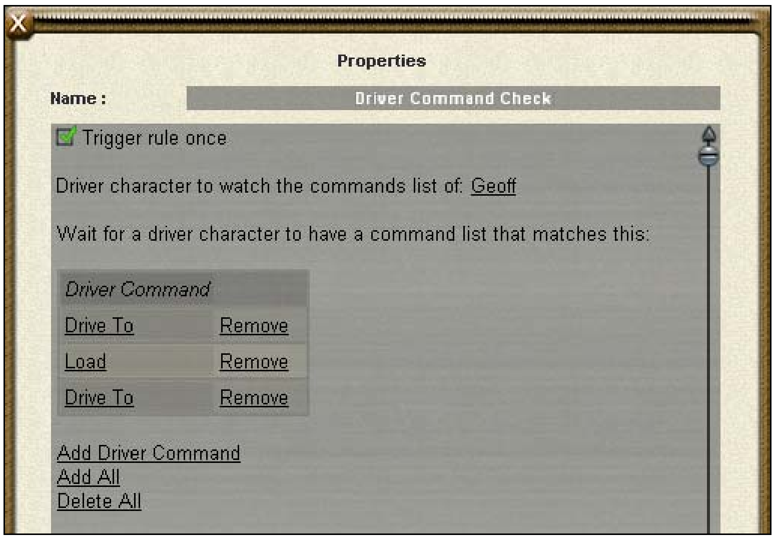
Driver Command Check

The *Driver Command Check* rule is a *Check* rule used to verify the commands list of a driver. It was written specifically for the TRS2006 tutorials and does not have conventional behavior so it may not be as useful and generic as some of the other rules.

In the properties of this rule, there is a target driver character and a list of driver commands.

To select the driver character to monitor, the asset that the targeted driver character is an instance of is selected from a list by clicking on the '*click to select*' link. This means that when looking for a driver, the rule will default to the first driver character it can find based on that driver asset, so be careful if more than one instance of that driver exists in the route.

The command list can be edited and modified with the 'Add Driver Command', 'Add All', 'Delete All' and 'Remove' links that can all be seen in *Screenshot 5.12*.


Properties

Name : Driver Command Check

✓ Trigger rule once

Driver character to watch the commands list of: Geoff

Wait for a driver character to have a command list that matches this:

Driver Command	
Drive To	Remove
Load	Remove
Drive To	Remove

Add Driver Command
Add All
Delete All

***Screenshot 5.12*:** Properties window of the *Driver Command Check* rule.

The 'Add Driver Command' link will add the selected driver command asset to the list. The comparison this rule does is on a command name basis only so there is no specific driver command configuration to setup. The order of the list determines how the driver character's command list is to appear.

The 'Remove' link adjacent to a command in the list will remove that command from the list. 'Delete All' will clear out the list entirely.

Once started, this rule will enter a monitoring cycle where it observes the targeted driver character's command list. When a change in that driver's command list is detected, the rule will compare that against the commands in the rule's configuration.

What happens next depends on the outcome of the comparison between this rules command list and the commands the driver has. This is where Driver Command Check begins to behave differently from most rules. *Diagram 5.1* illustrates the unique behavior of this rule.

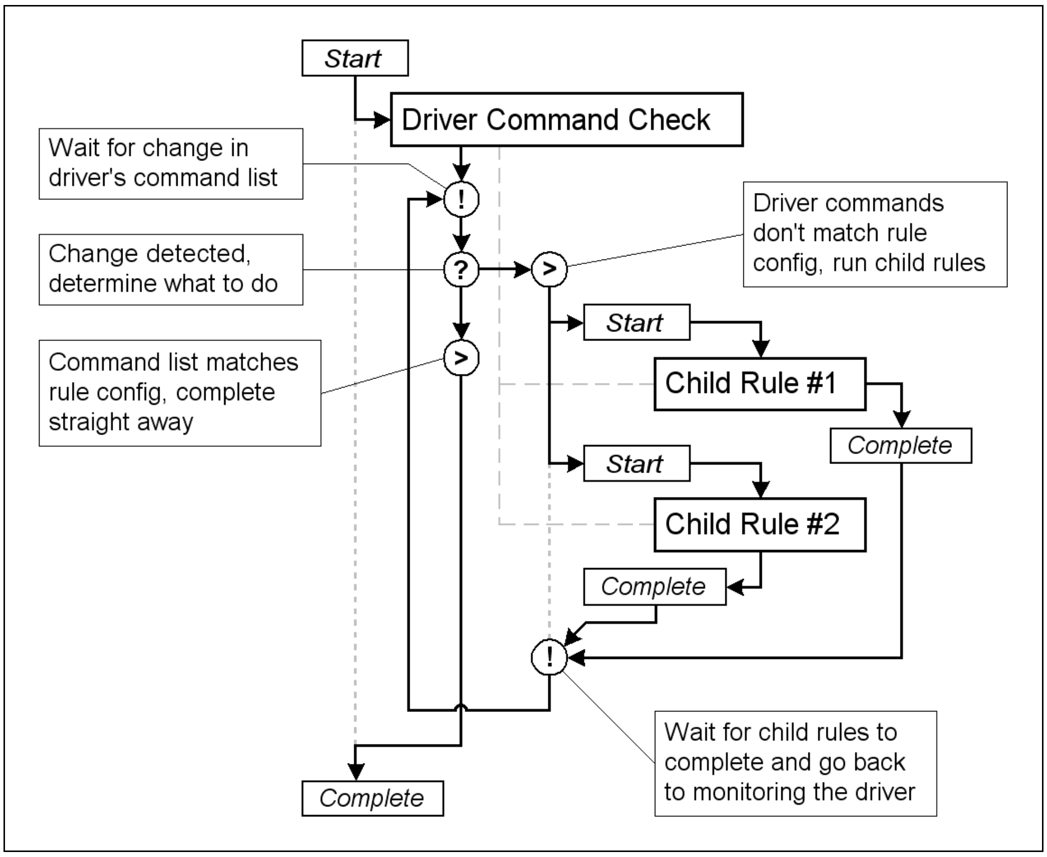

Start
Driver Command Check
Wait for change in driver's command list
Change detected, determine what to do
Command list matches rule config, complete straight away
Driver commands don't match rule config, run child rules
Start
Child Rule #1
Complete
Start
Child Rule #2
Complete
Wait for child rules to complete and go back to monitoring the driver
Complete

Diagram 5.1: Flow chart showing the operation of the *Driver Command Check* rule.

If there is a match, this rule's conditions will be satisfied and depending on the "Trigger rule once" setting it will either complete itself or go back to monitoring driver commands for the next occasion.

If a change in the driver's commands is detected but doesn't match, this rule will then run all the child rules to completion and then monitor the driver for the next time a chance is detected.

Driver Main Menu Options

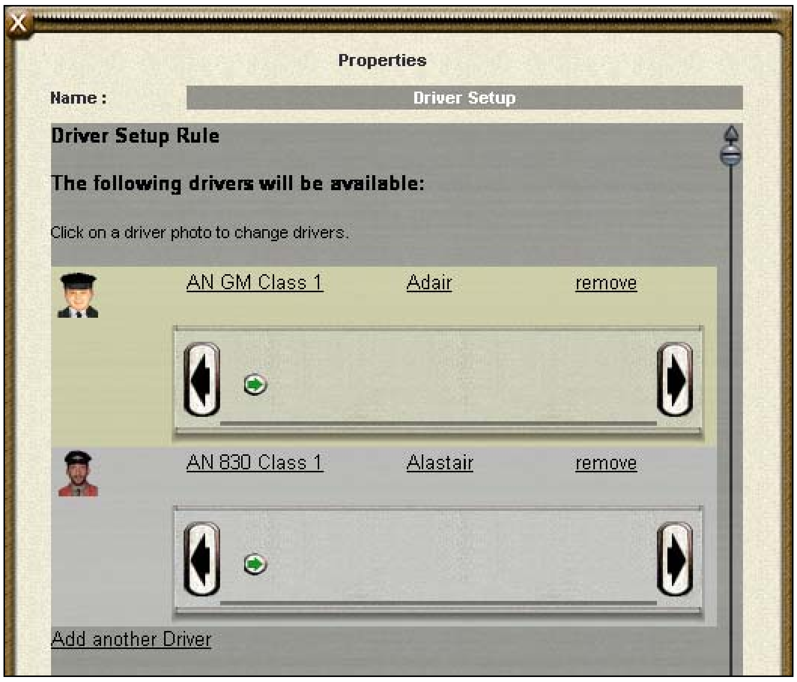
This rule either shows or hides the Driver Main Menu Bar located in the top left corner of the Driver screen. Once executed, it will show/hide the menu bar as set and complete straight away without running any child rules. It is a 'use once' *Task* rule that will need to be reset before being used again.

Driver Setup

The *Driver Setup* rule is used to assign driver characters and their command lists to trains. It is a 'use once' *Task* rule that is to be run as soon as the session is started so that the trains have drivers straight away. Once it has done its work of assigning drivers to trains, it will complete straight away – Trainz manages the drivers once they have been assigned.

Driver Setup is one of the three default rules TRS2006 will include in the session and if you just want to play with the trains, you can safely launch your session and it will automatically assign driver characters to the lead locomotive of all trains that have one. However none of these automatically assigned drivers will have commands assigned – this rule can't read your mind and determine what you actually want the drivers to do! (As nice as that might be ☺) If you want your trains to go into action following a list of commands, this rule will need to be configured.

When the properties window of *Driver Setup* is first opened, you will see that there is a panel with a driver character for each train with a locomotive on the route. *Screenshot 5.13* provides an example of such a default blank set of properties created for a session that has two train consists with locomotives.


Properties

Name : Driver Setup

Driver Setup Rule

The following drivers will be available:

Click on a driver photo to change drivers.

AN GM Class 1 Adair remove

AN 830 Class 1 Alastair remove

Add another Driver

***Screenshot 5.13*:** Properties window of the *Driver Setup* rule.

Across the top of each panel in the properties window, there are 3 properties that this rule assigns default settings for:

- Driver asset (icon);
- Locomotive the driver is assigned to; and
- Driver's name

The driver asset that the driver character is based on is represented as flattering portrait icon in the very top left corner of the panel. Clicking on the icon allows a different driver character asset to be selected from a list.

The name of the locomotive that the driver is to be assigned to is next across from the icon. Clicking on this name allows another locomotive to be selected.

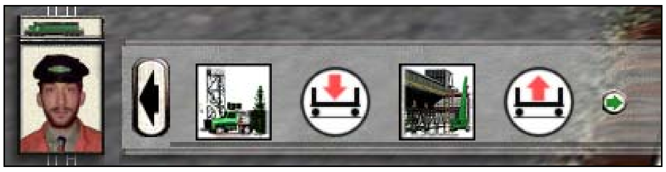
Note: A locomotive can only have one driver assigned to it at a time. Although it is possible to have multiple locomotives in a train consist, each with their own driver, there is also the potential for the different drivers to have conflicting commands and it is best to avoid such a situation.

The next property along form the locomotive is the name of this particular driver instance. This is the object name of the driver that is created based on the selected asset. The concept of a driver character being created with a name from a driver asset is equivalent to having a vehicle being created from a vehicle asset. The default name for a driver will be derived from the asset it is based on but it can be changed to whatever is desired.

The remove property at the far right of the panel removes that entire panel from this rule's list. This means driver settings and the commands list will be lost.

The driver command bar across the bottom of the panel is the command list that the driver will start performing as soon as this rule is started as the session is launched. The commands in this list are built in the properties window in the same way they would be from a Driver session as described in Section 9.4 of the TRS2006 Expanded Manual.

When the session is launched, you will find that each driver has a list of commands fully configured and ready to be run through as *Screenshot 5.14* shows.

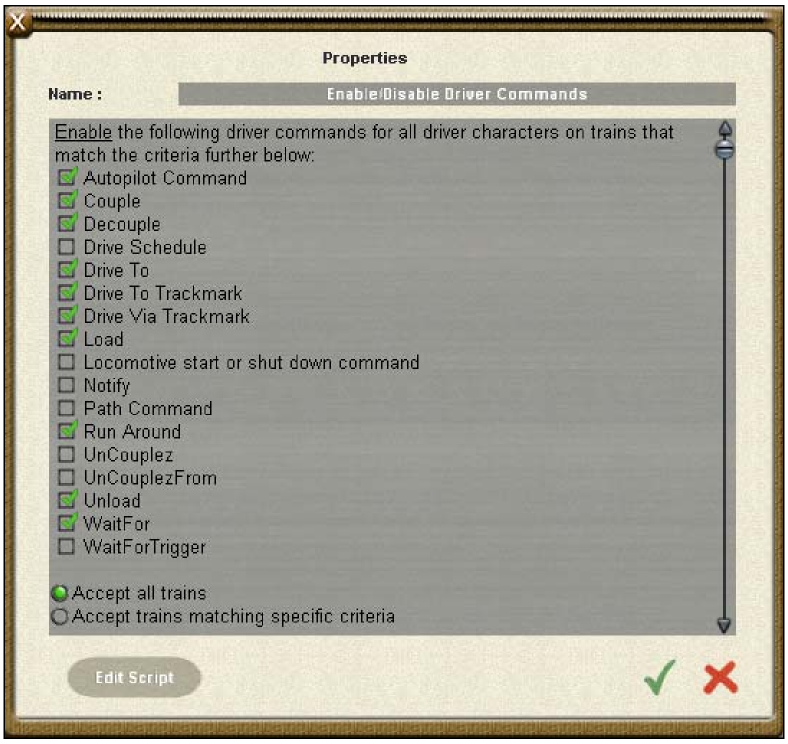

Screenshot 5.14**:** Driver command bar with commands the *Driver Setup* rule has assigned to the driver.

By giving a driver a pre-defined commands list, you can easily set a driver up to keep itself busy such that it can go about the route without the need for the user to interfere with it. This can be useful for populating a route with trains as dynamic scenery to make a session more interesting for example.

Note: It is possible to remove *Driver Setup* from the session's rule list entirely but this has several side effects. The main problem will be that drivers cannot be used in that session and none of the interface components that relate to driver characters will be available. The other consequence is that the camera may not be focused on a train. This however can be worked around with by using the *Set Camera* rule to target the camera to a specific locomotive.

Enable/Disable Driver Commands

This rule enables or disables the selected driver commands allowing specific driver commands to be removed/added to a driver character's list. A checkbox list of all available commands is provided to be enabled or disabled as shown in *Screenshot 5.15*.


Properties

Name : Enable/Disable Driver Commands

Enable the following driver commands for all driver characters on trains that match the criteria further below:

- [✓] Autopilot Command
- [✓] Couple
- [✓] Decouple
- [ ] Drive Schedule
- [✓] Drive To
- [✓] Drive To Trackmark
- [✓] Drive Via Trackmark
- [✓] Load
- [ ] Locomotive start or shut down command
- [ ] Notify
- [ ] Path Command
- [✓] Run Around
- [ ] UnCouplez
- [ ] UnCouplezFrom
- [✓] Unload
- [✓] WaitFor
- [ ] WaitForTrigger

[✓ Accept all trains]
Accept trains matching specific criteria

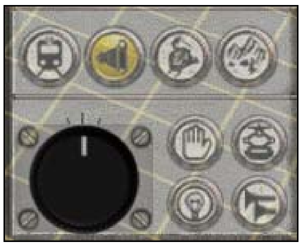
***Screenshot 5.15*:** Properties window of the *Enable/Disable Driver Commands* rule.

The *Enable/Disable Driver Commands* rule does not wait for anything or execute any child rules. Once started, it will enable or disable the driver commands as set and complete straight away. It is a 'use once' *Task* rule that needs to be reset if more than one execution is required.

Note: Even though the *Driver Command* rule is also capable of enabling/disabling access to driver commands, it is still a different rule with different purposes. The main difference is that *Driver Command* does not have a train filter so its settings will apply to all driver characters. The other major point to note is that *Driver Command* is meant to be used as an initialization rule run as the session starts while *Enable/Disable Driver Commands* is intended to be run further along into the session.

Enable/Disable HUD Icons

This rule enables or disables the selected HUD panel icons in Driver. A disabled icon will not respond to being clicked on and will appear grayed out as shown in *Screenshot 5.16*.

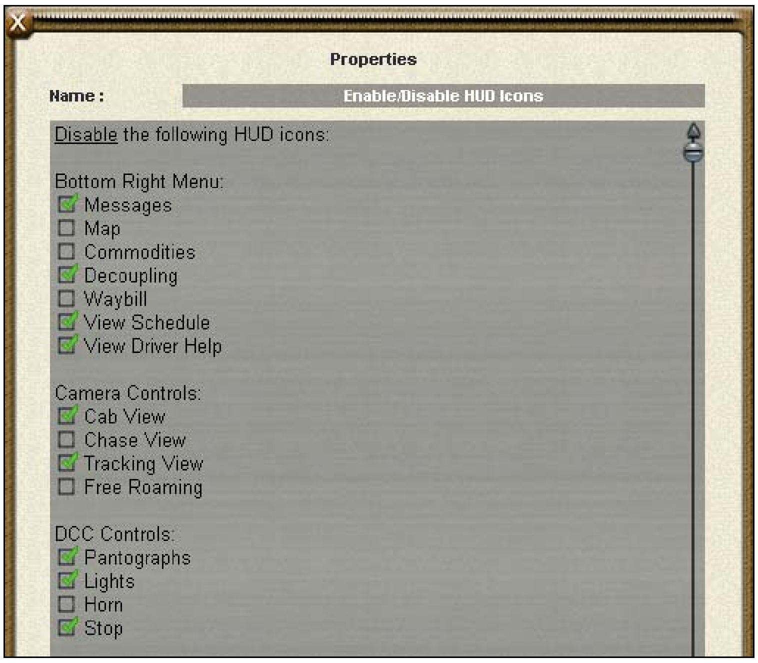


***Screenshot 5.16*:** Camera and DCC Control HUD panels with disabled icons.

The HUD panel icons that this rule can disable grouped by HUD are:

- *Bottom Right Menu:* Messages, Map, Commodities, Decoupling, Waybill, View Schedule & View Driver Help
- *Camera Controls:* Cab View, Chase View, Tracking View & Free Roaming
- *DCC Controls:* Pantographs, Lights, Horn & Stop

All icons that can be enabled/disabled are in the check box lists of this rule's properties window. Checked items will either be enabled or disabled, depending on this rule's settings.



Properties

Name : Enable/Disable HUD Icons

Disable the following HUD icons:

Bottom Right Menu:
- ✓ Messages
- ☐ Map
- ☐ Commodities
- ✓ Decoupling
- ☐ Waybill
- ✓ View Schedule
- ✓ View Driver Help

Camera Controls:
- ✓ Cab View
- ☐ Chase View
- ✓ Tracking View
- ☐ Free Roaming

DCC Controls:
- ✓ Pantographs
- ✓ Lights
- ☐ Horn
- ✓ Stop

***Screenshot 5.17*:** Properties window of the *Enable/Disable HUD* Icons rule.

The *Enable/Disable HUD Icons* rule does not wait for anything or execute any child rules. Once started, it will enable or disable the HUD icons as set and complete straight away. It is a 'use once' task rule that needs to be reset if more than one execution is required.

Note: If a HUD icon is disabled, the *Disabled HUD Icon Notification* rule can be used to detect clicks on it. The *Wait for Click on HUD Icon* rule cannot detect a click on a disabled icon.

End Scenario

This rule will terminate the current session the moment it is executed. There are no configuration properties for this rule. Other rules still running will be suspended and won't get a chance to complete if they haven't already done so.

Note: Caution is advised when using the *End Scenario* rule. Don't use it as a top-level rule executed when the session starts and when using it, always be careful – it exits the session without the user getting a chance to save!

Flash HUD Icon

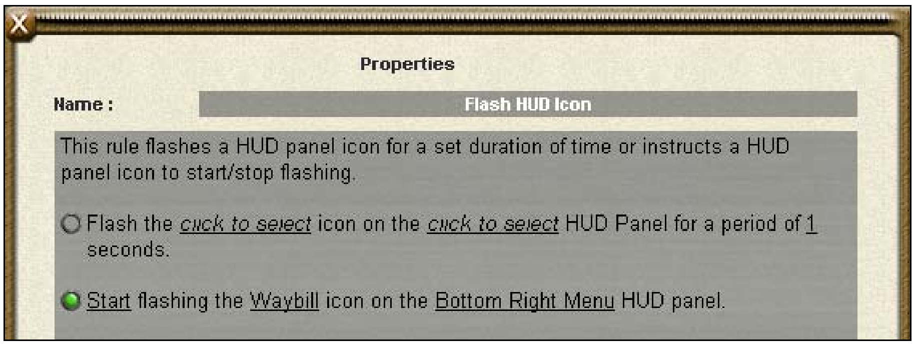
This rule allows the flashing of a specific icon on a HUD panel in the Driver interface. The capability to flash an icon either for a set duration of time in seconds or to switch flashing on/off is provided.

The HUD panel icons this rule can flash grouped by HUD panel are:

- *Bottom Right Menu:* Messages, Map, Commodities, Decoupling, Waybill, View Schedule & View Driver Help
- *Camera Controls:* Cab View, Chase View, Tracking View & Free Roaming
- *DCC Controls:* Pantographs, Lights, Horn & Stop
- *Driver Main Menu:* Exit, Pause, Save, Performance Settings, Imperial/Metric, Find & On Screen Help

Note: The Cabin Controls and Speed and Time Display HUD panels are not included because they don't have any icons on them.

Whether the rule flashes an icon for a set duration of time or switches an icon's flashing on or off is determined by the radio button selection in the rule's properties window as shown in *Screenshot 5.18*.



Properties

Name : Flash HUD Icon

This rule flashes a HUD panel icon for a set duration of time or instructs a HUD panel icon to start/stop flashing.

○ Flash the click to select icon on the click to select HUD Panel for a period of 1 seconds.

● Start flashing the Waybill icon on the Bottom Right Menu HUD panel.

Screenshot 5.18**:** Properties window of the *Flash HUD Icon* rule.

When flashing an icon for a set duration, the time can only be specified in whole seconds. The flashing of the icon will occur the moment the rule is executed if possible.

The Start/Stop option allows an icon to be switched on to flash indefinitely. Once an icon has started to flash through this option, its flashing can only be stopped by using another *Flash HUD Icon* rule instance to stop it. It is the responsibility of the session creator to ensure this is done properly if the flashing is to be stopped.

The Stop option instructs the Driver interface to stop flashing the specified icon. If that particular icon is not flashing or something else happens to be flashing, this is no real problem as the Driver interface will simply ignore the request.

If the panel the icon is on isn't visible, the request to flash it will be ignored. In the case of trying to flash a DCC panel icon when in cabin mode for instance, the request will be ignored because the DCC panel doesn't exist.

In both modes, if something is already flashing (either because of another *Flash HUD Panel* rule or *Flash HUD Icon*), then the request this rule sends to the Driver interface will be ignored with a message printed to the Trainz log file.

Once the *Flash HUD Icon* rule has sent its request to the Driver interface, it completes straight away. It doesn't wait for the timed flash to finish and won't execute any child rules. It is a 'use once' *Task* rule that needs to be reset if more than once execution is required.

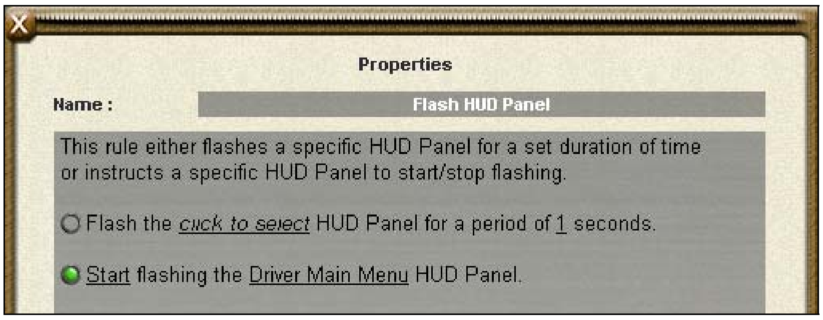
Flash HUD Panel

This rule allows the flashing of a HUD panel in the Driver interface. The capability to flash a HUD panel either for a set duration of time in seconds or to switch flashing on/off is provided.

The HUD panels this rule can flash are:

- Bottom Right Menu
- Cabin Controls
- Camera Controls
- DCC Controls
- Driver Main Menu
- Speed and Time Display

Whether the rule flashes a panel for a set duration of time or switches an panel's flashing on or off is determined by the radio button selection in the rule's properties window as shown in *Screenshot 5.19*.


Properties

Name : Flash HUD Panel

This rule either flashes a specific HUD Panel for a set duration of time or instructs a specific HUD Panel to start/stop flashing.

○ Flash the click to select HUD Panel for a period of 1 seconds.

● Start flashing the Driver Main Menu HUD Panel.

***Screenshot 5.19*:** Properties window of the *Flash HUD Panel* rule.

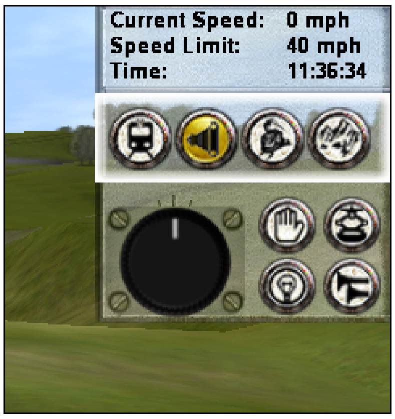
When flashing a panel for a set duration, the time can only be specified in whole seconds. The flashing of the panel will occur the moment the rule is executed if possible.

The Start/Stop option allows a panel to be switched on to flash indefinitely. Once a panel has started to flash through this option, its flashing can only be stopped by using another *Flash HUD Panel* rule instance to stop it. It is the responsibility of the session creator to ensure this is done properly if the flashing is to be stopped.

Note: A flashing HUD panel can be difficult to use, so from a usability perspective, leaving a panel to flash indefinitely without having a plan to stop it somehow is not recommended!

The Stop option instructs the Driver interface to stop flashing the selected panel. If that particular HUD panel is not flashing or something else happens to be flashing, this is no problem because the Driver interface will simply ignore the request.

If a panel is not visible, then the request to flash it will be ignored. In the case of trying to flash the Cabin Controls panel when in DCC mode for instance, the request will be ignored as well because the panel doesn't exist.



***Screenshot 5.20*:** Speed and Time HUD panel being flashed.

In both modes, if something is already flashing (either because of another *Flash HUD Panel* rule or *Flash HUD Icon*), then the request this rule sends to the Driver interface will be ignored with a message printed to the Trainz log file.

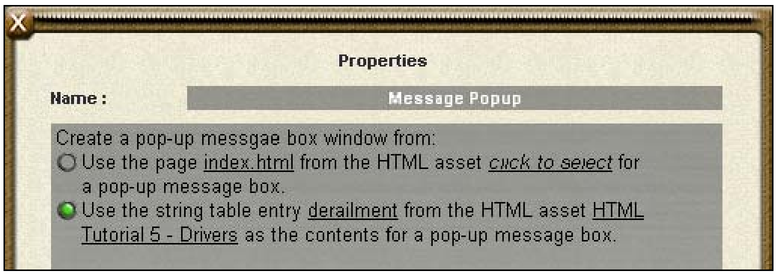
Once the *Flash HUD Panel* rule has sent its request to the Driver interface, it completes straight away. It doesn't wait for the timed flash to finish and won't execute any child rules. It is a 'use once' *Task* rule that needs to be reset if more than once execution is required.

Message Popup

This rule is used to display a message in a popup window that needs to be acknowledged by the user. When executed, this rule will create a browser window with the message content and an acknowledgement link that the user must click to close the browser and end the rule.

The browser created by *Message Popup* is of limited size and suited to small simple messages. It is not meant to be used to displaying large amounts of information like *Display HTML* or *Display HTML Pages* are.

In the *Message Popup* properties, there are two different ways to provide the content needed for the message window, either through a HTML asset and page or a string table entry from a HTML asset. *Screenshot 5.21* shows how radio buttons are provided to select which way the rule is going to get content for the browser window.


Properties

Name : Message Popup

Create a pop-up messgae box window from:

○ Use the page index.html from the HTML asset click to select for a pop-up message box.

● Use the string table entry derailment from the HTML asset HTML Tutorial 5 - Drivers as the contents for a pop-up message box.

Screenshot 5.21**:** Properties window of the *Message Popup* rule.

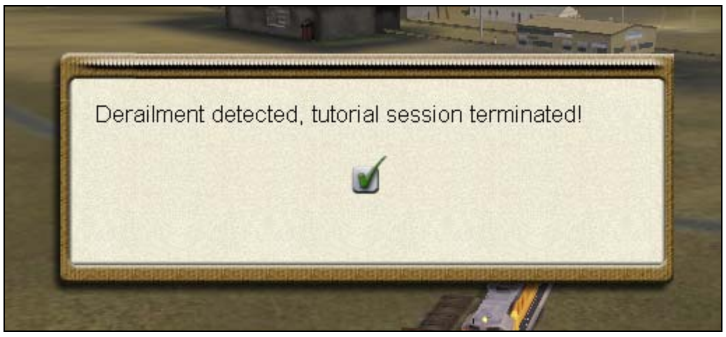
If the HTML page option is selected, a source HTML asset and HTML file in that asset's directory needs to be set. This is identical to the way a page is set in the *Display HTML* rule.

To work with Message Popup, the HTML page needs a 'done/close' link. This is the link that the rule waits for to close the browser window and complete. The URL for this link must be "**live://pages/messagepopup/done**", for example:

```
<a href='live://pages/messagepopup/done'>
  <img src=closewindow.tga>
</a>
```

There can be more than one link in the page and both text and images can be wrapped up in the link. It is up to the HTML asset author how they want to present the done/close link to the user.

If the string table option is being used, a string table entry and the HTML asset that hosts that entry must be specified. There isn't any real restriction on string content apart from the recommendation to keep it small (1-2 sentences) as the browser window used is quite small as *Screenshot 5.22* shows.


Derailment detected, tutorial session terminated!

Screenshot 5.22**:** Window created by *Message Popup* rule with text string and built-in checkmark for the close/done link.

There is no need to specify a 'done/close' link in the string table text as the *Message Popup* rule will insert a default close/done button at the bottom of the browser when using string table text.

No formatting tags are applied to the string table text, so it will appear white by default unless it is formatted. For example, to use black text like *Screenshot 5.22* has, the string table entry would look like this:

```
<font color=#000000>Derailment detected, tutorial session
 terminated!</font>
```

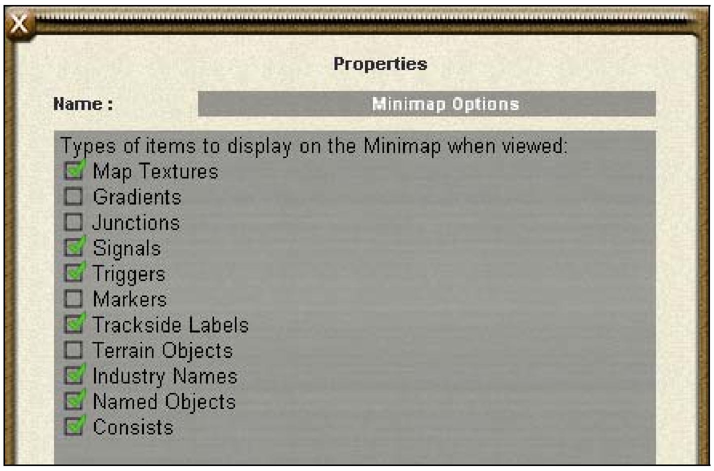
Other formatting tags like **<i>** (italics) and **<b>** (bold) can also be used in this way.

Once the *Message Popup* rule has detected the done/close link being clicked, it will close the browser window and complete itself. It will not execute any child rules and is a 'use once' *Task* rule that needs to be reset if the page is to be displayed again.

Minimap Options

This rule specifies what types of items are to be visible/invisible on the Minimap window in Driver.

All items that can be switched on/off are in the check box list of this rule's properties window as shown in *Screenshot 5.23*. Checked items will be visible and unchecked items will remain hidden.


Properties

Name : Minimap Options

Types of items to display on the Minimap when viewed:
- [✓] Map Textures
- [ ] Gradients
- [ ] Junctions
- [✓] Signals
- [✓] Triggers
- [ ] Markers
- [✓] Trackside Labels
- [ ] Terrain Objects
- [✓] Industry Names
- [✓] Named Objects
- [✓] Consists

***Screenshot 5.23*:** Properties window of the *Minimap Options* rule.

The items on the Minimap that this rule allows you to show/hide are:
- Map Textures
- Gradients
- Junctions
- Signals
- Triggers
- Markers
- Trackside Labels
- Terrain Objects
- Industry Names
- Named Objects
- Consists

The *Minimap Options* rule does not wait for anything or execute any child rules. Once started, it will set the options for the Minimap and complete straight away. This rule is a 'use once' *Task* rule that needs to be reset if more than one execution is required.

Note: This rule's settings do not stop the user from enabling/disabling the types of items they can see on the Minimap later on. It is only intended to set the Minimap window to a certain state.

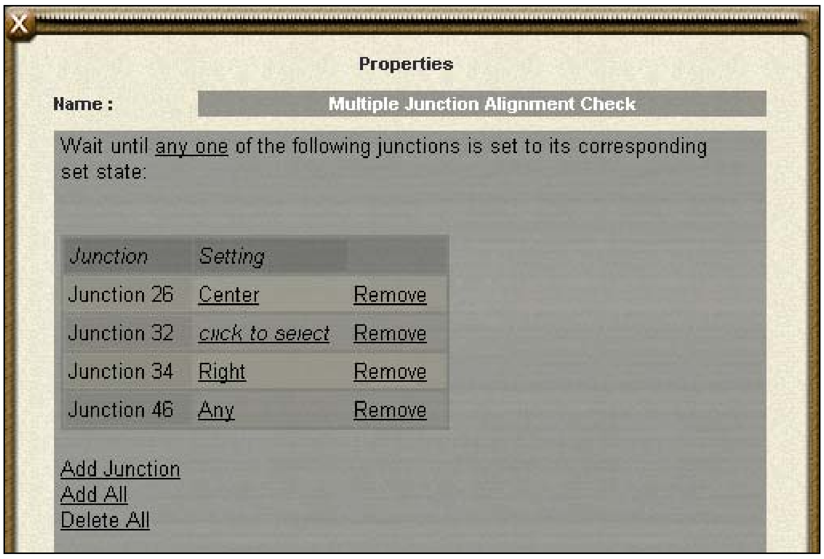
Multiple Junction Alignment Check

This rule is used to wait for one or more junctions to be changed to a specific state. Once the junction changes have been performed, the child rules will be started. When all child rules have completed, the *Multiple Junction Alignment Check* rule will complete itself.

Multiple Junction Alignment Check supports two different modes for its conditions to be satisfied:

- any one of the junctions is changed to its required state; or
- all of the junctions have been changed to their required state.

Click on the 'any one' (default) or 'all' links to switch between these modes at the top of the properties window.



Properties

Name : Multiple Junction Alignment Check

Wait until any one of the following junctions is set to its corresponding set state:

Junction	Setting	
Junction 26	Center	Remove
Junction 32	click to select	Remove
Junction 34	Right	Remove
Junction 46	Any	Remove

Add Junction
Add All
Delete All

Screenshot 5.24: Properties window of the *Multiple Junction Alignment Check* rule.

The properties window of this rule allows a list of junctions to be maintained as seen in *Screenshot 5.24*. The 'Add Junction' link provides a list of all junctions on the current route that are not already in the list and will append the selected junction to the list.

'Add All' adds all of the junctions on the route to the list. This is a handy way of locking every single junction on a route if there happen to be several hundred. The 'Delete All' link at the very bottom will delete all junctions in the list.

Note: Be careful when using the 'Delete All' link as your list of tediously configured junctions could vanish. Even if this does happen by accident, saving often so you can revert back without too much trauma is advised.

When monitoring a junction, this rule can be set to wait for that junction to reach several different states. The setting of a newly added junction will not be anything, hence the '*click to select*' link. Clicking this link will allow one of 4 possible junction states to be selected:

- Left
- Center
- Right
- Any

In the case of the Any option, this will mean the rule will be satisfied that the junction is changed, regardless of what position it is changed to – as long as the junction is actually changed to something. If the required state for a junction is not set to any of the options at all, then this rule will be unable to fulfill its completion requirements with that junction.

Once the required junction settings have been met, the child rules will be started. When all of the child rules have completed, *Multiple Junction Alignment Check* will complete itself. It is a 'use-once' *Event* rule that once complete, will sit idle and need to be reset if another run is required.

When the child rules are running, *Multiple Junction Alignment Check* will no longer monitor the junctions, so it is quite feasible the rule's conditions may no longer meet the set requirements before the child rules have completed running.

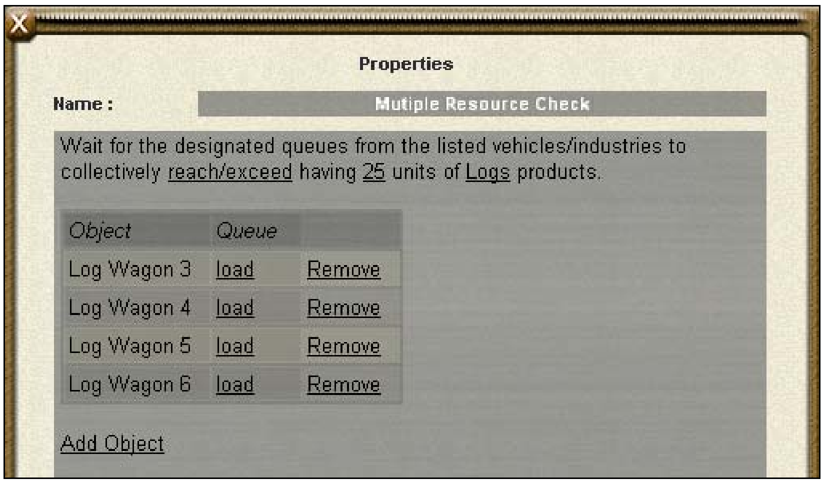
Tip: The state of one or more junctions can be set with the *Set Junctions* rule.

Multiple Resource Check

The *Multiple Resource Check* rule waits for the selected queues from multiple industries and vehicles to collectively reach a set total of a particular product type. Once the conditions are met, *Multiple Resource Check* will run all of the child rules and complete itself once all the child rules have finished.

This rule is similar to *Resource Check* with the main difference being that it allows multiple target objects and queues to be watched. This means that if multiple vehicles in a train are needed to reach a certain level for the train to be full, just one *Multiple Resource Check* rule can be used instead of having to add a *Resource Check* rule for each individual vehicle.

The properties window of this rule allows a list of objects and accompanying queue to be maintained as seen in *Screenshot 5.25*. The 'Add Object' link allows new industries or vehicles to be added from a list while the 'Remove' link adjacent to each object removes that object from the list.


Properties

Name : Mutiple Resource Check

Wait for the designated queues from the listed vehicles/industries to collectively reach/exceed having 25 units of Logs products.

Object	Queue	
Log Wagon 3	load	Remove
Log Wagon 4	load	Remove
Log Wagon 5	load	Remove
Log Wagon 6	load	Remove

Add Object

Screenshot 5.25: Properties window of the *Multiple Resource Check* rule.

Above the objects list at the top of the properties window is the resource requirements needed for this rule to activate and run the child rules. The rule conditions can be set for the queues to either collectively 'reach/exceed' or 'fall below' a set amount of units of a particular product.

The 'click to select' link refers to the type of product this rule is looking out for. As this rule is meant to be versatile, the list of products provided won't be restricted to what the queues in the list can or can't hold. It is assumed that the session author has done their homework on what types of products the queues can carry.

Tip: The properties window of a vehicle or industry in Surveyor can be used to determine the types and quantities of products a queue can carry.

Once the set conditions have been met, the child rules will be started. When all of the child rules have completed, *Multiple Resource Check* will complete itself. It is a 'use-once' *Event* rule that once complete, will sit idle. It will need to be reset if another run is required.

When the child rules are running, *Multiple Resource Check* will no longer monitor the target objects & queues so it is quite feasible for these queues to no longer meet the set requirements before the child rules have completed running.

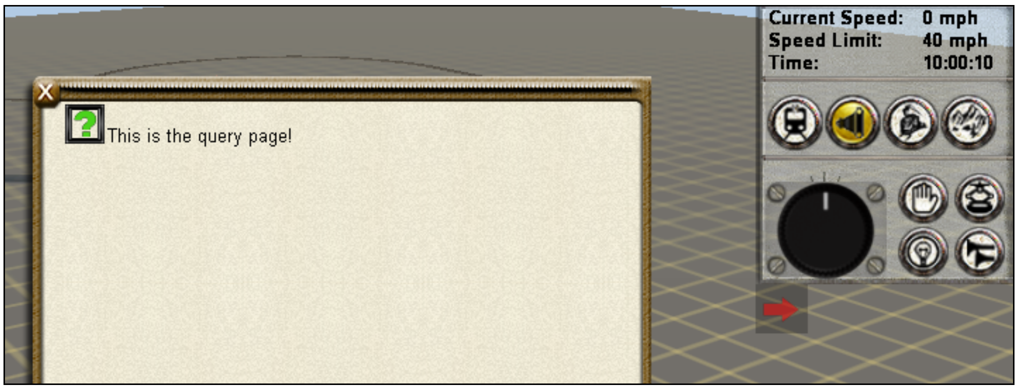
Objective Clear

This rule clears all objectives from the Driver interface. Once started, it will clear the objectives and complete itself straight away without waiting for anything or running any child rules. Objective Clear is a 'use once' *Task* rule that needs to be reset if more than one run of it is required.

Note: For details on objectives, see *Objective New*.

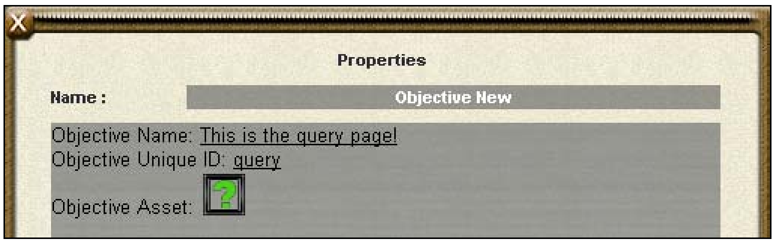
Objective New

This rule is used to create a new objective. An objective is simply a string of text and an icon in a window accessed in Driver by clicking on the red objective arrow. *Screenshot 5.26* shows an objective window opened by clicking on the red arrow icon located below the DCC Controls HUD panel.



***Screenshot 5.26*:** An Objective window in Driver opened by clicking on the red arrow.

The properties window of this rule has 3 properties that define the objective. The name is the text that appears in the objective window and the icon is the picture that appears adjacent to the text. The icon is selected from a list of existing icon assets and by selecting an asset, that icon can be previewed in the properties window.


Properties

Name : Objective New

Objective Name: This is the query page!
Objective Unique ID: query
Objective Asset:

***Screenshot 5.27*:** Properties window of the *Objective New* rule.

Along with the objective name/text and icon, there is also the unique ID. The unique ID is a reference name that is unique to that particular objective. This name will be needed when the objective is being removed through the Objective Remove rule.

This rule will create the objective and then complete itself straight away without waiting for anything or running any child rules. It is a 'use once' Task rule that needs to be reset if more than one run of it is required.

Once created, an objective will be managed by the Driver interface and it can be accessed multiple times as the user desires. To remove the objective, the *Objective Remove* rule must be used. Alternatively, all objectives can be removed at once with the *Objective Clear* rule.

Objective Remove

This rule removes the named objective from the objective panel. There is only one property and that is the name of the objective being removed. For this rule to work, the objective must have been created previously through the *Objective New* rule.

When started, *Objective Remove* will remove the named objective from the Driver interface if it exists and will then complete itself without waiting for anything or running any child rules. It is a 'use once' Task rule that needs to be reset if more than one run of it is required.

Note: For details on objectives, see *Objective New* rule’s description.

Online Chat

This rule is used to enable access to iTrainz Chat when running the session in Driver. Once started the *Online Chat* rule creates the Online Chat HUD panel that will be present for the rest of the session. This rule does not run any child rules and has an ongoing role in running the chat panel so it won’t be completing.

More information about using iTrainz Chat can be found in Chapter 15 of the TRS2006 Expanded Manual.

Ordered List

The *Ordered List* rule is a *List* rule that does not do anything directly itself, but rather acts as a way of controlling if and when a child rule is executed.

When started this particular rule executes the child rules one after the other. What makes *Ordered List* unique is that it will start the first child rule, wait for the first child rule to complete and then start the second child rule. It will progressively do this until it has gone through all child rules one after the other. As a result of this linear child rule execution, there will never be more than one child rule running at the same time.

Going through the child rules list and executing each one in order is referred to as the 'list cycle'. The final child rule must be complete for this list cycle to complete. This is not to be confused with the 'start and complete' cycle of a rule.

Once this list cycle of child rules is complete, the *Ordered List* rule will either complete itself or restart the cycle again. Whether the list cycle is restarted or the rule completes depends on its settings.

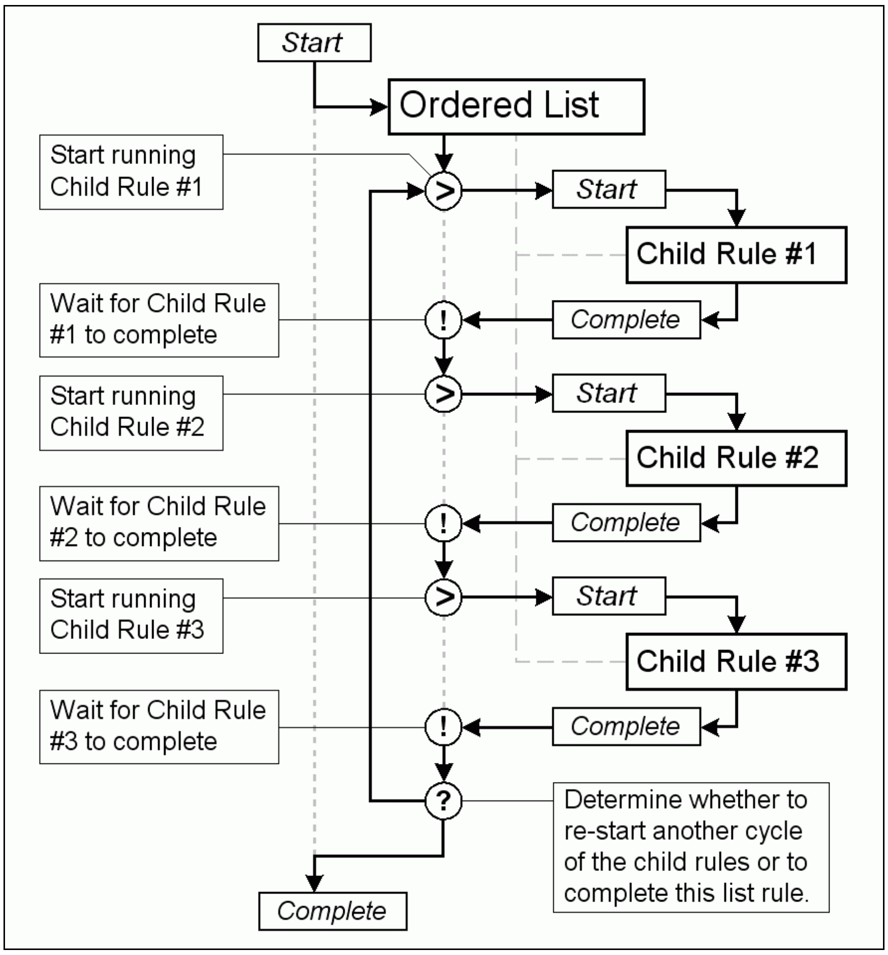

Start
Ordered List
Start running Child Rule #1
Start
Child Rule #1
Wait for Child Rule #1 to complete
Complete
Start running Child Rule #2
Start
Child Rule #2
Wait for Child Rule #2 to complete
Complete
Start running Child Rule #3
Start
Child Rule #3
Wait for Child Rule #3 to complete
Complete
Determine whether to re-start another cycle of the child rules or to complete this list rule.
Complete

Diagram 5.2: Flow chart showing the cycle of an *Ordered List* rule.

In *Diagram 5.2* above, the life cycle of an *Ordered List* rule is shown. What happens once the cycle completes depends on the rule's property settings, of which there are two options available:

- Run the child list cycle a set amount of cycles before completing
- Run the child list cycle indefinitely

If running the cycle for a set amount of times, *Ordered List* completes itself once it has finished the child list cycle X amount of times where X is set in the rule's properties configuration.

The default configuration for this rule is to run for 1 cycle only and complete, so the child rules will only be executed through once and this rule will then complete itself without running through another cycle.

The alternative to running through a set amount of cycles is to keep running through the child list cycle indefinitely. This means the *Ordered List* rule will always re-start the cycle again and again and never complete itself. However that does not mean an Ordered List rule running indefinitely can not be stopped as Trainz or a parent rule can pause/stop such a cycle as needed.

As an example, an ideal use for *Ordered List* would be when a particular rule (*Rule B*) has a dependency on another rule (*Rule A*) being complete before it can do its job. By placing both *Rule A* and *Rule B* as child rules of an *Ordered List* rule, it is possible to ensure *Rule B* is only run once *Rule A* has completed (assuming *Rule A* completes itself at an appropriate time).

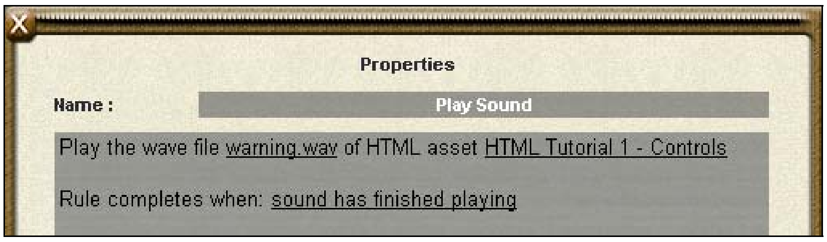
Tip: This is one of several list rules for handling child rules in a specific way. Other list rules include *Progressive List*, *Random List*, *Reset List* and *Simultaneous List*.

Play Sound

This rule plays the designated **.wav** sound file when started. 3D positional sounds are not supported by this rule, it simply plays the audio file 'as is'.

For Trainz to be able to play a **.wav** file, it needs the file name and the location where it can find the file. Currently *Play Sound* only supports playing sound files from the directory of a HTML asset. The rule's properties interface allows the host asset to be selected from a list while the file name must be entered.

To function properly, *Play Sound* is dependent on the user entering a valid sound file for the chosen HTML asset. If not, the rule's script will have an exception/crash causing an interruption to the session.


Properties

Name : Play Sound

Play the wave file warning.wav of HTML asset HTML Tutorial 1 - Controls

Rule completes when: sound has finished playing

***Screenshot 5.28*:** Properties window of *Play Sound* rule.

Note: When entering the file name, do not use the .wav extension as Trainz does not allow the full stop ('.') character to be entered in a text entry box. Instead, enter the file name without the ".wav" extension.

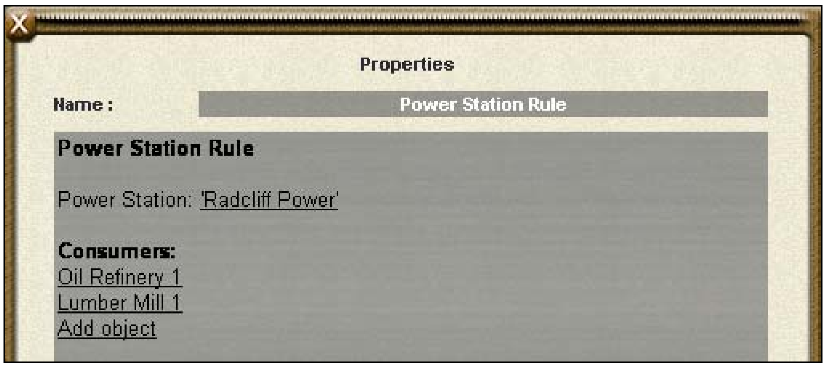
Depending on how it is set up, this rule will complete itself either the moment the sound starts to play or once the sound has completed playing. *Play Sound* does not run any child rules.

Power Station Rule

The *Power Station* rule continuously monitors a selected power station and switches dependent industries from a list on or off depending on whether the power station's coal consuming process is running or not.

This rule allows the creation of a virtual power grid such that the functioning of other industries is dependent on the power station. From here a session can be created where the user has to keep trains feeding the power station coal so other industries actually run.

The first property at the top of the properties window is the target power station that this rule will monitor. When the power station property is clicked (it is blank and '*click to select*' by default), a list of all industries on the route is provided for selection, but only power station industries made by Auran are guaranteed to work with this rule, so make sure an industry based on the "Powerstation" or "Powerstation Basic" assets is selected.


Properties

Name : Power Station Rule

Power Station Rule

Power Station: 'Radcliff Power'

Consumers:
Oil Refinery 1
Lumber Mill 1
Add object

***Screenshot 5.29*:** Properties window of the *Power Station* rule.

Below the power station property is a list of consumer industries. This list is empty be default but the 'Add Object' link allows a new industry to be added. An industry can be removed from the list by clicking on its name.

Although any industry can be added to the list, only the Auran industries written to work with this rule are guaranteed to work. These industries include "Airport", "Airport Basic", "Coal Mine", "Coal Mine Basic", "Container Station", "Container Station Basic", "Forestry", "Forestry Basic", "Hyde Pulp Mill", "Lumber Mill", "Lumber Mill Basic", "Multiple Industry", "Multiple Industry basic", "Multiple Industry New", "Oil Field", "Oil Field Basic", "Oil Refinery", "Oil Refinery Basic", "Powerstation", "Powerstation Basic", "Seaport", "Seaport Basic" and "Steam Filling Station". Other industries can be written to work with this rule however. How a particular industry responds to lack of power varies from industry to industry.

Once started, this rule will monitor the power station and send the messages to the consumer industries as needed whenever the power station's coal consumption process starts or stops. It does not finish and runs in a continuous cycle watching the power station and sending signals to the consuming industries without ever completing itself or running any child rules.

Progressive List

The *Progressive List* rule is one of the *List* rules that does not do anything directly itself, but rather acts as a way of controlling if and when a child rule is executed.

When started for the first time, this particular rule will run the first child rule. After the first child rule has completed, it will then complete itself. In order to run the second child rule, the *Progressive List* rule must be started again by a parent rule. This continues all the way through until the last rule is reached.

This is a similar sequential behavior to that of the *Ordered List* rule except here, the *Progressive List* rule needs to be started again to progress along to the next child rule. *Diagram 5.3* shows this progressive child rule execution:

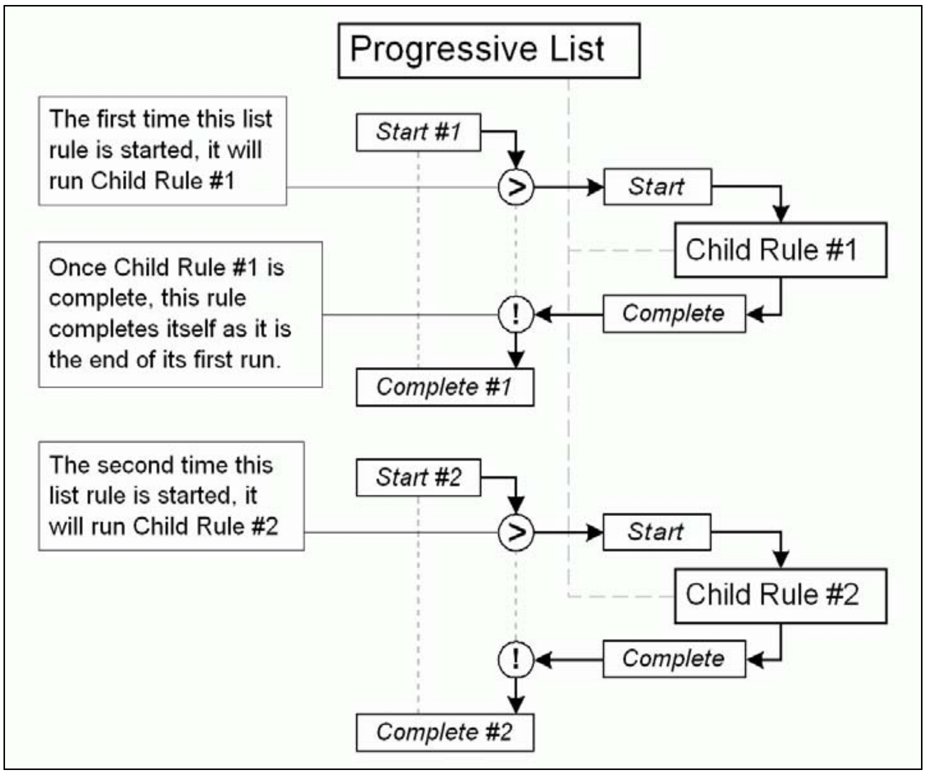

Progressive List
The first time this list rule is started, it will run Child Rule #1
Start #1
Start
Child Rule #1
Once Child Rule #1 is complete, this rule completes itself as it is the end of its first run.
Complete
Complete #1
The second time this list rule is started, it will run Child Rule #2
Start #2
Start
Child Rule #2
Complete
Complete #2

Diagram 5.3: Flow chart showing the cycle of a *Progressive List* rule.

After a *Progressive List* rule has done a complete cycle of its child rules, what happens when it is started again after completing its last rule depends on its completion setting. Three possible options for this setting are:

- Restart the cycle again with the first child rule.
- Run the last child rule again and every subsequent time this list rule is executed.
- No further child rules will be run. Even further attempts to start this list rule will not have any affect apart from it completing straight away.

The *Progressive List* rule is unique in that it can be executed and completed again and again and it has set behavior patterns for running children when used in such a way.

Tip: This is one of several list rules for handling child rules in a specific way. Other list rules include *Ordered List*, *Random List*, *Reset List* and *Simultaneous List*.

Random List

The *Random List* rule is one of the *List* rules that does not do anything directly itself, but rather acts as a way of controlling if and when a child rule is executed.

When started this particular rule randomly executes one of its child rules. Once the randomly chosen child rule is complete, *Random List* will complete itself. It behaves as a 'use once' *Task* rule that needs to be reset if more than one execution of it is required.

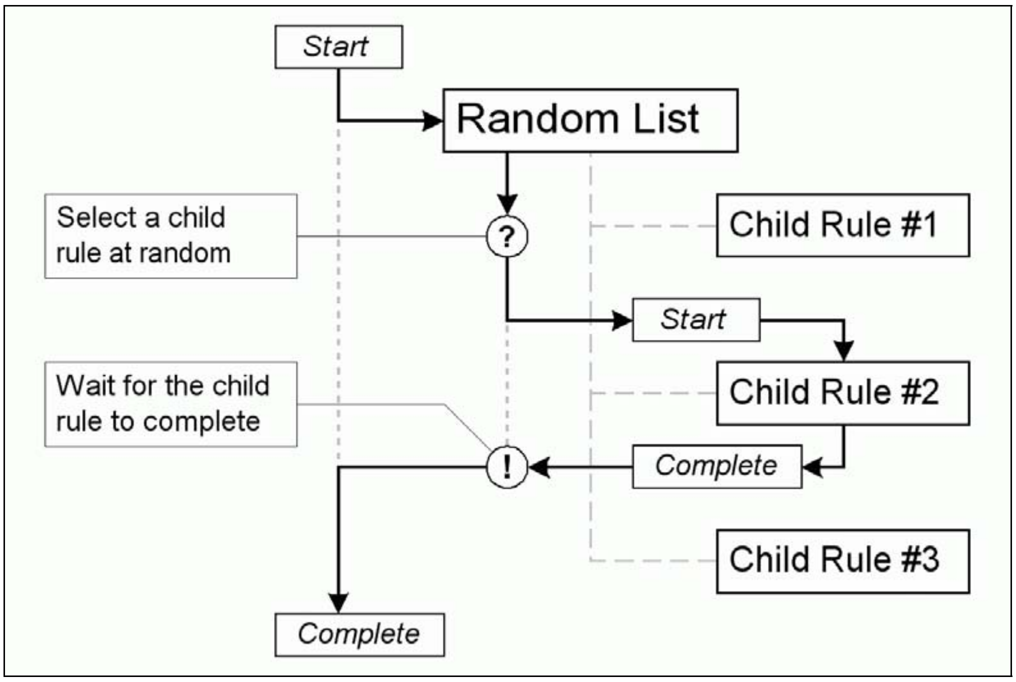


***Diagram 5.4*: Flow chart showing the cycle of a *Random List* rule.

The behavior of *Random List* is fixed and will always behave as described and illustrated above. There are no property options to be set or alternate modes this rule can operate in.

Note: As this rule is random in its nature, it is not a good idea to totally rely on one particular child rule being executed. There is no guarantee a particular child rule may ever be executed.

Tip: This is one of several list rules for handling child rules in a specific way. Other list rules include *Ordered List*, *Progressive List*, *Reset List* and *Simultaneous List*.

Reset List

The *Reset List* rule is a *List* rule that does not do anything directly itself, but rather acts as a way of controlling if and when a child rule is executed. In the case of *Reset List*, it continuously runs and re-runs all of its child rules in continuous cycles.

When started, *Reset List* will start running all child rules simultaneously. It then waits for all of the child rules to complete themselves. Once this has happened, it will reset all of the child rules and start running them again.

This is an indefinite cycle that the *Reset List* rule does for its lifetime. It will never complete itself. *Diagram 5.5* below shows how this rule operates.

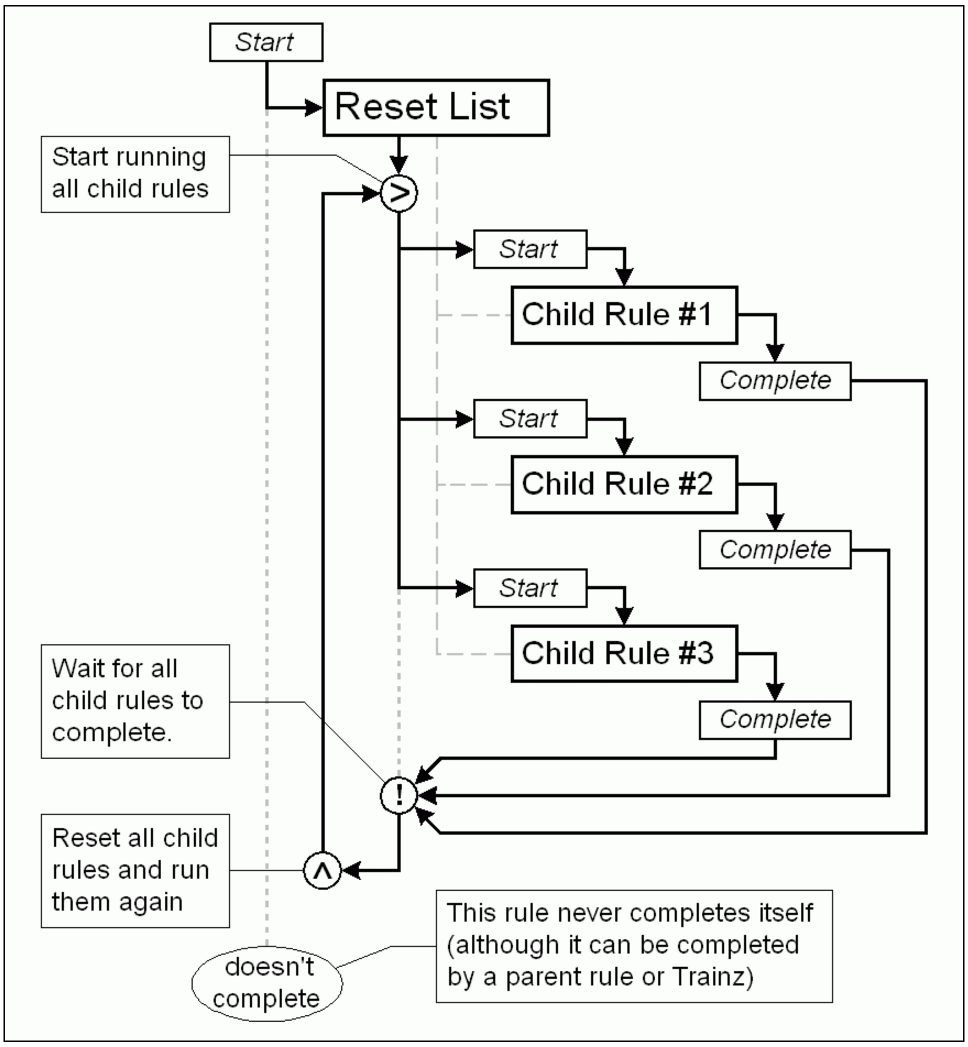

Start
Reset List
Start running all child rules
>
Start
Child Rule #1
Complete
Start
Child Rule #2
Complete
Start
Child Rule #3
Complete
Wait for all child rules to complete.
!
Reset all child rules and run them again
^
This rule never completes itself (although it can be completed by a parent rule or Trainz)
doesn't complete

Diagram 5.5: Flow chart showing the cycle of a *Reset List* rule.

There are no configuration properties for *Reset List*, it only has one pattern of behavior and that is the continuous cycle described above.

When a cycle of child rules is complete, the child rules are all reset before being started again. This means that the child rule is back in a refreshed initial state and can be run from the start like it was when first started. This reset affect applies to all rules, regardless of how they complete.

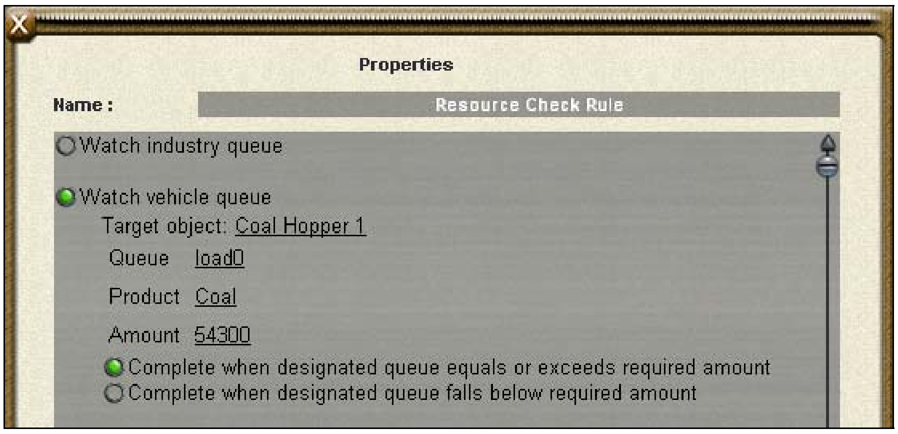
Although *Reset List* will never complete itself when the session is running in Driver, it can still be paused/stopped by a parent rule or Trainz.

Tip: This is one of several list rules for handling child rules in a specific way. Other list rules include *Ordered List*, *Progressive List*, *Random List* and *Simultaneous List*.

Resource Check

This rule waits for a queue on a target industry/vehicle to contain a certain level of a product and then runs any child rules once the conditions have been satisfied.

To configure this rule, the type of object being monitored must be selected by choosing the radio button for either "Watch industry queue" or "Watch vehicle queue". Each of these options will display a sub-section of further configuration options when selected as seen in *Screenshot 5.30*.


Properties

Name : Resource Check Rule

○ Watch industry queue

● Watch vehicle queue
Target object: Coal Hopper 1
Queue load0
Product Coal
Amount 54300
● Complete when designated queue equals or exceeds required amount
○ Complete when designated queue falls below required amount

***Screenshot 5.30*:** Properties window of the *Resource Check* rule.

To select a target object, click on the hyperlink to the right of the 'Target object:' property and a list of available vehicles or industries will appear to select from.

Along with a target object, this rule also needs to know which queue on the target object to monitor as well as the type and quantity of product needed. The 'Queue' property will only allow queues to be selected that exist on the chosen target object and the 'Product' property only allows products that the selected queue is allowed to carry.

The 'Amount' property is a positive whole number that indicates the target amount needed for the rule to complete. This amount is restricted to the capacity of the queue for the currently selected product.

How *Resource Check* completes ultimately depends on the 'Complete when..' option where it can be set to complete either when the queue equals or exceeds the amount (greater than or equal to) or falls below the amount (less than).

When the set conditions have been met, the child rules will be started. Once all of the child rules have completed, *Resource Check* will complete itself. It is a 'use-once' *Event* rule that once complete, will sit idle. It will need to be reset if another run is required.

When the child rules are running, *Resource Check* will no longer monitor the target object & queue so it is quite feasible for the queue to no longer meet the set requirements before the child rules have completed running.

It is possible that a target object and/or queue may not exist when the rule is run. If that is the case, *Resource Check* will simply terminate itself without ever running any child rules.

Tip: *Resource Check* can only monitor one target object at a time. The Multiple Resource Check rule allows multiple industries and vehicles to be collectively monitored for an overall amount.

Note: There is no rule to set the contents of a queue, however a queue can be initialized to contain certain products through the properties window of the host industry or vehicle.

Schedule Rule

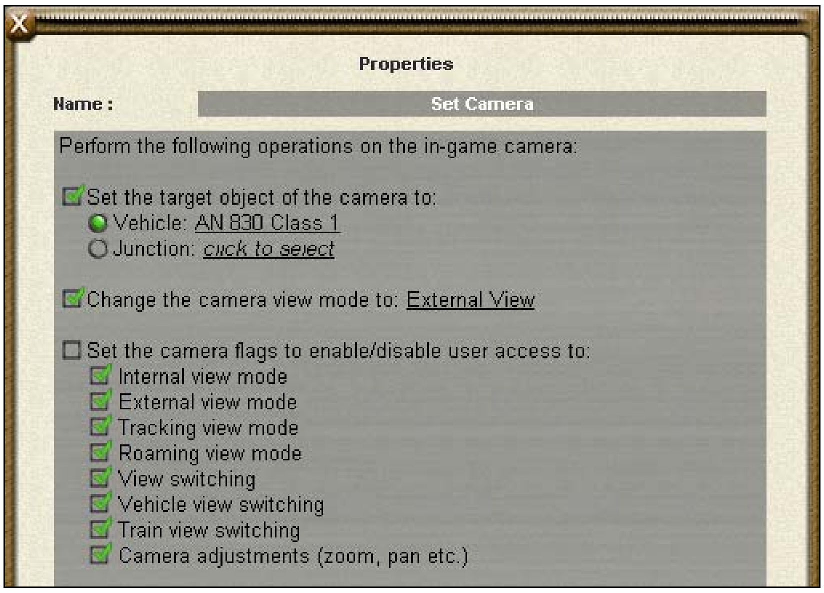
This rule is not documented yet. It is however functional and works.

Set Camera

This rule allows the various parameters of the camera in Driver to be set. There are three different groups of camera settings that this rule supports and they are:

- Set target (vehicle or junction)
- Change view mode
- Set flags

Checkboxes for each camera setting type is provided and if not checked, the rule will not change those particular camera parameters. *Screenshot 5.31* shows how the different groups of settings appear in the properties window.



Name : Set Camera

Perform the following operations on the in-game camera:

✓ Set the target object of the camera to:
- ● Vehicle: AN 830 Class 1
- ○ Junction: *click to select*

✓ Change the camera view mode to: External View

☐ Set the camera flags to enable/disable user access to:
- ✓ Internal view mode
- ✓ External view mode
- ✓ Tracking view mode
- ✓ Roaming view mode
- ✓ View switching
- ✓ Vehicle view switching
- ✓ Train view switching
- ✓ Camera adjustments (zoom, pan etc.)

***Screenshot 5.31*:** Properties window of the *Set Camera* rule.

The 'Set target' option allows the camera to be set to target a specific vehicle or junction. All available vehicles/junctions can be selected from a pop-up list. The initial setting for both of these possible targets is a blank '*click to select*' link.

The change view mode option allows the view mode of the camera to be changed to one of four available view modes:
- Internal (cab)
- External
- Tracking
- Roaming

The view mode setting selected will be applied to the current targeted object that the camera is focused on. If the Set target option is also enabled for this rule, then the chosen camera mode will be applied once the selected vehicle/junction is the camera's currently targeted object.

The Set camera flags option is used to enable/disable various modes and actions on the camera for the user. Camera properties that can be enabled/disabled are:
- Internal view mode
- External view mode
- Tracking view mode
- Roaming view mode
- View switching
- Vehicle view switching
- Train view switching
- Camera adjustments (zoom, pan etc.)

A checkbox is provided for each camera flag. If checked, that property will be enabled for the camera, otherwise will be disabled.

Note: The camera flags were implemented but have not been tested or verified because they did not end up being needed for the original purpose this rule was written for. There is no guarantee they will work so it is advised not to rely on them.

When started, the *Set Camera* rule will go through its properties and apply the selected options to the camera in Driver. Once that has been done, it will complete itself without waiting for anything or running any child rules. *Set Camera* is a 'use once' *Task* rule that needs to be reset if more than one run of it is required.

Note: This is not a totally instantaneous rule that completes straight away. It does have slight time delays (usually less than a second) to allow for a vehicle to actually exist in the Trainz world before being targeted.

Tip: The *Wait for Camera View Mode* rule can be used to wait for the camera to enter a particular mode.

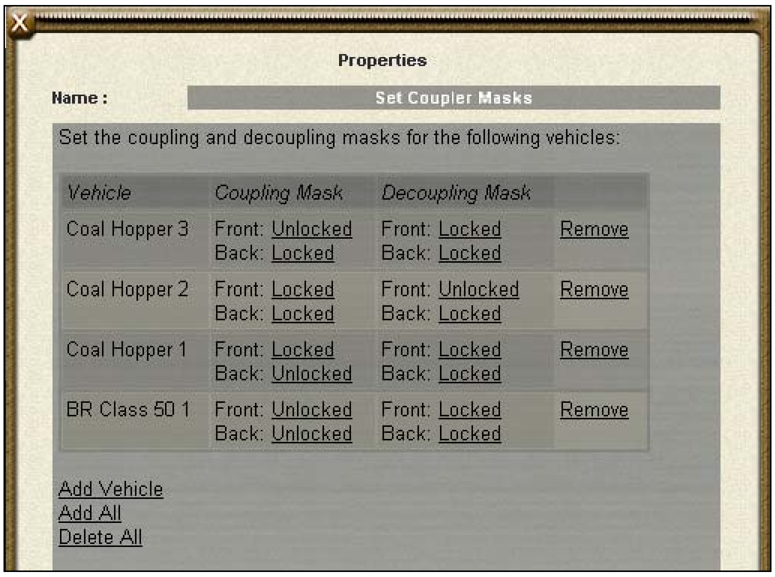
Set Coupler Masks

The Set Coupler Masks rule sets the coupling and decoupling masks of one or more chosen vehicles.

The properties window of this rule allows a list of vehicles to be maintained as seen in *Screenshot 5.32*. The 'Add Vehicle' link provides a list of all vehicles on the current route that are not already in the list and will append the selected vehicle to the list.

'Add All' adds all of the vehicles on the route to the list. The 'Delete All' link at the very bottom will delete all vehicles in the list.

Note: Be careful when using the 'Delete All' link as your list of tediously configured vehicles will vanish!



Properties

Name : Set Coupler Masks

Set the coupling and decoupling masks for the following vehicles:

Vehicle	Coupling Mask	Decoupling Mask	
Coal Hopper 3	Front: Unlocked Back: Locked	Front: Locked Back: Locked	Remove
Coal Hopper 2	Front: Locked Back: Locked	Front: Unlocked Back: Locked	Remove
Coal Hopper 1	Front: Locked Back: Unlocked	Front: Locked Back: Locked	Remove
BR Class 50 1	Front: Unlocked Back: Unlocked	Front: Locked Back: Locked	Remove

Add Vehicle
Add All
Delete All

Screenshot 5.32**:** Properties window of the *Set Coupler Masks* rule.

Each vehicle in the list will have several properties adjacent to it. These are the coupling and decoupling masks of the vehicle. Both of these settings have a lock state for the front and back couplers of the vehicle.

The front of the vehicle, regardless of its direction of travel or orientation relative to the train, will always be the same and this is where the front coupler is considered to be. This means that it is possible for the front coupler to be facing towards the rear of the train if the vehicle is not facing towards the front of the train.

The coupling mask determines whether a coupler can be coupled to. If a coupler's coupling mask is locked, that coupler cannot be coupled to. Applying a coupling mask means the vehicle cannot be shunted/switched.

The decoupling mask determines whether a coupler can be decoupled from the other vehicle's coupler that it is connected to. When locked, the coupler cannot be decoupled. Locking the decoupling masks of all vehicles in a train consist means a train consist cannot be broken up by the user. For example, this rule could be used to ensure the user doesn't break up vehicles that are meant to be permanently coupled up to each other like a steam locomotive and its tender or a TGV set.

Although they both relate to the same set of couplers, coupling masks and decoupling masks are separate and don't affect each other directly. A vehicle with its front and back decoupling masks locked and its front and back coupling masks unlocked can be coupled to, but once coupled, you won't be able to decouple from it.

Note: Be careful when locking the coupling/decoupling masks of vehicles as once they are locked, ***ONLY*** another *Set Coupler Masks* rule instance can be used to unlock them again. By not being careful here, there are potential complications with editing your consists in Surveyor as well when loading and saving sessions in progress.

Once started, *Set Coupling Masks* goes through the list and applies the mask settings as configured. It will then complete itself straight away without running any child rules. This is a 'use once' *Task* rule that will need to be reset before being run again.

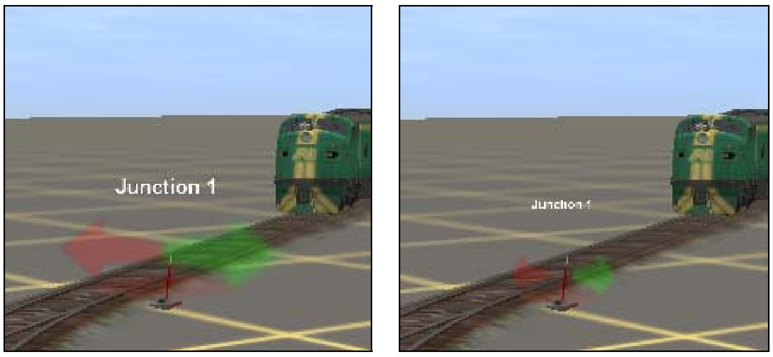
Tip: The *Consist Check* rule can be used to verify the vehicles in a train consist.

Set Helper Icon Size

This rule allows the size of helper icons (such as the green and red arrows that hover above junctions) to be scaled down in size from the usual Trainz default.

There is only one property in this rule to set and that is the percentage value that indicates the desired scale of helper icons relative to the default Trainz size.

Screenshot 5.33 illustrates how a setting of 50% with this rule will make the junction arrows appear compared to the default 100% setting. This scaling down of helper icons will also affect their appearance in the Minimap as well.



***Screenshot 5.33*:** Comparison of helper icon sizes with 100% (default) on the left and 50% on the right.

When started, the *Set Helper Icon Size* rule will instruct Trainz to scale the helper icons to the desired value and will then complete itself without waiting for anything or running any child rules. *Set Helper Icon Size* is a 'use once' *Task* rule that needs to be reset if more than one run of it is required.

Tip: The visibility of helper icons can be set with the Startup Options rule.

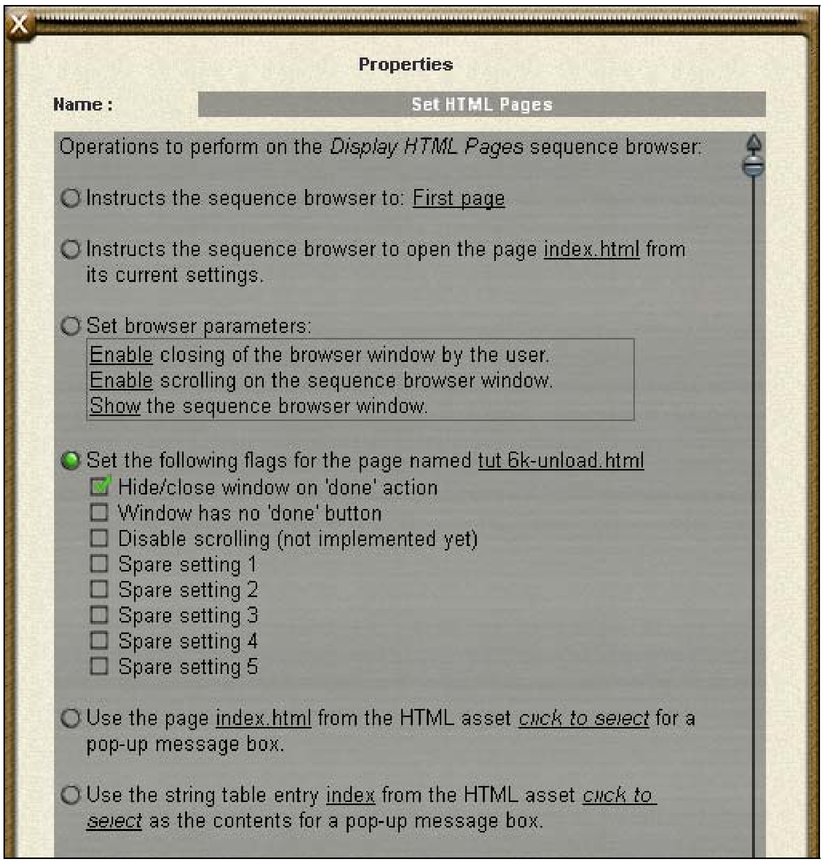
Set HTML Pages

This rule is used to send instructions that affect the properties of the sequence browser created by the *Display HTML Pages* rule. It was written specifically for the TRS2006 tutorial sessions and was not meant to be a general-purpose rule. Familiarity with *Display HTML Pages* is a prerequisite for using this rule.

There are 6 different things this rule can request of the sequence browser (and the pages in the sequence):

- To perform an operation
- Instruct the sequence browser to go to a specific page
- Set the parameters of sequence browser window
- Set the flag properties of a specific page
- Create a pop-up message box from a HTML page
- Create a pop-up message box from a string table entry

Set HTML Pages can only do one of these actions where the action to perform is selected via a radio button interface in the properties window as shown in *Screenshot 5.34*.


Properties

Name : Set HTML Pages

Operations to perform on the *Display HTML Pages* sequence browser:

○ Instructs the sequence browser to: First page

○ Instructs the sequence browser to open the page index.html from its current settings.

○ Set browser parameters:
Enable closing of the browser window by the user.
Enable scrolling on the sequence browser window.
Show the sequence browser window.

● Set the following flags for the page named tut 6k-unload.html
- [✓] Hide/close window on 'done' action
- [ ] Window has no 'done' button
- [ ] Disable scrolling (not implemented yet)
- [ ] Spare setting 1
- [ ] Spare setting 2
- [ ] Spare setting 3
- [ ] Spare setting 4
- [ ] Spare setting 5

○ Use the page index.html from the HTML asset click to select for a pop-up message box.

○ Use the string table entry index from the HTML asset click to select as the contents for a pop-up message box.

Screenshot 5.34: Properties window of the *Set HTML Pages* rule.

Performing an Operation

This action instructs the sequence browser to perform a particular navigation or window operation. The operations to perform are selected from a list and are simple 'actions' that don’t require any further configuration.

The operations that can be performed are:

• Close	- Close and destroy the sequence browser.
• Done check button	- Simulate a press on the 'done' checkmark button.
• Previous page	- Go to the previous page in the sequence.
• Next page	- Go to the next page in the sequence.
• First page	- Go to the first page in the sequence.
• Last page	- Go to the last page in the sequence.
• Hide window	- Hide the sequence browser window.
• Show window	- Show the sequence browser window.

There is no default operation selected, the '*click to select*' link must be clicked on to choose one. If an operation is not selected or the selected operation cannot be performed, then this rule will not be able to do anything and the sequence browser will remain unaffected.

Open a Page

This action instructs the sequence browser to open the specified page from its list. The property for this action (default "index.html") is the name of the HTML file to open. It must be specified without its .HTML extension exactly as it is in *Display HTML Pages* rule’s properties. If an invalid page is specified, the sequence browser will remain unchanged and error message will be logged to the **JetLog.txt** recording this error.

Setting Browser Parameters

There are three different features of the sequence browser that this mode can enable or disable:

- Closing of the browser window by the user (i.e. if disabled, there is no 'X' in top left corner)
- Window scrolling (if disabled, the scrollbar will not appear at all, even if the page is longer than the sequence browser window’s height)
- Visibility (show or hide the sequence browser window)

Note: When these settings are applied to the sequence browser window, they affect the browser for all pages it displays, not just the current one.

Setting these browser parameters via this rule has not been fully tested and is not guaranteed to work. It is best to set the browser up as required in the *Display HTML Pages* rule’s properties.

Setting flags for an individual page

This mode allows several on/off flag settings to be applied to the specified HTML page (default "index.html") from the sequence browser’s list. The page must be specified without the .HTML extension exactly as it is in the *Display HTML Pages* rule’s properties.

The choices that are functional and work are:
- Hide/Close on done
- Window has no 'done' button

If checked, the "Hide/Close on done" will cause the sequence browser to become invisible when that specified page’s 'done/check' link is clicked on.

Enabling the "Window has no 'done' button" flag means that the 'done/check' button will not be active or work for that page. It is assumed that the author of the HTML asset made their pages as specified in the Display HTML Pages rule description.

The remaining flags will have no effect as they are not currently used for anything. This may change in the future though, so always leave these flags unchecked as they may be used for something you don’t want in a future version!

HTML Page Message Popup
This mode instructs the sequence browser module to open a new browser window (separate from the sequence browser) with the named page from the specified HTML asset loaded into it. The only requirement for the page is that it must have a URL of "**live://pages/messagebox/done**" in it. This is required so that the pop-up browser window can actually be closed. What the link is wrapped in (text or images) is up to the author of the HTML asset.

String Table Message Popup
This mode instructs the sequence browser module to open a new browser window (separate from the sequence browser) with the string table entry from the specified HTML asset being used as the page’s contents. A 'done' checkmark link is automatically inserted after this. Keep in mind that this is small window and the same guidelines recommend in the Message Popup rule apply here.

When operating in either of the pop-up modes, the *Set HTML Pages Rule* actually instructs an internal interface module of Trainz to create that page and will complete itself straight away without worrying about the page any further. This differs from the *Message Popup* rule that creates and manages the browser window itself and completes once that browser window is closed.

If you want to wait for a message popup window to be acknowledged, use *Message Popup*. Using *Set HTML Pages* for this purpose however will mean that even if this rule is complete/reset from elsewhere, the message window will still be displayed as it has been handed over to a sperate module.

Regardless of which mode this rule is running in, it will always send the settings to the sequence browser to perform and complete itself straight away without waiting for anything or running any child rules. It is a 'use once' Task rule that will need to be reset if another run is required.

Tip: In most cases if something is attempted through this rule that the sequence browser doesn't like or thinks is invalid, a message will appear in the **JetLog.txt** indicating the error.

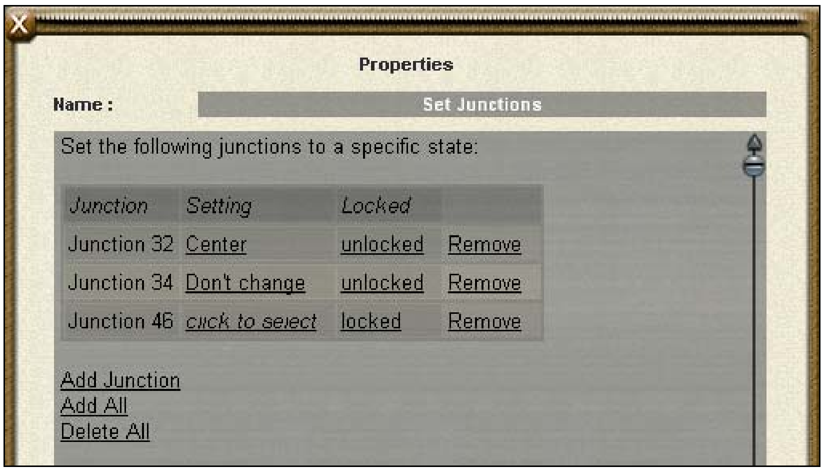
Set Junctions

The *Set Junctions* rule sets the state of one or more junctions. The position of a junction and its locked state can both be set by this rule.

The properties window of this rule allows a list of junctions to be maintained as seen in *Screenshot 5.35*. The 'Add Junction' link provides a list of all junctions on the current route that are not already in the list and will append the selected junction to the list.

'Add All' adds all of the junctions on the route to the list. This is a handy way of locking every single junction on a route if there happen to be several hundred. The 'Delete All' link at the very bottom will delete all junctions in the list.

Note: Be careful when using the 'Delete All' link as your list of tediously configured junctions will vanish. Even if this does happen by accident, saving often so you can revert back without too much trauma is advised.



Properties

Name : Set Junctions

Set the following junctions to a specific state:

Junction	Setting	Locked	
Junction 32	Center	unlocked	Remove
Junction 34	Don't change	unlocked	Remove
Junction 46	click to select	locked	Remove

Add Junction
Add All
Delete All

Screenshot 5.35: Properties window of the *Set Junctions* rule.

Each junction in the list will have several properties adjacent to it. These are the setting and locked states. The 'Remove' link is used to remove that junction from the list.

The setting of a newly added junction will not be anything, hence the '*click to select*' link. Clicking this link will allow one of 3 possible junction states to be selected:

- Left
- Center
- Right

If no setting for the junction is selected, the rule will leave the junction's direction unchanged.

Note: Not all junctions will have a center position. Only 3-way junctions have a center position.

The locked state refers to whether the user will be allowed to change that junction in Driver. Locking a junction will prevent the user from being able to change the junction in any circumstance. If a locked junction needs to be unlocked again later on in the session, another *Set Junctions* rule instance will be needed to unlock it.

Note: The user cannot change a locked junction, however the Driver AI and this rule can still change junctions.

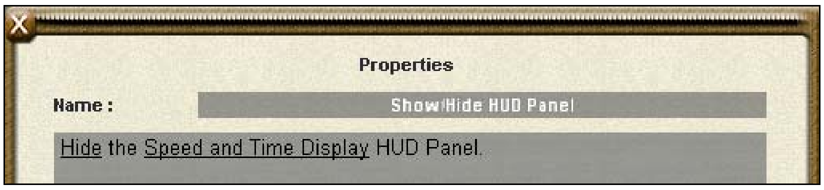
Set Junctions does not wait for anything. It changes the state of the junctions it is configured to and completes without executing any child rule. This is a 'use once' *Task* rule that will need to be reset before being run again.

Tip: To detect the change of a junction to a particular position, use the *Multiple Junction Alignment Check* rule.

Show/Hide HUD Panel

This rule either shows or hides the designated HUD panel. The HUD panels this rule can show/hide are:

- Bottom Right Menu
- Cabin Controls
- Camera Controls
- DCC Controls
- Speed and Time Display


Properties

Name : Show/Hide HUD Panel

Hide the Speed and Time Display HUD Panel.

***Screenshot 5.36*:** Properties window of the *Show/Hide HUD Panel* rule.

Once started, *Show/Hide HUD Panel* will show/hide the panel as set and complete straight away without running any child rules. It can be set to show/hide only one panel so multiple instances of the rule would be needed if more than one panel is to be shown or hidden. This is a 'use once' *Task* rule that will need to be reset before being run again.

Show/Hide Minimap

This rule either shows or hides the Driver Minimap window. Once started, it will show/hide the Minimap window as set and complete straight away without running any child rules. It is a 'use once' *Task* rule that will need to be reset before being run again.

Simultaneous List

The *Simultaneous List* rule is one of the *List* rules that does not do anything directly itself, but rather acts as a way of controlling if and when a child rule is executed.

When started for the first time, this particular rule will start running all of its child rules simultaneously. How a *Simultaneous List* rule completes itself depends on its settings.

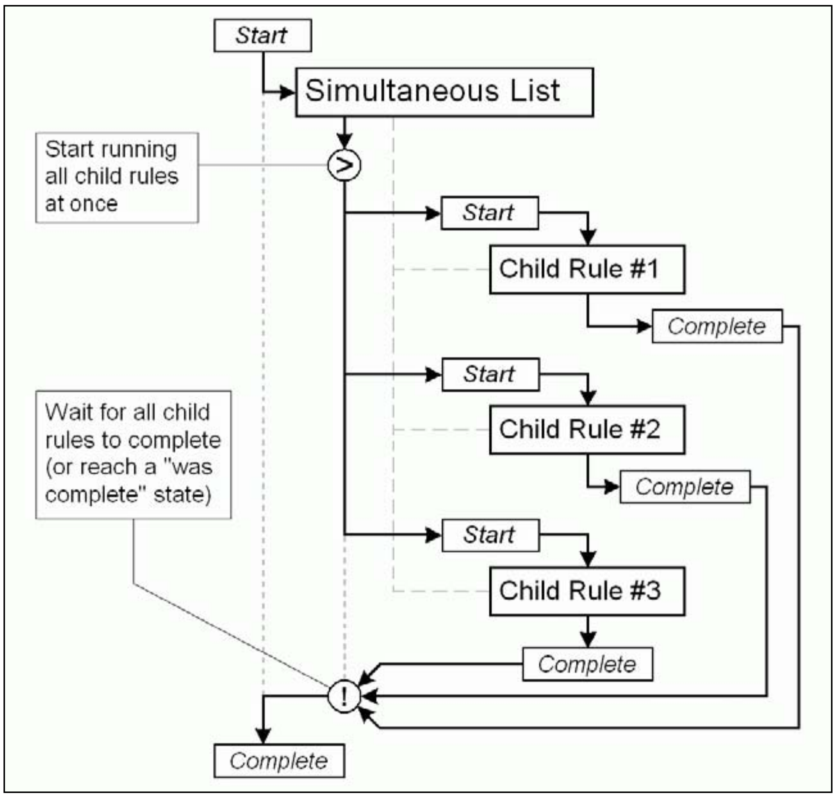


Diagram 5.6: Flowchart of the *Simultaneous List* rule showing completion when all of child rules have reached a "complete" (or "was complete") state.

The first choice is for this rule to complete itself after all child rules have reached a complete state (or a "was complete" state). This is shown illustrated in *Diagram 5.6*.

In the case of the "was complete" mode, this means that all child rules must have reached a complete state at least once already in their lifespan. They can still go on and do other things while still being considered as being in a "was complete" state.

The other way that the *Simultaneous List* rule can complete itself is to complete the moment one child rule completes. This means all other child rules will be reset and abandoned because the list rule is completing itself on the basis of whichever child rule completes first. *Diagram 5.7* below shows this behavior:

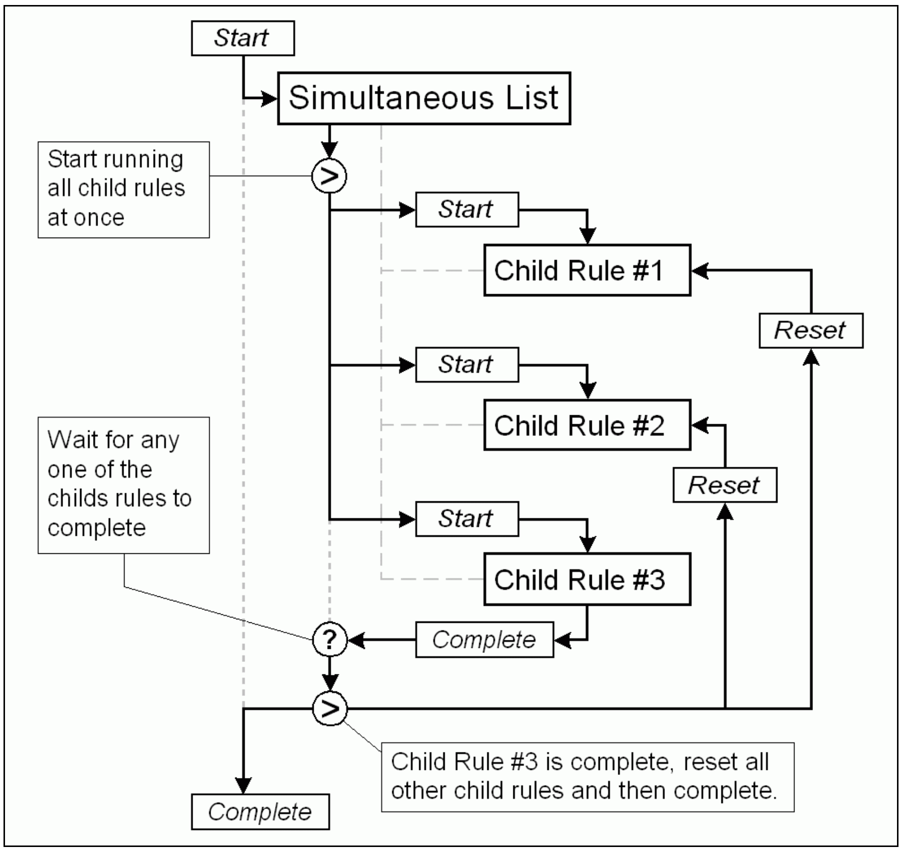

Start
Simultaneous List
Start running all child rules at once
Start
Child Rule #1
Reset
Start
Child Rule #2
Reset
Wait for any one of the childs rules to complete
Start
Child Rule #3
Complete
Child Rule #3 is complete, reset all other child rules and then complete.
Complete

Diagram 5.7: Flowchart of the *Simultaneous List* rule completing when any one of the child rules has completed.

Tip: This is one of several list rules for handling child rules in a specific way. Other list rules include *Ordered List*, *Progressive List*, *Random List* and *Reset List*.

Startup Options

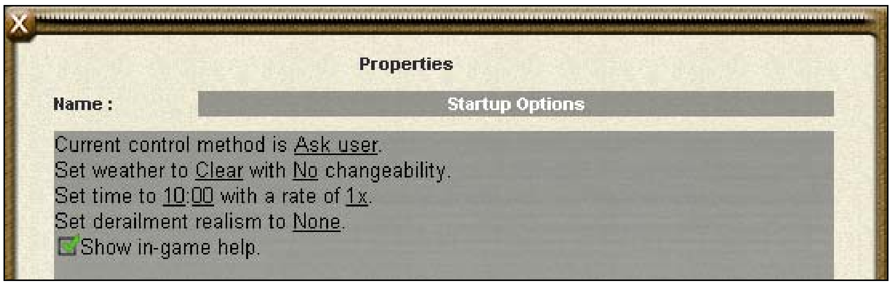
This rule sets various parameters that affect how the Trainz world in Driver appears and operates. It is mostly a grouping together of functionality that various other rules offer separately. Parameters this rule can set are:

- Control Mode (*Control Type* rule)
- Weather (*Weather* rule)
- Current time and time rate (*Time and Rate* rule)
- Derailment realism (*Derailment Realism* rule)
- In game-help

Note: For further details of each parameter in the list above, see the description of the rule mentioned in brackets.

Once started, this rule sets the Trainz world parameters according to its own properties. It will then complete straight away without executing any child rules. It has no role in the on-going maintenance/monitoring of the parameters it sets in the Trainz world.

If the control mode property is set to "Ask user", *Startup Options* completes when the user has selected a control mode. This means it may not necessarily complete straight away, although all the other properties would have been set as soon as it was started.


Properties

Name : Startup Options

Current control method is Ask user.
Set weather to Clear with No changeability.
Set time to 10:00 with a rate of 1x.
Set derailment realism to None.
✓ Show in-game help.

***Screenshot 5.37*:** Properties window of the *Startup Options* rule.

The "Show in-game help" option switches the in-game help on or off. In-game help refers to the visibility of things like junction direction arrows and names as described in Section 9.14 of the TRS2006 Expanded Manual.

Startup Options is a 'use once' *Task* rule that needs to be reset if more than one run of it is required.

Time and Rate

This rule sets the Trainz world time in Driver as well as the rate that time progresses. The time is set based on a 24-hour clock (i.e. no AM/PM) and the time rate ranges from 1X to 1440X.

Once started, this rule sets the Trainz world to run from its property settings and completes straight away. It has no on-going role in running the time in the Trainz world.

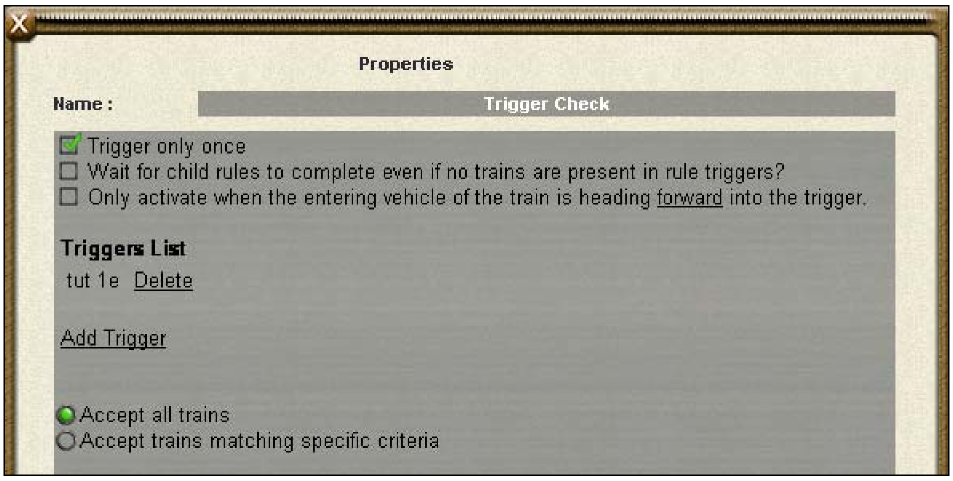
The *Time and Rate* rule does not wait for anything or execute any child rules. It is effectively a 'use once' task-rule that needs to be reset if another run is required.

Tip: The *Startup Options* rule can also set the time and rate that this rule can, as well as several other parameters in the Trainz world.

Trigger Check

The Trigger Check rule is an *Event* rule that waits for a train to enter a target trigger and then runs the child rules. It is a much more flexible and featured-rich version of *Trigger Rule* as it supports multiple triggers and train filtering.

At the top of the properties window shown in *Screenshot 5.38*, there are several checkbox options. These options affect the behavior of Trigger Check rule when running and will be examined in the diagrams further below.



Properties

Name : Trigger Check

[✓] Trigger only once
[ ] Wait for child rules to complete even if no trains are present in rule triggers?
[ ] Only activate when the entering vehicle of the train is heading forward into the trigger.

Triggers List

tut 1e Delete

Add Trigger

● Accept all trains
○ Accept trains matching specific criteria

Screenshot 5.38: Properties window of the *Trigger Check* rule.

To add a trigger to the list, click on the 'Add Trigger' link and select the desired trigger from the list. A trigger can be removed from the list by clicking on the 'Delete' link adjacent to its name in the list.

By default, *Trigger Check* does not filter trains and will activate when any train enters one of its targeted triggers. Filtering such that only certain types of trains will activate the rule is possible by enabling the "Accept trains matching specific criteria property" at the bottom of the properties window. Once enabled, a train filter interface appears that can be configured so this rule will only accept trains that contain a specific vehicle, vehicle type or driver character.

The first checkbox property in this rule is the "Trigger rule once" property. This refers to whether the *Trigger Check* rule waits for a train just once or operates in a continuous cycle where it will activate and run the child rules every time it is entered.

The "Wait for child rules" property determines how this rule handles the child rules once activated by a train entering one of the triggers. Depending on what the child rules actually are, this option can have a major affect on how these child rules run and if they even get a chance to complete.

When running, the *Trigger Check* rule has two states: active and idle. The idle state is when the rule has been started and no part of any train that matches filter conditions is anywhere within any of the monitored triggers. The moment a train matching filter conditions enters a monitored trigger, this rule enters an active state and starts running the child rules.

If the rule is already in an active state and another valid train enters one of its triggers, this will not initially affect the rule’s operations as it is already in an active state. The child rules are only started when the rule changes from an idle to active state.

To transition back into an idle state, all valid trains that have entered the triggers must have fully exited and be clear of the triggers. It doesn’t matter which particular train leaves first, the rule will only stop being in an active state when all triggers are clear.

Regardless of whether the "Wait for child rules" property setting is switched on or off, the rule will always become active whenever the first valid train enters a trigger. It is what happens when the rule becomes idle again that the "Wait for child rules" property affects.

Wait for child rules Disabled

When the "Wait for child rules" property is not enabled (the default setting), all child rules will be paused and reset when the rule changes from an active to idle state. *Diagram 5.8* shows how the child rules can be potentially cut off before they get a chance to complete themselves.

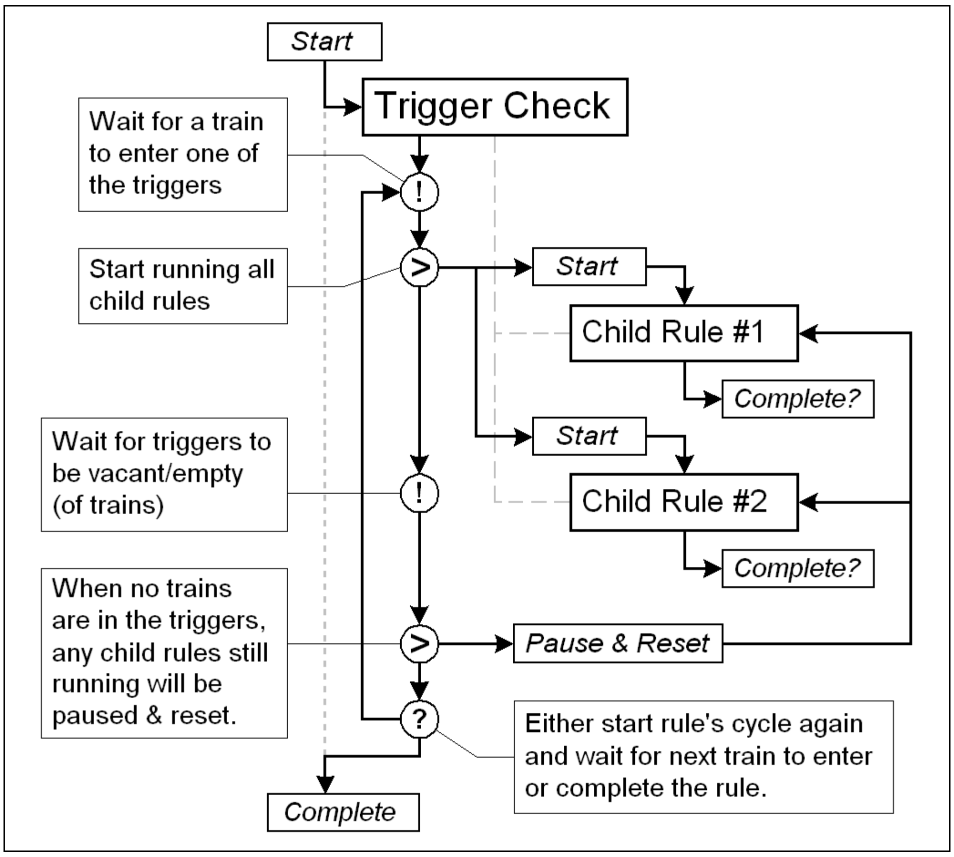

Start
Trigger Check
Wait for a train to enter one of the triggers
!
Start running all child rules
>
Start
Child Rule #1
Complete?
Start
Child Rule #2
Complete?
Wait for triggers to be vacant/empty (of trains)
!
When no trains are in the triggers, any child rules still running will be paused & reset.
>
Pause & Reset
?
Either start rule's cycle again and wait for next train to enter or complete the rule.
Complete

Diagram 5.8: Flowchart of the *Trigger Check* rule running with the "Wait for child rules" property disabled where child rules can be cut off from completing.

Once the rule is no longer active, any child rules still running will be paused and reset. If the "Trigger rule once" option is enabled, the rule will complete itself, otherwise it will go back to monitoring its triggers for next time.

Wait for child rules Enabled

When the "Wait for child rules" property is enabled, the *Trigger Check* rule will wait for all child rules to complete, regardless of whether the rule is in an active state or gone idle with no trains in its triggers. *Diagram 5.9* shows how the child rules will always be waited on to completion.

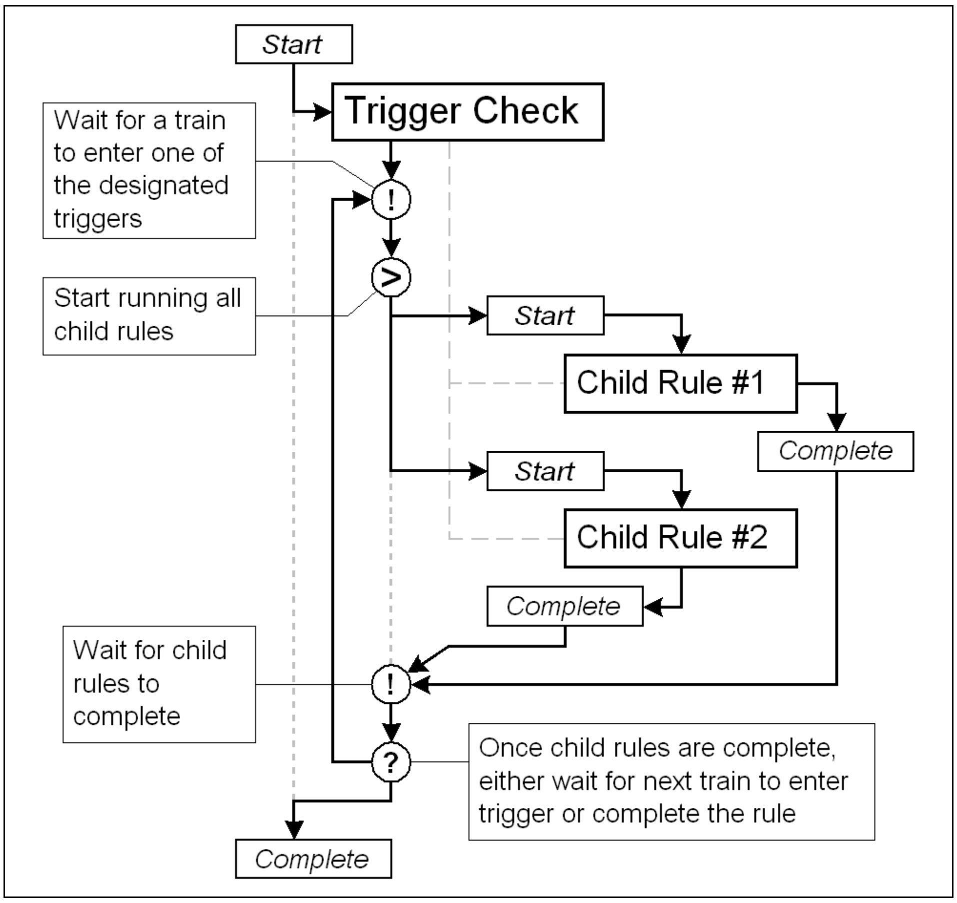

Start
Trigger Check
Wait for a train to enter one of the designated triggers
!
>
Start running all child rules
Start
Child Rule #1
Complete
Start
Child Rule #2
Complete
Wait for child rules to complete
!
?
Once child rules are complete, either wait for next train to enter trigger or complete the rule
Complete

***Diagram 5.9*:** Flowchart of the *Trigger Check* rule running with the "Wait for child rules" property enabled so the child rules will get a chance to complete.

After the child rules have completed, Trigger Check will either resume monitoring triggers again for the next occasion or complete itself, depending on whether the "Trigger once rule" option has been enabled.

Allowing the child rules to complete means that you can safely depend on the child rules getting their job done without having to worry about a train leaving a trigger and cutting things off. For example, consider a child rule that itself is an *Event* waiting rule and the event in the Trainz world it is waiting on may not occur in a predictable space of time.

Note: Trigger Check is one of the more internally complex rules and this is further complicated by how triggers work in the Trainz world. The trigger's scope for example, can be blinded by a junction and make a train within range invisible. Further information about triggers can be found in sections 11.8.7 & 11.8.8 of the TRS2006 Expanded Manual.

Trigger Rule

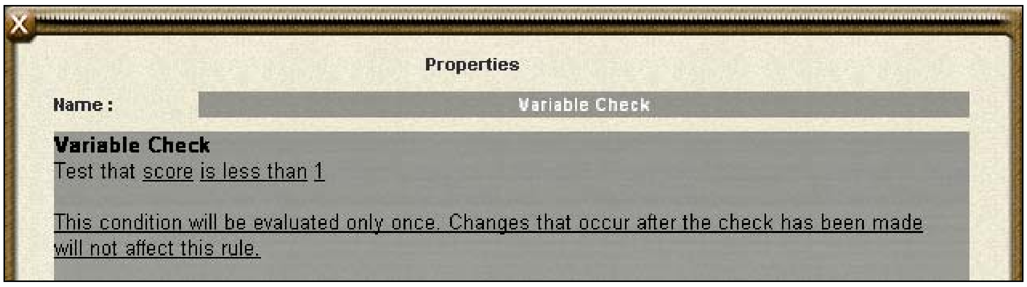
This rule has been made obsolete by the *Trigger Check* rule. It is included with TRS2006 as a legacy rule for backward compatibility reasons.

Variable Check

This rule is a *Check* rule that compares the designated variable value against a set value using a logical comparison operator.

Note: The concept of what variables are and how to create them are discussed in the description of the Variable Show rule.

Along with the name of the variable being checked, the type of check being performed on the variable and the value it is being compared against need to be set in the rule's properties window.


Properties

Name : Variable Check

Variable Check
Test that score is less than 1

This condition will be evaluated only once. Changes that occur after the check has been made will not affect this rule.

Screenshot 5.39: Properties window of the *Variable Check* rule.

The logical comparisons supported by this rule are:

- Less than (<)
- Less than or equal to (<=)
- Equal to (==)
- Greater than (>)
- Greater than or equal to (>=)
- Not equal to (!=)

These are the typical value comparison operators that most programming languages have. The default comparison is "is less than", so click on that link to change as desired. The comparison value is 1 by default.

Once started *Variable Check* will perform the check on the variable straightaway. If the check on variable is successful (i.e. it equates to true), this rule will run all child rules before completing. Otherwise it will complete itself without doing anything as the check has failed.

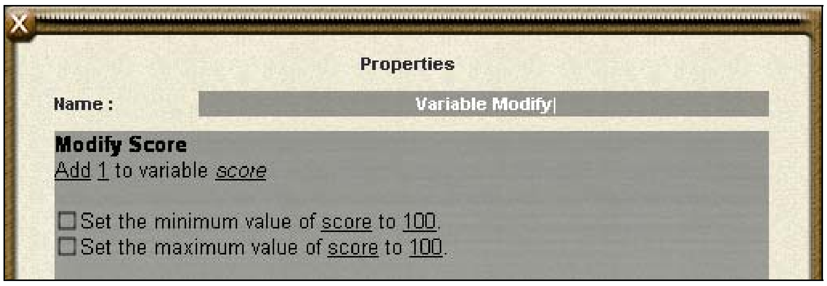
This rule does not wait for the variable check it is performing to success. It is a 'use once' *Event* rule that needs to be reset if it is to be run again to perform the check later on.

Note: For this rule to work, the target variable it is checking must already exist. Use *Variable Show* to create the variable and *Variable Modify* to initialize it as needed.

Variable Modify

This rule modifies the designated variable by a set value in conjunction with a selected operation. The three modify operations this rule supports are:

- Add (+)
- Subtract (-)
- Set value to (=)



Properties

Name : Variable Modify

Modify Score
Add 1 to variable *score*

☐ Set the minimum value of score to 100.
☐ Set the maximum value of score to 100.

Screenshot 5.40: Properties window of the *Variable Modify* rule.

To accompany the variable name and the operation properties, there is the value property that determines how the operation applies to the variable.

Variable Modify also has the option to impose an upper and lower limit on the variable such that the variable will stick to these limits, even if the modified results falls outside of the limits. These limits will apply to the variable not just for this modification, but on an on-going basis until it is changed by another rule elsewhere.

When started, this rule applies the modification to the variable and completes straight away. No child rules are executed. If an invalid variable is specified, this rule will complete without having modified anything. This rule is a 'use once' *Task* rule that will need to be reset if another run is required.

Note: For this rule to work, the target variable it is modifying must already exist. Use *Variable Show* to create the variable. *Variable Check* can be used to verify the current value of a variable.

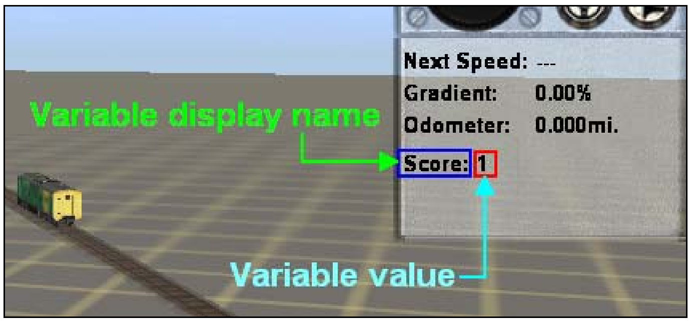
Variable Modify Continuous

This rule continuously modifies the designated variable by a set amount every second. Once started, this rule will keep modifying the variable and won't stop doing so until it is paused or reset by a parent rule or Trainz itself.

Note: For this rule to work, the target variable it is modifying must already exist. Use *Variable Show* to create the variable. *Variable Check* can be used to verify the current value of a variable.

Variable Show

A variable is a whole number value that has a unique name to identify it by and a display name used to label it on the Custom HUD panel as seen in *Screenshot 5.41*.

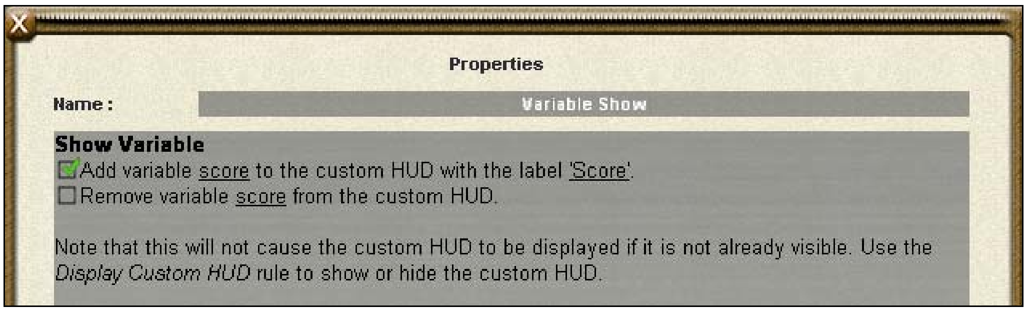


Screenshot 5.41: A variable displayed on the Custom HUD in Driver.

Variables don't exist unless they are explicitly created somehow and this is what the *Variable Show* rule is for. It can be configured to either create a new variable such that it is displayed on the Custom HUD or remove an existing variable from the Custom HUD.

The properties window of this rule (as seen in *Screenshot 5.42*) allows a new variable with label to be created and added to the Custom HUD. However, the value of the variable cannot be set here, that must be done with the *Variable Modify* rule later.

To remove a variable form the Custom HUD, only its name needs to be specified.



Properties

Name : Variable Show

Show Variable

✓ Add variable score to the custom HUD with the label 'Score'.

☐ Remove variable score from the custom HUD.

Note that this will not cause the custom HUD to be displayed if it is not already visible. Use the *Display Custom HUD* rule to show or hide the custom HUD.

Screenshot 5.42: Properties window of the *Variable Show* rule.

For the variable to be visible, the Custom HUD panel has to have been made visible with the *Display Custom HUD* rule. However, variables can still exist as virtual invisible values used by the session rules that the user needn't know about.

Once started, this rule will create/add and remove the variable values as configured. If the Custom HUD panel doesn't exist and the variable is being added, that variable will still exist in the Trainz world as a value the user can't see. After the variables have been added/removed, this rule completes itself straight away without running any child rules. It is a 'use once' *Task* rule that needs to be reset if another run is required.

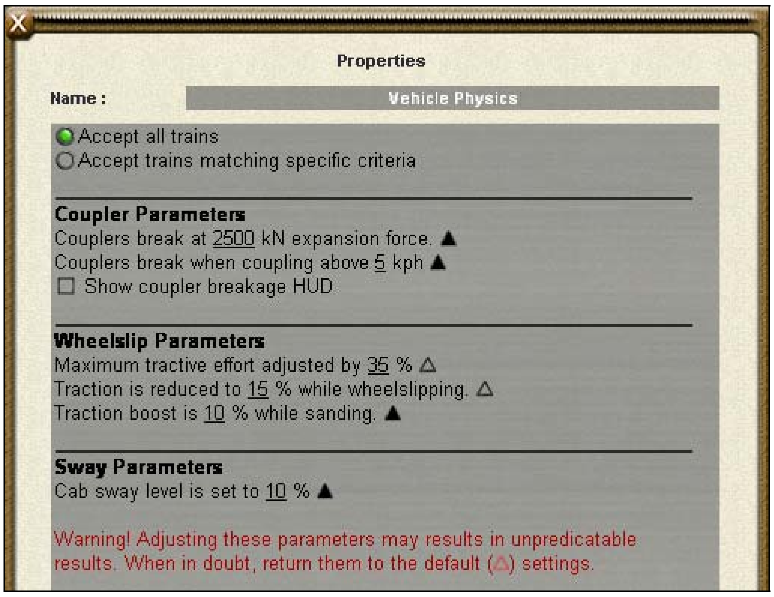
Tip: To manipulate and check on variable values, use the *Variable Check*, *Variable Modify* and *Variable Modify Continuous* rules.

Vehicle Physics

Vehicle Physics is a *Task* rule that is used to tweak the various physics parameters of the vehicles in the Trainz world. Once started, it will set the physics parameters and then complete itself.

Note: The parameters of this rule will only take affect when Driver is running in Cabin control mode.

This rule can either be applied to all vehicles in the Trainz World or to only vehicles on trains that match specific filtering criteria. The radio buttons at the top of the properties window allow this filtering to be turned on or off. Selecting the "Accept trains matching specific criteria" option will open up the filter where trains can be filtered on the basis of specific vehicles, vehicle types and driver characters.


Properties

Name : Vehicle Physics

● Accept all trains
○ Accept trains matching specific criteria

Coupler Parameters
Couplers break at 2500 kN expansion force. ▲
Couplers break when coupling above 5 kph ▲
☐ Show coupler breakage HUD

Wheelslip Parameters
Maximum tractive effort adjusted by 35 % △
Traction is reduced to 15 % while wheelslipping. △
Traction boost is 10 % while sanding. ▲

Sway Parameters
Cab sway level is set to 10 % ▲

Warning! Adjusting these parameters may results in unpredicatable results. When in doubt, return them to the default (△) settings.

Screenshot 5.43: Properties window of the *Vehicle Physics* rule.

Below the train filtering options in the properties window, there are three different groups of physics parameters that this rule will apply to the vehicles.

To the far right of each parameter is a triangle. If the triangle is a solid black, this means the parameter is at its default value. When the parameter is changed to something that varies from its default value, an unfilled triangle will appear as seen in *Screenshot 5.43*. Clicking on the empty triangle will reset that parameter back to its default value.

Coupler Parameters
Expansion force breakage refers to the limit of force that can be applied to a coupler in kilo Newtons (kN) before it will break. A typical modern knuckle-type coupler can handle approximately 2500kN of force before breaking (which is this rule’s default setting).

"Couplers break when coupling above" is the maximum safe coupling speed for the vehicles. If the vehicles are coupled at a speed greater than the one specified here, their couplers will break and become unusable.

The "Show coupler breakage HUD" option will show or hide the Coupler Breakage HUD panel. This property performs the same function as the *Coupler Breakage HUD* rule does.

Wheelslip Parameters
"Maximum tractive effort" is a percentage value that scales the effectiveness of the locomotive’s traction or pulling power. A setting of 100% will mean a locomotive has the complete traction that the asset was configured to have while 50% will mean the locomotive has half of its traction.

"Traction reduction while wheelslipping" refers to the amount of reduction in tractive power a locomotive will have when it encounters wheelslip. Wheelslip occurs in situations such as hauling a heavy load up gradient.

"Traction boost" is the increase in traction a locomotive will have when sanding is enabled. This means a locomotive experiencing wheelslip can get a traction boost by turning on sanding.

Sway Parameters
There is only one parameter in this grouping and that is the Cab sway level. This value is a percentage in the range of 0-100% that expresses the degree of cabin sway experienced. A setting of 0% will mean no cabin sway while 100% is the highest possible level of cabin sway.

This is not to be confused with super-elevation or tilting. Cabin sway is more the general “bounciness” experienced from internal cabin view and only affects the cabin view. The trains will not sway/bounce or be affected by the sway when viewed externally.

By allowing filtering of trains, different vehicles can have their own custom physics parameters set. This could be handy for example if a particular vehicle type is by default extremely over-powered so a reduction in tractive effort could be applied to all vehicles of that type to fix this to make a more realistic session.

Once started, *Vehicle Physics* will set the physics parameters as required and then complete itself straightaway without running any child rules. It is a 'use once' *Task* rule that will need to be reset if another run is required.

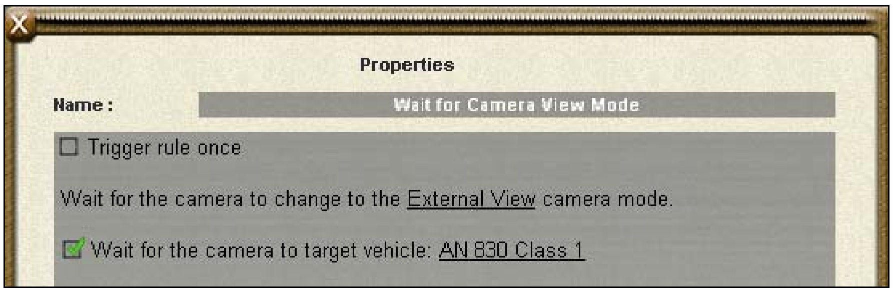
Wait for Camera View Mode

This rule waits for the Driver camera to be changed to either a designated viewing mode or target vehicle.

The 4 camera modes to choose from are:

- Internal (cab)
- External
- Tracking
- Roaming

If a mode isn't selected, this rule will be unable to run child rules or complete as it has nothing to wait for.


Properties

Name : Wait for Camera View Mode

☐ Trigger rule once

Wait for the camera to change to the External View camera mode.

✓ Wait for the camera to target vehicle: AN 830 Class 1

Screenshot 5.44: Properties window of the *Wait for Camera View Mode* rule.

Below the camera mode setting in the property window is the target vehicle setting that is enabled by clicking the checkbox next to it. From there, a desired target vehicle can be selected from a list. If this checkbox is enabled, the camera mode setting will be ignored and the rule waits for the selected vehicle to come into focus instead.

When started, this rule waits until the required camera mode/target is set. Once this has happened, all child rules will be started. When all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the "Trigger rule run once" property setting.

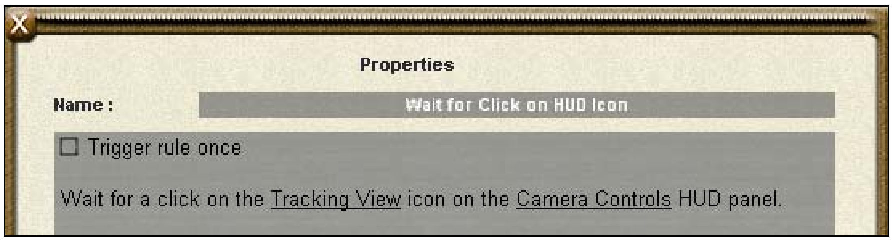
Tip: The camera mode can be changed with the *Set Camera* rule.

Wait for Click on HUD Icon

This rule waits for the designated HUD icon to be clicked on. The panel icons that *Wait for Click on HUD Icon* can monitor for a click on grouped by HUD panel are:

- *Bottom Right Menu*: Messages, Map, Commodities, Decoupling, Waybill, View Schedule & View Driver Help
- *Camera Controls*: Cab View, Chase View, Tracking View & Free Roaming
- *DCC Controls*: Pantographs, Lights, Horn & Stop

In the properties window, the appropriate HUD Panel (the last link in the sentence) must be selected before the icon (first link) can be chosen. *Screenshot 5.45* shows how the 'Tracking View' icon is selected from the 'Camera Controls' HUD Panel.


Properties

Name : Wait for Click on HUD Icon

☐ Trigger rule once

Wait for a click on the Tracking View icon on the Camera Controls HUD panel.

Screenshot 5.45**:** Properties window of the *Wait for Click on HUD Icon* rule.

When the designated HUD icon has been clicked, the child rules will be started. When all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the 'Trigger rule once' property setting.

If the HUD panel that the watched icon is on happens to be invisible, then it obviously can't be clicked. Unless the panel concerned becomes visible somehow, this rule will be waiting indefinitely.

Note: If the icon is disabled, this rule won't be able to detect an attempt to click on it. The *Disabled HUD Icon Notify* rule provides a way to detect a click on a disabled HUD icon while *Enable/Disable HUD Icons* can enable/disable HUD icons.

Tip: The *Flash HUD Icon* rule can flash the HUD icons.

Wait for Derailment

This rule is an *Event* rule that waits for a derailment to occur in the Trainz world. When a derailment is detected, it will run all child rules simultaneously and then wait for all of them to complete. Once the child rules are complete, *Wait for Derailment* either completes itself or resets the child rules and waits for the next derailment to occur.

There is only one option in this rule's properties and that is the "Trigger derailment watch once" setting. This refers to whether this rule is used once or operates continuously.

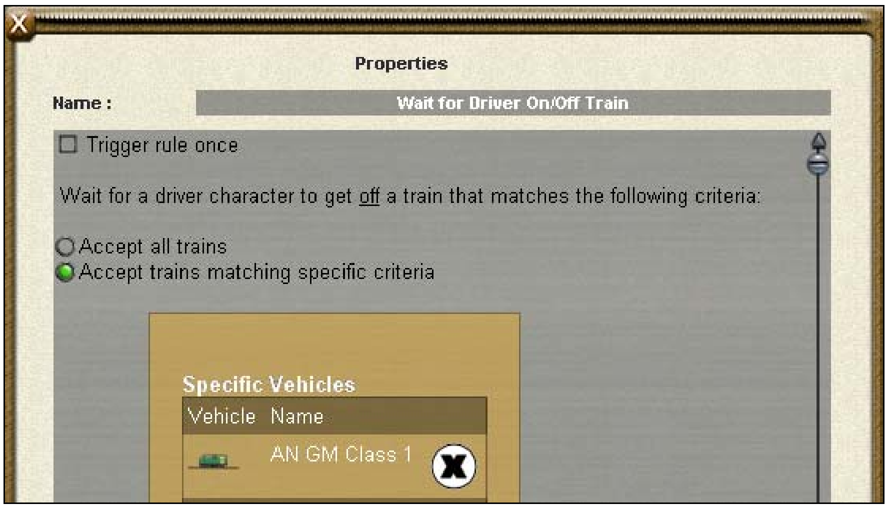
Trainz detects derailments on a per-vehicle basis, so a train will generate multiple derailments messages based on how many vehicles there are in that consist. However once this rule has detected a derailment, it will get on with running the child rules and won't be listening for further derailments.

When the child rules complete and if this rule is operating continuously, it will return to listening for derailment messages again but other derailments that may have occurred in the meantime will have been missed.

Wait for Driver On/Off Train

Wait for Driver On/Off Train is an *Event* rule that waits for a driver character to either get on or off a train. Once a driver has gotten on or off a train and activated this rule, all child rules will be run. When all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the "Trigger rule once" property setting.

If the "Accept all trains" property is selected, this rule will activate on the first possible occasion of a driver getting on/off a train without regard to what the consist has or who the driver character is.


Properties

Name : Wait for Driver On/Off Train

☐ Trigger rule once

Wait for a driver character to get off a train that matches the following criteria:

○ Accept all trains
● Accept trains matching specific criteria

Specific Vehicles

Vehicle	Name
	AN GM Class 1

***Screenshot 5.46*:** Properties window of the *Wait for Driver On/Off Train* rule.

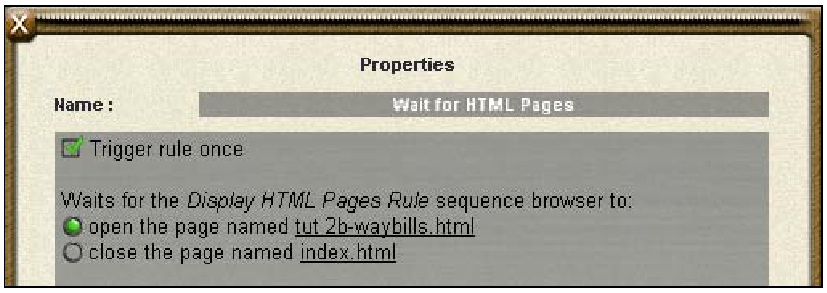
This rule can also be configured to only activate when a driver gets on or off a train consist that matches certain criteria. When the "Accept trains matching specific criteria" option is selected, the standard train filter property interface is displayed as shown in *Screenshot 5.46*.

Train consists can be filtered by the presence of specific vehicles and vehicle types. Filtering of driver characters is not functional at this stage.

Tip: The *Driver Setup* rule is responsible for assigning drivers to train consists.

Wait for HTML Pages

Wait for HTML Pages is an *Event* rule that waits for the sequence browser created by the *Display HTML Pages* rule to either open or close a particular HTML page.


Properties

Name : Wait for HTML Pages

✓ Trigger rule once

Waits for the *Display HTML Pages Rule* sequence browser to:
● open the page named tut 2b-waybills.html
○ close the page named index.html

Screenshot 5.47: Properties window of the *Wait for HTML Pages* rule.

The properties interface provides radio buttons to select whether the rule is to wait for a page to open/close. A HTML page is specified individually for the radio button it is adjacent to.

When specifying the page ("index.html" by default), the page name must be specified without its .html extension. This applies to all occasions when file names are entered in rules because Trainz doesn't allow full stop ('.') characters in text entry boxes.

Note: For this rule to activate on a closed page, the page's "Hide close window on 'done' action" setting flag must be enabled. This can be done with the *Set HTML Pages* rule

When started, this rule waits for the required page to be open/closed to be activated. Once this happens, all of the child rules will be started. After all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the "Trigger rule once" property setting.

Note: *Wait for HTML Pages* is a rule designed specifically to work with the *Display HTML Pages* and *Set HTML Pages* rules. It is not compatible with other HTML display rules.

Wait on Minimap Screen/Main Screen

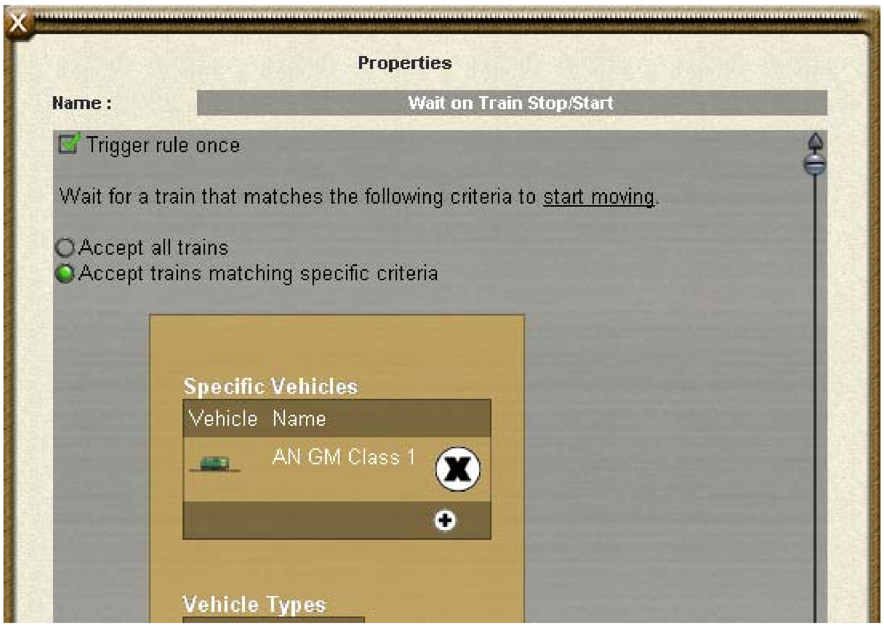
Wait on Minimap Screen/Main Screen is an *Event* rule that once started, waits for the Minimap window to be either opened or closed. Once this has happened, all child rules (if any) will be started. When all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the "Trigger rule run once" property setting.

Tip: The Minimap can be shown or hidden by using the *Show/Hide Minimap* rule while *Minimap Options* allows the types of items visible in the Minimap window to be set.

Wait on Train Stop/Start

Wait on Train Stop/Start is an *Event* rule that waits for a train consist to either stop or start moving. Once a train has started/stopped moving and activated this rule, all child rules will be run. When all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the "Trigger rule once" property setting.

If the "Accept all trains" property is selected, this rule will activate on the first possible occasion of any train starting or stopping without regard to what the consist has in it.


Properties

Name : Wait on Train Stop/Start

☑ Trigger rule once

Wait for a train that matches the following criteria to start moving.

○ Accept all trains
● Accept trains matching specific criteria

Specific Vehicles

Vehicle	Name
	AN GM Class 1

Vehicle Types

Screenshot 5.48: Properties window of the *Wait on Train Start/Stop* rule.

This rule can also be configured to only activate when a driver gets on or off a train consist that matches certain criteria. When the "Accept trains matching specific criteria" option is selected, the standard train filter property interface is displayed as shown in *Screenshot 5.48*.

Train consists can be filtered by the presence of specific vehicles, a certain type of vehicle and drivers.

For this rule to be activated, the train must start or stop moving after the rule has been started so the rule can detect this change of state in the train. If the train stops or starts before the rule is started, the rule will be unable to detect that start or stop. For example, if a train has stopped before the rule has started and the rule needs the train to stop to activate, the train will have to be started so it can stopped for the rule to detect it.

Tip: The contents of a consist can be detected with the *Consist Check* rule and *Wait for Driver On/Off Train* allows a driver getting on or off a train to be detected.

Wait on Waybill Screen/Main Screen

Wait on Waybill Screen/Main Screen is an *Event* rule that once started, waits for the Waybill window to be either opened or closed. Once this has happened, all child rules (if any) will be started. When all child rules have reached a complete state, this rule either completes or restarts itself for the next occasion, depending on the "Trigger rule run once" property setting.

Weather

This rule sets the weather conditions for the Trainz world in Driver. Weather conditions that can be set are:

- Clear
- Cloudy
- Drizzle
- Rain
- Stormy
- Light Snow
- Medium Snow
- Heavy Snow

As well as the weather conditions, the changeability of the weather condition can also be set. 3 levels of changeability are supported:

- Off
- Periodic
- Extreme

When started, this rule tells Trainz the weather settings desired for the world and completes itself straight away. It does not have an on-going role in the Trainz weather conditions or run any child rules. It is a 'use once' *Task* rule that needs to be reset if more than one run of it is required.

Tip: The *Startup Options* rule can also set the same weather conditions this rule can as well as several other parameters in the Trainz world.

Part 8 – Rule Asset Creation

Rule assets consist of script code that defines how the rule behaves. As this document was written for a general audience free of programming topics, rule asset creation isn't covered here. See the example rule source code from the **TRS2006 Script and API Reference** that will be made available shortly after this document has been released.