Railway.exe file specifications (updated July 2021)

The main files created by railway.exe are railway files (.dev during development and .rly when complete and ready for operation), session files (.ssn), timetable files (.ttb) and the configuration file Config.txt. These files are both written to and read from by the program, so users may wish to understand and edit them directly, either manually or by a supporting program. The structure and content of these files are specified in this document. There are other files such as formatted timetables, performance logs, images and so on, but these have standard formats for spreadsheets, text files and image files so they are not dealt with here.

Other files include error files (errorlog.err) and temporary timetable files (tmp). The error file is generated when a fatal error occurs and is accompanied by a message asking the user to send the file to railwayfeedback@gmail.com to help in diagnosing and correcting the error. The error file format is not specified in this document as it seems unlikely that anyone would want to edit it. Temporary timetable files are used during program execution and allow more than one instance of railway.exe to run at the same time. Most of these temporary files are deleted automatically when the program exits but may remain in some circumstances. They can be safely deleted when the program is not running. Temporary timetable files have the same structure as .ttb files.

All the above files can be read using Notepad++ (see below), and most can be read using a plain text editor or word processor.

Please do not edit any of the files with a text editor (or word processor) because all strings (e.g names) end with a NUL character (zero in ASCII), and text editors ignore NULs. The files can be edited directly using Notepad++ (https://notepad-plus-plus.org/), which is a free editor that displays and can copy all special characters, but note that to do this 'Paste special' must be used for both copying and pasting. The copying and pasting process is described in detail later in this document in 'Changing a timetable in a session file whilst retaining routes'.

In this specification file contents are indicated in red.

****

Contents

****

Railway files (.dev and .rly)

Broad structure:

a) General information
b) Active track elements (elements that have tracks)
c) Inactive track elements (other elements - parapets, concourses, platforms, level crossings and non-station named locations)
d) Text
e) Preferred directions
f) User graphics

Detailed structure:

a) General information:
Version number of the railway.exe program that saved the file
Home horizontal offset *(positive or negative)*
Home vertical offset *(positive or negative)*

The whole railway is divided up into 16 x 16 pixel squares, each capable of holding a single track element, where each square is identified by its horizontal location (HLoc) and vertical location (VLoc). Position 0-0 (horizontal and vertical locations both zero) corresponds to the top left hand corner square of the screen when the railway first started to be built, and never changes for that railway thereafter. Although the railway can be any size (limited only by computer memory and operating system limitations), the displayed area is often smaller than full size. In order to retain the same 'Home' screen when a railway is saved the top left hand corner square of the home screen is saved as a horizontal offset and a vertical offset.

b) Active track elements:
Number of active elements
Identifier *two forms*:
    Active elements *(if no user graphics present)*
    Active elements1 *(if there are user graphics)*

Individual element information
Element number 0 *for first*, (Number of active elements - 1) *for last*
SpeedTag *see SpeedTag notes below*
HLoc *see above note under General information*
VLoc *as above*
if a gapjump:
    ConnLinkPos[0] *see Link notes below*
    Conn[0] *see Link notes below*
if a signal, points or level crossing:
    Attribute *special variable used only for points, signals & level crossings, ignored otherwise; points 0=set to go straight, 1=set to diverge, or, for 'Y' shaped points where both legs diverge 0=set to left fork, 1=set to right fork; signals 0=red; 1=yellow; 2=double yellow; 3 = green; level crossings 0 = raised barriers = closed to trains, 1 = lowered barriers = open to trains; 2 = changing state = closed to trains*

if a signal:
    CallingOnSet *0 if a call on is not available, or 1 if it is*
if any other type of element: nothing

In a .rly file the signal and point Attribute and CallingOnSet values aren't needed, but they are included because the functions that create a .rly file are also used for building a session file, where they are needed.

Length01 *a track element can have two tracks (crossovers, bridges and points), or one (all other active elements). 01*
Length23 *corresponds to the first or single track, and 23 to the other*
SpeedLimit01 *as above*
SpeedLimit23 *as above*
LocationName *name not used for timetabling, only for identification purposes: platforms, non-station named locations, concourses and footcrossings have LocationNames*
ActiveTrackElementName *name used either in the timetable or for a continuation (continuation names are not used in timetables as trains can't stop there). Only active track elements where there are platforms or non-station named locations have ActiveTrackElementNames*
End of element marker
if 4 aspect signal 4*****
if 3 aspect signal 3*****
if 2 aspect signal 2*****
if ground signal G*****
if not a signal ******

c) Inactive track elements:
Number of inactive elements
Identifier
Inactive elements

Individual element information
Element number 0 *for first*, (Number of inactive elements - 1) *for last*
SpeedTag *see SpeedTag notes below*
HLoc *see above note under General information*
VLoc as above
LocationName *see description in Active track elements*
End of element marker
******

d) Text: *(no identifiers or end of item markers - individual items can be distinguished by text and font name)*
Number of text items *(includes location names)*
HPos *this and VPos correspond to the pixel position of the top left hand corner of the first text character*
VPos
Text string
Font name
Font size)
Font colour) integers
Font charset)
Font style)

e) Preferred directions:
A single preferred direction is an enhanced type of track element with additional information relating to the preferred direction. Each preferred direction element has only a single direction and a single track, so a bridge with both tracks having bidirectional preferred directions will be represented by four separate preferred direction elements.

Number of preferred direction elements *if no preferred directions are set and there are no user graphics this is zero and is the last entry in the file*

Individual element information
Element number 0 *for first, (Number of preferred direction elements - 1) for last*
TrackVectorPosition *the element number in the sequence of all active track elements that the preferred direction element relates to*
ELink *the entry link number- see Link notes*
ELinkPos *the entry link array position (in Link[0-3])*
XLink *the exit link number*
XLinkPos *the exit link array position (in Link[0-3])*
EXNumber *a number identifying the required graphic for display purposes*
CheckCount *an internal check value used when building preferred directions*
IsARoute *preferred directions are also used for routes, so this indicates whether it is a route (value 1) or not (value 0)*
AutoSignals *a marker for routes to indicate whether or not it's an automatic signal route element (value 1 if it is)*
PrefDirRoute *a marker for routes that are on preferred directions (value 1) or not (value 0).*
if the element is a route element and AutoSignals is set then PrefDirRoute is also set.
if the element is a route element and neither of the above is set then it's an unrestricted route element
End of element marker
If not last element ******
If last element ********** *with no user graphics present this is the last entry in the file*

f) User graphics:

Number of user graphics *This number is only included if there are user graphics present, as indicated by the active track element identifier being* **Active elements**1 - *see b) above*

Individual graphic information

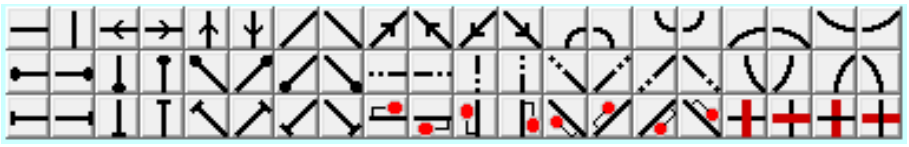
Graphic file name *must be of type .bmp, .jpg, .png or .gif*

HPos *this and VPos correspond to the pixel position of the top left hand corner of the graphic*

VPos

SpeedTag notes:

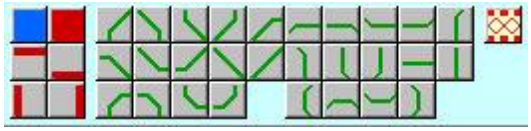
Windows includes a type of button called a SpeedButton, which is used for selecting particular elements when building a railway. This button has an attribute called a Tag, used to identify a particular SpeedButton in an application. In railway.exe this is called SpeedTag and is an integer. The diagram below shows all the track elements with their corresponding SpeedTag numbers.



1	2	125	126	127	128	18	19	140	142	141	143	3	4	5	6	20	21	22	23
88	89	90	91	92	93	94	95	80	81	82	83	84	85	86	87	26	27	24	25
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	129	130	145	146

(top row speedtag numbers)
(second row speedtags)
(bottom row speedtags)

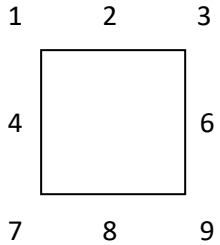
7	8	9	10	11	12	13	14	133	135	139	15	16	48	49	51	50
28	29	30	31	32	33	34	35	137	136	138	44	45	52	53	55	54
36	37	38	39	40	41	42	43	132	134		47	46	57	56	58	59



131	96	97	98	99	100	101	111	112	113	114	115	144
76	77	102	103	104	105	106	116	117	118	119	120	
78	79	107	108	109	110		121	122	123	124		

Link notes:

Each track element has a 4-integer array of links, corresponding to the positions where the track links to other elements. These are numbered as follows:-



Some elements have no links (platforms, concourses, parapets etc) but others have a maximum of four links (crossovers, bridges and points (although points only have 3 they are treated as having 4 where the leading link is listed for both tracks - it makes other program functions simpler). The simpler elements have just 2 links. For example a diagonal crossover would have numbers 1, 9, 3, and 7 in positions Link[0], Link[1], Link[2] and Link[3]. Number 5 is omitted for

better symmetry - all opposites add to 10, all diagonals are odd, all horizontals and verticals are even. Points with straight track 4 to 6 and diverging track 4 to 9 would have numbers 4, 6, 4, and 9 in the four array positions as above. Unused links take the value -1 to indicate that they are not set.

The order of the links is vitally important as the position of a link is often used to find specific information. For example all continuations, buffers and gaps use Link[0] for the continuation/buffer/gap end and Link[1] for the other end; all points use Link[0] & Link[2] for the leading end, Link[1] for the straight or left-hand trailing link and Link[3] for the diverging or right-hand trailing link; and all bridges use Link[0] & Link[1] for the top track and Link[2] and Link[3] for the bottom track and similarly for others. Link information is not provided in .rly files because it can be found from the SpeedTag number.

Each element needs to know which other elements it connects to so that preferred directions and routes can be set and train movements properly controlled. To allow this each element contains two more 4-integer arrays Conn[0-3] and ConnLinkPos[0-3]. Conn[0-3] indicates the connecting element (i.e. its number in the sequence of all active track elements) in the same order as the order of Link[0-3]. ConnLinkPos[0-3] indicates the connecting element link array position again in the same order as the order of Link[0-3]. All values for Conn[0-3] and ConnLinkPos[0-3] are set when the user clicks the  'Link all track together' button, and if any can't be set then an error message is given and the offending element highlighted.

Sample start of LU Metropolitan Line (JKWok).rly: (copied from Notepad++)

NUL = NUL character
CR LF = carriage return & line feed = new line

·v2.5.0NULCRLF	Saved by program version v2.5.0
-46CRLF	Home horizontal offset -46
-12CRLF	Home vertical offset -12
2034CRLF	2034 active track elements
Active·elements1NULCRLF	Identifier with '1' at end indicating that there are user graphics
0CRLF	first element (no. 0)
20CRLF	SpeedTag = 20 = 
-46CRLF	HLoc
0CRLF	VLoc
100CRLF	Length01 (100m)
-1CRLF	Length23 not set (= -1) since it's a simple single track element
48CRLF	SpeedLimit01 (48km/h)
-1CRLF	SpeedLimit23 not set
NULCRLF	No LocationName
NULCRLF	No ActiveTrackElementName
******NULCRLF	End of element marker - not a signal so ******
1CRLF	2nd element (no. 1)
69CRLF	SpeedTag = 69 and it's a ground signal - see below - so it displays as 
-45CRLF	HLoc
0CRLF	VLoc
0CRLF	It's a signal so this is its Attribute = 0 = red (all signals red for a .rly file)
0CRLF	Again a signal so CallingOnSet = 0 = not available (all same for .rly file)
100CRLF	Length01 = 100m
-1CRLF	Length23 not set
48CRLF	SpeedLimit01 = 48km/h
-1CRLF	SpeedLimit23 = not set
NULCRLF	No Location Name
NULCRLF	No ActiveTrackElementName
G*****NULCRLF	End of element marker - it is a ground signal so prefixed by 'G'

The above includes lines 1 to 29 of the file. It continues until line 57,864 - which is quite big.
The biggest so far is Xeon's Shanghai Metro with 1,266,667 lines - beat that!!

****

Session files (.ssn)

Broad structure:

a) Interface
b) Railway (Track elements [active & inactive], Text, Preferred directions & User graphics)
c) Remaining items

Detailed structure:

a) Interface:

Program version followed by : ***Interface*** followed by excess LC down minutes
'Excess LC down minutes' represents the accumulated excess minutes that a level crossing has been down during the session, used in calculating the performance score. It is stored here as part of a string as a convenient place to include it without disturbing the general structure of the file for compatibility purposes. It was included from v2.2.0 as it had been omitted in error earlier. As will be seen later additions now have their own area at the end of the file.

PreStart or NotPreStart
This information is needed in subsequent loads to allow some actions (for PreStart) that are not allowed otherwise, such as setting routes and train failure information.

Railway title
Timetable title
PreferredRoute *boolean for 'preferred direction route'. 0 (no) or 1 (yes) depending on whether the route type selected follows preferred directions or not*
PreferredRoute *Saved again - prior to v2.7.0 this used to be 'ConsecSignalsRoute', indicating that the route had to be set from signal/buffer/continuation to the next signal/buffer/continuation, i.e. consecutively, but up to then it always had the same value as PreferredRoute because preferred routes could only be set consecutively (the original design intent allowed for preferred routes to be set from signal/buffer/continuation to any following signal/buffer/continuation but it wasn't taken up until v2.7.0). ConsecSignalsRoute couldn't be set differently at this point in the session file because if it was loaded into an earlier program version the load would fail. To maintain backward compatibility the value for ConsecSignalsRoute is stored instead at the end of the session file where earlier program versions will ignore it - see later.*

AutoSigsFlag *0 (no) or 1 (yes) as appropriate.*

Horizontal offset of the display when saved *(all offsets can be positive or negative)*
Vertical offset of the display when saved
Horizontal 'Home' offset
Vertical 'Home' offset
Horizontal 'Zoom-out Home' offset
Vertical 'Zoom-out Home' offset

Offsets are explained above under 'General information' for railway files

Warning 1 *this group gives the warnings in red at the top of the display area, 10 in all*
Warning 2
Warning 3
Warning 4
Warning 5
Warning 6
Warning 7
Warning 8

Warning 9
Warning 10

On-time arrivals *this group gives all the performance indicators*
Late arrivals
Early arrivals
On-time passes
Late passes
Early passes
On-time departures
Late departures
Missed stops
Other missed events
Unexpected exits
Incorrect exits
SPAD events
SPAD risks
Crashed trains
Derailments
Total late mins
Total early mins
Total late pass mins
Total early pass mins
Total late departure mins

'Excess LC down minutes' should have been included in the above group but it was a late addition and would have altered the file structure if included later. It is now included with the string ***Interface*** (see above).

b) Railway (Track elements [active & inactive], Text, Preferred directions & User graphics):

Track
Track element information *(active & inactive) - same format as in railway files (see above)*
Text
Text information *same format as in railway files (see above)*
PrefDirs
Preferred direction information *same format as in railway files (see above)*
User graphic information *This is only included if there are user graphics present, as indicated by the active track element identifier being **Active elements**1 in the track element information. It takes the same format as in railway files (see above)*

c) Remaining items

Routes
Number of routes
Next route ID *this is the identification number for the next route to be set*

individual route information - this is a long section because each route consists of a series of route elements, each of which has the same structure as preferred direction elements as specified above for a railway file.
Route ID
Number of route elements

Individual route element information for the particular route (uses preferred direction elements with boolean IsARoute set)
Element number 0 *for first, (Number of preferred direction elements - 1) for last*
TrackVectorPosition *the element number in the sequence of all active track elements that the route element relates to*
ELink *the entry link number- see Link notes*
ELinkPos *the entry link array position (in Link[0-3])*

XLink *the exit link number*
XLinkPos *the exit link array position (in Link[0-3])*
EXNumber *a number identifying the required graphic for display purposes*
CheckCount *an internal check value used when building preferred directions and routes*
IsARoute *this indicates whether it is a route (value 1) or a preferred direction (value 0), in this case all have value 1*
AutoSignals *a marker for routes to indicate whether or not it's an automatic signal route element*
PrefDirRoute *a marker for routes that are on preferred directions (value 1) or not (value 0).*
if the element has AutoSignals set then PrefDirRoute is also set.
if neither of the above is set then it's an unrestricted route element

End of element marker
If not last element in the route ******
If last element in the route **********

The above element information is repeated for each element in the route, and then repeated again after the next Route ID for the next route and so on until all routes and all route elements have been specified

Locked routes
Number of locked routes

individual locked route information
Locked route number
TruncateTrackVectorPosition *a locked route is normally part of an existing route and will truncate that route when the lock times out; this is the position of the first (nearest to the start) element to be truncated*
LastTrackVectorPosition *the position of the end element in the route - all elements between TruncateTrackVectorPosition and LastTrackVectorPosition inclusive are locked*
LastXLinkPos *the exit link position of the last element in the locked route*
LockStartTime *the timetable clock value when the lock began*

ContinuationAutoSigEntries this represents the start of the list of continuation signal states. When a train has exited at a continuation, the signals before the exit change back to green in stages as the train moves out of track sections beyond the exit. This list stores the values needed to make the necessary signal aspect changes
Number of continuation exits that are in the process of changing signal aspects

individual continuation exit information
RouteNumber *the route number (position in the list of all routes) of the route that the continuation is in*
AccessNumber *the number of times the signal changing function has been accessed - starts at 0 and increments after each change*
FirstDelay } *Delays in seconds before consecutive signal change - these correspond to the times taken for trains*
SecondDelay } *to pass subsequent signals outside the boundaries of the railway. After the third delay the signal*
ThirdDelay } *nearest to the continuation that was red when the train passed it has changed to green. If the signal(s) are other than 4-aspect they will change to green after the second or first delay, and subsequent delay timeouts will have no effect.*
PassoutTime *the timetable clock time at which the train exits from the continuation (expressed as a floating point number representing the number of days and day fractions from the midnight before the timetable start time. The value is more than precise enough to indicate seconds.)*

BarriersDownVector *represents the start of the list of level crossings (LCs) with barriers down (i.e. open to trains) - sessions can't be saved when a level crossing is changing state so there is no need to store this information. Level crossings with barriers raised are in the normal state and are stored in the railway section of the session file.*
Number of LCs with barriers down

individual barrier down LC information

TypeOfRoute *indicates the type of route that lies across the barrier - 0 = unrestricted, 1 = preferred direction (can't have an automatic signal route across an LC), 2 no route (2 was added at v2.6.0 for manual operation where there need not be a route across a LC)*

ReducedTimePenalty *boolean marker (0 = no reduced penalty) that is set when a train is present on one of the elements of the LC - used to provide a 3 minute penalty allowance in the performance score*
BarrierState *can be Raising, Lowering, Up or Down, though in all cases here this will be down. There is no need for this value in a session file but it is an LC parameter used in other circumstances and it is easier from a programming point of view to store the LC parameters as a whole rather than miss some out.*
ChangeDuration *duration of the level crossing changing period*
BaseElementSpeedTag *SpeedTag value for the base element of the LC (may be horizontal or vertical)*
HLoc *horizontal position of the LC element*
VLoc *vertical position of the LC element*
StartTime *the starting time for LC changing - again not needed for a session file*

Timetable *marks the start of the stored timetable*
Timetable information *stored in the same format as a .ttb file - see later timetable file specification*
End *marks the end of the stored timetable*

Immediately following the timetable is the internal timetable data. The timetable isn't used directly by the program, it is first converted into an internal format consisting of a series of parameters for each train, the whole contained in a vector called the TrainDataVector, which contains all the timetable service entries (not every individual service because it excludes repeats) and is more easily accessible to the program.

The TrainDataVector includes within it another vector called theTrainOperatingDataVector, the size of which represents the total number of trains with the same data (i.e. number of repeats + 1). Each element of this vector contains three entries:
TrainID: which is -1 for trains that haven't yet entered the railway, otherwise the train's unique identification number;
EventReported: which is the last event for that train reported to the performance log, the value is 'NoEvent' (=0) for trains that haven't yet entered the railway; and
RunningEntry: which can take one of three values -'NotStarted' (=0) - not yet entered the railway; 'Running' (=1) - present on the railway; and 'Exited' (=2) - left the railway.

The Session File records the following values:

Total number of timetable service entries (size of TrainDataVector) = no. of distinct trains excluding repeats
then for each service entry
Total number of trains *(= repeats + 1)*
then for each train
TrainID *int*
EventReported *enum expressed in the file as a short integer*
RunningEntry *enum expressed in the file as a short integer*

TimetableClock
Clock value *stored as a floating point number representing the number of days and day fractions from the midnight before the timetable start time. The value is more than precise enough to indicate seconds.*

Trains
Number of trains currently on the railway

Individual train information: a great deal of information is needed for each train, all listed here and some of it quite complex. Rather than try to explain it all here anyone who is in need of specific information should contact me (Albert) either on Discord or at railwayfeedback@gmail.com and I shall try to clarify anything required.

HeadCode
RearStartElement
RearStartExitPos
StartSpeed
SignallerMaxSpeed
HoldAtLocationInTTMode
RepeatNumber
IncrementalMinutes
IncrementalDigits
Mass
FrontelementSpeedLimit
FrontElementLength
EntrySpeed
ExitSpeedHalf
ExitSpeedFull
TimetableMaxRunningSpeed
MaxRunningSpeed
MaxExitSpeed
MaxBrakeRate
BrakeRate
PowerAtRail

FirstHalfMove
OneLengthAccelDecel
EntryTime
ExitTimeHalf
ExitTimeFull
ReleaseTime
TRSTime
LastActionTime
CallingOnFlag
BeingCalledOn
DepartureTimeSet
TrainMode
TimetableFinished
LastActionDelayFlag
SignallerRemoved
TerminatedMessageSent
Derailed
DerailPending
Crashed
StoppedAtBuffers
StoppedAtSignal
StoppedAtLocation
SignallerStopped
StoppedAfterSPAD
StoppedForTrainInFront
NotInService
Plotted
TrainGone
SPADFlag
TimeTimeLocArrived
HOffset[0]
HOffset[1]
HOffset[2]
HOffset[3]
VOffset[0]
VOffset[1]
VOffset[2]
VOffset[3]
PlotElement[0]
PlotElement[1]
PlotElement[2]
PlotElement[3]
PlotEntryPos[0]
PlotEntryPos[1]
PlotEntryPos[2]
PlotEntryPos[3]
TrainCrashedInto
Straddle
NextTrainID
TrainID
LeadElement
LeadEntryPos
LeadExitPos
MidElement
MidEntryPos
MidExitPos
LagElement
LagEntryPos
LagExitPos
ColourNumber
ForwardHeadCode
TrainDataEntryValue
ActionVectorEntryValue
End of train marker

Performance file
Performance information *stored here as a plain text file, formatted as a performance log*
End of performance file

Additions after v2.3.1 *this is a marker for any more information that need to be stored in versions later than v2.3.1. It doesn't disturb the structure of the file because earlier versions stop loading after the performance file. It was added at v2.4.0 Beta when failed train information was required. Since then no further additions have been made (up to v2.6.1)*
Mean time between failures *the time as set when the session was in pre-start mode*
Number of failed trains
Failed Trains

individual failed train information
TrainID
OriginalPowerAtRail *the power of a failed train is zero, but the original power is stored so that it can be repaired if required*
-1 *marker for end of failed trains*
End of file at v2.4.0 *marker for the end of file at v2.4.0.*

ConsecSignalsRoute *added here in place of the earlier file location (after the first Preferred direction route) after v2.7.0 as it can now differ from that value.*

The three boolean values PreferredRoute, ConsecSignalsRoute and AutoSigsFlag have the following values (in that order) for each route type from v2.7.0:-

Automatic signals route	1, 1, 1
Preferred direction route - signal/buffer/continuation to next signal/buffer/continuation	1, 1, 0
Preferred direction route - signal/buffer/continuation to any following signal/buffer/continuation	1, 0, 0
Unrestricted route	0, 0, 0

End of file at v2.7.0 *marker for the end of file, as from v2.7.0*

EarlyExits *int*
OnTimeExits *int*
LateExits *int*

TotEarlyExitMins *double*
TotLateExitMins *double*
End of file at v2.9.1 *marker for the end of file, as from v2.9.1. This marker isn't read by the program so it may change if additional information is added in later versions. It won't affect earlier versions because they will already have stopped reading the file.*

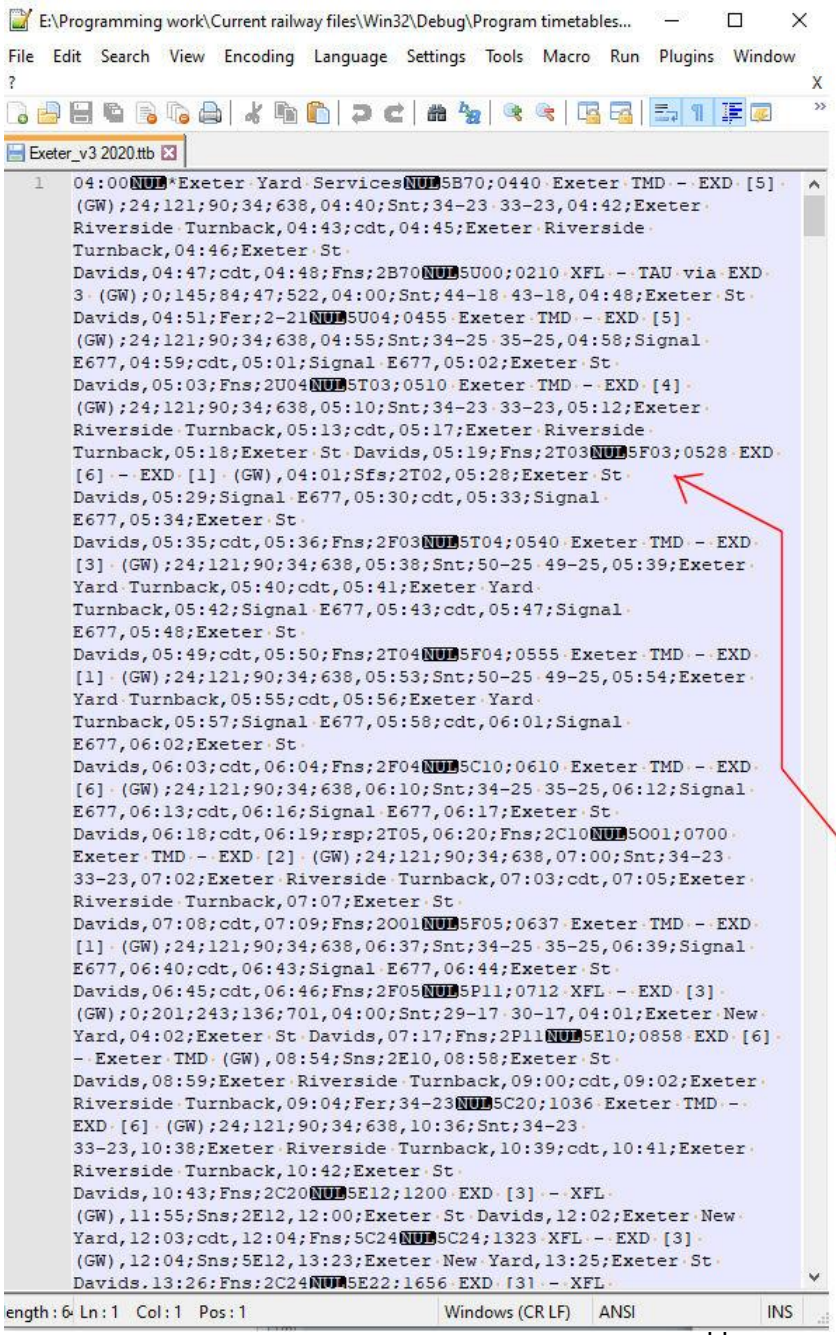
****

Timetable files (.ttb)

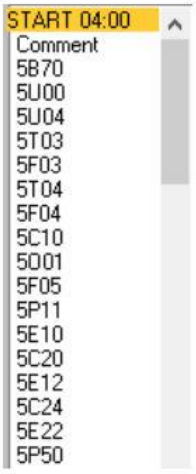
These contain the information that is provided when the program creates a timetable using the timetable editor and the format is the same apart from the end of line delimiters (special characters that define line ends in a text file).

The normal delimiter is CRLF (carriage return - character 13 in ASCII followed by line feed - character 10 in ASCII). In a .ttb file line ends between service events are delimited by commas, and between services, comments and the start time (called entries in the timetable editor) by NUL characters (0 in ASCII).

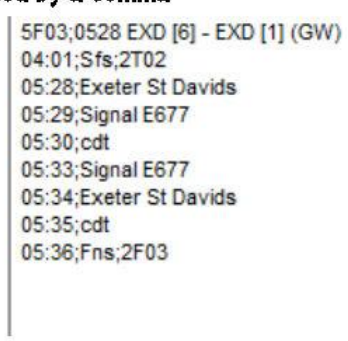
To show this the start of Oxalin's Exeter timetable (Exeter_v3 2020.ttb) is illustrated below in Notepad++. Note that wordwrap is turned on in Notepad++ in this illustration as without it the timetable is all on a single line because there are no CRLF characters. Alongside it are shown the timetable editor views.



This is the list of service entries as shown in the timetable editor. In the .ttb file each entry is delimited by a NUL character



This is an individual entry (for 5F03) as shown in the timetable editor. In the .ttb file each line is delimited by a comma



****

Config.txt

This is a very simple text file that stores information that is read when railway.exe runs. Currently (as at v2.6.1) the information consists of signal handedness (left or right), background colour (black, white or blue), and folder paths for railway, timetable and session files.

Typical contents are:-

Signals=right *anything other than 'right' in lower case is treated as left*
BgndCol=blue *anything other than 'white' or 'blue' in lower case is treated as black*
RLYLocn=E:\Programming work\Current railway files\Win32\Debug\Railways
TTBLocn=E:\Programming work\Current railway files\Win32\Debug\Program timetables
SSNLocn=E:\Programming work\Current railway files\Win32\Debug\Sessions
If these are edited please don't change anything before the = sign as the program looks for that text and anything else will cause the line to be ignored.

Changing a timetable in a session file whilst retaining routes

This is an example of how direct file manipulation can be helpful in some circumstances. Here a new timetable file is inserted into a session file.

There is no method within Railway Operation Simulator (ROS) to save a railway containing routes and then load a new timetable. This isn't normally a problem because automatic signal routes can be reset within a minute or two after a new timetable is loaded into a railway without routes. It is a problem however with very big railways like Krizar's London Underground, when automatic signal route setting can take a long time.

To get round this restriction session files can be manipulated directly, by temporarily saving all the routes and associated track etc, creating a new session file without routes but containing a new edited timetable, then inserting the routes and track back into this new session file. BUT, very importantly, a normal text editor such as Notepad or Wordpad can't be used, because all ROS files contain NUL characters (code '0' in ASCII), and normal text editors ignore these characters. Therefore Notepad++ (available from https://notepad-plus-plus.org/) is recommended. It is a free code editor that doesn't itself make any changes to loaded files.

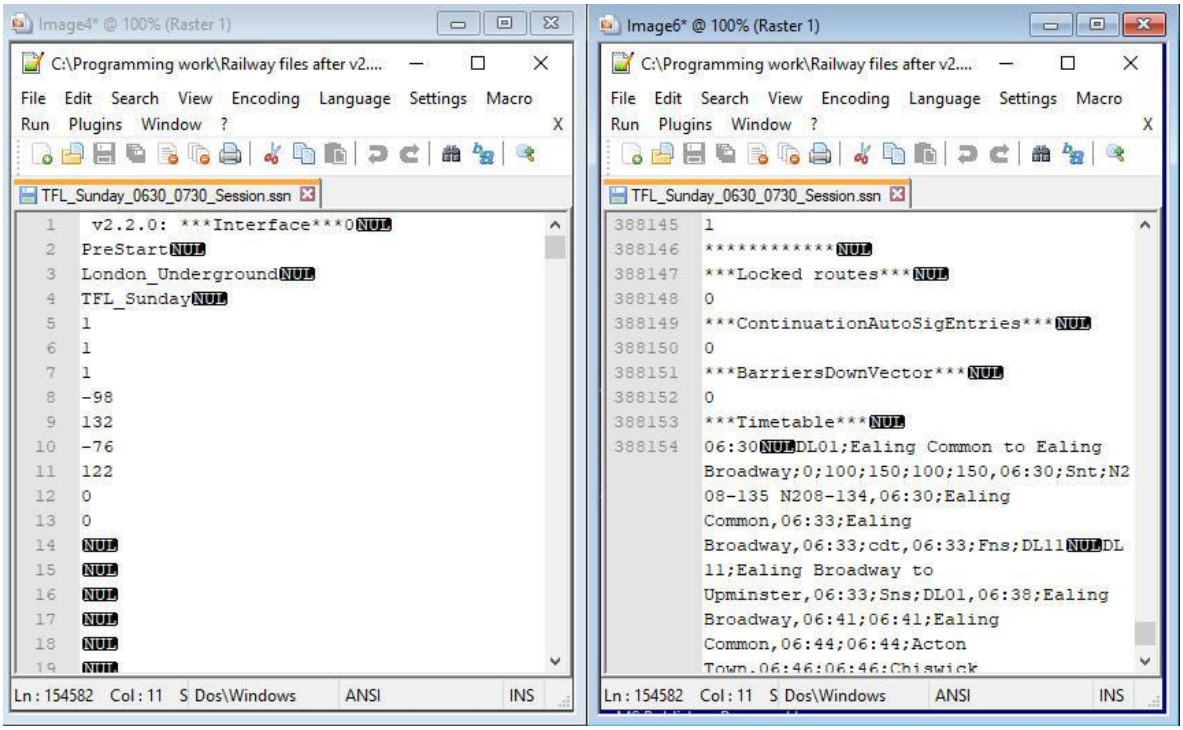
The following detailed description sounds very complicated but after a time or two it should become quite intuitive and therefore reasonably quick. Using Krizar's railway as an example, load the railway in ROS as normal, then edit the timetable that came with it. When the new timetable is as required and validates successfully save it, reload it, select 'Operate' and immediately save a session file without any routes.

Now load the original session file (TFL_Sunday_0630_0730_Session.ssn) into Notepad++ (or another suitable editor if you prefer - bearing in mind what was said earlier about text editors), and also load the new session file that you just saved. These will both load in Notepad++ under their own tabs.

Very important: remember which is which!!

Click the original session file tab (TFL_Sunday_0630_0730_Session.ssn) and scroll down to the timetable section which begins at line 388153. Now select everything up to line 388152 inclusive as described below so it can be inserted into the new session file.

This is what the start of the file and the timetable section look like in Notepad++:-

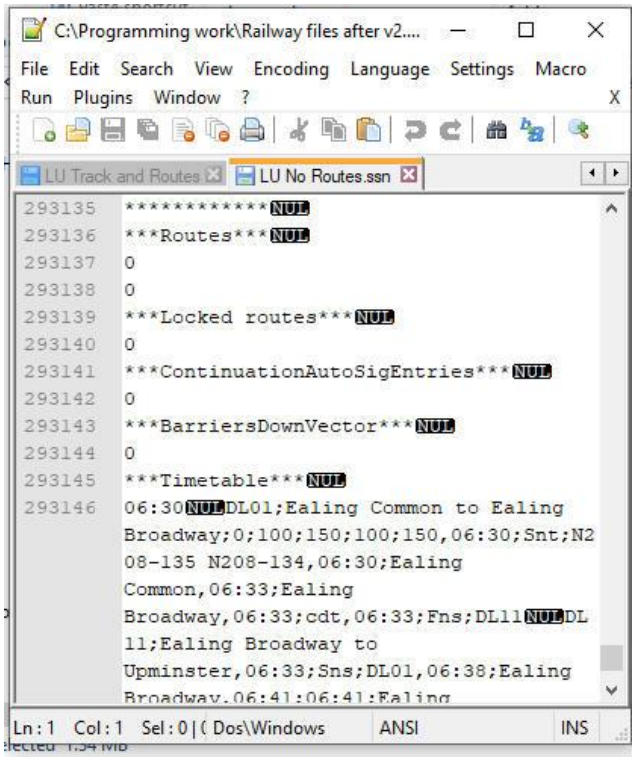


To find quickly the start of the timetable section hold the Ctrl key down and press 'f'. This brings up the search screen. Enter 'timetable' and press 'Enter'. This will find the timetable start line which is 388153 - '***Timetable***NUL'. Click the left mouse button at the start of this line then hold down Ctrl and Shift together and press Home. This will select everything before the timetable section.

Now, click 'Edit', 'Paste Special' (this no doubt sounds inappropriate as we want to copy, not paste, but Notepad++ uses this for copying the content as it is rather than as text), then 'Copy Binary Content'. Click 'File' then 'New', then 'Edit', 'Paste Special' and 'Paste Binary Content', and the selected text should be pasted (with all the necessary NULs) into this new file. Check it begins with the version number, ends with '***BarriersDownVector*** NUL' followed by a zero, and is 388152 lines long.

All good? Great! Now save it as it will probably be needed later on for future timetable changes. **But don't save as 'Normal text file (*.txt)' or the NULs will be lost, select 'All types (*.*)' and give it a name that doesn't end in .txt.**

Now go to the new session file you saved with no routes, and go to the line '***Timetable***NUL' using the search function as before. This will now be at line 293145 as shown below:-



Left click the mouse at the start of line 293145, press Ctrl, Shift and Home, and everything above the timetable section should be selected as before. Press 'Del' as we don't need this section. Now paste the earlier saved section - left click the mouse at the start of the file, i.e. the start of line 1, which contains '***Timetable***NUL'. Now go back to the saved track and routes file (here called 'LU Track and Routes') by clicking its tab, then 'Edit', 'Select All', 'Edit', 'Paste Special', 'Copy Binary Content', then return to the session file with no routes by clicking its tab, and the cursor should still be at the start of the file. Click 'Edit', 'Paste Special' and 'Paste Binary Content' and the earlier track and routes should now be before the timetable section, which should now be at line 388153 - as before. Now click 'File' and 'Save As', and give the revised session file an appropriate name, keeping .ssn at the end.

If all has gone well this should load satisfactorily into the program, and you now have a new timetable with all the routes as originally set.

Note that the session file loader is very unforgiving. A wrong line or a missing NUL and it won't load, and it won't be obvious why it hasn't. Be sure therefore to follow the above steps accurately.

Note also that this method won't work for a railway where trains have been moving, because the signal aspects won't match the new timetable for which trains haven't started to move.

If you have any problems please post any offending files on Discord in the feature-requests channel and I'll try to help. Also if you notice anything wrong in this note please let me know and I'll correct it - thanks.

Albert

July 2021