# 1. Introduction.

The timetable concept is not a replacement for the activity definition, but is an alternative way in defining both player and computer-controlled (AI and Static) trains.
In an activity, the player train is defined explicitly, and all AI trains are defined in a traffic definition. Static trains are defined separately.
In a timetable all trains are defined in a similar way. On starting a timetable run, the required player train is selected from the list of available trains. In the timetable definition itself, no distinction is made between running trains - all running trains can be selected as player train, and if not selected as such they will be run as AI trains. Static trains are also defined in the same way but cannot be selected as player train.
As a result, the number of different 'activities' that can be played using the same timetable file is equal to the number of trains which is defined in the timetable.

The development of the timetable concept is still very much a work in progress. This document details the state as it is at the moment, but also includes items yet to be produced, or items which have yet to be developed further.
To distinguish between these items, the following styles are used.
*Items shown in red italics are not yet included and are yet to be produced. Details of these items are likely not yet finalized and are indicative only.*
*Items shown in black italics are available but only in a provisional implementation, or in a limited context. Further development of these items is still required.*
**Important aspects where use of specific OR or MSTS items for timetables differ significantly from its use in an activity are shown in bold.**
Apart from the items indicated as above, it should be realised that as work continues, all items are still subject to change.

# 2. General.

## 2.1. Data definition.

The timetable data is defined in a Spreadsheet, and saved as a *.csv file (character separated file). As separation character, either ',' (comma) or ';' (semi-colon) must be used.
Do not select space or tab as separation character.
As ';' or ',' are possible separation character, these symbols must not be used anywhere within the actual data. Enclosure of text by quotes (either single or double) has no effect.

*At present, all trains must be defined in a single timetable file. Later on, it will be possible to split the timetable in multiple files, and create an overall list file detailing all required timetable files. This list file must be in JSON format.*

## 2.2. File structure.

The saved *.csv files must be renamed with extension *.timetable_or. The timetable files must be placed in a OpenRails subdirectory in the route's Activities directory.

*The timetable list file, in JSON format, must be placed in the same directory, and must have the extension *.timetablelist_or.*

## 2.3. File and train selection.

When starting a timetable run, the required timetable file must be selected.
*When using a timetable-list, the required actual timetable must be selected from the list of timetables contained in the timetable-list.*
After selecting the required timetable, a list of all trains contained in that timetable is generated and the required train can be selected.
*Further options will be made available to select which trains out of the timetable are included as AI trains.*
*As trains may be defined to run on specific days only, also the day of the week can be selected.*
***The options with relation to AI trains are being reconsidered and may well be removed from the final version.***
Season and weather can also be selected, these are not preset within the timetable definition.

# 3. Timetable definition.

## 3.1 General.

A timetable consists of a list of trains, and, per train, the required timing of these trains. The timing can be limited to just the start time, or it included intermediate times as well.
*At present, intermediate timings are limited to 'platform' locations as created using the MSTS Route Editor.*
Each column in the spreadsheet contains data for a train, each row represents a location. A cell at the intersection of a train and location contains the timing data for that particular train at that location.
Special rows and columns can be defined for general information or control commands.
The first row for each column contains the train definition.
The first column for each row contains the location definition.
The cell at the intersection of the first row and first column **must be empty**.

This paragraph only lists the main outline, full detailed description will follow in the next paragraphs.

## 3.2 Column definitions.

A column is defined by the contents of the first row.
Default, the first row defined the train name.
Special columns can be defined using the following syntax :

| | |
|---|---|
| #comment | : column contains comment only and is ignored when reading the timetable |
| <blank> | : column is extension of preceding column |

## 3.3 Row definitions.

A row is defined by the contents of the first column.
Default, the first column defines the stop location.
Special columns can be defined using the following syntax :

| | |
|---|---|
| #comment | : row contains comment only and is ignored when reading the timetable |
| <blank> | : row is extension of row above |
| #path | : defines train path |
| #consist | : defines train consist |
| *#create* | *: defines how train is created (if run as AI train)* |
| #start | : defines time when train is started |

| | |
|---|---|
| #note | : defines general notes for this train |
| #direction | : defines direction of this train |
| *#runs* | *: defines information on when this train should be included in the timetable* |
| #dispose | : defines how train is handled after it has terminated |

## 3.4 Timing details.

Each cell which is at an intersection of a train column and a location row, can contain timing details for that train at that location.
*Presently, only train stop details can be defined. Later on, passing times can also be defined, these passing times can be used to determine a train's delay.*
Control commands can be set at locations where the train stops, but can also be set for locations where no timing is inserted as the train passes through that location without stopping.

# 4. Timetable data details.

## 4.1 Timetable description.

Although #comment rows and columns are generally ignored, the contents of the cell at the intersection of the first #comment row and first #comment column is used as the timetable description.
*When using a multiple timetable list definition, the timetable description is used for selection of timetable from this list.*
*Also, when referencing a train from another timetable file, the train reference must contain the timetable description of the timetable which contains the referenced train.*

## 4.2 Train details.

The train name as defined in the first row must be unique for each train in a timetable file.
This name is also used when referencing this train in a train command, see details below.
The sequence of trains is not important.

## 4.3 Location details.

*At present, the possible locations are restricted to 'platforms' as defined in the MSTS Route Editor.*
Each location must be set to the 'Station Name' as defined in the platform definitions.
The name used in the timetable must exactly match the name as used in the route definition (*.tdb file), otherwise the location can not be found and therefor not processed.
Also, each location name must be unique, as otherwise its position in the train path could be ambiguous.
The sequence of the locations is not important, as the order in which the stations are passed by a train is defined in that train's path. For the same reason, a train's path can be set to just run in between some of the locations, or be set to bypass certain stations.

## 4.4 Timing details.

Each cell at an intersection of train and location can contain the timing details of that train at that location.
Times are defined as HH:mm, and the 24-hour clock must be used.
If a single time is inserted it is taken as the departure time (except at the final location).
If both arrival and departure time are to be defined, these must be separated by '-'.
Additional control commands can be included. Such commands can also be set for locations where the train does not stop and therefor has no timing details, but the train must pass through that location for the commands to be effective.

Although a location itself can be defined more than once in a timetable, it is not possible to defined timing details for trains for a location more than once. If a train follows a route which takes it through the same location more than once, the train must be 'split' into separate train entries.

## 4.5 Special columns.

### 4.5.1 #comment column.

A column with the #comment definition in the first row is a comment column and is ignored when reading the timetable, except for the cell at the intersection of the first comment column and the first comment row.

### 4.5.2 <blank> column.

A column with a blank (empty) cell in the first row is taken as a continuation of the preceding column. It can be used to insert control commands which apply to the details in the preceding column. This can be useful when timings are derived automatically through formula in the spreadsheet as inserting commands in the timing cell itself would exclude the use of such formula.

## 4.6 Special rows.

### 4.6.1 #comment row.

A row with the #comment definition in the first column is a comment row and is ignored when reading the timetable, except for the cell at the intersection of the first comment column and the first comment row.

### 4.6.2 <blank> row.

A row with a blank (empty) cell in the first column is taken as a continuation of the preceding row.

### 4.6.3 #path row.

The #path row defines the path of that train. The path must be a *.pat file as defined by the MSTS Activity Editor, and must be located in the route's Path directory. This field is compulsory.
The timetable uses the same paths as defined for activities.

**However, waiting points must not be defined in paths for use in timetables as the processing of waiting points is not supported in the timetable concept.**
**Waiting points within a timetable must be defined using the specific control commands.**
*Specific pathing option for use in timetables will be introduced later.*

### 4.6.4 #consist row.

The #consist row defined the consist used for that train. This field is compulsory.
However, if the train is run as an AI train and it is 'formed' out of another train (see below), and that other train is included in the actual running of the timetable, the consist information is ignored and the train uses the consist of the train out of which it was formed.
For the player train, the consist is always used even if the train is formed out of another train.
The consist definition must be a *.con file as defined by the MSTS Activity Editor, and must be stored in the defined consist directory.
*Specific consist selection options for use in timetables will be introduced later.*

### 4.6.5 #start row.

The #start row defines the time at which the train is started. It must be defined as HH:mm, and the 24 hour clock must be used. This field is compulsory.

Use of start time for AI trains :

- *When #create is set, the train is created as detailed in the create definition.*
  *The time defined in #start is the time the train becomes 'active', i.e. it starts clearing its route etc.*
- When a train is formed out of another train and this other train is included to run in the timetable, the time defined in #start is only used to define when the train becomes active.
- When #create is not defined and the train is not formed or the train out of which it is formed is not run in the timetable, the time defined in #start is taken as the time at which the train is created. The train is then active immediately after creation.

Use of start time for player train :

- The time as defined in #start is used as the start time of the 'activity'.

*Use of start time for static train :*

- *For static trains, the #start row value must be set to $static.*
- *If a #create row value is defined, the train is created as detailed in that value, otherwise the train is created from the start of the activity.*

If a train is formed out of another train and this train is included in the timetable, then if this train is delayed and has not arrived before the defined start time, the starting this train is also

delayed until the train out of which it is formed has arrived. **This applies to both AI and player train.** This means that the start of the player activity can be delayed.
For details on starting and running of trains around midnight see special paragraph.

### 4.6.6 #create row.

*The #create row defines how an AI train is created before it is started.*
*Details are yet to be defined.*

### 4.6.7 #note row.

The #note row can be used to defined control commands which are not location related but apply to the full run of the train. It can also be used to set commands for trains which do not stop at or pass through any defined location. This row is optional.

### 4.6.8 #direction row.

The #direction row can be used to indicate a train's general direction. Any value can be used, e.g. 'Up' and 'Down', 'East' and 'West' etc.
*The information is used to deselect AI trains which run in the same direction as the player train. If such AI trains are not to be included, all AI trains with the same value for #direction is excluded from the timetable. Note that this selection is not yet implemented.*
This row is optional.
***Use of this row is being evaluated - it could be it will not be included in the final version.***

### 4.6.9 #runs row.

*The #runs row defines when a train is run, i.e. when a train can be included in the timetable.*
*Details are not yet defined.*

### 4.6.10 #dispose row.

The #dispose row defines what happens to an AI train when it has reached the end of its run, i.e. it has reached the end of the defined path.
The information in the #dispose row can detail if the train is to be formed into another train, and, if so, how and where. For details see the commands as described further down.
This row is optional and if included, the use per train is also optional.

If the row is not included or the field is not set for a particular train, the train is removed from the activity after it has terminated.
*The #dispose row presently does not affect the end of the run for the player train. However, future development may include the use of this definition to allow the activity to continue using the formed train as next player train.*

## 4.7 Control Commands.

### 4.7.1 General.

Control commands can be set to control train and signalling behaviour and actions.
There are four sets of commands available :

- Location commands
- Train control commands
- Create commands
- Dispose commands

### 4.7.2 Command syntax.

All commands have the same basic syntax.
A commands consists of :

- Syntax name : defines the control command
- Syntax value : set the value related to the command
  Not all commands take a value.
- Syntax qualifiers : adds additional information to the command
  Not all commands have qualifiers.
  Some qualifiers may be optional but others may be compulsory, or compulsory only in combination with other qualifiers.
- Syntax qualifier values : a qualifier may require a value

Command syntax :

```
$name = value /qualifier=value
```

Multiple values may be set, separated by '+'.
Note that any qualifiers always apply to all values.

### 4.7.3 Train reference.

Many commands require a reference to another train.
This reference is the other train's name as defined in the first row.
*If the train is defined in another timetable file, the reference must contain both the train name and the description of the timetable in which this train is defined :* **<train>:<description>**.

### 4.7.4 Location commands.

Location commands are :

- $hold
- $forcehold
- *$nosignalwait*

These commands are also available as train control commands and are detailed in that paragraph.

### 4.7.5 Train control commands.

All available train control commands are detailed below.
These commands can be set for each timing cell, i.e. at each intersection of train column and location row. The commands will apply at and from the location onward (if applicable).
Some commands can also be set in the #note row, in which case they apply from the start of the train. These commands are indicated by an asterisk (*) behind the command name.
The commands $hold and $nosignalwait can also be set as location commands.

$hold, $nohold and $forcehold

If $hold is set, it defines that the exit signal for that location must be held at danger up to 2 mins. before train departure.
An exit signal is allocated to a platform if this signals is beyond the end platform marker (in the direction of travel), but is still within the same tracknode - so there must not be any points etc. between the platform marker and the signal.
**Default, the signal will not be held.**
If set per location, it will apply to all trains, but can be overridden for any specific train by defining $nohold in that train's column.
If set per train, it will apply to that train only.

$forcehold will set the first signal beyond the platform as 'hold' signal, even if this signal is not allocated to the platform as exit signal. This can be usefull at locations with complex layout where signals are not directly at the platform ends, but not holding the signals could lead to delay to other trains.

*$nosignalwait*

*By default, a train will no be allowed to depart if the exit signal for that station is still at danger. However, in particular for small stations that signal may be some distance removed from the actual platform. Setting $nosignalwait will allow the train to depart from that station regardless of the signal state.*
*This command can be set per train or per location in which case it applies to all trains.*

$callon

This will allow a train to 'call on' into a platform occupied by another train.
For full details, see paragraph on relation between signalling and timetable.

$connect

Syntax : $connect=<train> /maxdelay=n /hold=h

Defines that a train is to wait at a station until another train has arrived, so as to let passengers make the connection between the trains.
The train will be timetabled to allow this connection, and the $connect command is set to maintain this connection if the arriving train is running late.
Note that the $connect command will not lock the signal. If the paths of this train and the arriving train conflict before the arriving train reaches the station, additional $wait or $hold commands must be set to avoid deadlock.

**Command value** : reference to train which is to be waited for, this is compulsory.

**Command qualifiers** :

/maxdelay=n : n is the maximum delay (**in minutes**) of the arriving train for which this train is held.
If the delay of the arriving train exceeds this value the train will not wait. The maximum delay is independent from this train's own delay.
This qualifier and its value are compulsory.

/hold=h : h is the time (**in minutes**) the train is still held after the other train has arrived, and relates to the time required by the passengers to make the connection.
This qualifier and its value are compulsory.

$wait *

Syntax : $wait=<train> /maxdelay=n /notstarted

Defines that a train is to wait for the referenced train to allow this train to proceed first.
The referenced train can be routed in the same or the opposite direction as this train itself.
A search is done for the first track section which is common to both trains, starting at the location where the $wait is defined, or at the start of the path if defined in the #note row.
If the start location is already common for both trains, then first a search is done for the

first section which is not common to both trains, and the wait is applied to the next first common section beyond that.
If the wait is set, the section will not be cleared for this train until the referenced train has passed this section. This will force the train to wait. The referenced train must exist for the wait to be valid.
However, if /notstarted is set, the wait will also be set even if the referenced train has not yet been started. This can be used where the wait position is very close to the start position of the referenced train, and there is a risk that the train may clear the section before the referenced train is started.

Care should be taken when defining a $wait at a location where the train is to reverse. As the search is performed for the active subpath only, a $wait defined at a location where the train is to reverse will not be effective as the common section will be in the next subpath after the reversal. In such a situation, the train should be 'split' the train into two separate definitions, one up to the reversal location and another starting at that location.

**Command value** : referenced train, this is compulsory.

**Command qualifiers** :

/maxdelay=n : n is the maximum delay (**in minutes**) of the referenced train for which the wait is still valid.
This delay is compensated for any delay of the train which is to wait, e.g. if maxdelay is 5 minutes, the referenced train has a delay of 8 minutes but this train itself has a delay of 4 minutes, the compensated delay is 4 minutes and so the wait is still valid.
This parameter is optional, if not set a maxdelay of 0 minutes is set as default.

/notstarted : the wait will also be applied if the referenced train has not yet started.

$follow *

Syntax : $follow=<train> /maxdelay=n

This command is very similar to the $wait command, but in this case it is applied to each common section of both trains beyond a part of the route which was not common.
The train is controlled such that at each section where the paths of the trains rejoin after a section which was not common, the train will only proceed if the referenced train has passed that position. The command therefor works as a $wait which is repeated for each such section.
The command can only be set for trains routed in the same direction.
When a wait location is found and the train is due to be held, a special check is performed to ensure the rear of the train is not in the path of the referenced train or, if it is, the referenced train has already cleared that position. Otherwise, a deadlock would result, with the referenced train not being able to pass the train which is waiting for it.

**Command value** : referenced train, this is compulsory.

**Command qualifiers** :

/maxdelay=n : n is the maximum delay (**in minutes**) of the referenced train for which the wait is still valid.
This delay is compensated by any delay of the train which is to wait, e.g. if maxdelay is 5 minutes, the referenced train has a delay of 8 minutes but this train itself has a delay of 4 minutes, the compensated delay is 4 minutes and thus the wait is still valid.
This parameter is optional, if not set a maxdelay of 0 minutes is set as default.

$waitany *

Syntax : $waitany=<path> /both

This command will set a wait for any train which is on the path section as defined.
If the qualifier /both is set, the wait will be applied for any train regardless of its direction, otherwise the wait is set only for trains heading in the same direction as the definition of the path.
The path defined in the waitany command must have a common section with the path of the train itself, otherwise no waiting position can be found.
This command can be set to control trains to wait beyond the normal signal or deadlock rules.
For instance, it can be used to perform a check for a train which is to leave a siding or yard, checking the line the train is to join for any trains approaching on that line, for a distance further back than signalling would normally clear, so as to ensure it does not get into the path of any train approaching on that line.
With the /both qualifier set, it can be used at the terminating end of single track lines to ensure a train does not enter that section beyond the last passing loop if there is another train already in that section as this could lead to irrecoverable deadlocks.

### 4.7.6 Create commands.

*To be specified.*

### 4.7.7 Dispose commands.

Dispose commands can be set in the #dispose row to define what is to be done with the train after is has terminated.
See special notes below on the behaviour of the player train when it is formed out of another train by a dispose command, or when the player train itself has a dispose command.

$forms

Syntax : $forms=<train> /runround=<path> /rrime=time /setstop

$forms defines which new train is to be formed out of this train when the train terminates. The consist of the new train is formed out of the consist of the terminating train and any consist definition for the new train is ignored.
The new train will be 'static' until the time as defined in #start row for that train. This means that the new train will not try to clear its path, signals etc., and will not move even if it is not in a station.
If the incoming train is running late, and its arrival time is later as the start time of the new train, the start of the new train is also delayed but the new train will immediately become active as soon as it is formed.
For loco hauled trains, it can be defined that the engine(s) must run round the train in order for the train to move in the opposite direction. The runround qualifier needs a path which defines the path the engine(s) is to take when performing the runround. If the train has more than one leading engine, all engines will be run round. Any other power units within the train will not be moved.
For specific rules and conditions for runround to work, see paragraph on relation between signalling and timetable concept.
If runround is defined, the time at which the runround is to take place can be defined. If this time is not set, the runround will take place immediately on termination of the incoming train.

**Command value** : referenced train, this is compulsory.

**Command qualifiers** :

| | |
|---|---|
| /runround=<path> | : <path> is the path to be used by the engine to perform the runround.<br>This qualifier is optional; if set, the value is compulsory. |
| /rrtime=time | : time is the definition of the time at which the runround is to take place. The time must be defined in HH:mm and must use the 24 hour clock.<br>This qualifier is only valid in combination with the /runround qualifier, is optional but if set, the value is compulsory. |
| /setstop | : if this train itself has no station stops defined but the train it is to form starts at a station, this command will copy the details of the first station stop of the formed train, to ensure this train will stop at the correct location.<br>For this qualifier to work correctly, the path of the incoming train must terminate in the platform area of the departing train.<br>This qualifier is optional and takes no values. |

$triggers

Syntax : $triggers=<train>

$triggers also defines which new train is to be formed out of this train when the train terminates.
However, when this command is used, the new train will be formed using the consist definition of the new train and the existing consist is removed.

**Command value** : referenced train, this is compulsory.

$static

Syntax : $ static

The train will become a 'static' train after it has terminated.

**Command value** : none.

$stable

Syntax : $stable /out_path=<path> /out_time=time /in_path=<path> /in_time=time /static /runround=<path> /rrtime= time /rrpos=<runround position> /forms=<train> /triggers=<train>

$stable is an extended form of either $forms, $triggers or $static, where the train is moved to another location before the related command is performed. In case of /forms or /triggers, the train can move back to the same or to another location where the new train actually starts. Note that in these cases, the train has to make two moves, outward and inward.
A runround can be performed in case /forms  is defined.
If /triggers is defined, the change of consist will take place at the 'stable' position. Any reversal(s) in the inward path, or at the final inward position, are taken into account when the new train is build, such that the consist is facing the correct direction when the new train is formed at the final inward position.
The $stable can be used where a train forms another train but when the train must clear the platform before the new train can be formed to allow other trains to use that platform.
It can also be used to move a train to a siding after completing its last duty, and be 'stabled' there as static train.
Separate timings can be defined for each move; if such a time is not defined, the move will take place immediately when the previous move is completed.
If timings are defined, the train will be 'static' after completion of the previous move until that required time.

If the formed train has a valid station stop and the return path of the stable command (in_path) terminates in the area of the platform of the first station stop of the formed train, the 'setup' check (see setup qualifier in $forms command) will automatically be added.

**Command value** : none.

**Command qualifiers** :

/out_path=<path> : <path> is the path to be used by the train to move out to the 'stable' position. The start of the path must match the end of the path of the incoming train.

/out_time = time : time definition when the outward run must be started. Time is defined as HH:mm and must use the 24 hour clock.

/in_path=<path> : <path> is the path to be used by the train for the inward run from the 'stable' position to the start of the new train.
The start of the path must match the end of the out_path, the end of the path must match the start of the path for the new train.

/in_time = time : time definition when the inward run must be started. Time is defined as HH:mm and must use the 24 hour clock.

/runround=<path> : <path> is the path to be used by the engine to perform the runround. For details, see the $forms command.

/rrtime=time : time is the definition of the time at which the runround is to take place. The time must be defined in HH:mm and must use the 24 hour clock.

/rrpos = <runround position> : the position within the 'stable' move at which the runround is to take place.
Possible values :

| | |
|---|---|
| out | : the runround will take place before the outward move is started. |
| stable | : the runround will take place at the 'stable' position. |
| in | : the runround will take place after completion of the inward move. |

/static : train will become a 'static' train after completing the outward move.

/forms=<train> : train will form the new train after completion of the inward move. See the $forms command for details.

/triggers=<train>: train will trigger the new train after completion of the inward move. The train will change to the consist of the new train at the 'stable' position. See the $triggers command for details.

**Use of command qualifiers :**

In combination with /static :

| | |
|---|---|
| /out_path | : compulsory |
| /out_time | : optional |

In combination with /forms :

| | |
|---|---|
| /out_path | : compulsory |
| /out_time | : optional |
| /in_path | : compulsory |
| /in_time | : optional |
| /runround | : optional |
| /rrtime | : optional, only valid if /runround is set |
| /rrpos | : compulsory if /runround is set, otherwise not valid |

In combination with /triggers :

| | |
|---|---|
| /out_path | : compulsory |
| /out_time | : optional |
| /in_path | : compulsory |
| /in_time | : optional |

# 5. Additional notes.

## 5.1 Timetable commands and player train.

### 5.1.1 Termination of a timetable run.

On reaching the end of a timetabled run, the program will not be terminated automatically but has to be terminated by the player.
*If the player train is to be formed into another service (using dispose command), the player can continue the run playing the next train by issuing the 'form next service' command (used key allocation to be defined).*

## 5.2 Signalling requirements and timetable concept.

### 5.2.1 General.

The timetable concept is more demanding of the performance of the signalling system than 'normal' activities. The main reason for this is that the timetable will often have AI trains running in both directions, including trains running ahead of the player train in the same direction as the player train. There are very few activities with such situations as no effort would ofcourse be made to define trains in an activity which would never be seen, but also because MSTS could not always properly handle such a situation.
Any flaws in signalling, e.g. signals clearing the path of a train too far ahead, will immediately have an effect on the running of a timetable.
If signals clear too far ahead on a single track line, for instance, it means trains will clear through passing loops too early, which leads to very long waits for trains in the opposite direction. This, in turn, can lead to lock-ups as multiple trains start to converge on a single set of passing loops.
Similar situations can occur at large, busy stations - if trains clear their path through such a station too early, it will lead to other trains being kept waiting to enter or exit the station.
If 'forms' or 'triggers' is used to link reversing trains, the problem is exacerbated as any delays for the incoming train will work through on the return working.

### 5.2.2 Call On signal aspect.

Signalling systems may allow a train to 'call on', i.e. allow a train onto a section of train already occupied by another train (also knows as permissive working).
The difference between 'call on' and 'permissive signals (STOP and PROCEED aspects) is that the latter is also allowed if the train in the section is moving (in the same direction), but 'call on' generally is only allowed if the train in the section is at a standstill.
When a signal allows 'call on', AI trains will always pass this signal and run up to a pre-defined distance behind the train in the section.
In station areas, this can lead to real chaos as trains may run into platforms occupied by other trains such that the total length of both trains far exceeds the platform length, so the second train will block the 'station throat' stopping all other trains. This can easily lead to a complete

lock-up of all traffic in and around the station.
To prevent this, calling on is blocked in station areas even if the signalling would allow it. To allow train to call on when this is required in the timetable, the $callon command must be set which overrules the overall block. This applies to both AI and player train
*In case the train is to attach to another train in the platform, different rules apply.*
Because of the inability of AI trains in MSTS to stop properly behind another train if 'called on' onto an occupied track, most signalling systems do not support 'call on' aspects but instead rely on the use of 'permission requests'. AI trains cannot issue such a request, therefor in such systems $callon will not work.
*In this situation, attach commands can also not work in station areas.*
Note that the 'runround' command also requires 'call on' ability for the final move of the engine back to the train to attach to it. Therefor, when performed in station areas, also the runround can only work if the signalling supports 'call on'.

Note that the restriction on 'call on' only applies to station areas, that is to track sections where a platform is defined. Permissive working or 'call on' is always allowed on plain track, yards, sidings etc., if supported by the signalling system.

### 5.2.3 Wait commands and Passing paths.

From the location where the 'wait' or 'follow' is defined, a search is made for the first common section for both trains, following on from a section where the paths are not common.
However, on single track routes with passing loops where 'passing paths' are defined for both trains, the main path of the trains will run over the same tracks in the passing loops and therefor no not-common sections are found. As a result, the waiting point cannot find a location for the train to wait and therefor will not work.
If waiting points are used on single track lines, the trains must have their paths running over different tracks through the passing loop in order for the waiting points to work properly.
It is a matter of choice by the timetable creator to either preset passing locations using the wait commands, or let the system work out the passing locations using the passing paths.

### 5.2.4 Wait commands and Permissive signals.

The 'wait' and 'follow' commands are processed through the 'blockstate' of the signal control.
If at the location where the train is to wait permissive signals are used, and these signals allow a 'proceed' aspect on blockstate JN_OBSTRUCTED, the 'wait' or 'follow' command will not work as the train will not be stopped.

### 5.1.2 Running trains around midnight.

A timetable can be defined for a full 24 hour day, and so would include trains running around midnight.

The following rules apply for the player train :

- Train booked to start before midnight will be started at the end of the day, but will continue to run if terminating after midnight.
- Trains formed out of other trains starting before midnight will NOT be started if the incoming train is delayed and as a result the start time is moved after midnight. In this situation, the activity is aborted.
- Trains booked to start after midnight will be started at the beginning of the day.

The following rules apply for AI trains :

- Trains booked to start before midnight will be started at the end of the day, but will continue to run if terminating after midnight.
- Trains formed out of other trains starting before midnight will still be started if the incoming train is delayed and as a result the start time is moved after midnight.
- Trains booked to start after midnight will be started at the beginning of the day.

As a result of these rules, it is not really possible to run an activity around or through midnight with all required AI trains included.
*It may be possible to set definitions such that in an activity starting just before midnight, trains starting after midnight will also be included or that activities starting just after midnight will include trains which started just before midnight.*
*The main problem in such a situation however is how to handle trains which terminated and formed 'static' trains, in combination with trains which are created after midnight (through the #create statement or due to the start time) where the 'static' train should actually form the 'created' train.*