EEP specific Lua variables and functions

Up to date until EEP 18.1 incl. plug-in 1 (ed 18/10/2025)

1 System variables

EEPVer		EEPVer

Requires	EEP 10.2 Plug-in 2	`if EEPVer >= 11 then`
		`  print("Train control via Lua possible!")`
		`end`

Purpose	Provides the version number of the installed EEP version.	

EEPTime		EEPTime

Requires	EEP 10.2 Plug-in 2	`if EEPTime == previoustime + 50 then`
		`  print("50 Seconds elapsed")`
		`  previoustime = EEPTime`
		`elseif EEPTime > previoustime + 50 then`
		`  print("More than 50 seconds elapsed")`
		`  previoustime = EEPTime`
		`else`
		`  print("50 seconds have not elapsed yet")`
		`end`

Purpose	Provides the current time within the EEP layout. The value corresponds to the number of seconds passed since midnight (EEP time). **Note:** Compared to the real time, the EEP time lags by -100 seconds every hour.	

EEPTimeH		EEPTimeH
Requires	EEP 10.2 Plug-in 2	`print("It's now: "..EEPTimeH..":"..EEPTimeM)`
Purpose	Provides the hour component of EEPTime, expressed as a value between 0 and 23.	

EEPTimeM		EEPTimeM
Requires	EEP 10.2 Plug-in 2	`print("It's now: "..EEPTimeH..":"..EEPTimeM)`
Purpose	Provides the minute component of EEPTime, expressed as a value between 0 and 59.	

EEPTimeS		EEPTimeS

Requires	EEP 10.2 Plug-in 2	`if EEPTimeS == 15 then`
		`  -- turn traffic light 1 to green`
		`  EEPSetSignal(1, 1)`
		`elseif EEPTimeS == 45 then`
		`  -- turn traffic light 1 to red`
		`  EEPSetSignal(1, 2)`
		`end`

Purpose	Provides the second component of EEPTime, expressed as a value between 0 and 59.	

EEPLng		EEPLng

Requires	EEP 17	`if EEPLng == "GER" then`
		`  print("Dies ist eine deutsche EEP-Version")`
		`elseif EEPLng == "ENG" then`
		`  print("This is an English version of EEP")`
		`elseif EEPLng == "FRA" then`
		`  print("Il s'agit d'une version française de l'EEP")`
		`end`

Purpose	Provides the abbreviation for the language of the installed EEP version: GER = German version, ENG = English version, FRA = French version.	

2 System functions

clearlog()	clearlog()
Parameters	none
Returns	none
Requires	EEP 10.2 Plug-in 2
Purpose	Clears the Lua event window
Notes	• This function is called by EEP without any parameters.

```
clearlog()
```

print()	print("Text1", "Text2", ... , TextN)
Parameters	various
Returns	one
Requires	EEP 10.2 Plug-in 2
Purpose	Displays in the event window the text in brackets
Notes	• All variables are converted to strings. • Accepts multiple parameters as long as they are separated with a comma « , ». New line with « \n ». • This function returns the entire print as one string.

```
print("It's now: ", EEPTimeH, ":", EEPTimeM)
```

EEPMain()	EEPMain()
Parameters	none
Returns	none
Requires	EEP 10.2 Plug-in 2
Purpose	This function is called by EEP every 200 milliseconds (i.e. 5 times per Second). Suits repetitive actions.
Notes	• Declaration of this function is mandatory in every script. Otherwise EEP won't connect to Lua. • This function is called by EEP without any parameters. • The function must return any number other than 0. • Returning a 0 interrupts the repeated calling of the concerned function. The other functions in your Lua script remain active. • If this function returns something other than a number, or doesn't return anything, an error message will appear and the connection to Lua will be interrupted.

```
function EEPMain()
  return 1
end
```

EEPPause()		EEPPause(Status)
Parameters	one	Status = 1 Pause = EEPPause(Status) Pause = EEPPause(0)
Returns	one	
Requires	EEP 14.1 Plug-in 1	
Purpose	Activates and deactivates the break-status of EEP.	
Notes	• This parameter sets the break-status: 0 resumes EEP 1 stops EEP **except** the Lua scripts. 2 stops EEP **as well as** the Lua scripts. To resume there is no other possibility left than pressing the P key. • Return value is 0 if the simulation is running, or 1 if it has been paused.	

EEPSetTime()		EEPSetTime(Hour, Minute, Second)
Parameters	three	Hour = 9 Minute = 24 Second = 45 ok = EEPSetTime(Hour, Minute, Second) ok = EEPSetTime(14, 35, 2)
Returns	one	
Requires	EEP 15	
Purpose	Changes the EEP time to the desired time.	
Notes	• First parameter sets the hours. • Second parameter sets the minutes. • Third parameter sets the seconds. • **All three parameters are mandatory** • The return value is true when the execution was successful, false if not.	

EEPGetFramesPerSecond()		EEPGetFramesPerSecond()
Parameters	none	fps = EEPGetFramesPerSecond()
Returns	one	
Requires	EEP 17.2 Plug-in 2	
Purpose	Returns the current frame rate (fps).	
Notes	• This function is called by EEP without any parameters. • The return value is the current frame rate (number of frames per second) as a number.	

EEPGetCurrentFrame()		EEPGetCurrentFrame()
Parameters	none	`FrameNumber = EEPGetCurrentFrame()`
Returns	one	
Requires	EEP 17.2 Plug-in 2	
Purpose	Returns the current value of the frames counter since system start without the images during editing and pause mode.	
Notes	• This function is called by EEP without any parameters. • The return value is the current value of the frames counter starting with the loading of the layout file (.anl3) as a number, whereby the images in edit and pause mode were **not** counted.	

EEPGetCurrentRenderFrame()		EEPGetCurrentRenderFrame()
Parameters	none	`FrameNumber = EEPGetCurrentRenderFrame()`
Returns	one	
Requires	EEP 17.2 Plug-in 2	
Purpose	Returns the total number of images rendered since the start of the layout, including images rendered during edit and pause modes.	
Notes	• This function is called by EEP without any parameters. • The return value is the current value of the frames counter starting with the loading of the layout file (.anl3) as a number, whereby the images in edit and pause mode were also **counted**.	

EEPGetTimeLapse()		EEPGetTimeLapse()
Parameters	none	`Time-lapse_factor = EEPGetTimeLapse()`
Returns	one	
Requires	EEP 17.2 Plug-in 2	
Purpose	Returns the time-lapse factor currently set in EEP.	
Notes	• This function is called by EEP without any parameters. • The return value is the time-lapse factor currently set in EEP, 1, 5 or 10 as a number.	

EEPSetColourFilter()		`EEPSetColourFilter()`
Parameters	four	`EEPSetColourFilter(hue, saturation, brightness, contrast)`
Returns	none	
Requires	EEP 17.3 Plug-in 3	
Purpose	This function is only used to temporarily set the colours e.g. after a change of the camera shot, and not to permanently change the programme settings	
Notes	• First parameter corresponds to the hue setting in the programme settings. • Second parameter corresponds to the saturation setting in the programme settings. • Third parameter corresponds to the brightness setting in the programme settings. • Fourth parameter corresponds to the contrast setting in the programme settings. • **Attention: The parameters are not transferred to the programme settings** • This function has no return value.	

3 Signal functions

EEPSetSignal()		**EEPSetSignal(ID, Aspect [, Callback])**

Parameters	two or three	`-- switches signal 0023 to 1`
		`-- (aspect depends on signal)`
		`Result = EEPSetSignal(23, 1)`
		
		`-- switches signal 0045 to 1 and`
		`-- call EEPOnSignal_45()`
		`Result = EEPSetSignal(45, 1, 1)`

Returns	one	
Requires	EEP 10.2 Plug-in 2 EEP 14.1 Plug-in 1	
Purpose	Switches a signal	
Notes	• First parameter is the signal's ID • Second parameter is the signals aspect. Values from 1 onwards immediately activate the corresponding signal position in the selection list under the signal's "properties" section. The value 0 causes the signal to adopt the next aspect. **Since EEP 14.1 Plug-in 1**, you can switch back to the previous position with a negative value. For pre-set route signals position 1 corresponds to "Release", position 2 corresponds to "pre-set route #1" as position n corresponds to "pre-set route #n-1" • **Since EEP 14.1 :** A third optional parameter with the value 1 allows to start the execution of the function *EEPOnSignal_x()*. This third parameter only has an effect on the signal if it has been registered and an *EEPOnSignal_x()* action has been defined. **Please use this function with caution!** There is a risk of setting up an incorrect chain of command running in a loop which can lead to freezing both EEP and Lua. • Return value is 1 if the signal and the demanded aspect exist, 0 if either one of the two couldn't be found.	

EEPGetSignal()		**EEPGetSignal(ID)**

Parameters	one	`Aspect = EEPGetSignal(1)`
		`if Aspect == 0 then`
		`  print("Signal 1 doesn't exist")`
		`elseif Aspect == 1 then`
		`  print("Signal 1 is set to Danger/Stop")`
		`elseif Aspect == 2 then`
		`  print("Signal 1 is set to Clear/Main")`
		`end`
		`-- ATTENTION: With older signals position 1`
		`-- can be Clear/Main and position 2 Danger/Stop.`

Returns	one	
Requires	EEP 10.2 Plug-in 2	
Purpose	Returns the position of a signal in relation to the prevailing train control (see last comment).	
Notes	• The parameter is the ID of the signal. • Return value relates to the signal's current aspect. It refers to the corresponding line in the list of the different possible signal aspects. This list is available in the signal's "properties" section. For pre-set route signals values greater than 1 indicate which (value – 1) pre-set route originating from the signal is on "Clear/Main" (e.g. value = 4: the 3rd route). When the signal queried does not exist, then 0 will be the returned value. • **Be aware that:** The EEPGetSignal() function will only return the new state of a signal set by using *EEPSetSignal()* once a new *EEPMain()* cycle has begun. • **Attention:** If an activation delay is set in the signal, the changed 'signal position' is only displayed after it has expired. This means that even if, for example, the signal is already visibly indicating movement, but the train is still 'held' at the signal by the activation delay, the function will continue to return stop.	

EEPRegisterSignal()		EEPRegisterSignal(ID)
Parameters	one	`Result = EEPRegisterSignal(1)`

Returns	one	`function EEPOnSignal_1(signal)`
		`  print("Signal 1 is set to "..Aspect)`
		`end`

Requires	EEP 10.2 Plug-in 2	
Purpose	Registers a signal for the callback function *EEPOnSignal_x()* This mandatory registration prevents signals from triggering the *EEPOnSignal_x()* callback function while no order has been defined for them in the script.	
Notes	• The registration is mandatory for those signals which you want to trigger the *EEPOnSignal_x()* function whenever their aspect changes. • The parameter is the ID of the signal. • Return value is 1 if the signal exists, 0 if it doesn't.	

EEPOnSignal_x()		EEPOnSignal_x(Aspect)

Parameters	one	`function EEPOnSignal_1(Aspect)`
		`  print("Signal 1 is set to  "..Aspect)`
		`end`
		`EEPRegisterSignal(1)  -- is mandatory!`

Returns	none	
Requires	EEP 10.2 Plug-in 2	
Purpose	Every aspect change induced by a contact or by manual operation (directly or in a link chain) triggers this callback function if the signal has been registered for callbacks. In the script, one defines the corresponding function and thus determines what is to be done when the signal position is changed. **Important:** Changing this or a linked signal's state by *EEPSetSignal()* will not trigger the callback, unless the third parameter in that function was a 1.	
Notes	• The name of the function must not end in _x but with the signal's ID. For signal 0012 the correct name of the function would be EEPOnSignal_12() **Please note:** The leading 00 must be omitted in the function's name. • The parameter is the new signal aspect. The number matches the position in the signal's aspect list as found in the signal's properties. You can use a variable of your choice between the brackets to store this value for later use. • EEP requires no return value when calling this function.	

EEPGetSignalTrainsCount()		EEPGetSignalTrainsCount(ID)
Parameters	one	`Number = EEPGetSignalTrainsCount(3)`
Returns	one	
Requires	EEP 13.2 Plug-in2	
Purpose	Return the number of "vehicles composite" held by the specified signal.	
Notes	• The parameter is the ID of the signal. • Return value is the number of "vehicles composite" held by the signal. In EEP, 1 "vehicles composite" is defined as several vehicles coupled together to form a unit (e.g. 1 train or 1 truck with trailer) but also as 1 single vehicle.	

EEPGetSignalTrainName()	EEPGetSignalTrainName(ID, Position)
Parameters	two
Returns	one
Requires	EEP 13.2 Plug-in2
Purpose	Returns the name of the specified "vehicles composite" held by the specified signal.
Notes	• First parameter is the ID of the signal. • Second parameter is the position number of the "vehicles composite" held by the signal. In EEP, 1 "vehicles composite" is defined as several vehicles coupled together to form a unit (e.g. 1 train or 1 truck with trailer) but also as 1 single vehicle. • Return value is the name of the specified "vehicles composite".

```
Name = EEPGetSignalTrainName(3, 1)
```

EEPSetSignalStopDistance()	EEPSetSignalStopDistance(ID, Stop_Distance)
Parameters	two
Returns	one
Requires	EEP 17.3 Plug-in3
Purpose	Assigns a stop distance to the specified signal.
Notes	• First parameter is the ID of the signal. • Second parameter is the new stop distance in meter. If a negative value is entered, the stop distance is set to zero. • Return value is true if the signal exists, false if it doesn't.

```
ok = EEPSetSignalStopDistance(13, 45)
```

EEPGetSignalStopDistance()	EEPGetSignalStopDistance(ID)
Parameters	one
Returns	two
Requires	EEP 17.3 Plug-in3
Purpose	Returns the stop distance to the specified signal.
Notes	• Parameter is the ID of the signal. • First return value is true if the signal exists, false if it doesn't. • Second return value is the stop distance in meter. • After *EEPSetSignalStopDistance()*, EEPGetSignalStopDistance() immediately returns the new stop distance.

```
ok, Stop_Distance = EEPGetSignalStopDistance(9)
print("Signal 9 has a stop distance of ",
Stop_Distance, " meter.")
```

EEPOnTrainStoppedOnSignal()		EEPOnTrainStoppedOnSignal(sigID, Train)

Parameters	two	function EEPOnTrainStoppedOnSignal(sigID, train)
		  print("Train "..Train.." stopped at signal "..
		  sigID)
		end

Returns	none	
Requires	EEP 17.3 Plug-in3	
Purpose	Is called whenever a "vehicles composite" stops at a signal.	
Notes	• First parameter is the ID of the signal. • Second parameter is the name of "vehicles composite" That has stopped at the signal. • The parameter names can be freely selected. • EEP does not expect a return value when this function is called.	

EEPGetSignalFunctions()		EEPGetSignalFunctions(ID)
Parameters	one	
Returns	two	ok, Number = EEPGetSignalFunctions(14)
Requires	EEP 17.3 Plug-in3	
Purpose	Returns the number of signal functions defined in the NOS.	
Notes	• Parameter is the ID of the signal. • First return value is true if the signal exists, false if it doesn't. • Second return value is the number of signal functions defined in the NOS. This is identical to the number of signal positions in the selection box in the object properties of the signal.	

EEPGetSignalFunction()		EEPGetSignalFunction(ID, Position)
Parameters	two	
Returns	two	ok, Function = EEPGetSignalFunction(14, 3)
Requires	EEP 17.3 Plug-in3	
Purpose	Returns the function entered by the designer in the NOS for the specified position in the selection box in the object properties of the defined signal.	
Notes	• Parameter is the ID of the signal. • First return value is true if the signal and the demanded position exist and false if either one of the two couldn't be found. • Second return value return value is the function defined in the NOS. This means: 1 -> Danger/Stop 2 -> Clear/Main/Run and, if the signal has a speed control, e.g. 1025 -> Max. speed 25 km/h 1040 -> Max. speed 40 km/h 1100 -> Max. speed 100 km/h depending on the design of the signal. • The return values correspond to the values entered in the NOS by the designer.	

EEPGetSignalItemName()	EEPGetSignalItemName(ID, [false | true])
Parameters	one or two
Returns	two
Requires	EEP 17.3 Plug-in3

```
ok, Name = EEPGetSignalItemName(3)
ok, Name = EEPGetSignalItemName(3, false)
-- Name => "Ks_Sig_A_screen (GK3)"

ok, Name = EEPGetSignalItemName(3, true)
-- Name => "Signale\Signale\KsSigASch_GK3.3dm"
```

Purpose	Returns the model name of the signal from the ini file or the name of the 3dm file including its path.

Notes	• First parameter is the ID of the signal.
	• Second parameter is optional and can be entered as true or false.
	  ○ If false or **non-existent**, the model name from the ini file is returned in the second return value.
	  ○ If true the path with the name of the 3dm file is returned in the second return value.
	• First return value is true if the signal exists, false if it doesn't.
	• Second return value is either the model name or the path with the name of the 3dm file of the specified signal depending on the optional second parameter.

EEPCheckSetRoute()	EEPCheckSetRoute(RouteStartSignalID, RouteDestinationNumber)
Parameters	two
Returns	one
Requires	EEP 18.1 - Plug-in 1

```
ok = EEPCheckSetRoute(33, 2)

if ok then
  EEPSetSignal(33, 3, 1)
end
```

Purpose	Checks whether the route from the start signal to the finish line is clear or occupied.
Notes	• First parameter is the signal ID of the route start signal • Second parameter is the destination number of the route, as specified in the object properties of the start signal. • The return value is true if the track is free and the route could therefore be switched, or false if the track is occupied. • **ATTENTION**: If you subsequently switch the route, remember that the signal position in *EEPSetSignal*() is "destination number + 1", as position 1 means "cancel route", and if the callback function *EEPOnSignal_x()* is to be called subsequently, a 1 must be set as the third parameter..

4 Switch point functions

EEPSetSwitch()		EEPSetSwitch(ID, Direction [, Callback])

Parameters	two or three	`-- set switch point 0067 to 1 (Main line)`
		`Result = EEPSetSwitch(67, 1)`
		
		`-- set switch point 0089 to 1 and`
		`-- call EEPOnSwitch_89()`
		`Result = EEPSetSwitch(89, 1, 1)`

Returns	one	
Requires	EEP 10.2 Plug-in 2 EEP 14.1 Plug-in 1	
Purpose	Switches a switch point	
Notes	• First parameter is the ID of the switch point. • Second parameter is the switch direction to be adopted. Values from 1 onwards immediately activate the corresponding switch direction position in the selection list under the switch point's “properties” section. The value 0 causes the switch to adopt the next switch direction. **Since EEP 14.1 Plug-in 1**, you can switch back to the previous position with the value -1. • **Since EEP 14.1 :** A third optional parameter with the value 1 allows to start the execution of the function EEPOnSwitch_x(). This third parameter only has an effect on the switch if it has been registered and an EEPOnSwitch_x() action has been defined. **Please use this function with caution!** There is a risk of setting up an incorrect chain of command running in a loop which can lead to freezing both EEP and Lua. • Return value is 1 if the switch and the demanded switch direction exist, 0 if either one of the two couldn’t be found.	

EEPGetSwitch()		EEPGetSwitch(ID)

Parameters	one	`Direction = EEPGetSwitch(1)`
		`if Direction == 0 then`
		`  print("Switch point 1 doesn’t exist")`
		`elseif Direction == 1 then`
		`  print("Switch point 1 is set to main line (straight)")`
		`elseif Direction == 2 then`
		`  print("Switch point 1 is set to branch line (turnout)")`
		`end`

Returns	one	
Requires	EEP 10.2 Plug-in 2	
Purpose	Provides the current state of a switch point	
Notes	• Parameter is the ID of the switch point. • Return value is a numeric representation of the switch’s current direction. The value reflects the direction’s position in the effects list of the switch point’s properties. When the switch queried does not exist, then 0 will be the returned value. • **Be aware that:** The EEPGetSwitch() function will only provide the new state of a switch set by using *EEPSetSwitch()* once a new *EEPMain()* cycle has begun.	

EEPRegisterSwitch()		EEPRegisterSwitch(ID)
Parameters	one	`Result = EEPRegisterSwitch(3)`

Returns	one	`function EEPOnSwitch_3(Direction)`
		`  print("switch point 3 changed to "..Direction)`
		`end`

Requires	EEP 10.2 Plug-in 2	
Purpose	Registers a switch point for the callback function *EEPOnSwitch_x()* This mandatory registration prevents switches from triggering the *EEPOnSwitch_x()* callback function while no order has been defined for them in the script.	
Notes	• The registration is mandatory for those switches which you want to trigger the *EEPOnSwitch_x()* function whenever their direction changes. • The parameter is the ID of the switch. • Return value is 1 if the switch exists, 0 if it doesn't.	

EEPOnSwitch_x()		EEPOnSwitch_x(Direction)

Parameters	one	`function EEPOnSwitch_34(Direction)`
		`  print("Switch point 34 changed to "..Direction)`
		`end`
		`EEPRegisterSwitch(34)  -- is mandantory!`

Returns	none	
Requires	EEP 10.2 Plug-in 2	
Purpose	Every direction change induced by a contact or by manual operation (directly or in a link chain) triggers this callback function if the switch point has been registered for callbacks. In the script, one defines the corresponding function and thus determines what is to be done when the switch direction is changed. **Important:** Changing this or a linked switch's state by *EEPSetSwitch()* will not trigger the callback, unless the third parameter in that function was a 1	
Notes	• The name of the function must not end in _x but with the switch's ID. For switch 0034 the correct name of the function would be EEPOnSwitch_34() **Please note:** The leading 00 must be omitted in the function's name! • This function is called by the new switch direction. The number matches the position in the switch's direction list as found in the switch's properties. Use a variable of your choice between the brackets to store this value for later use. • EEP requires no return value when calling this function.	

5 Storage functions

EEPSaveData()		EEPSaveData(Slot, Boolean|Value|"String"|nil)

Parameters	two	`-- store "true" in slot 1`
		`ok = EEPSaveData(1, true)`
		
		`-- store value 42 in slot 2`
		`ok = EEPSaveData(2, 42)`
		
		`-- store string "I am slot 3" in slot 3`
		`ok = EEPSaveData(3, "I am slot 3")`
		
		`-- deletes slot 4`
		`ok = EEPSaveData(4, nil)`

Returns	one	
Requires	EEP 11.0	
Purpose	Saves to a specific storage location. Is automatically saved and loaded with the installation	
Notes	• There are 1000 storage locations ("slots") numbered from 1 to 1000. • It is possible to store booleans, numbers, or strings. A maximum of 999 characters are available for strings, which may not contain any formatting characters. • First parameter is the storage slot location. • Second parameter is the content to be stored. It is possible to delete the content of the storage slot with nil. • The returned values are true when the storage was successful or false if not. • When the installation is saved, EEP of course links the contents of the storage slots to the corresponding script. This part of the script is not visible in any editor when the EEP layout is open, but it is visible in an external editor **when the EEP layout is closed** and can then be edited with it.	

EEPLoadData()		EEPLoadData(Slot)

Parameters	one	`ok, data = EEPLoadData(1)`
		
		`if ok then`
		`  print("Slot 1 contains: "..data)`
		`else`
		`  print("Slot 1 is empty")`
		`end`

Returns	two	
Requires	EEP 11.0	
Purpose	Loads content from data slot. This function is used to restore data when you reload your script.	
Notes	• There are 1000 storage locations ("slots") numbered from 1 to 1000. • It is possible to store booleans, numbers, or strings. A maximum of 999 characters are available for character strings, which may not contain any formatting characters. • The parameter is the storage slot location. • First returned value is true, if the slot contains data, or false if not. • Second returned value is the data contained in the slot • When the installation loads, EEP reads the contents of the storage slots mentioned in the script. By calling EEPLoadData() it is possible to query and assign these values to variables.	

6 Train functions

EEPSetTrainSpeed()	EEPSetTrainSpeed("#Name", Speed [, false|true])
Parameters	two or three
Returns	one
Requires	EEP 11.0 EEP 17.2 - Plug-in 2

```
ok = EEPSetTrainSpeed("#Passenger train", 80)

ok = EEPSetTrainSpeed("#Rheingold", 80, false)
ok = EEPSetTrainSpeed("#Rheingold", 80, true)
```

Purpose

Assigns a new speed to a specified "vehicles composite" (e.g. a train).

Notes

- First parameter is the name of the "vehicles composite" as a string preceded by #.
- Second parameter is the target speed. A negative value will result in a reverse gear.
- **Since EEP 17.2 Plug-in 2**, an optional third parameter can be entered with true or false respectively **1** or **0**.
  - If false or **0** or **non-existence**, a new **target speed** is assigned - as before - whereby any signal influence is cancelled (i.e. a "vehicles composite" that is currently being held up by a signal **sets off**).
  - If true or **1**, a **"cruising" speed** is assigned to the "vehicles composite" while maintaining a signal influence (i.e. a "vehicles composite" currently stopped by a signal will **continue to stop**)
- Return value is true if the targeted "vehicles composite" exists, or false if it doesn't.

EEPGetTrainSpeed()	EEPGetTrainSpeed("#Name" [, false|true])
Parameters	one or two
Returns	two
Requires	EEP 11.0 EEP 17.2 - Plug-in 2

```
ok, TargetSpeed = EEPGetTrainSpeed("#VT98;001")
ok, TargetSpeed = EEPGetTrainSpeed("#VT98;001", false)

ok, CruisingSpeed = EEPGetTrainSpeed("#VT98;001", true)
```

Purpose

Returns the target or cruising speed of a specified "vehicles composite" (e.g. a train)

Notes

- (First) parameter is the name of the "vehicles composite" as a string preceded by #.
- **Since EEP 17.2 Plug-in 2**, an optional second parameter can be entered with true or false respectively **1** or **0**.
  - If false or **0** or **non-existence**, the **current actual speed** is returned - as before.
  - If true or **1**, the **"cruising" speed** is returned (even if it is waiting in front of a signal!).
- First return value is true if the targeted "vehicles composite" exists, or false if not.
- Second return value is the determined speed of the "vehicles composite".
- **Attention:** After *EEPSetTrainSpeed()* EEPGetTrainSpeed() returns the new value in the next cycle of *EEPMain()* at the earliest

EEPSetTrainRoute()		EEPSetTrainRoute("#Name", "Route")
Parameters	two	ok = EEPSetTrainRoute("#Passenger train", "Dortmund")
Returns	one	
Requires	EEP 11.2 Plug-in 2	
Purpose	Assigns a route to the specified "vehicles composite" (e.g. a train).	
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • Second parameter is the route as a string. **Please note:** The route must have been previously defined in EEP. • Return value is true, when the concerned "vehicles composite" and the desired route exist, or false if at least one of them does not exist.	

EEPGetTrainRoute()		EEPGetTrainRoute("#Name")
Parameters	one	ok, Route = EEPGetTrainRoute("#Passenger_train")
Returns	two	
Requires	EEP 11.2 Plug-in 2	
Purpose	Returns the current route of a specified "vehicles composite" (e.g. a train).	
Notes	• Parameter is the name of the "vehicles composite" as a string preceded by #. • First return value is true, if the targeted "vehicles composite" exists or false if not. • Second return value is the route.	

EEPSetTrainLight()		EEPSetTrainLight("#Name", true|false [, Source])
Parameters	two or three	- Switching the lights on ok = EEPSetTrainLight("#Rheingold", true) ok = EEPSetTrainLight("#Rheingold", true, o) - Right turn signal off ok = EEPSetTrainLight("#Ford Transit", false, 2) - Left turn signal off ok = EEPSetTrainLight("#Ford_Transit", true, 1) - Brake lights on ok = EEPSetTrainLight("#Ford_Transit", true, 3) - Brake lights off ok = EEPSetTrainLight("#Ford_Transit", false, 3)
Returns	one	
Requires	EEP 11.2 Plug-in 2 EEP 15 EEP 16.3 Plug-in 3	
Purpose	Turns on or off the lights and indicators on a "vehicles composite" (e.g. a train). **Please note:** The distinction between lights and indicators has been added since EEP 15. **Since EEP 16.3 Plug-in 3** the automatic switching on of the vehicle's brake lights can be switched on or off during braking. The old command with only two parameters remains valid.	
Notes	• First parameter is the name of the "vehicles composite" preceded by #. • Second parameter is either true (lights on) or false (lights off) • Third parameter is the source of the light 0 for the normal driving lights and the interior illumination. 1 for the left turn signal. **(since EEP 15)** 2 for the right turn signal. **(since EEP 15)** 3 for the brake lights **(since EEP 16.3 Plug-in 3)** • Return value is true when the execution was successful and false otherwise.	

EEPGetTrainLight()		EEPGetTrainLight("#Name" [, Source])
Parameters	one or two	ok, IsBurning = EEPGetTrainLight("#Rheingold") ok, IsBurning = EEPGetTrainLight("#Rheingold", o) ok, IsBurning = EEPGetTrainLight("#Ford Transit", 2)
Returns	two	
Requires	EEP 18.0	
Purpose	Determines whether the lights or indicators are on or off in a 'vehicle composite' (e.g. a train).	
Notes	• First parameter is the name of the "vehicles composite" preceded by #. • Second (optional) parameter defines the light source: 0 for the normal driving lights and the interior illumination, 1 for the left turn signal, 2 for the right turn signal, 3 for the brake lights. • First return value is true when the execution was successful and false otherwise • Second return value is true if the specified light source is on, otherwise false.	

EEPSetTrainSmoke()		EEPSetTrainSmoke("#Name", true|false)
Parameters	two	ok = EEPSetTrainSmoke("#Passenger train", true)
Returns	one	
Requires	EEP 11.2 Plug-in 2	
Purpose	Turns the smoke and steam of the specified "vehicles composite" (e.g. a train) on or off.	
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • Second parameter is either true (=smoke on) or false (=smoke off) • Return value is true if the "vehicles composite" exists, false if not.	

EEPSetTrainHorn()		EEPSetTrainHorn("#Name", true|false)
Parameters	one or two	ok = EEPSetTrainHorn("#Passenger_train") ok = EEPSetTrainHorn("#Passenger_train", true) ok = EEPSetTrainHorn("#Passenger_train", false)
Returns	one	
Requires	EEP 11.2 Plug-in 2	
Purpose	Turns the warning sound of the specified "vehicles composite" (e.g. a train) on or off.	
Notes	• First parameter is the name of the "vehicles composite" as a string • Second parameter is true to sound the horn of the "vehicles composite" and false to mute it before end. As true is set as default, the horn will sound even if second parameter is omitted. • Return value is true if the "vehicles composite" exists, false if not.	

EEPSetTrainCouplingFront()	EEPSetTrainCouplingFront("#Name", true|false)
Parameters	two
Returns	one
Requires	EEP 11.2 Plug-in 2
Purpose	Sets the front coupler of a "vehicles composite" (e.g. a train) to coupling or decoupling
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by # • Second parameter is true (= coupling) or false (= decoupling). • Return value is true if the "vehicles composite" exists, false if not.

```
ok = EEPSetTrainCouplingFront("#Freight_train", true)
```

EEPGetTrainCouplingFront()	EEPGetTrainCouplingFront("#Name")
Parameters	one
Returns	one
Requires	EEP 18.0
Purpose	Determines whether the front coupler of a "vehicles composite" (e.g. a train) is set to coupling or decoupling
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by # • First return value is true if the "vehicles composite" exists, false if not. • Second return value is the position of the front coupler: 1 = coupling 2 = decoupling

```
ok = EEPGetTrainCouplingFront("#Freight_train")
```

EEPSetTrainCouplingRear()	EEPSetTrainCouplingRear("#Name", true|false)
Parameters	two
Returns	one
Requires	EEP 11.2 Plug-in 2
Purpose	Sets the rear coupler of a "vehicles composite" (e.g. a train) to coupling or decoupling
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by # • Second parameter is true (= coupling) or false (= decoupling). • Return value is true if the "vehicles composite" exists, false if not.

```
ok = EEPSetTrainCouplingRear("#Freight_train", true)
```

EEPGetTrainCouplingRear()	EEPGetTrainCouplingRear("#Name")
Parameters	one
Returns	one
Requires	EEP 18.0
Purpose	Determines whether the rear coupler of a "vehicles composite" (e.g. a train) is set to coupling or decoupling
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by # • First return value is true if the "vehicles composite" exists, false if not. • Second return value is the position of the rear coupler: 1 = coupling 2 = decoupling

```
ok = EEPGetTrainCouplingRear("#Freight_train")
```

EEPTrainLooseCoupling()		EEPTrainLooseCoupling("#Name", true|false, Number, "#Name")

Parameters	three or four	`ok = EEPTrainLooseCoupling("#Freight_train", true, 3)`
		
		`ok = EEPTrainLooseCoupling("#Freight_train", true, 3, "#wagons")`

Returns	one	
Requires	EEP 11.2 Plug-in 2	
Purpose	Separates a "vehicles composite" (e.g. a train) at the specified point.	
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • Second parameter defines if you count vehicles from the front or the rear. true = front false = rear • Third parameter determines the number of rollingstock to be uncoupled from the front or rear. • An optional fourth parameter gives the detached section a new name (a string preceded by #). • Return value is true if the "vehicles composite" exists, false if not.	

EEPOnTrainCoupling()		EEPOnTrainCoupling("Train_A", "Train_B", "New_Train")

Parameters	three	`function EEPOnTrainCoupling(Train_A, Train_B, New_Train)`
		`  print(..Train_A.." and "..Train_B.." became "..New_Train)`
		`end`

Returns	none	
Requires	EEP 14.1 Plug-in 1	
Purpose	Is always executed when coupling two "vehicles composite" (e.g. two trains)	
Notes	• First parameter is the name of the moving "vehicles composite" which has transferred its ID to the new assembled "vehicles composite". If both "vehicles composite" were moving, then the fastest one will be taken into account. • Second parameter is the name of the "vehicles composite" that was at standstill (or has moved slower) and lost its ID • Third parameter is the name of the new "vehicles composite" being formed. • EEP requires no return value when calling this function.	

EEPOnTrainLooseCoupling()		EEPOnTrainLooseCoupling("Train_A", "Train_B", "Previous_train")

Parameters	three	`function EEPOnTrainLooseCoupling(Train_A, Train_B, Previous_train)`
		`  print(Previous_train.." divided in "..train_A.." and "..Train_B)`
		`end`

Returns	None	
Requires	EEP 14.1 Plug-in 1	
Purpose	Is always executed when decoupling a "vehicles composite" (e.g. a train).	
Notes	• First parameter is the name of the "vehicles composite" whose coupler has been configured to uncouple. This part of the train keeps its previous ID. • Second parameter is the name of the train that has been uncoupled. A new ID is assigned to this part. • Third parameter is the name of the original train. • EEP requires no return value when calling this function.	

EEPSetTrainHook()	EEPSetTrainHook("#Name", true|false)
Parameters	two
Returns	one
Requires	EEP 11.2 Plug-in 2
Purpose	Turns the cargo hook function on a "vehicles composite" (e.g. a train) on or off.
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • Second parameter is either true (= hook on) or false (= hook off). • Return value is true if the targeted "vehicles composite" exists, false if not.

```
ok = EEPSetTrainHook("#Rail_crane", true)
```

EEPSetTrainHookGlue()	EEPSetTrainHookGlue("#Name", true|false)
Parameters	two
Returns	one
Requires	EEP 11.2 Plug-in 2
Purpose	Influences the behavior of goods on a cargo hook of a "vehicles composite" (e.g. a train).
Notes	• First parameter is the name of the "vehicles composite" preceded by #. • Second parameter determines the behavior false = Goods are swinging (appropriate for hooks) true = Goods are fixed (appropriate for grabbers) • Return value is true, when the execution was successful, false if not.

```
ok = EEPSetTrainHookGlue("#Rail_crane", true)
```

EEPSetTrainAxis()	EEPSetTrainAxis("#Name", "Axis", Position" [, false|true])
Parameters	three or four
Returns	one
Requires	EEP 11.2 Plug-in 2 EEP 17.2 Plug-in 2
Purpose	Animates one or more selected axles of the specified "vehicles composite" (e.g. a train).
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • Second parameter is the name of the axis as a string. • Third parameter is the target position of the axis. • **Since EEP 17.2 Plug-in 2**, an optional fourth parameter can be entered with true or false respectively **1** or **0**. o If true or 1, the term entered as the 2nd parameter serves as a filter for all axis names that **begin** with the filter fragment. If, for example, "dome cover" is entered as the second parameter, the function sets the axes "dome cover 1", "dome cover 2", "dome cover 3", etc. (if they exist) to the position entered as the third parameter. o If false or **if the 4th parameter does not exist**, the complete axis name must always be entered as the second parameter. • Return value is true if the targeted "vehicles composite" and the axis exist, false if not.

```
ok = EEPSetTrainAxis("#Rail_crane", "raise/lower
the boom", 100)

ok = EEPSetTrainAxis("#DustyGoodsTrain",
"Dome cover", 100, true)
```

EEPSetTrainName()		EEPSetTrainName("#OldName", "#NewName")
Parameters	two	
Returns	one	ok = EEPSetTrainName("#RE 1", "#RE 5")
Requires	EEP 14.1 Plug-in 1	
Purpose	Assigns a name to a "vehicles composite" (e.g. a train).	
Notes	• First parameter is the actual name of the "vehicles composite" as a string preceded by #. • Second parameter is the new name of this "vehicles composite" as a string preceded by #. • Return value is true, when the execution was successful, false if not.	

EEPGetTrainLength()		EEPGetTrainLength("#Name")
Parameters	one	
Returns	two	ok, length = EEPGetTrainLength("#Freight_train")
Requires	EEP 15.1 Plug-in 1	
Purpose	Returns the entire length of a "vehicles composite" (e.g. a train).	
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • First return value is true if the targeted "vehicles composite" exists, false if not • Second return value is the length of the "vehicles composite" in meters.	

EEPTrainChangeOrientation()		EEPTrainChangeOrientation("#Name")
Parameters	one	
Returns	one	ok = EEPTrainChangeOrientation("#Freight_train")
Requires	EEP 18.0	
Purpose	Rotates the orientation and direction of travel of a "vehicles composite" (e.g. a train).	
Notes	• Parameter is the name of the "vehicles composite" as a string preceded by #. • Return value is true if the targeted "vehicles composite" exists, false if not • **Note:** The "vehicles composite" is also rotated when it is travelling, **but not in a virtual depot**.	

7 Rollingstock functions

EEPRollingstockSetCouplingFront()	EEPRollingstockSetCouplingFront("Name", Coupler)
Parameters	two
Returns	one
Requires	EEP 11.0

```
ok = EEPRollingstockSetCouplingFront("Castor 1;001", 1)
```

Purpose	Sets the front coupler of the specified rollingstock to coupling or decoupling
Notes	• First parameter is the entire name of the rollingstock as a string • Second parameter is the desired condition of the front coupler: 1 = activate coupler (couples on contact when opposing coupler is also active), 2 = deactivate coupler • Return value is true if the rollingstock exists, false if not.

EEPRollingstockGetCouplingFront()	EEPRollingstockGetCouplingFront("Name")
Parameters	one
Returns	two
Requires	EEP 11.0

```
ok, Coupler = EEPRollingstockGetCouplingFront("Castor 1;001")
```

Purpose	Determines the current condition of the front coupler of the specified rollingstock.
Notes	• Parameter is the entire name of the rollingstock as a string • First return value is true, if the rollingstock exists, false if not. • Second return value refers to the behavior of the coupler 1 = ready to couple 2 = in decoupling position 3 = coupled

EEPRollingstockSetCouplingRear()	EEPRollingstockSetCouplingRear("Name", Coupler)
Parameters	two
Returns	one
Requires	EEP 11.0

```
ok = EEPRollingstockSetCouplingRear("fals 175 Kalk", 1)
```

Purpose	Sets the rear coupler of the specified rollingstock to coupling or decoupling
Notes	• First parameter is the entire name of the rollingstock as a string • Second parameter is the desired condition of the front coupler: 1 = activate coupler (couples on contact when opposing coupler is also active), 2 = deactivate coupler. • Return value is true if the rollingstock exists, false if not.

EEPRollingstockGetCouplingRear()	EEPRollingstockGetCouplingRear("Name")
Parameters	one
Returns	two
Requires	EEP 11.0

```
ok, Coupler =
EEPRollingstockGetCouplingRear("fals 175 Kalk")
```

Purpose	Determines the current condition of the rear coupler of the specified rollingstock.
Notes	• Parameter is the entire name of the rollingstock as a string. • First return value is true, if the rollingstock exists, false if not. • Second return value refers to the behavior of the coupler 1 = ready to couple 2 = in decoupling position 3 = coupled

EEPRollingstockSetAxis()	EEPRollingstockSetAxis("Name", "Axis", Position [, true|false])
Parameters	three or four
Returns	one
Requires	EEP 11.0 EEP 17 2 - Plug-in 2

```
Name = "Dispolok 189-917 EpVI"
Axis = "Pantograph DE"
ok = EEPRollingstockSetAxis(Name, Axis, 100)

ok = EEPRollingstockSetAxis(
"Dispolok 189-917 EpVI", "Pantograph", 0, true)
```

Purpose	Moves the axle named by axle name of the specified rollingstock to the desired position.

Notes	• First parameter is the name of the rollingstock as a string.
	• Second parameter is the name of the axis as a string.
	• Third parameter is the target position of the axis.
	• **Since EEP 17.2 Plug-in 2**, an optional fourth parameter can be entered with true or false respectively **1** or **0**.
	  ○ If true or 1, the term entered as the second parameter serves as a filter for all axis names that **begin** with the filter fragment. If, for example, " Pantograph " is entered as the second parameter, the function sets the axes "Pantograph 1", "Pantograph 2", "Pantograph 3", etc. (if they exist) to the position entered as the third parameter.
	  ○ If false or **if the 4th parameter does not exist**, the complete axis name must always be entered as the second parameter.
	• Return value is either true, if the rollingstock and the axis exist, false if at least one is missing.

EEPRollingstockGetAxis()	EEPRollingstockGetAxis("Name", "Axis")
Parameters	two
Returns	two
Requires	EEP 11.0

```
Name = "Dispolok 189-917 EpVI"
Axis = "Pantograph DE"
ok, Position = EEPRollingstockGetAxis(Name,
Axis)
```

Purpose	Determines the current position of an axle named by axle name of the specified rollingstock.
Notes	• Parameter is the name of the rollingstock as a string • Second parameter is the name of the axis as a string. • First return value is true if the rollingstock and the axis exist, false if at least one is missing. • Second return value is the current position of the axis as a number.

EEPRollingstockSetAxisByNumber()	EEPRollingstockSetAxisByNumber("Name", AxisNumber, Position)
Parameters	three
Returns	one
Requires	EEP 18.0
Purpose	Moves the axle named by axle number of the specified rollingstock to the desired position.
Notes	• First parameter is the name of the rollingstock as a string. • Second parameter is the axis number of the axis to be moved. It is in the ini file of the model and is the same in all EEP language versions. • Third parameter is the target position of the axis. • Return value is either true, if the rollingstock and the axis exist, false if at least one is missing.

```
ok = EEPRollingstockSetAxisByNumber(
"Dispolok 189-917 EpVI", 26, 100)
```

EEPRollingstockGetAxisByNumber()	EEPRollingstockGetAxisByNumber("Name", "Axis")
Parameters	two
Returns	two
Requires	EEP 18.0
Purpose	Determines the current position of an axle named by axle number of the specified rollingstock.
Notes	• Parameter is the name of the rollingstock as a string • Second parameter is the axis number of the axis to be moved. It is in the ini file of the model and is the same in all EEP language versions. • First return value is true if the rollingstock and the axis exist, false if at least one is missing. • Second return value is the current position of the axis as a number.

```
Name = "Dispolok 189-917 EpVI"

ok, Position =
EEPRollingstockGetAxisByNumber(Name, 26)
```

EEPRollingstockSetSlot()	EEPRollingstockSetSlot("Name", AxesGroup)
Parameters	two
Returns	one
Requires	EEP 11.0
Purpose	Moves all axes of the specified rollingstock to the position saved in the axes group (slot).
Notes	• It is necessary first to set all axes to the desired position and save this state in one of the 16 axes groups available for each rollingstock (context menu). If this function is then executed later, all axes move from their current position to the position stored in the corresponding axes group (slot). • First parameter is the name of the rollingstock as a string. • Second parameter is the ID of the axes group in which the values related to the desired positions of the axes are stored. • Return value is either true if the rollingstock and the axes group exist, or false if at least one of them is missing. This function does not check if data has been saved in this axes group storage slot.

```
ok = EEPRollingstockSetSlot("PortalCrane_2", 1)
```

EEPGetRollingstockItemsCount()	EEPGetRollingstockItemsCount("#Name")
Parameters	one
Returns	one
Requires	EEP 13.2 Plug-in2
Purpose	Specifies how many rollingstocks are composing the "vehicles composite" (e.g. train)
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • Return value is the number of rollingstock composing the "vehicles composite". No distinction is made between engine, tender, wagon or special items like separate bogeys. If the "vehicles composite" doesn't exist, 0 will be the returned value.

```
Number =
EEPGetRollingstockItemsCount("#Freight_train")
```

EEPGetRollingstockItemName()	EEPGetRollingstockItemName("#Name", Number)
Parameters	two
Returns	one
Requires	EEP 13.2 Plug-in2
Purpose	Returns the name of a specified rollingstock in a "vehicles composite" (e.g. train).
Notes	• First parameter is the name of the "vehicles composite" as a string preceded by #. • Second parameter is the place of the rollingstock in the train set. **Please note** that the first rollingstock is given the number 0. • Return value is the name of the rollingstock. If the "vehicles composite" or the rollingstock doesn't exist, then EEP will return an empty string.

```
Name =
EEPGetRollingstockItemName("#Freight_train", 3)
```

EEPRollingstockGetTrainName()	EEPRollingstockGetTrainName("Name")
Parameters	one
Returns	two
Requires	EEP 15
Purpose	Returns the name of the "vehicles composite" (e.g. train) to which the specified rollingstock belongs.
Notes	• First parameter is the name of the rollingstock. • First return value is true when the execution was successful, false if not. • Second return value is the name of the "vehicles composite" including this rollingstock.

```
ok, Name = EEPRollingstockGetTrainName("Castor 1")
```

EEPRollingstockGetLength()		EEPRollingstockGetLength("Name")
Parameters	one	ok, Length = EEPRollingstockGetLength("Container 0100")
Returns	two	
Requires	EEP 15	
Purpose	Returns the length of a specified rollingstock.	
Notes	• First parameter is the name of the specified rollingstock. • First return value is true when the execution was successful, false if not. • Second return value is the length of the rollingstock in meters from coupler to coupler.	

EEPRollingstockGetMotor()		EEPRollingstockGetMotor("Name")
Parameters	one	ok, Motor = EEPRollingstockGetMotor("DB_360_339")
Returns	two	
Requires	EEP 15	
Purpose	Specifies if the rollingstock is motorized or not by giving its number of gears.	
Notes	• The parameter is the name of the rollingstock • First return value is true when the execution was successful, false if not. • Second return value is the number of gears available on that specified rollingstock. A non motorized vehicle will return 0 gear.	

EEPRollingstockGetTrack()		EEPRollingstockGetTrack("Name")
Parameters	one	ok, Track_ID, Position, Direction, System = EEPRollingstockGetTrack("BR 212 376-8")
Returns	five	
Requires	EEP 15	
Purpose	Returns the current position of a rollingstock on the track system.	
Notes	• The parameter is the name of the rollingstock. • First return value is true when the execution was successful, false if not. • Second return value is the ID of the track on which the rollingstock is located. • Third return value is the distance (in meters) between the rollingstock and the beginning of the track. • Fourth return value specifies the traffic direction: 1= in the direction of traffic 0 = in the opposite direction of traffic • Fifth return value specifies the type of track on which the rollingstock travels: 1 = railroad tracks 2 = roads 3 = subway and tram tracks 4 = waterways / other tracks	

EEPRollingstockGetModelType()	EEPRollingstockGetModelType("Name")
Parameters	one
Returns	two
Requires	EEP 15
Purpose	Provides information on the model category to which a rollingstock belongs.
Notes	• The parameter is the name of the rollingstock. • First return value is true when the execution was successful, false if not. • Second return value refers to the category in which the constructor has classified his model: 1 = Tank locomotive 2 = Locomotive with a separate, hauled tender 3 = Tender 4 = Electric locomotive 5 = Diesel locomotive 6 = Rail car 7 = Subway or commuter rail 8 = Tram 9 = Freight wagon 10 = Passenger wagon 11 = Aircraft 12 = Machine (e.g. crane) 13 = Ship 14 = Truck 15 = Car

```
ok, Type = EEPRollingstockGetModelType("Castor 1")
```

EEPRollingstockChangeOrientation()	EEPRollingstockChangeOrientation("Name")
Parameters	one
Returns	one
Requires	EEP 18.0
Purpose	Turns the orientation of a rollingstock inside a "vehicles composite" (e.g. train).
Notes	• Parameter is the name of the rollingstock • Return value is true when the execution was successful, false if not. • **Note:** The rollingstock is also rotated while travelling in a "vehicles composite".

```
ok =
EEPRollingstockChangeOrientation("DB Gbrs")
```

EEPRollingstockGetOrientation()	EEPRollingstockGetOrientation("Name")
Parameters	one
Returns	two
Requires	EEP 15.1 Plug-in 1
Purpose	Returns the orientation of a rollingstock inside a "vehicles composite" (e.g. train).
Notes	• First parameter is the name of the rollingstock • First return value is true when the execution was successful, false if not. • Second return value is true if the rollingstock is orientated in the direction of traffic, false if not.

```
ok, Direction =
EEPRollingstockGetOrientation("DB Gbrs")
```

EEPRollingstockSetHook()	EEPRollingstockSetHook("Name", true|false)
Parameters	two
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Activates or deactivates the merchandise hook of the specified rollingstock.
Notes	• First parameter is the entire name of the rollingstock as a string • Second parameter is either true (= hook on) or false (= hook off). • Return value is true when the execution was successful, false if not

```
ok = EEPRollingstockSetHook("Crane truck", true)
```

EEPRollingstockGetHook()	EEPRollingstockGetHook("Name")
Parameters	one
Returns	two
Requires	EEP 16.1
Purpose	Indicates whether or not the merchandise hook of a specific rollingstock is activated
Notes	• Parameter is the name of the rollingstock as a string. • First return value is true when the execution was successful, false if not • Second return value indicates whether the hook is : 0 = deactivated 1 = activated 3 = currently in use

```
ok, Status = EEPRollingstockGetHook("Crane truck")
```

EEPRollingstockSetHookGlue()	EEPRollingstockSetHookGlue("Name", true|false)
Parameters	two
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Influences the behavior of goods on the hook of a rollingstock
Notes	• First parameter is the name of the rollingstock as a string • Second parameter is either: true = hook glue on, i. e. the goods are fixed (ideal for clamps) false = hook glue off, i. e. the goods are swinging (ideal for hooks) • Return value is true when the execution was successful, false if not

```
ok = EEPRollingstockSetHookGlue("Crane truck",
true)
```

EEPRollingstockGetHookGlue()	EEPRollingstockGetHookGlue("Name")
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Indicates the behavior of the goods on the hook of a rollingstock
Notes	• Parameter is the entire name of the rollingstock as a string • Return value indicates whether the hook glue is: 0 = deactivated, the goods are swinging 1 = activated, the goods are fixed

```
ok = EEPRollingstockGetHookGlue("Crane truck")
```

EEPRollingstockGetHookPosition()		EEPRollingstockGetHookPosition("Name")
Parameters	one	ok, PosX, PosY, PosZ = EEPRollingstockGetHookPosition("Bridge crane - hook")
Returns	four	
Requires	EEP 18.1 - Plug-in 1	ok, PosX, PosY, PosZ = EEPRollingstockGetHookPosition("Bridge crane - 3 hooks", 2)
Purpose	Determines the position of the crane hook of the rolling stock in the EEP coordinate system.	
Notes	• Parameter is the full name of the rollingstock as a string. • The optional second parameter is the hook number if there are multiple hooks. • First return value is true if the execution was successful, otherwise false. • Second return value is the X-position in meter. • Third return value is the Y-position in meter. • Fourth return value is the Z-position in meter.	

EEPRollingstockSetSmoke()		EEPRollingstockSetSmoke("Name", true | false)
Parameters	two	
Returns	one	ok = EEPRollingstockSetSmoke("vehicle", true)
Requires	EEP 16.1 Plug-in 1	
Purpose	Turns the rollingstock's smoke on or off	
Notes	• First parameter is the name of the rollingstock as a string • Second parameter switches the smoke on with true or off with false. • Return value is true when the execution was successful, false if not	

EEPRollingstockGetSmoke()		EEPRollingstockGetSmoke("Name")
Parameters	one	
Returns	two	ok, Smoke = EEPRollingstockGetSmoke("vehicle")
Requires	EEP 16.1 Plug-in 1	
Purpose	Indicates if the smoke of a rollingstock is currently turned on or off.	
Notes	• Parameter is the name of the rollingstock as a string • First return value is true when the execution was successful, false if not • Second return value is false if the smoke is turned off or true if it is turned on.	

EEPRollingstockSetHorn()	EEPRollingstockSetHorn("Name" [, true | false])
Parameters	One or two
Returns	one
Requires	EEP 16.1 Plug-in 1

```
ok = EEPRollingstockSetHorn("Car")
ok = EEPRollingstockSetHorn("Car", true)
ok = EEPRollingstockSetHorn("Car", false)
```

Purpose	Turns the warning sound of the specified rollingstock on or off.
Notes	• First parameter is the name of the rollingstock as a string • Second optional parameter allows the warning tone to sound with true or stops it with false before it has faded away. As true, is set by default, the horn will sound even without second argument. • Return value is true when the execution was successful, false if not

EEPRollingstockGetMileage()	EEPRollingstockGetMileage("Name")
Parameters	one
Returns	two
Requires	EEP 16.1 Plug-in 1

```
ok, Distance =
EEPRollingstockGetMileage("vehicle")
```

Purpose	Returns the covered distance of a specified rollingstock.
Notes	• Parameter is the name of the rollingstock as a string • First return value is true when the execution was successful, false if not • Second return value is the distance in meters that the rollingstock has traveled since its use in EEP.

EEPRollingstockGetPosition()	EEPRollingstockGetPosition("Name")
Parameters	one
Returns	four
Requires	EEP 16.1 Plug-in 1

```
ok, PosX, PosY, PosZ =
EEPRollingstockGetPosition("vehicle")
```

Purpose	Returns the position of the rollingstock in the EEP coordinate system
Notes	• Parameter is the name of the rollingstock as a string • First return value is true when the execution was successful false if not • Second return value indicates its X position in meters • Third return value indicates its Y position in meters • Fourth return value indicates its Z position in meters

EEPRollingstockGetRotation()	EEPRollingstockGetRotation("Name")
Parameters	one
Returns	four
Requires	EEP 18.1 - Plug-in 1

```
ok, RotX, RotY, RotZ =
EEPRollingstockGetRotation("DB_Sdgkms_3180409-355")
```

Purpose	Determines the orientation of the rollingstock in the EEP coordinate system.
Notes	• Parameter is the full name of the rollingstock as a string. • First return value is true if the execution was successful, otherwise false. • Second return value is the rotation around the around the X-axis in degrees. • Third return value is the rotation around the around the Y-axis in degrees. • Fourth return value is the rotation around the around the Z-axis in degrees.

8 Structure functions

EEPStructureSetSmoke()		EEPStructureSetSmoke("#Lua-Name", true|false)
Parameters	two	Name = "#1_Lauscha_manufacture" ok = EEPStructureSetSmoke(Name, true) ok = EEPStructureSetSmoke("#1", true)
Returns	one	
Requires	EEP 11.1 Plug-in 1	
Purpose	Turns the smoke of a structure on or off.	
Notes	• First parameter is the **Lua name** of the structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter turns the smoke on with true, or off with false. • Return value is either true when the structure exists and has a smoke emission function, or false if at least one of the two is missing.	

EEPStructureGetSmoke()		EEPStructureGetSmoke("#Lua-Name")
Parameters	one	Name = "#1_Lauscha_Manufacture" ok, Smoke = EEPStructureSetSmoke(Name, true) ok, Smoke = EEPStructureGetSmoke("#1")
Returns	two	
Requires	EEP 11.1 Plug-in 1	
Purpose	Specifies whether smoke emission on a structure is enabled or disabled.	
Notes	• Parameter is the **Lua name** of the specified structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true if the structure exists and has a smoke emission function, false if at least one of the two is missing. • Second return value is true if the smoke function is activated, false if not. • **Be aware that** once the smoke emission of a structure has been modified by using *EEPStructureSetSmoke()* you'll have to wait until a new *EEPMain()* cycle has begun to ask with EEPStructureGetSmoke() whether the structure is emitting smoke or not.	

EEPStructureSetLight()		EEPStructureSetLight("#Lua-Name", true|false)
Parameters	two	Name = "#13_Industrial_buildings" ok = EEPStructureSetLight(Name, true) ok = EEPStructureSetLight("#13", true)
Returns	one	
Requires	EEP 11.1 Plug-in 1	
Purpose	Turns the lights of a structure on or off.	
Notes	• Parameter is the **Lua name** of the specified structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter turns the lights on with true, or off with false. • Return value is either true if the structure exists and has a light feature, or false if at least one of the two is missing.	

EEPStructureGetLight()		EEPStructureGetLight("#Lua-Name")
Parameters	one	Name = "#13_Industrial_buildings" ok, Lights = EEPStructureSetLight(Name, true) ok, Lights = EEPStructureGetLight("#13")
Returns	two	
Requires	EEP 11.1 Plug-in 1	
Purpose	Specifies whether the lights of a structure are turned on or off.	
Notes	• Parameter is the **Lua name** of the specified structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true if the structure exists and has a light feature, false if at least one of the two is missing. • Second return value is either true when the lights are turned on, or false if the lights are turned off or set to automatic mode. • **Be aware that** once the lights of a structure have been turned on or off by using *EEPStructureSetLight()* you'll have to wait until a new *EEPMain()* cycle has begun to get their new status with EEPStructureGetLight().	

EEPStructureSetFire()		EEPStructureSetFire("#Lua-Name",true|false)
Parameters	two	Name = "#15_Burning_house_01_SB1" ok = EEPStructureSetFire(Name, true) ok = EEPStructureSetFire("#15",true)
Returns	one	
Requires	EEP 11.1 Plug-in 1	
Purpose	Turns the fire feature of a structure on or off.	
Notes	• First parameter is the **Lua name** of the specified structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter activates the fire with true or deactivates it with false. • Return value is either true if the structure exists and has a fire feature, or false if at least one of the two is missing.	

EEPStructureGetFire()		EEPStructureGetFire("#Lua-Name")
Parameters	one	Name = "#15_Burning_house_01_SB1" ok, Fire = EEPStructureGetFire(Name) ok, Fire = EEPStructureGetFire("#15")
Returns	two	
Requires	EEP 11.1 Plug-in 1	
Purpose	Specifies whether the fire feature of a structure is turned on or off.	
Notes	• Parameter is the **Lua name** of the specified structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true if the structure exists and has a fire feature, or false if at least one of the two is missing. • Second return value is either true if the fire function is turned on, or false if it's turned off. • **Be aware that** once the fire function of a structure has been turned on or off by using *EEPStructureSetFire()* you'll have to wait until a new *EEPMain()* cycle has begun to get its new status with EEPStructureGetFire().	

EEPStructureSetAxis()		EEPStructureSetAxis("#Lua-Name", "Axis", Position)
Parameters	three	Name = "#83_Turntable" Axis = "Bridge" ok = EEPStructureSetAxis(Name, Axis, 50) ok = EEPStructureSetAxis("#83", "Bridge", 50)
Returns	one	
Requires	EEP 11.1 Plug-in 1	
Purpose	Sets the axis of the structure or track-side object defined by the axis name to a new position.	
Notes	• First parameter is the **Lua name** of the specified structure or track-side object as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the entire name of the axis as a string. • Third parameter is the new position of the specified axis. • Return value is either true if the structure and the specified axis exist, or false if at least one of the two is missing.	

EEPStructureGetAxis()		EEPStructureGetAxis("#Lua-Name", "Axis")
Parameters	two	Name = "#83_Turntable" Axis = "Bridge" ok, Orientation = EEPStructureGetAxis(Name, Axis) ok, Orientation = EEPStructureGetAxis("#83", "Bridge")
Returns	two	
Requires	EEP 11.1 Plug-in 1	
Purpose	Returns the position of an axis defined by the axis name of the specified structure or track-side object	
Notes	• First parameter is the **Lua name** of the specified structure or track-side object as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the entire name of the axis as a string. • First return value is true, if the structure and the axis exist, false if at least one of the two is missing. • Second return value is the current position of the axis as a number. • **Be aware that** once the position of the axis has been modified on or off by using *EEPStructureSetAxis()* you'll have to wait until a new *EEPMain()* cycle has begun to retrieve its new position with EEPStructureGetAxis().	

EEPStructureSetAxisByNumber()		EEPStructureSetAxisByNumber("#Lua-Name", "Axis", Position)
Parameters	three	Name = "#83_Turntable" ok = EEPStructureSetAxisByNumber(Name, 1, 50) ok = EEPStructureSetAxisByNumber("#83", 1, 50)
Returns	one	
Requires	EEP 18.0	
Purpose	Sets the axis of the structure or track-side object defined by the axis number to a new position.	
Notes	• First parameter is the **Lua name** of the specified structure or track-side object as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the number of the axis. • Third parameter is the new position of the specified axis. • Return value is either true if the structure and the specified axis exist, or false if at least one of the two is missing.	

EEPStructureGetAxisByNumber()		EEPStructureGetAxisByNumber("#Lua-Name", "Axis")
Parameters	two	Name = "#83_Turntable" ok, Orientation = EEPStructureGetAxisByNumber(Name, 1) ok, Orientation = EEPStructureGetAxisByNumber("#83", 1)
Returns	two	
Requires	EEP 18.0	
Purpose	Returns the position of an axis defined by the axis number of the specified structure or track-side object	
Notes	• First parameter is the **Lua name** of the specified structure or track-side object as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the number of the axis. • First return value is true, if the structure and the axis exist, false if at least one of the two is missing. • Second return value is the current position of the axis as a number. • **Be aware that** once the position of the axis has been modified on or off by using *EEPStructureSetAxisByNumber()* you'll have to wait until a new *EEPMain()* cycle has begun to retrieve its new position with EEPStructureGetAxisByNumber().	

EEPStructureAnimateAxis()		EEPStructureAnimateAxis("#Lua-Name", "Axis", Position)
Parameters	three	ok = EEPStructureAnimateAxis("#17_Windmill", "Rotor", 1000) ok = EEPStructureAnimateAxis("#17", "Rotor", 1000)
Returns	one	
Requires	EEP 11.1 Plug-in 1	
Purpose	Animates the specified axis of the structure or track-side object.	
Notes	• First parameter is the **Lua name** of the specified structure or track-side object as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the entire name of the axis as a string • Third parameter is the number of steps (positive or negative) the axis shall move. A value of 1000 or -1000 starts an endless motion if the model was built for it. A value of 0 stops any motion. • Return value is either true if the structure and the axis exist, or false if at least one of the two is missing.	

EEPStructureIsAxisAnimate()		EEPStructureIsAxisAnimate("#Lua-Name", "Axis")
Parameters	two	ok, Status = EEPStructureIsAxisAnimate("#17", "Rotor") if Status > 0 then print("The rotor of the windmill is rotating") end
Returns	two	
Requires	EEP 15	
Purpose	Specifies whether the moving part of the concerned structure is currently in motion and whether this motion is continuous or not.	
Notes	• First parameter is the **Lua name** of the specified structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the entire name of the axis. • First return value is true, when the execution was successful, false if not • Second return value is : 0, if the axis is not moving 1, if the axis is moving in an infinite loop 2, if the axis is moving according to a given number • **Be aware that** once the axis has been animated by using *EEPStructureAnimateAxis()* you'll have to wait until a new *EEPMain()* cycle has begun to be informed if the axis is currently animated thanks to the EEPStructureIsAxisAnimate() function.	

EEPStructureSetPosition()	EEPStructureSetPosition("#Lua-Name", Pos_X, Pos_Y, Pos_Z)
Parameters	four
Returns	one
Requires	EEP 11.1 Plug-in 1

```
Name = "#55_Strawball"
ok = EEPStructureSetPosition(Name, 11, 17, 3)

ok = EEPStructureSetPosition("#55", 11, 17, 3)
```

Purpose: Places the specified structure or track-side object at a new position.

Notes:
- First parameter is the **Lua name** of the specified structure or landscape element as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument.
- Second parameter is the new X position of the structure in meters.
- Third parameter is the new Y position of the structure in meters.
- Fourth parameter is the Z position of the structure in meters.
- Structures cannot be placed outside the layout's boundaries.
- Return value is true if the structure exists, false if it doesn't or if the new position is outside the layout.

EEPStructureGetPosition()	EEPStructureGetPosition("#Lua-Name")
Parameters	one
Returns	four
Requires	EEP 15

```
Ok, Pos_X, Pos_Y, Pos_Z =
EEPStructureGetPosition("#55_Strawball")

Ok, Pos_X, Pos_Y, Pos_Z =
EEPStructureGetPosition("#55")
```

Purpose: Returns the current position of a structure or a landscape element.

Notes:
- Parameter is the **Lua name** of the specified structure or landscape element as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument.
- First return value is true when the execution was successful, false if not.
- Second return value is the X position of the structure in meters.
- Third return value is the Y position of the structure in meters.
- Fourth return value is the Z position of the structure in meters.
- **Be aware that** once the structure has been moved by using *EEPStructureSetPosition()* you'll have to wait until a new *EEPMain()* cycle has begun to get its new position thanks to the EEPStructureGetPosition() function.

EEPStructureSetRotation()	EEPStructureSetRotation("#Lua-Name", Rot_X, Rot_Y, Rot_Z)
Parameters	four
Returns	one
Requires	EEP 11.1 Plug-in 1

```
Name = "#55_Strawball"
ok = EEPStructureSetRotation(Name, 0, 0, 25)

ok = EEPStructureSetRotation("#55", 0, 0, 25)
```

Purpose: Rotates the relevant element to give it a new orientation.

Notes:
- First parameter is the **Lua name** of the specified structure or landscape element as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument.
- Second parameter is the rotation of the structure around the X-axis in degrees.
- Third parameter is the rotation of the structure around the Y-axis in degrees.
- Fourth parameter is the rotation of the structure around the Z-axis in degrees.
- Return value is true if the structure exists, false if not.

EEPStructureGetRotation()		EEPStructureGetRotation("#Lua-Name")
Parameters	one	ok, Pos_X, Pos_Y, Pos_Z = EEPStructureGetRotation("#55_Strawball")
Returns	four	ok, Pos_X, Pos_Y, Pos_Z = EEPStructureGetRotation("#55")
Requires	EEP 11.1 Plug-in 1	
Purpose	Returns the current orientation of a structure or a landscape element.	
Notes	• Parameter is the **Lua name** of the specified structure or landscape element as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true when the execution was successful false if not • Second return value is the rotation of the structure around the X-axis in degrees. • Third return value is the rotation of the structure around the Y-axis in degrees. • Fourth return value is the rotation of the structure around the Z-axis in degrees. • **Be aware that** once the structure has been rotated by using *EEPStructureSetRotation()* you'll have to wait until a new *EEPMain()* cycle has begun to get its new orientation thanks to the EEPStructureGetRotation() function.	

EEPStructureGetModelType()		EEPStructureGetModelType("#Lua-Name")
Parameters	one	
Returns	two	ok, Type = EEPStructureGetModelType("#23")
Requires	EEP 15	
Purpose	Returns the model category type of a structure or landscape element	
Notes	• Parameter is the **Lua name** of the specified structure or landscape element as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true when the execution was successful, false if not. • Second return value is the category that was assigned to this element by the constructor: 16 = Track objects: railway tracks 17 = Track objects: tram tracks 18 = Track objects: road tracks 19 = Track objects: water or air tracks. 22 = Real estate 23 = Landscape elements: fauna 24 = Landscape elements: flora 25 = Landscape elements: terra 38 = Landscape elements: 3D ground textures	

EEPStructureSetLightingColour()	EEPStructureSetLightingColour("#Lua-Name", R, G, B)
Parameters	four
Returns	one
Requires	EEP 18.1 -Plug-in 1

```
Name = "#13_Buckingham_Palace"
ok = EEPStructureSetLightingColour(Name, 128, 255, 128)

ok = EEPStructureSetLightingColour("#13", 128, 255, 128)
```

Purpose

Sets alternative lighting colour for named structure.

Notes

- First parameter is the **Lua name** of the specified structure as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument.
- Parameters 2 – 4 determine the colour from the proportions of red, green and blue, each in the range from 0 to 255.
- Return value is true if the execution was successful, otherwise false.

EEPStructurePlaySound()	Enables or disables the sound of a sound model from the category landscape elements/sound.	16-1
EEPStructureSetTextureText()	Adds a new text to the writable area of a landscape or structure element.	17-1
EEPStructureGetTextureText()	Returns the text of a writable area of a landscape or structure element.	17-4
EEPStructureSetTagText()	Changes the tag-text of a structure or a landscape element.	18-1
EEPStructureGetTagText()	Returns the tag text of a property or a landscape element.	18-1
EEPChangeInfoStructure()	Assigns a new tip text to a structure or landscape element.	14-1
EEPShowInfoStructure()	Turns the tip text of the specified structure or landscape element on or off.	14-1

9 Cargo functions

EEPGoodsSetPosition ()		EEPGoodsSetPosition("#Lua-Name", Pos_X, Pos_Y, Pos_Z)
Parameters	four	ok = EEPGoodsSetPosition("#66_boxes", Pos_X, Pos_Y, Pos_Z) ok = EEPGoodsSetPosition("#66", Pos_X, Pos_Y, Pos_Z)
Returns	one	
Requires	EEP 15	
Purpose	Changes the position of a cargo model on the layout.	
Notes	• First parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the X position of the cargo in meters. • Third parameter is the Y position of the cargo in meters. • Fourth parameter is the Z position of the cargo in meters. • Return value is true when the execution was successful, false if not.	

EEPGoodsGetPosition ()		EEPGoodsGetPosition("#Lua-Name)
Parameters	one	ok, Pos_X, Pos_Y, Pos_Z = EEPGoodsGetPosition("#66_cartons") ok, Pos_X, Pos_Y, Pos_Z = EEPGoodsGetPosition("#66")
Returns	four	
Requires	EEP 15	
Purpose	Returns the current position of a cargo	
Notes	• Parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true when the execution was successful, false if not. • Second return value is the X position of the cargo in meters. • Third return value is the Y position of the cargo in meters. • Fourth return value is the Z position of the cargo in meters. • **Be aware that** once the cargo has been moved by using *EEPGoodsSetPosition()* you'll have to wait until a new *EEPMain()* cycle has begun to get its new position thanks to the EEPGoodsGetPosition() function.	

EEPGoodsSetRotation ()	EEPGoodsSetRotation("#Lua-Name", Rot_X, Rot_Y, Rot_Z)
Parameters	four
Returns	one
Requires	EEP 15

```
ok = EEPGoodsSetRotation("#66_boxes", 0, 0, 30)

ok = EEPGoodsSetRotation("#66", 0, 0, 30)
```

Purpose	Changes the orientation of a cargo on the layout
Notes	• First parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the rotation of the cargo around the X-axis in degrees. • Third parameter is the rotation of the cargo around the Y-axis in degrees. • Fourth parameter is the rotation of the cargo around the Z-axis in degrees. • Return value is true when the execution was successful, false if not.

EEPGoodsGetRotation()	EEPGoodsGetRotation("#Lua-Name")
Parameters	one
Returns	four
Requires	EEP 16.1 Plug-in 1

```
ok, Pos_X, Pos_Y, Pos_Z =
EEPGoodsGetRotation("#66_boxes")

ok, Pos_X, Pos_Y, Pos_Z =
EEPGoodsGetRotation("#66")
```

Purpose	Returns the current orientation of a cargo
Notes	• Parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true when the execution was successful false if not • Second return value is the rotation of the cargo around the X-axis in degrees. • Third return value is the rotation of the cargo around the Y-axis in degrees. • Fourth return value is the rotation of the cargo around the Z-axis in degrees. • **Be aware that** once the cargo has been rotated by using *EEPGoodsSetRotation()* you'll have to wait until a new *EEPMain()* cycle has begun to get its new position thanks to the EEPGoodsGetRotation() function.

EEPGoodsGetModelType()	EEPGoodsGetModelType("#Lua-Name")
Parameters	one
Returns	two
Requires	EEP 15

```
ok, Type = EEPGoodsGetModelType("#78")
```

Purpose	Returns the model category type of a cargo
Notes	• Parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • First return value is true when the execution was successful, false if not. • Second return value is the category that was assigned to this cargo by the constructor : 20 = boxes (cubic cargoes) 21 = cylinders (cylindrical cargoes)

EEPGoodsSetAxis()		EEPGoodsSetAxis("#Lua-Name", "Axis", Position)
Parameters	three	Name = "#83_IF_ctnr_OT20_C1 (MS7)" Axis = "Cargo level" ok = EEPGoodsSetAxis(Name, Axis, 50) ok = EEPGoodsSetAxis("#83", "Cargo level", 50)
Returns	one	
Requires	EEP 18.0	
Purpose	Sets the axle of the named cargo defined by the axle name to a specific position (without animation).	
Notes	• Parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the entire name of the axis as a string. • Third parameter is the new position of the specified axis. • Return value is either true if the cargo and the specified axis exist, or false if at least one of the two is missing.	

EEPGoodsGetAxis()		EEPGoodsGetAxis("#Lua-Name", "Axis")
Parameters	two	Name = "#83_IF_ctnr_OT20_C1 (MS7)" Axis = "Cargo level" ok, Position = EEPGoodsGetAxis(Name, Axis) ok, Position = EEPGoodsGetAxis("#83", "Cargo level")
Returns	two	
Requires	EEP 18.0	
Purpose	Returns the position of an axle of the named cargo defined by the axle name.	
Notes	• Parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the entire name of the axis as a string. • First return value is true, if the cargo and the axis exist, false if at least one of the two is missing. • Second return value is the current position of the axis as a number. • **Be aware that** once the position of the axis has been modified on or off by using *EEPGoodsSetAxis()* you'll have to wait until a new *EEPMain()* cycle has begun to retrieve its new position with EEPGoodsGetAxis().	

EEPGoodsSetAxisByNumber()		EEPGoodsSetAxisByNumber("#Lua-Name", "Axis", Position)
Parameters	three	Name = "#83_IF_ctnr_OT20_C1 (MS7)" ok = EEPGoodsSetAxisByNumber(Name, 1, 50) ok = EEPGoodsSetAxisByNumber("#83", 1, 50)
Returns	one	
Requires	EEP 18.0	
Purpose	Sets the axle of the named cargo defined by the axle number to a specific position (without animation).	
Notes	• Parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the axis number of the axis to be moved. It is in the ini file of the model and is the same in all EEP language versions. • Third parameter is the new position of the specified axis. • Return value is either true if the cargo and the specified axis exist, or false if at least one of the two is missing.	

EEPGoodsGetAxisByNumber()	EEPGoodsGetAxisByNumber("#Lua-Name", "Axis")
Parameters	two
Returns	two
Requires	EEP 18.0

```
Name = "#83_IF_ctnr_OT20_C1 (MS7)"
ok, Position = EEPGoodsGetAxisByNumber(Name, 1)

ok, Position = EEPGoodsGetAxisByNumber("#83", 1)
```

Purpose	Returns the position of an axle of the named cargo defined by the axle number.
Notes	• Parameter is the **Lua name** of the cargo as a string. It is in the object properties and differs from the model name by the prefixed ID. The ID preceded by # is already sufficient for the argument. • Second parameter is the axis number of the axis to be moved. It is in the ini file of the model and is the same in all EEP language versions. • First return value is true, if the cargo and the axis exist, false if at least one of the two is missing. • Second return value is the current position of the axis as a number. • **Be aware that** once the position of the axis has been modified on or off by using *EEPGoodsSetAxisByNumber()* you'll have to wait until a new *EEPMain()* cycle has begun to retrieve its new position with EEPGoodsGetAxisByNumber().

10 Track functions

EEPRegisterRailTrack()		EEPRegisterRailTrack(ID)
Parameters	one	ok = EEPRegisterRailTrack(123)
Returns	one	
Requires	EEP 11.3 - Plug-in 3	
Purpose	Registers a rail track for a future occupancy enquiry.	
Notes	• Parameter is the ID of the track to be registered. • Return value is true if the track exists, false if not. • Registration is mandatory for *EEPIsRailTrackReserved()* enquiry.	

EEPIsRailTrackReserved()		EEPIsRailTrackReserved(ID [,true | number])
Parameters	one or two	
Returns	two or three	ok, Occupied = EEPIsRailTrackReserved(123)
Requires	EEP 11.3 - Plug-in 3 EEP 13.2 - Plug-in 2 EEP 17.2 - Plug-in 2	ok, Occupied, Name = EEPIsRailTrackReserved(123, true) ok, Occupied, Name = EEPIsRailTrackReserved(123, 2)
Purpose	Determines whether a track element is occupied or whether it is occupied with the xth "vehicles composite" and then returns its name.	
Notes	• Parameter is the ID of the rail track to be checked. • **Since EEP 13.2 Plug-in 2**, an optional second parameter true or **1** can be given so that the function returns the name of the foremost "vehicles composite" in the track direction as the third return value. **Since EEP 17.2 Plug-in 2**, **numbers greater than 1** can also be entered so that the function outputs the name of the "vehicles composite" on the corresponding rear position as the third return value. • First return value is true if the targeted rail track exists and has been registered beforehand, false if not. • Second return value is true without the second parameter if the track is occupied, otherwise false. With the second parameter it returns true if there is a "vehicles composite" at the location designated in the second parameter (where true = 1 there), otherwise false. • Third **(optional)** return value is the name of the "vehicles composite" occupying the track at the position specified as the second parameter (where true = 1). If a number is entered as the third parameter which is greater than the number of "vehicles composites" on the track, the third return value is an empty character string. **ATTENTION:** The third return value is **correct as long as the trains are not moving!** But if the vehicles are moving, the Lua interpreter can deliver wrong names, because the list with the names changes e.g. with 60 fps, but Lua runs asynchronously in another thread (CPU) in order not to slow down EEP. • Remember to register the track before using this function.	

EEPRegisterRoadTrack()	EEPRegisterRoadTrack(ID)
Parameters	one
Returns	one
Requires	EEP 11.3 - Plug-in 3
Purpose	Registers a road for a future occupancy enquiry.
Notes	• Parameter is the ID of the road to be registered. • Return value is true if the road exists, false if not. • Registration is mandatory for *EEPIsRoadTrackReserved()* enquiry.

```
ok = EEPRegisterRoadTrack(211)
```

EEPIsRoadTrackReserved()	EEPIsRoadTrackReserved(ID [,true | number])
Parameters	one or two
Returns	two or three
Requires	EEP 11.3 - Plug-in 3 EEP 13.2 - Plug-in 2 EEP 17.2 - Plug-in 2
Purpose	Determines whether a road track element is occupied or whether it is occupied with the xth "vehicles composite" and then returns its name.
Notes	• Parameter is the ID of the road track to be checked. • **Since EEP 13.2 Plug-in 2**, an optional second parameter true or **1** can be given so that the function returns the name of the foremost "vehicles composite" in the road direction as the third return value. **Since EEP 17.2 Plug-in 2**, **numbers greater than 1** can also be entered so that the function outputs the name of the "vehicles composite" on the corresponding rear position as the third return value. • First return value is true if the targeted road exists and has been registered beforehand, false if not. • Second return value is true without the second parameter if the road track is occupied, otherwise false. With the second parameter it returns true if there is a "vehicles composite" at the location designated in the second parameter (where true = 1 there), otherwise false. • Third **(optional)** return value is the name of the "vehicles composite" occupying the road track at the position specified as the second parameter (where true = 1). If a number is entered as the third parameter which is greater than the number of "vehicles composites" on the road track, the third return value is an empty character string. **ATTENTION:** The third return value is **correct as long as the trains are not moving!** But if the vehicles are moving, the Lua interpreter can deliver wrong names, because the list with the names changes e.g. with 60 fps, but Lua runs asynchronously in another thread (CPU) in order not to slow down EEP. • Remember to register the road before using this function.

```
ok, Occupied = EEPIsRoadTrackReserved(211)

ok, Occupied, Name =
   EEPIsRoadTrackReserved(211, true)

ok, Occupied, Name =
   EEPIsRoadTrackReserved(211, 3)
```

EEPRegisterTramTrack()	EEPRegisterTramTrack(ID)
Parameters	one
Returns	one
Requires	EEP 11.3 - -Plug-in 3
Purpose	Registers a tramway track for a future occupancy enquiry.
Notes	• Parameter is the ID of the tramway track to be registered. • Return value is true if the tramway track exists, false if not. • Registration is mandatory for *EEPIsTramTrackReserved()* enquiry.

```
ok = EEPRegisterTramTrack(187)
```

EEPIsTramTrackReserved()	EEPIsTramTrackReserved(ID [,true | number])
Parameters	one or two
Returns	two or three
Requires	EEP 11.3 - Plug-in 3 EEP 13.2 - Plug-in 2 EEP 17.2 - Plug-in 2
Purpose	Determines whether a tramway track element is occupied or whether it is occupied with the xth "vehicles composite" and then returns its name.
Notes	• Parameter is the ID of the tramway track to be checked. • **Since EEP 13.2 Plug-in 2**, an optional second parameter true or **1** can be given so that the function returns the name of the foremost "vehicles composite" in the tram track direction as the third return value. **Since EEP 17.2 Plug-in 2**, **numbers greater than 1** can also be entered so that the function outputs the name of the "vehicles composite" on the corresponding rear position as the third return value. • First return value is true if the targeted tramway track exists and has been registered beforehand, false if not. • Second return value is true without the second parameter if the tram track is occupied, otherwise false. With the second parameter it returns true if there is a "vehicles composite" at the location designated in the second parameter (where true = 1 there), otherwise false. • Third **(optional)** return value is the name of the "vehicles composite" occupying the tram track at the position specified as the second parameter (where true = 1). If a number is entered as the third parameter which is greater than the number of "vehicles composites" on the tram track, the third return value is an empty character string. **ATTENTION:** The third return value is **correct as long as the trains are not moving!** But if the vehicles are moving, the Lua interpreter can deliver wrong names, because the list with the names changes e.g. with 60 fps, but Lua runs asynchronously in another thread (CPU) in order not to slow down EEP. • Remember to register the tramway track before using this function.

```
ok, Occupied = EEPIsTramTrackReserved(187)

ok, Occupied, Name = EEPIsTramTrackReserved(187, true)

ok, Occupied, Name = EEPIsTramTrackReserved(187, 2)
```

EEPRegisterAuxiliaryTrack()		EEPRegisterAuxiliaryTrack(ID)
Parameters	one	ok = EEPRegisterAuxiliaryTrack(321)
Returns	one	
Requires	EEP 11.3 - Plug-in 3	
Purpose	Registers an auxiliary track for a future occupancy enquiry.	
Notes	• Parameter is the ID of the auxiliary track to be registered. • Return value is true if the auxiliary track exists, false if not. • Registration is mandatory for *EEPIsAuxiliaryTrackReserved()* enquiry.	

EEPIsAuxiliaryTrackReserved()		EEPIsAuxiliaryTrackReserved(ID [,true | number])
Parameters	one or two	ok, Occupied = EEPIsAuxiliaryTrackReserved(321) ok, Occupied, Name = EEPIsAuxiliaryTrackReserved(321,true) ok, Occupied, Name = EEPIsAuxiliaryTrackReserved(321,3)
Returns	two or three	
Requires	EEP 11.3 - Plug-in 3 EEP 13.2 - Plug-in 2 EEP 17.2 - Plug-in 2	
Purpose	Determines whether an auxiliary track element is occupied or whether it is occupied with the xth "vehicles composite" and then returns its name.	
Notes	• Parameter is the ID of the auxiliary track to be checked. • **Since EEP 13.2 Plug-in 2**, an optional second parameter true or **1** can be given so that the function returns the name of the foremost "vehicles composite" in the track direction as the third return value. **Since EEP 17.2 Plug-in 2**, **numbers greater than 1** can also be entered so that the function outputs the name of the "vehicles composite" on the corresponding rear position as the third return value • First return value is true if the targeted auxiliary track exists and has been registered beforehand, false if not. • Second return value is true without the second parameter if the auxiliary track is occupied, otherwise false. With the second parameter it returns true if there is a "vehicles composite" at the location designated in the second parameter (where true = 1 there), otherwise false. • Third **(optional)** return value is the name of the "vehicles composite" occupying the auxiliary track at the position specified as the second parameter (where true = 1). If a number is entered as the third parameter which is greater than the number of "vehicles composites" on the auxiliary track, the third return value is an empty character string. **ATTENTION:** The third return value is **correct as long as the trains are not moving!** But if the vehicles are moving, the Lua interpreter can deliver wrong names, because the list with the names changes e.g. with 60 fps, but Lua runs asynchronously in another thread (CPU) in order not to slow down EEP. • Remember to register the auxiliary track before using this function.	

EEPRegisterControlTrack()	EEPRegisterControlTrack(ID)
Parameters	one
Returns	one
Requires	EEP 11.3 - Plug-in 3
Purpose	Registers a control track element for a future occupancy enquiry.
Notes	• Parameter is the ID of the section to be registered. • Return value is true if the auxiliary track exists, false if not. • Registration is mandatory for *EEPIsControlTrackReserved()* enquiry.

```
ok = EEPRegisterControlTrack(333)
```

EEPIsControlTrackReserved()	EEPIsControlTrackReserved(ID [,true | number])
Parameters	one or two
Returns	two or three
Requires	EEP 11.3 - Plug-in 3 EEP 13.2 - Plug-in 2 EEP 17.2 - Plug-in 2
Purpose	Determines whether a control track element is occupied or whether it is occupied with the xth "vehicles composite" and then returns its name
Notes	• Parameter is the ID of the section of the control track system to be checked. • **Since EEP 13.2 Plug-in 2**, an optional second parameter true or **1** can be given so that the function returns the name of the foremost "vehicles composite" in the control track direction as the third return value. **Since EEP 17.2 Plug-in 2**, **numbers greater than 1** can also be entered so that the function outputs the name of the "vehicles composite" on the corresponding rear position as the third return value • First return value is true if the targeted control track system exists and has been registered beforehand, false if not. • Second return value is true without the second parameter if the contol track is occupied, otherwise false. With the second parameter it returns true if there is a "vehicles composite" at the location designated in the second parameter (where true = 1 there), otherwise false. • Third **(optional)** return value is the name of the "vehicles composite" occupying the control track at the position specified as the second parameter (where true = 1). If a number is entered as the third parameter which is greater than the number of "vehicles composites" on the control track, the third return value is an empty character string. **ATTENTION:** The third return value is **correct as long as the trains are not moving!** But if the vehicles are moving, the Lua interpreter can deliver wrong names, because the list with the names changes e.g. with 60 fps, but Lua runs asynchronously in another thread (CPU) in order not to slow down EEP. • Remember to register the control track system before using this function.

```
ok, Occupied = EEPIsControlTrackReserved(333)

ok, Occupied, Name =
   EEPIsControlTrackReserved(333, true)

ok, Occupied, Name =
   EEPIsControlTrackReserved(333, 1)
```

EEPRailTrackChangeAppearance()		EEPRailTrackChangeAppearance(RailTrackID, LayerNumber)
Parameters	two	
Returns	one	`ok = EEPRailTrackChangeAppearance(97, 2)`
Requires	EEP 18.1 - Plug-in 1	
Purpose	Changes the layer of a multi-layer rail track and thus the appearance of the track surface.	
Notes	• First parameter is the ID of the rail section, whose layer should be changed. • Second parameter is the number of the desired layer. If a number greater than the number of existing layers is specified, the input is internally set to the highest layer number. • Return value is true when the execution was successful, false if not.	

EEPRoadTrackChangeAppearance()		EEPRoadTrackChangeAppearance(RoadTrackID, LayerNumber)
Parameters	two	
Returns	one	`ok = EEPRoadTrackChangeAppearance(87, 3)`
Requires	EEP 18.1 - Plug-in 1	
Purpose	Changes the layer of a multi-layer road and thus the appearance of the road surface.	
Notes	• First parameter is the ID of the road section, whose layer should be changed. • Second parameter is the number of the desired layer. If a number greater than the number of existing layers is specified, the input is internally set to the highest layer number. • Return value is true when the execution was successful, false if not.	

EEPTramTrackChangeAppearance()		EEPTramTrackChangeAppearance(TramTrackID, LayerNumber)
Parameters	two	
Returns	one	`ok = EEPTramTrackChangeAppearance(87, 3)`
Requires	EEP 18.1 - Plug-in 1	
Purpose	Changes the layer of a multi-layer tram track and thus the appearance of the track surface.	
Notes	• First parameter is the ID of the tram track, whose layer should be changed. • Second parameter is the number of the desired layer. If a number greater than the number of existing layers is specified, the input is internally set to the highest layer number. • Return value is true when the execution was successful, false if not.	

EEPAuxiliaryTrackChangeAppearance()		EEPAuxiliaryTrackChangeAppearance(AuxTrackID, LayerNumber)
Parameters	two	
Returns	one	`ok = EEPAuxiliaryTrackChangeAppearance(87, 3)`
Requires	EEP 18.1 - Plug-in 1	
Purpose	Changes the layer of a multi-layer auxiliary track and thus the appearance of the track surface.	
Notes	• First parameter is the ID of the auxiliary track category element, whose layer should be changed. • Second parameter is the number of the desired layer. If a number greater than the number of existing layers is specified, the input is internally set to the highest layer number. • Return value is true when the execution was successful, false if not	

EEPRailTrackSetTextureText()	Adds a new text to the writable area of a rail track.	17-3
EEPRailTrackGetTextureText()	Returns the text of a writable area of a rail track.	17-5
EEPRoadTrackSetTextureText()	Adds a new text to the writable area of a road.	17-3
EEPRoadTrackGetTextureText()	Returns the text of a writable area of a road	17-5
EEPTramTackSetTextureText()	Adds a new text to the writable area of a tram track.	17-3
EEPTramTackGetTextureText()	Returns the text of a writable area of a tram track	17-6
EEPAuxiliaryTackSetTextureText()	Adds a new text to the writable area of a auxiliary track.	17-3
EEPAuxiliaryTackGetTextureText()	Returns the text of a writable area of a auxiliary track.	17-6

11 Camera functions

EEPSetCamera()		EEPSetCamera(Type, "Name")
Parameters	two	
Returns	one	`ok = EEPSetCamera(0, "Station")`
Requires	EEP 11.3 Plug-in 3	
Purpose	Choose one of the registered cameras among the list	
Notes	• First parameter is the type of camera : 0 = static 1 = dynamic 2 = moving camera. • Second parameter is the name of the camera as a string. • Return value is true if the camera exists, false if not.	

EEPSetCameraPosition()		EEPSetCameraPosition(Pos_X, Pos_Y, Pos_Z)
Parameters	three	
Returns	one	`ok = EEPSetCameraPosition(3, 4,  5)`
Requires	EEP 16.1 Plug-in 1	
Purpose	Sets the camera position	
Notes	• First parameter is the X position of the camera in meters • Second parameter is the Y position of the camera in meters • Third parameter is the Z position of the camera in meters • Return value is true when the execution was successful, false if not	

EEPGetCameraPosition()		EEPGetCameraPosition()
Parameters	none	
Returns	four	`ok, Pos_X, Pos_Y, Pos_Z = EEPGetCameraPosition()`
Requires	EEP 16.1 Plug-in 1	
Purpose	Returns the current camera position	
Notes	• First return value is true when the execution was successful, false if not • Second return value is the X position of the camera in meters • Third return value is the Y position of the camera in meters • Fourth return value is the Z position of the camera in meters • **Be aware that** once the camera position has been set by using *EEPSetCameraPosition()* you'll have to wait until a new *EEPMain()* cycle has begun to get its new position thanks to the EEPGetCameraPosition() function.	

EEPSetCameraRotation()	EEPSetCameraRotation(Rot_X, Rot_Y, Rot_Z)
Parameters	three
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Sets the camera orientation
Notes	• First parameter is the rotation of the camera around the X-axis in degrees. • Second parameter is the rotation of the camera around the Y-axis in degrees. • Third parameter is the rotation of the camera around the Z-axis in degrees. • Return value is true when the execution was successful, false if not.

```
ok = EEPSetCameraRotation(30, 45, 25)
```

EEPGetCameraRotation()	EEPGetCameraRotation()
Parameters	none
Returns	four
Requires	EEP 16.1 Plug-in 1
Purpose	Returns the current camera orientation
Notes	• First return value is true when the execution was successful, false if not • Second return value is the rotation of the camera around the X-axis in degrees. • Third return value is the rotation of the camera around the Y-axis in degrees. • Fourth return value is the rotation of the camera around the Z-axis in degrees. • **Be aware that** once the camera orientation has been set by using *EEPSetCameraRotation()* you'll have to wait until a new *EEPMain()* cycle has begun to get its new position thanks to the EEPGetCameraRotation() function.

```
ok, Rot_X, Rot_Y, Rot_Z = EEPGetCameraRotation()
```

EEPSetPerspectiveCamera()		EEPSetPerspectiveCamera(Camera_position [, "#Name"])
Parameters	one or two	ok = EEPSetPerspectiveCamera(3, "#Passenger train") - Since EEP 15: ok = EEPSetPerspectiveCamera(6) - Since EEP 16.4 Plug-in 4: ok = EEPSetPerspectiveCamera(10)
Returns	one	
Requires	EEP 11.3 Plug-in 3 EEP 15 EEP 16.4 Plug-in 4	
Purpose	Activates a camera connected to the specified or active "vehicles composite".	
Notes	• First parameter is the position of the camera – corresponds to the keys 1 to 9 for camera selection (since EEP 16.4 Plug-in 4 also to key 0) 1 = Straight left to the "vehicles composite" 2 = Straight right to the "vehicles composite" 3 = Sideways from above on the left side of the "vehicles composite" 4 = Sideways from above on the right side of the "vehicles composite" 5 = At the front of the "vehicles composite" in the direction of travel 6 = Front view on the "vehicles composite" 7 = Activates the nearest automatic camera to the rollingstock. 8 = Cockpit camera 9 = Above the "vehicles composite" in or against the direction of travel respectively if present the tracking camera specified by the designer (outside the driver's cab) respectively if defined the last camera position defined by the user with *EEPRollingstockSetUserCamera()* **Since EEP 16.4 Plug-in 4 :** 10 = view of the “old cabin” (i.e. 0 key) or a different view of the 8 • Second parameter is the name of the "vehicles composite" as a string preceded by #. **Since EEP 15** the second parameter can be omitted. Then the tracking camera is aimed at the active "vehicles composite". • Return value is true when the execution was successful, false if not. • **Attention:** If a driver's cab is to be called up with EEPSetPerspectiveCamera(8) [or (10)] in the case of a "vehicles composite" with more than 1 vehicle, but currently another vehicle in the "vehicles composite" is active (in manual mode), the vehicle with the driver's cab must be activated beforehand with *EEPRollingstockSetActive()*.	

EEPGetPerspectiveCamera()		EEPGetPerspectiveCamera(["#Name"])
Parameters	one or none	ok, Camera_position = EEPGetPerspectiveCamera("#Passenger train") ok, Camera_position = EEPSetPerspectiveCamera()
Returns	Two	
Requires	EEP 18.0	
Purpose	Indicates which perspective camera are selected for the specified or active "vehicles composite".	
Notes	• The optional parameter is the name of the "vehicles composite" (preceded by the # character) as a string. If the name is not specified, the second return value is the perspective camera aligned to the active "vehicles composite". • First return value is true when the execution was successful, false if not. • Second return value is the position of the camera – corresponds to the keys 0 to 9 for camera selection: 1 = Straight left to the "vehicles composite" 2 = Straight right to the "vehicles composite" 3 = Sideways from above on the left side of the "vehicles composite" 4 = Sideways from above on the right side of the "vehicles composite" 5 = At the front of the "vehicles composite" in the direction of travel 6 = Front view on the "vehicles composite" 7 = Activates the nearest automatic camera to the rollingstock. 8 = Cockpit camera 9 = Above the "vehicles composite" in or against the direction of travel respectively if present the tracking camera specified by the designer (outside the driver's cab) respectively if defined the last camera position defined by the user with *EEPRollingstockSetUserCamera()* 10 = view of the “old cabin” (i.e. 0 key) or a different view of the 8	

EEPRollingstockSetUserCamera()	EEPRollingstockSetUserCamera("Fahrzeugname", Pos_X, Pos_Y, Pos_Z, Rot_H, Rot_V[, 1)]
Parameters	Six or seven
Returns	one
Requires	EEP 16.1 Plug-in 1

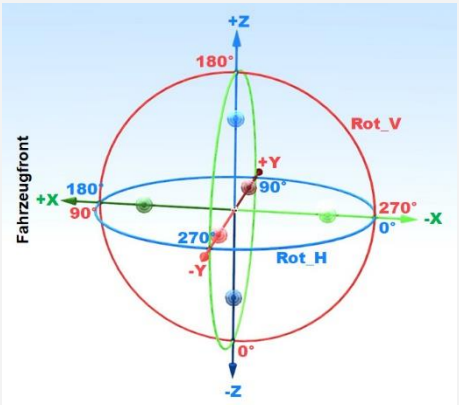
```
ok = EEPRollingstockSetUserCamera("#Pullman", -4, 0, 2, 180, 90, 1) -- immediate execution

ok = EEPRollingstockSetUserCamera("#Pullman", -4, 0, 2, 180, 90)  -- only definition
```

Purpose

Sets the user-defined tracking camera (called with key 9 or by *EEPSetPerspectiveCamera(9, "#Name")* or **since EEP 15** by *EEPSetPerspectiveCamera(9)*)

Notes

- First parameter is the complete name of the vehicle as string.
- Second parameter is the camera position on the X-axis (green) in meters in relation to the zero point of the vehicle (positive forward).
- Third parameter is the camera position on the Y-axis (red) in meters in relation to the zero point of the vehicle (positive to the right).
- Fourth parameter is the camera position on the Z-axis (blue) in meters in relation to the zero point of the vehicle (positive upwards).
- Fifth parameter is the **horizontal** camera orientation (rotation) around the Z-axis (blue circle) in degrees from 0 to 359.9 (view to the rear = 0.0, further counterclockwise).
- Sixth parameter is the **vertical** camera orientation (rotation) around the Y-axis (red circle) in degrees from 0 to 359.9 (looking down = 0.0, further clockwise).



- An **optional** seventh parameter with the value 1 causes the camera setting to be adopted immediately. Without the 7th parameter [Default] only the values are defined. To execute this, the key 9 or the function *EEPSetPerspectiveCamera(9, "#Name")* in the Lua script or, from EEP 15 onwards, only *EEPSetPerspectiveCamera(9)* is required.
- A rotation around the X-axis (green circle) and thus the view of a lying person is not possible for technical reasons.
- Return value is true when the execution was successful, false if not.
- **Important:** If there is **more than 1 vehicle in the "vehicles composite"**, the following must be considered:
  - If the camera is to refer to a vehicle other than the one in front, this vehicle must be activated beforehand. Otherwise, EEP will not respond to the following camera settings. However, it is advisable - in order to always act in the same way - to apply this to the vehicle in front as well.
  - Since the vehicle in front changes when the direction of travel is changed, EEP independently switches to certain predefined camera positions. If this is not desired, it is essential to activate the vehicle responsible for the camera position after the change with EEPSetRollingstockActive() and reset the camera position with EEPSetPerspectiveCamera(9 [, "#Name"]) in the same function [before or after EEPSetTrainSpeed()].It does not matter whether it is the same (as before) or a different, new vehicle. For the same vehicle, it is not necessary to redefine the position with EEPRollingstockSetUserCamera().
  - If a camera movement defined with EEPRollingstockSetUserCamera() is to be followed by a driver's cab ride with EEPSetPerspectiveCamera(8 [, "#Name"]) or EEPSetPerspectiveCamera(10 [, "#Name"]) and the vehicle defined for the user camera is not the foremost vehicle, the foremost vehicle must be activated beforehand with EEPSetRollingstockActive() [see also *EEPSetPerspectiveCamera()*].

EEPRollingstockGetUserCamera()	EEPRollingstockGetUserCamera("Vehicle_name")
Parameters	one
Returns	six
Requires	EEP 17

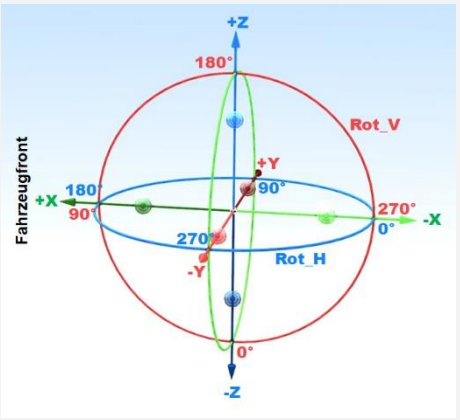
```
ok, Pos_X, Pos_Y, Pos_Z, Rot_H, Rot_V =
EEPRollingstockGetUserCamera("Pullman")
```

Purpose

Determines (if present) the position of the tracking camera (outside the driver's cab) specified by the designer **or preferentially** the tracking camera defined by the user with *EEPRollingstockSetUserCamera()*.

Notes

- The parameter is the complete name of the vehicle as string.
- First return value is true when the execution was successful, false if not.
- Second return value is the camera position on the X-axis (green) in meters in relation to the zero point of the vehicle.
- Third return value is the camera position on the Y-axis (red) in meters in relation to the zero point of the vehicle.
- Fourth return value is the camera position on the Z-axis (blue) in meters in relation to the zero point of the vehicle.
- Fifth return value is the **horizontal** camera orientation (rotation) around the Z-axis (blue circle) in degrees.
- Sixth return value is the **vertical** camera orientation (rotation) around the Y-axis (red circle) in degrees.



- **Attention:** This function does not determine the position of the other tracking cameras available in EEP.

12 Layout functions

EEPLoadProject()		EEPLoadProject("[sub_folder / | \\] File_name")
Parameters	one	ok = EEPLoadProject("My_project.anl3")
Returns	one	ok = EEPLoadProject("Tutorials/Tutorial_49_LUA.anl3")
Requires	EEP 11.3 Plug-in 3	ok = EEPLoadProject("Tutorials\\Tutorial_54_LUA.anl3")
Purpose	Loads a layout from the "Resourcen\Anlagen" folder or the layout folder path entered under *program settings* or their sub-folders.	
Notes	• Parameter is the sub-folder (if necessary) followed by the file name with its .anl3 suffix. The separator between folder and filename is either a slash (/) or a double backslash (\\) • Return value is true if the layout exists, false if not.	

EEPOnSaveAnl()		EEPOnSaveAnl(saving_path)
Parameters	none	function EEPOnSaveAnl(saving_path) print("Layout has been saved to "..saving_path) end
Returns	one	
Requires	EEP 16.1 Plug-in 1	
Purpose	EEP automatically calls this function during saving and returns the path in which the system was saved in the self-selected parameter.	
Notes	• Parameter is the saving path with the name of the file as a string. A self-defined variable in the function brackets takes this value for further use. • EEP does not expect any kind of feedback when using this function. • **Caution:** This function is **not** suitable for specifying what to do before saving. This can be done in the function *EEPOnBeforeSaveAnl()*.	

EEPOnBeforeSaveAnl()		EEPOnBeforeSaveAnl()
Parameters	none	function EEPOnBeforeSaveAnl() EEPSaveData(1, "Lua makes it possible!") end
Returns	none	
Requires	EEP 17	
Purpose	EEP automatically calls this function before each saving. In the function one can define what has to be done before saving the layout.	
Notes	• This function is called by EEP without any parameters. • EEP does not expect any kind of feedback when using this function.	

EEPGetAnlVer()		EEPGetAnlVer()

Parameters	none	`LayoutVersion = EEPGetAnlVer()`
		
		`print("The layout was last saved in EEP ", LayoutVersion, " before opening.")`

Returns	one	
Requires	EEP 17	
Purpose	Returns the EEP version with which the layout was last saved before opening.	
Notes	• This function is called by EEP without any parameters. • Return value is the version number of EEP in which the layout was last saved. It is up-to-date immediately after each saving.	

EEPGetAnlLng()		EEPGetAnlLng()

Parameters	none	`LayoutLanguage = EEPGetAnlLng()`
		
		`if LayoutLanguage == "GER" then`
		`  print("The layout language for axes is German.")`
		`elseif LayoutLanguage == "ENG" then`
		`  print("The layout language for axes is English.")`
		`elseif LayoutLanguage == "FRA" then`
		`  print("The layout language for axes is French.")`
		`end`

Returns	one	
Requires	EEP 17	
Purpose	Returns the "layout language" related to axis names. This is assigned by EEP with the first insertion of a model with axes and does not change thereafter. As long as no model with axes has been inserted, this "layout language" corresponds to the language of the current EEP version, i.e. the EEP-specific variable *EEPLng*.	
Notes	• This function is called by EEP without any parameters. • The return value is the abbreviation for the "layout language" assigned in relation to axis names (for definition see above under Purpose): GER = German version, ENG = English version, FRA = French version.	

EEPGetAnlName()		EEPGetAnlName()

Parameter	none	`Name = EEPGetAnlName()`
		
		`print(Welcome to the ", Name, "layout")`

Rückgabewerte	one	
Voraussetzung	EEP 17.1 - Plug-in 1	
Zweck	Returns the name with which the layout was last saved.	
Bemerkungen	• This function is called by EEP without any parameters. • Return value is the name of the layout as string.	

EEPGetAnlPath()		EEPGetAnlPath()

Parameters	none	`Path = EEPGetAnlPath()`
		
		`dofile(Path.."\\Filename.lua")`

Returns	one	
Requires	EEP 18.1 - Plug-in 1	
Purpose	Returns the storage path of the layout file (.anl3). This allows, for example, Lua files stored in the layout folder to be integrated into the layout Lua file..	
Notes	• The function call by EEP is performed without parameters. • Return value is the path of the layout file (without its name) as a string.	

13 Virtual depot functions

EEPGetTrainFromTrainyard()		EEPGetTrainFromTrainyard(Depot, "#Name", Position [, Direction])
Parameters	three or four	ok = EEPGetTrainFromTrainyard(1, "#Rheingold", 0) ok = EEPGetTrainFromTrainyard(2, "", 4, 1)
Returns	one	
Requires	EEP 11.3 Plug-in 2 EEP 15	
Purpose	Sends a specified "vehicles composite" from the specified virtual depot.	
Notes	• First parameter is the ID of the virtual depot. You can find the ID in the header of the depot's properties window. • Second parameter is the name of the "vehicles composite" as a string preceded by #. If you enter an empty string as a name, the "vehicles composite" will be determined by the third argument. • Third parameter is the position of the "vehicles composite" in the depot list. This parameter only applies if no name is specified in the second argument. If a name is given, this number is arbitrary, but **still required**. Enter 0 if you use the name of the "vehicles composite". • **Since EEP 15** a fourth **optional** parameter defines the direction of travel of the "vehicles composite": 0 (or without 4th argument) = Direction of traffic as specified by the depot. 1 = Forward motion. 2 = Backwards motion 3 = Travel in the opposite direction to that specified by the depot. • Return value is true if the targeted depot and "vehicles composite" exist, regardless if the "vehicles composite" is currently in the depot and available or not! It is false if either the depot doesn't exist or the specified "vehicles composite" is not listed in this depot.	

EEPOnTrainExitTrainyard()		EEPOnTrainExitTrainyard(Depot, Name)
Parameters	two	function EEPOnTrainExitTrainyard(Depot, Name) print(Name.." has left depot "..Depot) end
Returns	none	
Requires	EEP 14.1 Plug-in 1	
Purpose	Called if a "vehicles composite" leaves a virtual depot.	

Notes	• Two self-defined variables in the function brackets take the following values for further use:
	  ○ In the first parameter EEP transfers the ID of the virtual depot as a number from which something has left.
	  ○ In the second parameter EEP transfers the name of the "vehicles composite" that has left.
	• EEP requires no return value when calling this function.	

EEPOnTrainEnterTrainyard()	EEPOnTrainEnterTrainyard(DepotID, Name)
Parameters	two
Returns	none
Requires	EEP 18.1 - Plug-in 1

```
function EEPOnTrainEnterTrainyard(DepotID, Name)
  print(Name.." has entered depot "..DepotID)
end
```

Purpose	Called whenever a "vehicles composite" enters a virtual depot.

Notes	• Two self-defined variables in the function brackets take the following values for further use :
	  ○ In the first parameter, EEP transfers the ID of the virtual depot as a number into which something has driven.
	  ○ In the second parameter, EEP transmits the name of the " vehicles composite" that has entered.
	• EEP does not expect a return value when this function is called..

EEPGetTrainyardItemsCount()	EEPGetTrainyardItemsCount(Depot)
Parameters	one
Returns	one
Requires	EEP 13.2 Plug-in 2

```
Number = EEPGetTrainyardItemsCount(2)
```

Purpose	Returns the number of "vehicles composite" managed in the virtual depot
Notes	• Parameter is the ID of the virtual depot. You can find the ID in the header of the depot's properties window. • Return value is the number of "vehicles composite" listed in the depot. **Be aware that** this value does not correspond to the number of waiting (present) "vehicles composite". This must be determined with the help of the function *EEPGetTrainyardItemStatus()*.

EEPGetTrainyardItemName()	EEPGetTrainyardItemName(Depot, Position)
Parameters	two
Returns	one
Requires	EEP 13.2 Plug-in 2

```
Name = EEPGetTrainyardItemName(1, 2)
```

Purpose	Returns the name of a "vehicles composite" in the virtual depot
Notes	• First parameter is the ID of the virtual depot. You can find the ID in the header of the depot's properties window. • Second parameter is its position on the depot list. • Return value is the name of the "vehicles composite" at the specified position.

EEPGetTrainyardItemStatus()		EEPGetTrainyardItemStatus(Depot, "#Lua_Name", Position)

Parameters	three	`Status = EEPGetTrainyardItemStatus(1, "#Freight_train", 0)`
		
		`Status = EEPGetTrainyardItemStatus(1, "", 3)`
		
		`if EEPGetTrainyardItemStatus(3, "#Rheingold", 0) == 1 then`
		`  print("The train is waiting in the depot")`
		`else`
		`  print("The train has already left the depot")`
		`endif`

Returns	one	
Requires	EEP 13.2 Plug-in 2	
Purpose	Returns the status of a "vehicles composite" in a specified virtual depot	
Notes	• First parameter is the ID of the virtual depot. You can find the ID in the header of the depot's properties window. • Second parameter is the name of the "vehicles composite" as a string preceded by #. If you enter an empty string as a name, the "vehicles composite" will be determined by the third argument. • Third parameter is the position of the "vehicles composite" in the depot list. This parameter only applies if no name is specified in the second argument. If a name is given, this number is arbitrary, but **still required**. Enter 0 if you use the name of the "vehicles composite". • Return the status of the specified "vehicles composite": 0 = running (i.e. not present at the depot, only reported) 1 = waiting (present in the depot.)	

EEPIsTrainInTrainyard()		EEPIsTrainInTrainyard(Name)
Parameters	one	`ok, DepotID = EEPIsTrainInTrainyard("#Rheingold")`
Returns	two	
Requires	EEP 18.1 - Plug-in 1	
Purpose	Returns the number of the virtual depot where the "vehicles composite" is located, or the number zero if it is located in the layout.	
Notes	• Parameter is the name of the the "vehicles composite" whose location is to be determined. • First return value is `true`, if the "vehicles composite" exists, otherwise `false` if it does not. • Second return value is the number of the virtual depot in which the '"vehicles composite" is located, or 0 if it is located in the layout. A return value of -1 means that the "vehicles composite" exists in the layout but is not registered in any depot..	

EEPPutTrainToTrainyard()	EEPPutTrainToTrainyard (DepotID, Name)
Parameters	two
Returns	one
Requires	EEP 18.1 - Plug-in 1

```
ok = EEPPutTrainToTrainyard(3, "#Rheingold")

ok = EEPPutTrainToTrainyard(3, "")
```

Purpose	Moves a "vehicles composite" into a virtual depot
Notes	• First parameter is the ID of the virtual depot. You can find the ID in the header of the depot's properties window. • Second parameter is the name of the "vehicles composite", which is to be moved to the selected depot. (**Important:** The "vehicles composite" must be registered there beforehand.) If you enter an empty string (""), all "vehicles composites" registered there will be moved back to the specified virtual depot. • Return value is true if the execution was successful, otherwise false.

14 Tip text functions

EEPChangeInfoStructure()	EEPChangeInfoStructure("#Lua_Name", "Text")
Parameters	two
Returns	one
Requires	EEP 13
Purpose	Assigns a new tip text to a structure or landscape element
Notes	• First parameter is the **Lua name** of a structure or a landscape element as a string. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter is the new text. Use \n for line feeds. • Return value is true if the structure or the landscape element exists, false if not.

```
ok = EEPChangeInfoStructure("#23", "Switching station")
```

EEPShowInfoStructure()	EEPShowInfoStructure("#Lua_Name",true|false)
Parameters	two
Returns	one
Requires	EEP 13
Purpose	Turns the tip text of the specified structure or landscape element on or off.
Notes	• First parameter is the **Lua name** of a structure or a landscape element as a string. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter switches the tip text on with true or off with false. • Return value is true if the structure or the landscape element exists, false if not.

```
ok = EEPShowInfoStructure("#23", true)
```

EEPChangeInfoSignal()	EEPChangeInfoSignal(ID, "Text")
Parameters	two
Returns	one
Requires	EEP 13
Purpose	Assigns a new tip text to a signal
Notes	• First parameter is the ID of the signal. • Second parameter is the new text. Use \n for line feeds. • Return value is true if the signal exists, false if not.

```
EEPChangeInfoSignal(74, "Exit signal track 2")
```

EEPShowInfoSignal()	EEPShowInfoSignal(ID, true|false)
Parameters	two
Returns	one
Requires	EEP 13
Purpose	Turns the tip text of the specified signal on or off
Notes	• First parameter is the ID of the signal. • Second parameter switches the tip text on with true or off with false. • Return value is true if the signal exists, false if not.

```
ok = EEPShowInfoSignal(74, true)
```

EEPChangeInfoSwitch()	EEPChangeInfoSwitch(ID, "Text")
Parameters	two
Returns	one
Requires	EEP 13

```
Text = "Junction to \nManchester"
ok = EEPChangeInfoSwitch(63, Text)
```

Purpose	Assigns a new tip text to a specified switch
Notes	• First parameter is the ID of the switch. • Second parameter is the new text. Use \n for line feeds. • Return value is true if the switch exists, false if not.

EEPShowInfoSwitch()	EEPShowInfoSwitch(ID, true|false)
Parameters	two
Returns	one
Requires	EEP 13

```
ok = EEPShowInfoSwitch(63, true)
```

Purpose	Turns the tip text of the specified switch on or off
Notes	• First parameter is the ID of the switch. • Second parameter switches the tip text on with true or off with false. • Return value is true if the switch exists, false if not.

15 Info text functions

EEPShowInfoTextTop()	EEPShowInfoTextTop(r, g, b, sz, t, j, "Text")

Parameters seven
Returns one
Requires EEP 13 Plug-in 1
	`r = 1 -- red`
	`g = 1 -- green`
	`b = 1 -- blue`
	`sz = 1 -- size`
	`t = 10 -- time`
	`j = 1 -- alignment`
	`Text = "White on top and centered 10 seconds"`
	`ok = EEPShowInfoTextTop(r, g, b, sz, t, j, Text)`

Purpose	Displays the specified text at the top of the 3D window.
Notes	• The first three parameters set the color from the proportions for red, green and blue. • The forth parameter sets the size. (from x0,5 to x2) • The fifth parameter sets the time the text is displayed for in seconds (minimum is 5 seconds). • The sixth parameter justifies the text: 0 = justified 1 = centered 2 = left aligned 3 = right aligned • The seventh parameter is the text to be displayed as a string Use \n for line feeds. • Return value is true when the execution was successful, false if not.

EEPShowInfoTextBottom()	EEPShowInfoTextBottom(r, g, b, sz, t, j, "Text")

Parameters seven
Returns one
Requires EEP 13 Plug-in 1
	`r = 1 -- red`
	`g = 1 -- green`
	`b = 0 -- blue`
	`sz = 0.75 -- size`
	`t = 15 -- time`
	`j = 2 -- alignment`
	`Text = "Yellow on bottom left aligned 15 seconds"`
	`ok = EEPShowInfoTextBottom(r, g, b, sz, t, j, Text)`

Purpose	Displays the specified text at the bottom of the 3D window.
Notes	• The first three parameters set the color from the proportions for red, green and blue. • The forth parameter sets the size. (from x0,5 to x2) • The fifth parameter sets the time the text is displayed for in seconds (minimum is 5 seconds). • The sixth parameter justifies the text: 0 = justified 1 = centered 2 = left aligned 3 = right aligned • The seventh parameter is the text to be displayed as a string. Use \n for line feeds. • Return value is true when the execution was successful, false if not.

EEPShowScrollInfoTextTop()		EEPShowScrollInfoTextTop(r, g, b, sz, t, j, sp, "Text")

Parameters	eight	`r = 0 -- red`
		`g = 1 -- green`
		`b = 0.7 -- blue`
		`sz = 1.2 -- size`
		`t = 20 -- time`
		`j = 0 -- alignment`
		`sp = 0.2 -- speed`
		`Text = "Turkish blue on top scrolling text for 20 seconds"`
		`ok = EEPShowScrollInfoTextTop(r, g, b, sz, t, j, sp, Text)`

Returns	one	
Requires	EEP 13 Plug-in 1	
Purpose	Displays the specified scrolling text at the top of the 3D window.	
Notes	• The first three parameters set the color from the proportions for red, green and blue. • The forth parameter sets the size. (from x0,5 to x2) • The fifth parameter sets the time the text is displayed for in seconds (minimum is 5 seconds). • The sixth parameter (to justify the text) is not relevant in this context, but is **still required**. Please enter a **number** (0 is fine). • The seventh parameter sets scrolling speed. • The eighth parameter is the text to be displayed as a string. Use \n for line feeds. • Return value is true when the execution was successful, false if not.	

EEPShowScrollInfoTextBottom()		EEPShowScrollInfoTextBottom(r, g, b, sz, t, j, sp, "Text")

Parameters	eight	`r = 1 -- red`
		`g = 0 -- green`
		`b = 1 -- blue`
		`sz = 0.6 -- size`
		`t = 10 -- time`
		`j = 0 -- alignment`
		`sp = 0.1 -- speed`
		`Text = "Magenta on bottom scrolling text for 10 seconds"`
		`ok = EEPShowScrollInfoTextBottom(r, g, b, sz, t, j, sp, Text)`

Returns	one	
Requires	EEP 13 Plug-in 1	
Purpose	Displays the specified scrolling text at the bottom of the 3D window.	
Notes	• The first three parameters set the color from the proportions for red, green and blue. • The forth parameter sets the size. (from x0,5 to x2) • The fifth parameter sets the time the text is displayed for in seconds (minimum is 5 seconds). • The sixth parameter (to justify the text) is not relevant in this context, but is **still required**. Please enter a **number** (0 is fine). • The seventh parameter sets scrolling speed. • The eighth parameter is the text to be displayed as a string Use \n for line feeds. • Return value is true when the execution was successful, false if not.	

EEPHideInfoTextTop()		EEPHideInfoTextTop()
Parameters	none	
Returns	one	`ok = EEPHideInfoTextTop()`
Requires	EEP 13 Plug-in 1	
Purpose	Hides the info text at the top of the 3D window.	
Notes	• This function requires no arguments. • Return value is true when the execution was successful, false if not.	

EEPHideInfoTextBottom()		EEPHideInfoTextBottom()
Parameters	none	
Returns	one	`ok = EEPHideInfoTextBottom()`
Requires	EEP 13 Plug-in 1	
Purpose	Hides the info text at the bottom of the 3D window.	
Notes	• This function requires no arguments. • Return value is true when the execution was successful, false if not.	

16 Sound functions

EEPPlaySound()	EEPPlaySound("[Path / | \\] File_name")
Parameters	one
Returns	one
Requires	EEP 13 Plug-in 1
Purpose	Plays a sound unrelated to any location.
Notes	• Parameter is the path (relative to the Resources/Sounds folder) and the file name of a suitable mono-wav file as a string. The separator between folder and filename is either a **slash** (/) or a **double backslash** (\\) • Return value is true when the execution was successful, false if not.

```
ok = EEPPlaySound("User/Bimmel.wav")

ok = EEPPlaySound("EEXP\\Routel.wav")
```

EEPStructurePlaySound()	EEPStructurePlaySound("#Name Lua", true|false)
Parameters	two
Returns	one
Requires	EEP 13 Plug-in 1
Purpose	Enables or disables the sound of a sound model from the category landscape elements/sound
Notes	• First parameter is the **Lua name** of the sound model as a string. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter switches the sound on with true or off with false. • Return value is true when the execution was successful, false if not. • If this sound is to be managed exclusively by Lua, then it is recommended to set the activation distance in the model to 0.

```
ok = EEPStructurePlaySound("#33", true)
```

17 Texture text functions

EEPStructureSetTextureText()	EEPStructureSetTextureText("#Lua_Name", Area_number, "Text")
Parameters	three
Returns	one
Requires	EEP 15

```
ok = EEPStructureSetTextureText("#1", 1, "Neustadt")
```

Purpose	Adds a new text to the writable area of a landscape or structure element.
Notes	• First parameter is the **Lua name** of the structure or landscape element. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.

EEPRollingstockSetTextureText()	EEPRollingstockSetTextureText("Name", Area_number, "Text")
Parameters	three
Returns	one
Requires	EEP 15

```
ok = EEPRollingstockSetTextureText("BR481", 1, "Not in service")
```

Purpose	Adds a new text to the writable area of a rollingstock.
Notes	• First parameter is the name of the rollingstock as a string. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.

EEPRollingstockGetTextureText()	EEPRollingstockGetTextureText("Name", Area_number)
Parameters	two
Returns	two
Requires	EEP 16.3 Plug-in 3

```
ok, Text = EEPRollingstockGetTextureText("BR481", 1)
```

Purpose	Returns the text being displayed on the writable area of a rollingstock
Notes	• First parameter is the name of the rollingstock as a string. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.

EEPSignalSetTextureText()	EEPSignalSetTextureText(ID, Area_number, "Text")
Parameters	three
Returns	one
Requires	EEP 15
Purpose	Adds a new text to the writable area of a signal.
Notes	• First parameter is the ID of the signal. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.

```
ok = EEPSignalSetTextureText(88, 1, "Fire exit")
```

EEPGoodsSetTextureText()	EEPGoodsSetTextureText("#Lua_Name", Area_number, "Text")
Parameters	three
Returns	one
Requires	EEP 15
Purpose	Adds a new text to the writable area of a cargo.
Notes	• First parameter is the **Lua name** of the cargo as a string. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.

```
ok = EEPGoodsSetTextureText("#137", 2, "Lua Logistics")
```

EEPRailTrackSetTextureText()	EEPRailTrackSetTextureText(ID, Area_number, "Text")
Parameters	three
Returns	one
Requires	EEP 15
Purpose	Adds a new text to the writable area of a rail track.
Notes	• First parameter is the ID of the rail section. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.

```
ok = EEPRailTrackSetTextureText(67, 1, "Rail")
```

EEPRoadTrackSetTextureText()		EEPRoadTrackSetTextureText(ID, Area_number, "Text")
Parameters	three	ok = EEPRoadTrackSetTextureText(128, 1, "Road")
Returns	one	
Requires	EEP 15	
Purpose	Adds a new text to the writable area of a road.	
Notes	• First parameter is the ID of the road section. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.	

EEPTramTrackSetTextureText()		EEPTramTrackSetTextureText(ID, Area_number, "Text")
Parameters	three	ok = EEPTramTrackSetTextureText(213, 1, "Tram track")
Returns	one	
Requires	EEP 15	
Purpose	Adds a new text to the writable area of a tram track.	
Notes	• First parameter is the ID of the tram track. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.	

EEPAuxiliaryTrackSetTextureText()		EEPAuxiliaryTrackSetTextureText(ID, Area_number, "Text")
Parameters	three	ok = EEPAuxiliaryTrackSetTextureText(197, 1, "Construction site fencing")
Returns	one	
Requires	EEP 15	
Purpose	Adds a new text to the writable area of an auxiliary track.	
Notes	• First parameter is the ID of the auxiliary track category element. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • Third parameter is the text to be displayed. Use \n for line feeds. • Return value is true when the execution was successful, false if not.	

EEPStructureGetTextureText()		EEPStructureGetTextureText("#Lua-Name", Area_number)
Parameters	two	ok, Text = EEPStructureGetTextureText("#147", 1)
Returns	two	
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the text being displayed on the writable area of a landscape or structure element.	
Notes	• First parameter is the **Lua name** of the structure or landscape element. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter is the number of the area that is supposed to display the text. some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.	

EEPSignalGetTextureText()		EEPSignalGetTextureText(Signal-ID, Area_number)
Parameters	two	ok, Text = EEPSignalGetTextureText(88, 1)
Returns	two	
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the text being displayed on the writable area of a signal.	
Notes	• First parameter is the ID of the signal. • Second parameter is the number of the area that is supposed to display the text. some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.	

EEPGoodsGetTextureText()		EEPGoodsGetTextureText("#Lua-Name", Area_number)
Parameters	two	
Returns	two	`ok, Text = EEPGoodsGetTextureText("#137", 1)`
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the text being displayed on the writable area of cargo.	
Notes	• First parameter is the **Lua name** of the cargo as a string. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter is the number of the area that is supposed to display the text. some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.	

EEPRailTrackGetTextureText()		EEPRailTrackGetTextureText(Gleis-ID, Area_number)
Parameters	two	
Returns	two	`ok, Text = EEPRailTrackGetTextureText(67, 1)`
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the text being displayed on the writable area of rail track.	
Notes	• First parameter is the ID of the rail section. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.	

EEPRoadTrackGetTextureText()		EEPRoadTrackGetTextureText(Strassen-ID, Area_number)
Parameters	two	
Returns	two	`ok, Text = EEPRoadTrackGetTextureText(128, 1)`
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the text being displayed on the writable area of road.	
Notes	• First parameter is the ID of the road section • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.	

EEPTramTrackGetTextureText()		EEPTramTrackGetTextureText(Strassenbahngleis-ID, Area_number)
Parameters	two	
Returns	two	`ok, Text = EEPTramTrackGetTextureText(213, 1)`
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the text being displayed on the writable area of a tram track.	
Notes	• First parameter is the ID of the tram track. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.	

EEPAuxiliaryTrackGetTextureText()		EEPAuxiliaryTrackGetTextureText(Spline-ID, Area_number)
Parameters	two	
Returns	two	`ok, Text = EEPAuxiliaryTrackGetTextureText(197, 1)`
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the text being displayed on the writable area of an auxiliary track.	
Notes	• First parameter is the ID of the auxiliary track category element. • Second parameter is the number of the area that is supposed to display the text. Some models have more than just one writable area. • First return value is true when the execution was successful, false if not. • Second return value is the text being displayed on the specified area.	

18 Tag text functions

EEPStructureSetTagText()		EEPStructureSetTagText("#Lua_Name", "Text")
Parameters	two	ok = EEPStructureSetTagText("#112", "occupied: 2,3,5,8")
Returns	one	
Requires	EEP 15	
Purpose	Changes the tag-text of a structure or a landscape element. Each structure can carry an arbitrary string of maximal 1024 characters. These strings are saved and loaded with the layout.	
Notes	• First parameter is the **Lua name** of the structure or landscape element. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter is the desired text to be saved. • Return value is true when the execution was successful, false if not. **Note:** As of EEP 18, a tag text can also be assigned to a structure or landscape element via its object properties or this can be changed.	

EEPStructureGetTagText()		EEPStructureGetTagText("#Lua_Name")
Parameters	one	ok, Text = EEPStructureGetTagText("#112")
Returns	two	
Requires	EEP 15	
Purpose	Returns the tag text of a property or a landscape element. By means of tag texts, real estate and landscape elements can be used as permanent storage for relevant information.	
Notes	• First parameter is the **Lua name** of the structure or landscape element. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • First return value is true when the execution was successful, false if not. • Second return value is the tag text that was assigned to the structure. • **Be aware that:** The EEPStructureGetTagText() function will only return the new tag text of a structure set by using *EEPStructureSetTagText()* once a new *EEPMain()* cycle has begun. **Note:** As of EEP 18, a tag text entered for a structure or landscape element can also be read out via its object properties.	

EEPRollingstockSetTagText()	EEPRollingstockSetTagText("Name","Text")
Parameters	two
Returns	one
Requires	EEP 15

```
ok = EEPRollingstockSetTagText("DB Zcs-Eva", "Tank car")
```

Purpose	Changes the tag-text of a rollingstock. Each rollingstock can carry an arbitrary string of maximal 1024 characters. These strings are saved and loaded with the layout. As these strings are individually assigned to each rollingstock, they won't get lost on coupling or uncoupling trains, in contrast to routes.
Notes	• First parameter is the name of the rollingstock as a string. • Second parameter is the desired text to be saved. • Return value is true when the execution was successful, false if not. **Note:** As of EEP 18, a tag text can also be assigned to a rollingstock via its object properties or this can be changed.

EEPRollingstockGetTagText()	EEPRollingstockGetTagText("Name")
Parameters	one
Returns	two
Requires	EEP 15

```
ok, Text = EEPRollingstockGetTagText("DB Zcs-Eva")
```

Purpose	Returns the tag text of a rollingstock. By means of tag texts, real estate and landscape elements can be used as permanent storage for relevant information. For example, you can save wagon types or destinations there.
Notes	• First parameter is the name of the rollingstock as a string. • First return value is true when the execution was successful, false if not. • Second return value is the tag text that was assigned to the rollingstock. • **Be aware that:** The EEPRollingstockGetTagText() function will only return the new tag text of a rollingstock set by using *EEPRollingstockSetTagText()* once a new *EEPMain()* cycle has begun. **Note:** As of EEP 18, a tag text entered for a rollingstock can also be read out via its object properties.

EEPSignalSetTagText()	EEPSignalSetTagText("Name","Text")
Parameters	two
Returns	one
Requires	EEP 17.1 - Plug-in 1

```
ok = EEPSignalSetTagText(87, "Route_C")
```

Purpose	Changes the tag-text of a signal. Each signal can carry an arbitrary string of maximal 1024 characters. These strings are saved and loaded with the layout. As these strings are individually assigned to each signal, they won't get lost.
Notes	• First parameter is the signal's ID. • Second parameter is the desired text to be saved. • Return value is true when the execution was successful, false if not. **Note:** As of EEP 18, a tag text entered for a signal can also be read out via its object properties

EEPSignalGetTagText()		EEPSignalGetTagText("Name")
Parameters	one	ok, Text = EEPSignalGetTagText(87)
Returns	two	
Requires	EEP 17.1 - Plug-in 1	
Purpose	Returns the tag text of a signal. By means of tag texts, e.g. information on preset routes or level crossings can be stored directly in the signals instead of in data slots.	
Notes	• The parameter is the signal's ID. • First return value is true when the execution was successful, false if not. • Second return value is the tag text that was assigned to the signal. • **Be aware that:** The EEPSignalGetTagText() function will only return the new tag text of a signal set by using *EEPSignalSetTagText()* once a new *EEPMain()* cycle has begun. **Note:** As of EEP 18, a tag text entered for a signal can also be read out via its object properties.	

EEPGoodsSetTagText()		EEPGoodsSetTagText("#Lua_Name", "Text")
Parameters	two	ok = EEPGoodsSetTagText("#44", "DB Schenker")
Returns	one	
Requires	EEP 18.0	
Purpose	Changes the tag-text of a cargo. Each cargo can carry an arbitrary string of maximal 1024 characters. These strings are saved and loaded with the layout.	
Notes	• First parameter is the **Lua name** of the cargo. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • Second parameter is the desired text to be saved. • Return value is true when the execution was successful, false if not. **Note:** You can also enter a tag text via the object properties of the good.	

EEPGoodsGetTagText()		EEPGoodsGetTagText("#Lua_Name")
Parameters	one	ok, Text = EEPGoodsGetTagText("#44")
Returns	two	
Requires	EEP 18.0	
Purpose	Returns the tag text of a cargo. By means of tag texts, cargos can be used as permanent storage for relevant information.	
Notes	• First parameter is the **Lua name** of the cargo. The difference between the Lua name and the model name is the presence of the ID in the header. The ID preceded by # is already sufficient for the argument. • First return value is true when the execution was successful, false if not. • Second return value is the tag text that was assigned to the cargo. • **Be aware that:** The EEPGoodsGetTagText() function will only return the new tag text of a cargo set by using *EEPGoodsSetTagText()* once a new *EEPMain()* cycle has begun. **Note:** You can also read an entered tag text via the object properties of the good.	

EEPSwitchSetTagText()		**EEPSwitchSetTagText(SwitchID,"Text")**
Parameters	two	
Returns	one	`ok = EEPSwitchSetTagText(87, "Turnout occupied")`
Requires	EEP 18.1 - Plug-in 1	
Purpose	Changes the tag text for a switch point. Each switch can carry its own string of up to 1024 characters in length. These strings are saved and loaded with the layout. Since the texts are individually assigned to each switch, they are not lost.	
Notes	• First parameter is the ID of the switch point. • Second parameter is the desired text. • Return value is true if the execution was successful, otherwise false.	

EEPSwitchGetTagText()		**EEPSwitchGetTagText(WeichenID)**
Parameters	one	
Returns	two	`ok, Text = EEPSwitchGetTagText(87)`
Requires	EEP 18.1 - Plug-in 1	
Purpose	Reads the tag text of a switchpoint . Tag texts can also be used to store relevant information permanently on switches.	
Notes	• Parameter is the ID of the switch point. • First return value is true if the execution was successful, otherwise false. • Second return value is the tag text that was assigned to the switch. If the first return value is false, the second return value is nil. After *EEPSwitchSetTagText()*, *EEPSwitchGetTagText()* returns the new, changed tag text in the same cycle of *EEPMain()*.	

19 Control panel functions

EEPRollingstockSetActive()	EEPRollingstockSetActive("Name")
Parameters	one
Returns	one
Requires	EEP 15.1 Plug-in 1
Purpose	Selects the specified rollingstock and opens the control panel in manual mode.
Notes	• The parameter is the name of the rollingstock as a string. • Return value is true when the execution was successful false if not

```
ok = EEPRollingstockSetActive("BR 212 376-8")
```

EEPRollingstockGetActive()	EEPRollingstockGetActive()
Parameters	none
Returns	one
Requires	EEP 15.1 Plug in 1
Purpose	Returns the name of the rollingstock that is currently selected in the control panel
Notes	• Return value is the name of the rollingstock that is currently selected in the control panel. An empty string will be returned if the control panel is in automatic mode while using this function.

```
Name = EEPRollingstockGetActive()
```

EEPSetTrainActive()	EEPSetTrainActive("#Name")
Parameters	none
Returns	one
Requires	EEP 15.1 Plug-in 1
Purpose	Selects the specified "vehicles composite" and opens the control panel in automatic mode.
Notes	• The parameter is the name of the "vehicles composite" as a string preceded by #. • Return value is true when the execution was successful false if not

```
ok = EEPSetTrainActive("#freight_train")
```

EEPGetTrainActive()	EEPGetTrainActive()
Parameters	none
Returns	one
Requires	EEP 15.1 Plug-in 1
Purpose	Returns the name of the "vehicles composite" that is currently selected in the control panel
Notes	• Return value is the name of the "vehicles composite" that is currently selected in the control panel. If the control panel is in manual mode while using this function, it will return the name of the currently selected rollingstock.

```
Name = EEPGetTrainActive()
```

20 Environment functions

EEPSetCloudIntensity()	EEPSetCloudsIntensity(cloud_intensity)
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Switches to a "normal" cloud image and sets the cloud intensity between 10 % and 100 % (corresponding to the settings under "Environment settings").
Notes	• Parameter is the clouds intensity in percent. If the value is below 10 % then the proportion of clouds will be reduced to 0. The new value is set immediately, but the cloud image changes gradually. • Return value is true when the execution was successful false if not

```
ok = EEPSetCloudsIntensity(75)
```

EEPSetDarkCloudIntensity()	EEPSetDarkCloudsIntensity(cloud_intensity)
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Switches to a dark, grey cloud image and sets the cloud intensity between 10 % and 100 % (corresponding to the settings under "Environment settings")
Notes	• Parameter is the cloud intensity in percent. If the value is below 10 % then the proportion of clouds will be reduced to 0. The new value is set immediately, but the cloud image changes gradually. **Note:** A value of 10 is already sufficient to produce a completely grey sky in a short time (as with a change under "Environment settings"). • Return value is true when the execution was successful false if not

```
ok = EEPSetDarkCloudsIntensity(10)
```

EEPGetCloudIntensity()	EEPGetCloudsIntensity()
Parameters	none
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Returns cloud intensity
Notes	• Return value is the cloud intensity in percent regardless of how the cloud percentage was previously set, whether under "Setting the Environment" or with the Lua functions *EEPSetCloudsIntensity()* or *EEPSetDarkCloudsIntensity()*. • **Be aware that** once the clouds intensity has been set by using *EEPSetCloudsIntensity()* or *EEPSetDarkCloudsIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetCloudsIntensity() function..

```
Clouds_intensity = EEPGetCloudsIntensity()
```

EEPSetWindIntensity()	EEPSetWindIntensity(wind_intensity)
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Sets wind intensity between 10 % and 100 % (corresponding to the settings under "Environment settings")
Notes	• Parameter is wind intensity in percent. If the value is below 10 % then the wind intensity will be reduced to 0. This setting will immediately be applied. • Return value is true when the execution was successful false if not

```
ok = EEPSetWindIntensity(60)
```

EEPGetWindIntensity()	EEPGetWindIntensity()
Parameters	none
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Returns wind intensity
Notes	• Return value is wind intensity in percent. • **Be aware that** once the wind intensity has been set by using *EEPSetWindIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetWindIntensity() function.

```
Wind_intensity = EEPGetWindIntensity()
```

EEPSetRainIntensity()	EEPSetRainIntensity(rain_intensity)
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Sets rain intensity between 10 % and 100 % (corresponding to the settings under "Environment settings")
Notes	• Parameter is wind intensity in percent. If the value is below 10 % then the wind intensity will be reduced to 0. The changeover to the new intensity is smoothly done within 20 seconds. • Return value is true when the execution was successful false if not

```
ok = EEPSetRainIntensity(50)
```

EEPGetRainIntensity()	EEPGetRainIntensity()
Parameters	none
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Returns rain intensity
Notes	• Return value is rain intensity in percent. During the transition, values below 10% may be returned. • **Be aware that** once the rain intensity has been set by using *EEPSetRainIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetRainIntensity() function.

```
Rain_intensity = EEPGetRainIntensity()
```

EEPSetSnowIntensity()	EEPSetSnowIntensity(snow_intensity)
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Sets snow intensity between 10 % and 100 % (corresponding to the settings under "Environment settings")
Notes	• Parameter is snow intensity in percent. If the value is below 10 % then the snow intensity will be reduced to 0. The changeover to the new intensity is smoothly done within 20 seconds. • Return value is true when the execution was successful false if not

```
ok = EEPGetSnowIntensity(50)
```

EEPGetSnowIntensity()	EEPGetSnowIntensity()
Parameters	none
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Returns snow intensity
Notes	• Return value is snow intensity in percent. During the transition, values below 10% may be returned. • **Be aware that** once the snow intensity has been set by using *EEPSetSnowIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetSnowIntensity() function.

```
Snow_intensity = EEPGetSnowIntensity()
```

EEPSetHailIntensity()	EEPSetHailIntensity(hail_intensity)
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Sets hail intensity between 10 % and 100 % (corresponding to the settings under "Environment settings")
Notes	• Parameter is hail intensity in percent. If the value is below 10 % then the hail intensity will be reduced to 0. The changeover to the new intensity is smoothly done within 20 seconds. • Return value is true when the execution was successful false if not

```
ok = EEPSetHailIntensity(55)
```

EEPGetHailIntensity()	EEPGetHailIntensity()
Parameters	none
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Returns hail intensity
Notes	• Return value is hail intensity in percent. During the transition, values below 10% may be returned. • **Be aware that** once the hail intensity has been set by using *EEPSetHailIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetHailIntensity() function.

```
Hail_intensity = EEPGetHailIntensity()
```

EEPSetFogIntensity()	EEPSetFogIntensity(fog_intensity)
Parameters	one
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Sets fog intensity between 10 % and 100 % (corresponding to the settings under "Environment settings")
Notes	• Parameter is fog intensity in percent. If the value is below 10 % then the fog intensity will be reduced to 0. This setting will immediately be applied. • Return value is true when the execution was successful false if not

```
ok = EEPSetFogIntensity(40)
```

EEPGetFogIntensity()	EEPGetFogIntensity()
Parameters	none
Returns	one
Requires	EEP 16.1 Plug-in 1
Purpose	Returns fog intensity
Notes	• Return value is fog intensity in percent. • **Be aware that** once the fog intensity has been set by using *EEPSetFogIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetFogIntensity() function.

```
Fog_intensity = EEPGetFogIntensity()
```

EEPSetZonePos()		EEPSetZonePos(zone_number, pos_X, pos_Y, pos_Z, radius)
Parameters	five	
Returns	one	`ok = EEPSetZonePos(3, 250, -160, 0, 300)`
Requires	EEP 17.1 - Plug-in 1	
Purpose	Moves the named weather zone to a new position and/or changes its radius.	
Notes	• First parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window. • Second parameter is the position of the zone centre in X-direction. • Third parameter is the position of the zone centre in X-direction. • Fourth parameter is the position of the zone centre in X-direction. • Fifth parameter is the radius of the weather zone. • **Be aware that** the positioning of the centre of the weather zone may only be done within the system boundaries minus 1 pixel in each direction. The function is not executed beyond this. There is no error message! • Return value is true if the zone exists, false if not.	

EEPGetZonePos ()		EEPGetZonePos(zone_number)
Parameters	one	
Returns	five	`ok, pos_X, pos_Y, pos_Z, radius = EEPGetZonePos(3)`
Requires	EEP 17.1 - Plug-in 1	
Purpose	Returns the current position of a weather zone and its radius.	
Notes	• Parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window. • First return value is true when the execution was successful, false if not. • Second return value is the position of the zone centre in X-direction. • Third. return value is the position of the zone centre in Y-direction. • Fourth return value is the position of the zone centre in Z-direction. • Fifth return value is the radius of the zone. • **Be aware that** once the zone has been moved by using *EEPSetZonePos()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new mode thanks to the EEPGetZonePos() function.	

EEPSetZoneWindIntensity()		EEPSetZoneWindIntensity(zone_number, wind_intensity)
Parameters	two	
Returns	one	`ok = EEPSetZoneWindIntensity(2, 60)`
Requires	EEP 17.1 Plug-in 1	
Purpose	Sets wind intensity in a weather zone between 10 % and 100 % (corresponding to the settings under its properties window).	
Notes	• First parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • Second parameter is wind intensity in percent. If the value is below 10 % then the wind intensity will be reduced to 0. This setting will immediately be applied. • Return value is true when the execution was successful false if not	

EEPGetZoneWindIntensity()		EEPGetZoneWindIntensity(zone_number)
Parameters	one	
Returns	two	`ok, wind_intensity = EEPGetZoneWindIntensity(2)`
Requires	EEP 17.1 Plug-in 1	
Purpose	Returns wind intensity in a weather zone.	
Notes	• Parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • First return value is true when the execution was successful, false if not. • Second return value is wind intensity in percent. • **Be aware that** once the wind intensity has been set by using *EEPSetZoneWindIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetZoneWindIntensity() function.	

EEPSetZoneRainIntensity()		EEPSetZoneRainIntensity(zone_number, rain_intensity)
Parameters	two	
Returns	one	`ok = EEPSetZoneRainIntensity(1, 50)`
Requires	EEP 17.1 Plug-in 1	
Purpose	Sets rain intensity in a weather zone between 10 % and 100 % (corresponding to the settings under its properties window)	
Notes	• First parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • Second parameter is wind intensity in percent. If the value is below 10 % then the wind intensity will be reduced to 0. The new value is set immediately, but rain and clouds change gradually. • Return value is true when the execution was successful false if not	

EEPGetZoneRainIntensity()	EEPGetZoneRainIntensity(zone_number)
Parameters	one
Returns	two
Requires	EEP 17.1 Plug-in 1
Purpose	Returns rain intensity in a weather zone.
Notes	• First return value is true when the execution was successful, false if not. • Second return value is rain intensity in percent. • **Be aware that** once the rain intensity has been set by using *EEPSetZoneRainIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetZoneRainIntensity() function.

```
ok, rain_intensity = EEPGetZoneRainIntensity(1)
```

EEPSetZoneSnowIntensity()	EEPSetZoneSnowIntensity(zone_number, snow_intensity)
Parameters	two
Returns	one
Requires	EEP 17.1 Plug-in 1
Purpose	Sets snow intensity in a weather zone between 10 % and 100 % (corresponding to the settings under its properties window)
Notes	• First parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • Second parameter is snow intensity in percent. If the value is below 10 % then the snow intensity will be reduced to 0. The new value is set immediately, but snow and clouds change gradually. • Return value is true when the execution was successful false if not

```
ok = EEPGetZoneSnowIntensity(3, 35)
```

EEPGetZoneSnowIntensity()	EEPGetZoneSnowIntensity(zone_number)
Parameters	one
Returns	two
Requires	EEP 17.1 Plug-in 1
Purpose	Returns snow intensity in a weather zone.
Notes	• Parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • First return value is true when the execution was successful, false if not. • Second return value is snow intensity in percent. • **Be aware that** once the snow intensity has been set by using *EEPSetZoneSnowIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetZoneSnowIntensity() function.

```
ok, snow_intensity = EEPGetZoneSnowIntensity(3)
```

EEPSetZoneHailIntensity()	EEPSetZoneHailIntensity(zone_number, hail_intensity)
Parameters	two
Returns	one
Requires	EEP 17.1 Plug-in 1
Purpose	Sets hail intensity in a weather zone between 10 % and 100 % (corresponding to the settings under its properties window)
Notes	• First parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • Second parameter is hail intensity in percent. If the value is below 10 % then the hail intensity will be reduced to 0. The new value is set immediately, but hail and clouds change gradually. • Return value is true when the execution was successful false if not

```
ok = EEPSetZoneHailIntensity(4, 55)
```

EEPGetZoneHailIntensity()	EEPGetZoneHailIntensity(zone_number)
Parameters	one
Returns	two
Requires	EEP 17.1 Plug-in 1
Purpose	Returns hail intensity in a weather zone.
Notes	• Parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • First return value is true when the execution was successful, false if not. • Second return value is hail intensity in percent. • **Be aware that** once the hail intensity has been set by using *EEPSetZoneHailIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetZoneHailIntensity() function.

```
ok, hail_intensity = EEPGetZoneHailIntensity(4)
```

EEPSetZoneFogIntensity()	EEPSetZoneFogIntensity(zone_number, fog_intensity)
Parameters	two
Returns	one
Requires	EEP 17.1 Plug-in 1
Purpose	Sets fog intensity in a weather zone between 10 % and 100 % (corresponding to the settings under its properties window)
Notes	• First parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • Second parameter is fog intensity in percent. If the value is below 10 % then the fog intensity will be reduced to 0. This setting will immediately be applied. • Return value is true when the execution was successful false if not

```
ok = EEPSetZoneFogIntensity(2, 40)
```

EEPGetZoneFogIntensity()	EEPGetZoneFogIntensity(zone_number)
Parameters	one
Returns	two
Requires	EEP 17.1 Plug-in 1
Purpose	Returns fog intensity in a weather zone
Notes	• Parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window • First return value is true when the execution was successful, false if not. • Second return value is fog intensity in percent. • **Be aware that** once the fog intensity has been set by using *EEPSetZoneFogIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new intensity thanks to the EEPGetZoneFogIntensity() function.

```
ok, fog_intensity = EEPGetZoneFogIntensity(2)
```

EEPSetZoneClouds ()	EEPSetZoneClouds(zone_number, mode)
Parameters	two
Returns	one
Requires	EEP 17.1 - Plug-in 1
Purpose	Defines if there are clouds in a weather zone and what type they are.
Notes	• First parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window. • Second parameter is the clouds mode of the weather zone: 0 = No clouds, 1 = Clouds, 2 = Dark clouds. • Return value is true when the execution was successful, false if not.

```
ok = EEPSetZoneClouds(1, 2)
```

EEPGetZoneClouds()	EEPGetZoneClouds(zone_number)
Parameters	one
Returns	two
Requires	EEP 17.1 - Plug-in 1
Purpose	Returns whether there are clouds in a weather zone and what kind they are.
Notes	• Parameter is the number of the existing weather zone as a numeric number. It is in the top line of its properties window. • Second return value is the clouds mode of the weather zone: 0 = No clouds, 1 = Clouds, 2 = Dark clouds. • Return value is true when the execution was successful, false if not. • **Be aware that** once the mode has been set by using *EEPSetZoneClouds()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new mode thanks to the EEPGetZoneClouds() function.

```
ok, mode = EEPGetZoneClouds(1)
```

EEPGetCloudsMode()		EEPGetCloudsMode()
Parameters	none	`mode = EEPGetCloudsMode()`
Returns	one	
Requires	EEP 17.1 - Plug-in 1	
Purpose	Returns whether there are clouds in the sky globally (outside any weather zones) and what kind they are.	
Notes	• This function is called by EEP without any parameters. • Return value is the global clouds mode: 0 = No clouds, 1 = Clouds, 2 = Dark clouds. • **Be aware that** once the mode has been set by using *EEPSetCloudsIntensity()* respectively. *EEPSetDarkCloudsIntensity()* you'll have to wait until a new *EEPMain()* cycle has begun to get the new mode thanks to the EEPGetCloudsMode() function.	

EEPSetSeason()		EEPSetSeason()
Parameters	one	`EEPSetSeason(3)`
Returns	none	
Requires	EEP 18.0	
Purpose	Changes the season setting in the system.	
Notes	• Parameter is the season set in the layout as a number: 1 = Spring, 2 = Summer, 3 = Autumn, 4 = Winter. • This function has no return value.	

EEPGetSeason()		EEPGetSeason()
Parameters	none	`Season = EEPGetSeason()`
Returns	one	
Requires	EEP 17.2 - Plug-in 2	
Purpose	Returns the season set in the layout.	
Notes	• This function is called by EEP without any parameters. • The return value is the season set in the layout as a number: 1 = Spring, 2 = Summer, 3 = Autumn, 4 = Winter.	

21 Control desk functions

EEPActivateCtrlDesk()		EEPActivateCtrlDesk("Name")
Parameters	one	
Returns	one	`ok = EEPActivateCtrlDesk("main-station")`
Requires	EEP 16.1 Plug-in 1	
Purpose	Calls up the control desk in the radar window	
Notes	• Parameter is the control desk name as a string. **Be careful**, it is not its Lua name. The name of the control panel is in the field of the same name of its properties window. • Return value is true when the execution was successful false if not	

Impressum

TREND Redaktions- und Verlagsgesellschaft mbH
Leo-Wohleb-Str. 8,
79098 Freiburg.
Germany

Service E-Mail: info@trendverlag.com

Managing director: Jürgen Ludwig
USt.-Id.Nr.: DE 142476761
HRB-Nr.: 300347
Register court: Freiburg

Programming: Romuald Bacza
Text and Layout: Fried Sauert