ModelDesk

Traffic Object Management

For ModelDesk 5.5

Release 2021-A – May 2021

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: http://www.dspace.com/go/locations
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
  Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: http://www.dspace.com/go/supportrequest. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/go/patches for software updates and patches.

Important Notice

Contents

About This Document

Contents

This document introduces you to the management of traffic objects.

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	Indicates tips that can make your work easier.
	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Special folders

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

```
%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>
```

or

```
%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>
```

Documents folder A standard folder for user-specific documents.

```
%USERPROFILE%\Documents\dSPACE\<ProductName>\
<VersionNumber>
```

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

```
%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\
<ProductName>
```

Accessing dSPACE Help and PDF Files

After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com/go/help.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the icon in dSPACE Help. The PDF opens on the first page.

Basics and Instructions

Where to go from here

Information in this section

Basics of Traffic Objects

Introduction

Traffic objects are used to simulate and visualize the other traffic participants of a traffic scenario or the static objects in a MotionDesk scene. Traffic objects are visualized by an assigned 3-D object. In a simulation they can be detected by object detection sensors.

Using traffic objects

Traffic objects are used when object detection sensors for a driver assistance system are simulated. You can use traffic objects in two areas of applications:

- To simulate and visualize other traffic participants, such as cars or persons, in traffic scenarios. The movement of such traffic objects is defined with the Traffic Editor, calculated in the simulation, and visualized in MotionDesk.
- To visualize static objects (e.g., traffic signs) in a scenery of a road network, such traffic objects are integrated into the scenery with the Road Generator and visualized in MotionDesk after scene generation.

Managing traffic objects

Traffic objects are created and specified with the Traffic Object Manager. The traffic objects are stored in the `Objects` folder of the Pool.

To structure your traffic object collection, you can create folders with the Traffic Object Manager.

To use traffic objects in several ModelDesk projects, you can export and import the traffic objects.

The traffic objects in the Pool can be accessed by the Traffic Editor and the Road Generator.

Traffic objects in the simulation

The ASM Traffic blockset contains the Sensors subsystem which simulates the object detection sensors for the driver assistance system. The subsystem simulates several sensor types that can detect the traffic objects which correspond to the sensor type. For details on the Sensors subsystem, refer to Sensors (ASM Traffic Reference 🕮).

Traffic objects in the visualization

To visualize the traffic objects in a MotionDesk scene, you can assign any 3-D object of the dSPACE Objects library or custom object library to a traffic object. The 3-D object specifies the geometry of the traffic object and its states. If a 3-D object is updated, you can trigger a reload to get the latest parameters.

Tip

The dSPACE Objects library already contains suitable 3-D objects. If there are not enough objects for you, you can import your own 3-D objects into the custom object library in MotionDesk. Refer to How to Import Objects into the Custom Objects Library (MotionDesk Custom Object Library Management 🕮).

Support for state objects

ModelDesk supports traffic objects that reference state objects. The traffic objects specify the states of the subobjects of the state objects. These settings are also synchronized when you synchronize the traffic objects of the scene with MotionDesk.

When traffic objects are used as Shape objects of the Repeating type, the settings of state objects are not considered.

When traffic objects are used as fellows, the settings of state objects are only partly considered.

You can specify the states of a state object only when you create a traffic object.

Tip

If you want to use state objects in different configurations, you must create traffic objects for each configuration.

Related topics

Basics

Using State Objects and Animated Characters in the Scene (MotionDesk Scene Animation 🕮)

How to Use the dSPACE Objects Library on a PC Without a MotionDesk Installation

Objective

The dSPACE objects library is installed together with MotionDesk on a MotionDesk PC. If you want to use 3-D objects of this library for the traffic objects and MotionDesk is not installed on the same PC as ModelDesk, you must make the library available to ModelDesk manually.

Basics

You can use geometries of 3-D objects only when they are installed on the same PC as ModelDesk. If MotionDesk and ModelDesk are not installed on the same PC, you can manually copy the library from a MotionDesk PC to the ModelDesk PC.

You can copy the whole library or a part of it to the ModelDesk PC. However, when you copy the files, do not change the folder structure. Otherwise, MotionDesk is not able to find the used 3-D objects.

Possible methods

There are two methods to get the dSPACE objects library on a ModelDesk PC.

- You can copy the whole library or a part of it to the ModelDesk PC. Refer to Method 1 on page 14.
- You can install the MotionDesk product set on the ModelDesk PC even if you have no MotionDesk licenes. The dSPACE objects library is part of the MotionDesk product set but is not license-protected, so it is available after installation. Refer to Method 2 on page 14.

Precondition

To copy the library, a temporary memory, for example, a network drive or an USB memory stick, must be available. Note that the temporary memory must have a size greater than 4 GB when you copy the whole library.

Method 1

To use the dSPACE objects library on a PC without a MotionDesk installation using copy & paste

1. On the MotionDesk PC, copy the files from the `<RCP_HIL_InstallationPath>\3DLibrary\dSPACE Objects` folder to a temporary memory.

   Tip

   You can copy the whole library or only subfolders if only certain 3-D objects are required.

2. On the ModelDesk PC, copy the files from the temporary memory to the RCP and HIL installation folder. The folder structure must be the same as on the MotionDesk PC: `<RCP_HIL_InstallationPath>\3DLibrary\dSPACE Objects`.

Method 2

To use the dSPACE objects library on a PC without a MotionDesk installation

1. Install MotionDesk as described in Installing dSPACE Software 🕮.

Result

The dSPACE objects library is available on the ModelDesk PC. When you start ModelDesk, you can use the geometries to create traffic objects.

Some objects of the dSPACE objects library, especially the animated characters, require a valid license during animation. If your project has unlicensed objects, they will not move during animation.

Related topics

HowTos

How to Synchronize When MotionDesk and ModelDesk Run on Different PCs (ModelDesk Scene Synchronization 🕮)

Basics of the Traffic Object Manager

Introduction

The Traffic Object Manager helps you create and manage the traffic objects.

Features of the Traffic Object Manager

The Traffic Object Manager can be used to select a geometry file for the traffic object and specify its settings for the object detection sensor.

Geometry selection The 3-D Library Browser is used to access 3-D objects from the dSPACE objects library or the custom objects library. For information on how to work with it, refer to How to Work with the 3-D Library Browser on page 16.

Sensor definition To detect a traffic object in the simulation, the sensor type must be defined for it. You can enable the sensor type in the Properties pane or the Traffic Objects pane. The properties of the sensor type are specified in the Properties pane.

Folders You can create folders to structure the traffic objects. You can use drag & drop to move traffic objects from one folder to another. Folders can be collapsed and expanded to keep a clear overview.

Managing traffic objects The Traffic Objects pane has several commands in the context menu for you to manage the traffic objects. You can:

- Delete, copy, paste, and cut traffic objects
- Save one or all modified traffic objects
- Revert to the last saved settings of one or all traffic objects

Pool The traffic objects are stored in the Pool of a ModelDesk project. It is possible to export and import the settings of traffic projects from the Pool, so you can reuse your specified traffic objects in other ModelDesk projects.

Related topics

HowTos

How to Create a Traffic Object 17

How to Work with the 3-D Library Browser

Objective

The 3-D Library Browser is used to access 3-D objects from the dSPACE objects library or custom objects library.

You can enable/disable the hierarchical view or filter the objects in the 3-D Library Browser to work more efficiently.

Views in the 3-D Library Browser

The 3-D Library Browser provides different views of objects. You can switch between the following options:

- Hierarchical view: A folder-based view of objects
- Flat view: A flat view of objects

Filtering

You can filter the objects that are displayed in the 3-D Library Browser. The filter string can be used on the object or folder names, keywords, or both.

Possible methods

The 3-D Library Browser supports different tasks:

- To define the view of the library, refer to Method 1 on page 16.
- To filter objects, refer to Method 2 on page 16.

Method 1

To define the view of the 3-D Library Browser

1. To switch between the Hierarchical and Flat view, open the context menu of the object browser and select HierarchicalView.

   You can also use buttons:

   - Click to set the Hierarchical view.
   - Click to set the Flat view.

Method 2

To filter objects

1. In the Filter By section, select a category to apply a filter to Object \Folder Name, Keyword or Both.
2. Enter the filter term in the edit field.

   The object names, folder names, or keywords that match the filter term are listed. Click one of the entries to use it as filter term. Otherwise the entered filter term is used.

The **3-D Library Browser** displays only objects that match the filter term and objects that are in a folder that match the filter term.

3 To remove the filter, click .

The **3-D Library Browser** again displays all the objects in the libraries.

4 You can also select a term from the list of previously used filters by clicking the edit field arrow.

Result

The MotionDesk Library Manager is organized as you require.

Related topics

References

3-D Library Browser 27

How to Create a Traffic Object

Objective

When you want to simulate and visualize traffic scenarios, you need traffic objects.

Traffic objects for fellows

If you want to use a traffic object as a fellow, do not use a grouped object. The scene generation fails if fellows have a geometry of grouped objects.

OpenDRIVE import

When you want to import roads in the OpenDRIVE format in ModelDesk, traffic objects can automatically be assigned to OpenDRIVE signal objects. To be able to do this, the country, type, subtype, and value of the objects must be identical. Refer to Basics of Importing Road Models in OpenDRIVE Format (ModelDesk Road Creation 🕮).

Preconditions

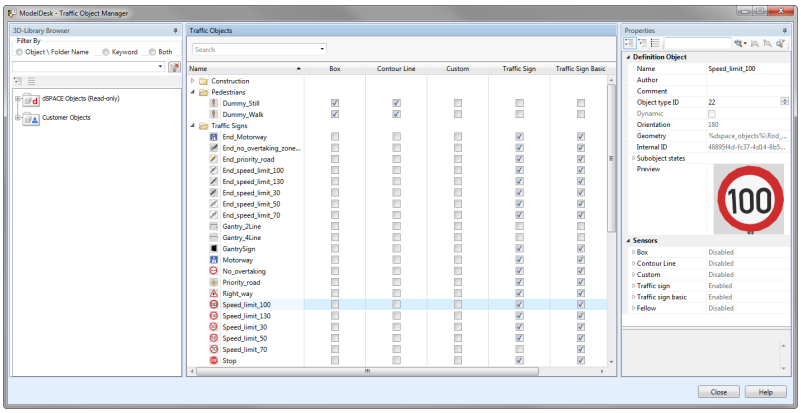
A project must be loaded in ModelDesk.

Method

To create a traffic object

1 On the Environment ribbon, click Traffic Objects – Manage.

The Traffic Object Manager opens.



2 To structure the traffic object library, create folders:

1. In the Traffic Objects pane, open the context menu and choose New – Folder.

   A new folder is created at the current position.

2. Repeat the previous step until the structure is complete.

3 From the 3-D Library Browser, drag the 3-D object whose geometry you want to use for the traffic object and drop it to a folder in the Traffic Objects pane.

A new traffic object is created.

4 In the Traffic Objects pane, select the new traffic object.

The Properties pane displays the properties of the selected traffic object.

5 In the Properties pane, specify the properties according to your needs.

To use a traffic object for a sensor type, you must enable the corresponding sensor type first and then specify the properties afterwards. You can enable the sensor type in the Property pane or the Traffic Objects pane.

For a contour line sensor, you must specify the contour line of the traffic object, refer to How to Specify a Contour Line on page 19.

To use the traffic object as fellow vehicle with the traffic model, you must enable it in the Fellow group and specify the properties of the group.

For a description of the properties, refer to Definition Object Properties on page 32, Sensor Properties on page 34 and Fellow Properties on page 33.

6 To save the new traffic object, open the context menu of the new traffic object in the Traffic Objects pane and choose Save.

Result

The traffic object is created and saved in the pool.

Related topics

HowTos

How to Work with the 3-D Library Browser..... 16

References

3-D Library Browser..... 27
Manage Traffic Objects..... 29

How to Specify a Contour Line

Objective

So that a traffic object can be detected by an object sensor, you must enable the contour sensor property and define the contour of the traffic object. You specify the contour line with the Contour Line Editor.

Basics

When you specify a contour line, you specify a list of points which altogether make up the contour line. You can enter the points in the Contour Line Editor. When you enter the points, the Contour Line Editor displays a preview of the contour line.

If you have also a box sensor for the traffic object, you can initialize the contour line with the values of the box sensor.

Tip

It is possible to export the contour line into a file and import it in the Contour Line Editor. So you can reuse a specified contour line in other projects.

Preconditions

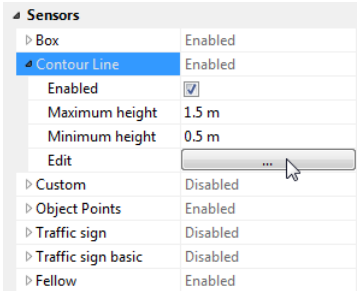
The Traffic Object Manager is open.

Method

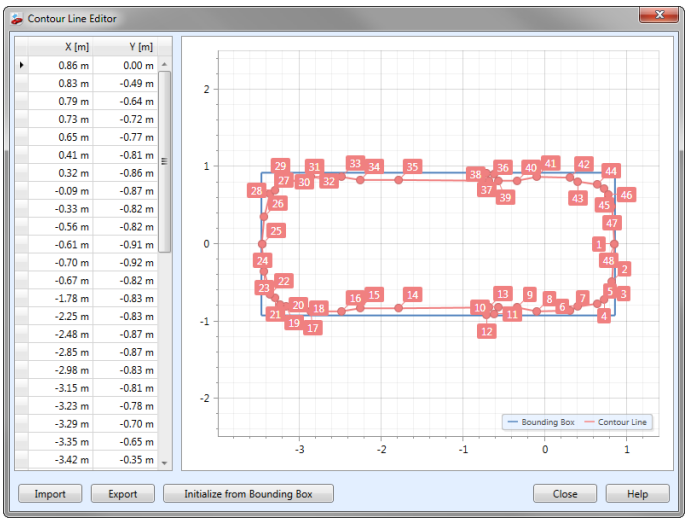
To specify a contour line

1. In the Traffic Object Manager, select the traffic object whose contour line should be detected by a sensor.
2. In the Properties pane, enable the Sensors – Contour Line property of the traffic object.

3 Click the button in the **Edit** property to open the **Contour Line Editor**.



The **Contour Line Editor** dialog opens.



4 If the box sensor is enabled in addition to the contour line sensor, you can initialize the contour line by using the values of the box sensor. To do this, click **Initialize from Bounding Box**.

5 To specify a point for the contour line, select an entry in the list, open its context menu and choose **Insert**.

A point is inserted the selected position. The index values are adapted automatically.

6 Specify the **X** and **Y** coordinates of the new point.

The preview in the **Contour Line Editor** is updated with the values of the new point.

7 Repeat the previous steps until the whole contour line is specified.

8 Click **Close**.

Result

The contour line of the traffic object is specified.

Related topics

References

Contour Line Editor Dialog.....26
Object Sensor 2-D (ASM Traffic Reference 🕮)
Object Sensor 3-D (ASM Traffic Reference 🕮)
Sensor Properties.....34

How to Specify Points for Object Points Sensors

Objective

So that a traffic object can be detected by an object point sensor, you must enable the object points sensor property and define the points of the traffic object.

Basics

You can specify three kind of points for a traffic object:

- Radar reflection points
- NCAP reference point
- Custom points

The Object Point Editor shows the traffic object and the points. Most traffic objects can be displayed transparently so you can see points which are placed within the object. To get information on the point positions, the view can also display the coordinate system.

Preconditions

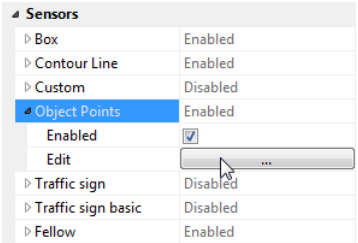
- The Traffic Object Manager is open.
- To display the preview of the traffic object, you must have installed a compatible graphics card und the newest driver for that card. The 3-D View requires at least OpenGL version 2.1 and OpenGL Shading Language version 1.4.

Method

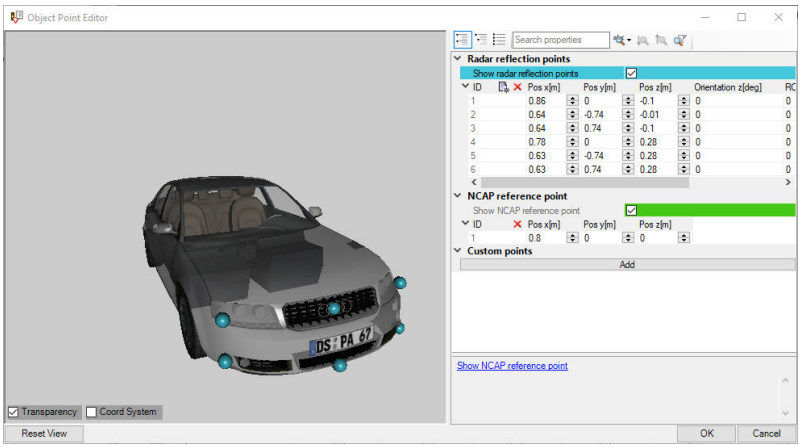
To specify points for object points sensors

1. In the Traffic Object Manager, select the traffic object whose object points should be detected by a sensor.
2. In the Properties pane, enable the Sensors – Object Points property of the traffic object.

3 Click the button in the **Edit** property to open the **Object Point Editor**.



The **Object Points Editor** dialog opens.



4 To specify a new point, click the **Add** button (Add or) ModelDesk insert a new point in the list. The index values are adapted automatically.

5 Specify the coordinates and other properties of the point if required. For details, refer to Object Point Properties on page 36.

The view in the **Object Points Editor** is updated with the new point.

Tip

To get a better view to the points, set the traffic object transparent or add the trafic object's coordinate system to the view.

6 Repeat the previous steps until all the points are specified.

7 Click **OK**.

Result

The points of the traffic object are specified. However, the points are saved only if you save all the settings of the traffic object in the **Traffic Object Manager**.

Related topics

References

Custom Points (ASM Traffic Reference)
Object Point Editor.....28
Object Point Properties.....36

Radar Reflection Points (ASM Traffic Reference 🕮)

Reference Information

Where to go from here

Information in this section

Traffic Object Manager Dialog and Commands

Introduction

ModelDesk provides various commands and dialogs for specifying traffic objects.

Where to go from here

Information in this section

Contour Line Editor Dialog

Access

You can access this command via:

Ribbon	None
Context menu of	None
Shortcut key	None
Icon	None
Button	Edit button in the properties of a contour line sensor

Purpose

To specify a contour line.

Description

The dialog lets you specify the points of a contour line for a contour line sensor.

Contour Editor

Table of points Lists the points of the contour line. The values of Index is specified automatically. The values of X and Y are the coordinates of the points which form the contour line.

The table has a context menu with the following commands:

Purpose	Command
To insert a point to the contour line.	Insert
To delete the selected point.	Delete
To close the contour line.	Close Contour

Import Lets you import a list of points.

Export Lets you export the specified list of points.

Initialize from Bounding Box Lets you use the values of the bounding box to initialize the contour line.

Related topics

HowTos

How to Specify a Contour Line..... 19

3-D Library Browser

Access

You can access the 3-D Library Browser via:

Ribbon	None
Context menu of	None
Shortcut key	None
Icon	None
Window	Traffic Object Manager

Purpose

To select 3-D objects for further actions.

Result

3-D objects are found and selected for further actions.

The 3-D Library Browser options

The 3-D Library Browser offers the following view options:

Filter options Filters reduce the number of displayed 3-D objects so that finding 3-D objects is less time-consuming. 3-D objects are filtered by a string entered in an edit field. The Filter By option bar provides the following filter options:

Filter By	Description
Object \ FolderName	Filters the list of 3-D objects by a specified string. Names of 3-D objects and folder that match the string are listed.
Keyword	Filters the list of 3-D objects by a specified string. Keywords of 3-D objects that match the string are listed.
Both	Filters the list of 3-D objects by a specified string. Only 3-D objects whose names or keywords include the string are listed.

Additional Filter Elements	Description
Edit field	Lets you enter the filter string. If more than 2 characters are specified, 3-D objects containing the specified characters are listed.
Drop-down arrow	Lists all previously entered filter strings.
	Displays all available 3-D objects without filtering.

Objects View The Objects View displays the available filtered/unfiltered 3-D objects. The 3-D objects are represented by their thumbnail preview and their file names. The Objects View can be hierarchical or flat. You can toggle between the views with the Hierarchical View command of the Objects View's context menu or using the (hierarchical view) and (flat view) buttons.

Related topics

HowTos

How to Work with the 3-D Library Browser..... 16

Object Point Editor

Access

You can access this command via:

Ribbon	None
Context menu of	None
Shortcut key	None
Icon	None
Button	Edit button in the properties of an object point sensor

Purpose

To specify points of a traffic object for an object point sensor, such as radar reflection points, the NCAP reference point, and custom points.

Description

The dialog lets you specify the points of object point sensors. You can add and remove points and specify their positions and other properties.

The dialog has a view that visualized the positions of the points as small spheres. The color of the spheres symbolized the kind of point:

- Cyan: Radar reflection points
- Green: NCAP reference point
- Yellow: Custom point
- Red: Currently selected point

You can turn the 3-D object by clicking on the view and dragging the mouse.

Dialog settings

Transparency Enables the transparency.

Coord System Shows the coordinate system.

Reset View Resets the view. You can also press the **Space** key.

Properties In the Properties list, you can add and remove points and specify their properties.

: Adds a point.

: Deletes the selected point.

You can specify the position and other properties in each line of a point. For information on the properties, refer to Object Point Properties on page 36.

Related topics

HowTos

References

Manage Traffic Objects

Access

You can access this command via:

Ribbon	Environment – Traffic Objects – Manage
Context menu of	None
Shortcut key	None
Icon	
Button	▪ Manage Object in the properties of an object instance

	▪ Manage Objects in the Scenario Editor ▪ Manage Objects in the Fellow Properties dialog of the Scenario Editor

Purpose

To open the Traffic Object Manager for creating and specifying traffic objects.

Traffic Object Manager

The Traffic Object Manager has three panes.

3-D Library Browser pane With the 3-D Library Browser you can select a 3-D object from the dSPACE objects library or custom objects library and use its geometry. You can filter the lists by the object name and/or keyword. Refer to 3-D Library Browser on page 27.

Traffic Objects The Traffic Objects pane lists the traffic objects defined in the current ModelDesk project. The context menu of the pane has several commands. See the following table.

Command	Purpose
New – Traffic Object	To create a new traffic object.
New – Folder	To create a new folder.
Copy	To copy the selected traffic object or folder to the Clipboard.
Cut	To cut the selected traffic object or folder.
Paste	To paste a traffic object or folder from the Clipboard.
Delete	To delete the selected traffic object or folder.
Reload Geometry	To read updated properties from the reference 3-D geometry.
Save	To save the selected traffic object.
Revert	To discard the changes in the settings of the selected traffic object.
Save all	To save all the traffic objects.
Revert all	To discard the changes in the settings of all the traffic objects.
Expand All	To expand all the folders.
Collapse All	To collapse all the folders.

You can use the mouse to create or modify the traffic objects. See the following table.

Purpose	Mouse Operation
To create a new traffic object in the root folder.	Drag a 3-D object from the 3-D Library Browser to anywhere in the Traffic Objects pane, but not to an existing traffic object or folder.
To create a new traffic object in a folder.	Drag a 3-D object from the 3-D Library Browser to the folder in the Traffic Objects pane.

Purpose	Mouse Operation
To replace the geometry of an existing traffic object.	Drag the 3-D object with the new geometry from the **3-D Library Browser** to an existing traffic object in the **Traffic Objects** pane.
To restructure the traffic objects.	Drag the traffic object to another folder in the **Traffic Objects** pane.
To enable a sensor of a traffic object.	Select the checkbox of the sensor in the **Traffic Objects** pane. You can also select the corresponding **Supported** property in the **Properties** pane.

Properties pane The **Properties** pane lets you modify the properties of the selected traffic object or folder. For information on the properties, refer to Definition Object Properties on page 32 and Sensor Properties on page 34.

Related topics

HowTos

How to Create a Traffic Object..... 17

Traffic Object Properties

Purpose

ModelDesk's **Properties** pane shows the properties of the selected traffic object.

Where to go from here

Information in this section

Definition Object Properties

Purpose

To specify general properties of a traffic object.

Definition object

Author Lets you specify the author of the traffic object.

Comment Lets you specify a comment for the traffic object.

Dynamic Lets you specify whether the state of the traffic object is controlled by the simulation. You can set the flag only for geometries that have exactly one state object, for example, a traffic light signal or gantry sign.

3D Object Displays the path of the 3-D object that is used in MotionDesk for visualization.

Internal ID Displays the internal ID of the traffic object. The internal ID is required if you want to import a road model in the OpenDRIVE format. Refer to Basics of Importing Road Models in OpenDRIVE Format (ModelDesk Road Creation 🕮).

Name Lets you specify the name of the traffic object.

Object type ID Lets you specify the identifier for the traffic object. The identifier must be unique.

Orientation Displays the orientation of the traffic object.

Preview Displays the thumbnail preview of the traffic object's geometry.

Subobject states Displays the states of the subobject. Refer to Using State Objects and Animated Characters in the Scene (MotionDesk Scene Animation 🕮).

Related topics

HowTos

How to Create a Traffic Object..... 17

Fellow Properties

Purpose

To specify dynamics and geometry properties for a traffic object so that it can be used as fellow vehicle in a traffic scenario with a traffic driver.

General properties

Enabled Lets you enable the use of the traffic object as fellow vehicle. Only if this option is enabled, you can use the traffic object in a traffic scenario with the traffic model.

Dynamics properties

Power Lets you specify the engine power of the fellow in kW.

Vehicle mass Lets you specify the vehicle mass of the fellow in kg.

Maximum speed Lets you specify the maximum possible speed of the fellow in km/h.

Maximum deceleration Lets you specify the maximum deceleration of the fellow in m/s^2.

Geometry properties

Wheel radius Lets you specify the wheel radius of the fellow in meters.

Rear track width Lets you specify track width of the fellow's rear axle in meters.

Front track width Lets you specify track width of the fellow's front axle in meters.

Wheelbase Lets you specify the wheelbase in meters.

Related topics

HowTos

How to Create a Traffic Object..... 17

Sensor Properties

Purpose

To specify the sensor properties.

General properties

Enabled Lets you enable the sensor for the traffic object. You can specify a sensor only if it is enabled.

Box

Lets you specify the properties of a bounding box.

Center point Lets you specify the position of the bounding box center point in Cartesian coordinates (X, Y, Z) in meters relative to the geometry's origin.

Height Lets you specify the height of the bounding box in meters.

Length Lets you specify the length of the bounding box in meters.

Width Lets you specify the width of the bounding box in meters.

Contour line

Lets you specify the properties of a contour line.

Maximum height Lets you specify the maximum height.

Minimum height Lets you specify the minimum height.

Edit Lets you specify the contour line. When you click the button, the Contour Editor dialog opens. Refer to Contour Line Editor Dialog on page 26.

Custom

Lets you specify the properties of a custom sensor.

User data Lets you specify names and values for the sensor.

Object points

Edit Lets you specify radar reflection points, the NCAP reference point, and custom points of the traffic object. When you click the button, the Object Point Editor opens. Refer to Object Point Editor on page 28.

Traffic sign

Lets you specify the properties for the traffic sign sensor. These properties are based on the plates of the traffic signs.

State-dependent values (Only available if the traffic object is marked as dynamic) Lets you enable a state-dependent use of the traffic sign.

If a traffic object supports several states, you can specify the properties of the Sensor category (Country, Type, SubType, Value, Text, and Visibility) for each state separately.

State – Display (Only if State-dependent values is enabled) Lets you select the state for which the parameters should be shown.

Sensor – Country Lets you specify the country code. The country code is a three-letter string according to ISO 3166 ALPHA-3. The following table shows some examples:

Country Code	Country
CHN	China
DEU	Germany
FRA	France
GBR	United Kingdom
JPN	Japan
USA	United States of America

Sensor – Type Lets you specify the numerical main type.

Sensor – SubType Lets you specify the numerical sub type.

Sensor – Value Lets you specify the value of the traffic sign. The meaning of the value depends on the type of the traffic sign, for example, if it specifies a speed limit or mass.

Sensor – Text Lets you specify text associated with the traffic sign.

Sensor – Visibility Lets you specify the visibility of the traffic sign. You can lower the visibility to simulate dirty traffic signs. The value does not influence the visualization in MotionDesk.

Encoding Displays whether the sensor is enabled in the simulation.

Encoding – State Displays the value of the state as used in the simulation.

Encoding – Country Displays the value for the country as used in the simulation according to ISO 3166-1 numeric.

Encoding – Visibility Displays the value of the visibility as used in the simulation.

Traffic sign basic

Lets you specify the properties for the basic traffic sign sensor.

Category Lets you select the category of the traffic sign.

Sensor – Identifier Lets you specify the identifier of the traffic sign.

Sensor – Visibility Lets you specify the visibility of the traffic sign. You can lower the visibility to simulate dirty traffic signs. The value does not influence the visualization in MotionDesk.

Sensor – Type Lets you select the sensor type.

Sensor –Value (Only when the sensor type is speed limit or custom) Lets you specify the numerical value.

Encoding Displays whether the sensor is enabled in the simulation.

Encoding - Category Displays the sensor value for the category which is used in the simulation.

Encoding –Value Displays the value that is used in the simulation.

Encoding – Identifier Displays the value of the identifier which is used in the simulation.

Encoding – Visibility Displays the value of the visibility which is used in the simulation.

Related topics

HowTos

How to Create a Traffic Object.....17

References

Object Point Editor.....28
Traffic Sign Sensor Calculation (ASM Traffic Reference 🕮)

Object Point Properties

Purpose

To specify points of a traffic object for object point sensors, such as radar reflection points, the NCAP reference point, and custom points.

General properties

ID Displays the index of the point.

Pos x[m] Lets you specify the x coordinate of the point in meters.

Pos y[m] Lets you specify the y coordinate of the point in meters.

Pos z[m] Lets you specify the z coordinate of the point in meters.

Radar reflection points	**Opening angle** Lets you specify the opening angle in meters. **Orientation z[deg]** Lets you specify the orientation in the z direction in degrees. **RCS** Lets you specify the radar cross section in dB/m^2. **Reflection points** Lists the radar reflection points. **Show radar reflection points** Shows the radar reflection points in the view.
NCAP reference point	**Show NCAP reference point** Shows the NCAP reference point in the view. **Reference point** Lets you specify the position of the NCAP reference point.
Custom points	**Custom points** Lists the custom points. **Show custom points** Shows the specified custom points in the view. **Data <n>** (n = 1 … 5) Lets you specify a data value.
Related topics	HowTos How to Specify Points for Object Points Sensors ..... 21 References Object Point Editor 28

Automation

Where to go from here

Information in this section

Programming ModelDesk Automation

Where to go from here

Information in this section

Managing Traffic Objects

Introduction

You can manage the traffic objects of a project using the automation of the Traffic Object Manager.

Features

The tool automation provides the following features:

- Managing (creating, modifying, or removing) traffic objects.
- Specifying the properties for the sensor:
  - Box properties
  - Contour line properties
  - Custom properties
  - Object points (radar, NCAP reference points, and custom)
  - Traffic sign properties
  - Traffic sign basic properties
- Specifying the properties to use traffic objects as fellows.

Example of Working with the Traffic Object Manager

Introduction

You can manage traffic objects using the tool automation.

Accessing the traffic object manager

The traffic objects are specified for a ModelDesk project. You must therefore start ModelDesk and open a project.

```
from win32com.client import Dispatch
# Start ModelDesk and load a project
Application = Dispatch("ModelDesk.Application")
Application.Visible = True
MyProject = Application.OpenProject(r"C:\ExamplePath\Example_001\Example_001.CDP")
TOM = MyProject.TrafficObjectManager
```

Adding a new traffic object

The following listing shows how to create a folder, add a new traffic object and move the traffic object to the folder.

```
# Add a new folder
TOM.AddFolder('MyObjects')
# Create a new traffic object
Name = 'MyObject'
Object3D = '%dspace_objects%\\Car_Compact\\$CompactCarWheels\\CompactCarWheels.dae'
NewObject = TOM.AddTrafficObject(Name,Object3D)
# Move the new traffic object to the new folder.
TOM.MoveTrafficObject(NewObject,'MyObjects')
# Create a new traffic object in a folder
Name = 'MyObjects\\MyObject2'
NewObject2 = TOM.AddTrafficObject(Name,Object3D)
```

Modifying the properties of a traffic object

The following listing shows how you can modify the properties of a traffic object. In this case, it is enabled for a box sensor and the dimensions of the box is specified.

```
# Enable the box sensor
NewObject.Sensors.Box.Enabled = True
# Specify the dimensions of the box
NewObject.Sensors.Box.Height = 1.2
NewObject.Sensors.Box.Length = 4.4
NewObject.Sensors.Box.Width = 1.8
# Specify the position of the center point
NewObject.Sensors.Box.CenterPoint.X = 3.4
NewObject.Sensors.Box.CenterPoint.Y = 0.9
NewObject.Sensors.Box.CenterPoint.Z = 0.2
# Save the settins
TOM.Save()
```

```
# Enable the box sensor
NewObject.Sensors.Box.Enabled = True
# Specify the dimensions of the box
NewObject.Sensors.Box.Height = 1.2
NewObject.Sensors.Box.Length = 4.4
NewObject.Sensors.Box.Width = 1.8
# Specify the position of the center point
NewObject.Sensors.Box.CenterPoint.X = 3.4
NewObject.Sensors.Box.CenterPoint.Y = 0.9
NewObject.Sensors.Box.CenterPoint.Z = 0.2
# Save the settings
TOM.Save()
```

Adding a stop sign

The following listing shows how you can add a stop sign and enable the basic traffic sign sensor for it.

```
# Import the enumerations provided by ModelDesk.
import dspace.com
Enums = dspace.com.Enums(Application)
Name = 'StopSign'
Object3D = '%dspace_objects%\\Rod_Signs\\$Stop\\Stop.dae'
StopSign = TOM.AddTrafficObject(Name,Object3D)
TOM.AddFolder('MyTrafficSigns')
TOM.MoveTrafficObject(StopSign,'MyTrafficSigns')
# Enable the traffic sensor basic and specify properties
StopSign.Sensors.TrafficSignBasic.Enabled = True
StopSign.Sensors.TrafficSignBasic.Category = Enums.TrafficSignTypes.Priority
StopSign.Sensors.TrafficSignBasic.SensorValues.Type = Enums.PriorityTypes.Stop
StopSign.Sensors.TrafficSignBasic.SensorValues.Visibility = 80
# Save the settings
TOM.Save()
```

Related topics

Basics

Overview of the Object Model for the Traffic Object Manager.....42

References

Classes for Managing Traffic Objects.....45

Overview of the Object Model for the Traffic Object Manager

Traffic object manager

The following table gives an overview of the object model classes for working with traffic objects:

Class	Level
TrafficObjectManager on page 92	0
FolderList on page 62	1

Class	Level
Folder on page 61	2
FolderList on page 62	3
TrafficObjectList on page 89	
TrafficObjectList on page 89	1
TrafficObject on page 87	2
SubObjectStates on page 83	3
SubObjectStateValues on page 85	4
Sensors on page 79	3
BoxProperties on page 47	4
CubicPoint on page 48	5
ContourLineProperties on page 49	4
PolygonLine on page 72	5
Point on page 71	6
CustomProperties on page 56	4
CustomSensorUserData on page 57	5
CustomSensorUserDataEntry on page 58	6
ObjectPointsSensor on page 70	4
RadarReflectionPoints on page 76	5
RadarReflectionPoint on page 75	6
Point on page 71	7
NCAPReferencePoints on page 67	5
NCAPReferencePoint on page 66	6
Point on page 71	7
CustomPoints on page 53	5
CustomPoint on page 52	6
Point on page 71	7
TrafficSignProperties on page 104	4
TrafficSignSensorData on page 106	5
TrafficSignSensorValues on page 110	6
TrafficSignSensorEncoding on page 107	
StateDependentValues on page 81	5
StateDependentValue on page 80	6
TrafficSignBasicProperties on page 102	4
TrafficSignBasicSensorValues on page 103	5

Class	Level
TrafficSignSensorPriorityValues on page 108	6
TrafficSignSensorSpeedLimitValues on page 109	
TrafficSignSensorNoOvertakingValues on page 108	
TrafficSignSensorCustomValues on page 105	
TrafficSignBasicSensorEncoding on page 102	5
FellowProperties on page 61	3
FellowDynamics on page 59	4
FellowGeometry on page 60	

Classes for Managing Traffic Objects

Where to go from here

Information in this section

BoxProperties

Purpose

To specify the properties for a box sensor.

Class Description (BoxProperties)

Syntax

```
BoxProperties = Sensors.Box
```

Purpose

To specify the properties for a box sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
CenterPoint	CubicPoint[1)]	To get/set the center point.
Enabled	Boolean	To get/set the enable state.
Height	Double	To get/set the height.
Length	Double	To get/set the length.
Width	Double	To get/set the width.

[1)] Refer to CubicPoint on page 48.

Methods –

Related topics References

Sensors.....79

CubicPoint

Purpose To specify a cubic point.

Class Description (CubicPoint)

Syntax

```
CubicPoint = BoxProperties.CenterPoint
```

Purpose To specify a cubic point.

Attributes The class contains the following attributes:

Attributes	Type	Purpose
X	Double	To get/set the x coordinate.
Y	Double	To get/set the y coordinate.
Z	Double	To get/set the z coordinate.

Methods –

Related topics References

BoxProperties.....47

ContourLineProperties

Purpose

To specify the properties for a contour line sensor.

Where to go from here

Information in this section

Class Description (ContourLineProperties)

Syntax

```
ContourLineProperties = Sensors.ContourLine
```

Purpose

To specify the properties for a contour line sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Enabled	Boolean	To get/set the enable state.
MaximumHeight	Double	To get/set the maximum height.
MinimumHeight	Double	To get/set the minimum height.
PolygonLine	PolygonLine[1)]	To get/set the polygon line.

[1)] Refer to PolygonLine on page 72.

Methods

The class contains the following methods:

Method	Purpose
Import	To import a contour line. Refer to Import on page 51.
Export	To export a contour line. Refer to Export on page 50.

Method	Purpose
InitializeFromBoundingBox	To initialize the contour line using the values of the bounding box. Refer to InitializeFromBoundingBox on page 51.

Related topics

References

Sensors.....79

Export

Class

ContourLineProperties

Syntax

```
RetVal = ContourLineProperties.Export(FilePath)
```

Purpose

To export a contour line.

Parameters

The method uses the following parameters:

Parameter	Type	Description
FilePath	String	The absolute path of a MAT file containing data of the contour line.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (ContourLineProperties).....49

Import

Class

ContourLineProperties

Syntax

```
RetVal = ContourLineProperties.Import(FilePath)
```

Purpose

To import a contour line.

Parameters

The method uses the following parameters:

Parameter	Type	Description
FilePath	String	The absolute path of a MAT file containing data of the contour line.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (ContourLineProperties) 49

InitializeFromBoundingBox

Class

ContourLineProperties

Syntax

```
ContourLineProperties.InitializeFromBoundingBox()
```

Purpose

To initialize the contour line using the values of the bounding box.

Parameters

–

Return value

–

Related topics

References

Class Description (ContourLineProperties)..... 49

CustomPoint

Purpose

To specify a custom point.

Class Description (CustomPoint)

Syntax

```
CustomPoint = CustomPoints.Item(Index)
CustomPoint = CustomPoints.Add()
```

Purpose

To specify a custom point.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Data1	Double	To get/set a data value.
Data2	Double	To get/set a data value.
Data3	Double	To get/set a data value.
Data4	Double	To get/set a data value.
Data5	Double	To get/set a data value.
ID	Integer	To get the ID of the point.
Position	Point[1)]	To get the position.

[1)] Refer to Point on page 71.

Methods –

Related topics

References

CustomPoints.....53

CustomPoints

Purpose To manage custom points.

Where to go from here

Information in this section

Class Description (CustomPoints)

Syntax

```
CustomPoints = ObjectPointsSensor.CustomPoints
```

Purpose To manage custom points.

Attributes The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of custom points.

Methods

The class contains the following methods:

Method	Purpose
Item	To get a custom point. Refer to Item on page 54.
Append	To append a custom point. Refer to Append on page 55.
Remove	To remove a custom point. Refer to Remove on page 55.

Related topics

References

ObjectPointsSensor.....70

Item

Class

CustomPoints

Syntax

```
CustomPoint  = CustomPoints.Item(Index)
```

Purpose

To get a custom point.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	The index of the custom point.

Return value

The method returns the following parameter:

Type	Description
CustomPoint[1)]	The custom point.

[1)] Refer to CustomPoint on page 52.

Related topics

References

Class Description (CustomPoints).....53

Append

Class

CustomPoints

Syntax

```
CustomPoint  = CustomPoints.Append()
```

Purpose

To append a custom point.

Parameters

–

Return value

The method returns the following parameter:

Type	Description
CustomPoint[1)]	The new custom point.

[1)] Refer to CustomPoint on page 52.

Related topics

References

Class Description (CustomPoints)..... 53

Remove

Class

CustomPoints

Syntax

```
RetVal  = CustomPoints.Remove(Index)
```

Purpose

To remove a custom point.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The custom point to be removed. To specify the custom point, you can use the index (integer type) or a reference to the object.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (CustomPoints)..... 53

CustomProperties

Purpose

To specify the properties for a custom sensor.

Class Description (CustomProperties)

Syntax

```
CustomProperties = Sensors.Custom
```

Purpose

To specify the properties for a custom sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Enabled	Boolean	To get/set the enable state.
UserData	CustomSensorUserData[1)]	To get the user data of the custom sensor.

[1)] Refer to CustomSensorUserData on page 57.

Methods

–

Related topics

References

Sensors..... 79

CustomSensorUserData

Purpose

To specify user data of a custom sensor.

Where to go from here

Information in this section

Class Description (CustomSensorUserData)

Syntax

```
CustomSensorUserData = CustomProperties.UserData
```

Purpose

To specify user data of a custom sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of entries.

Methods

The class contains the following methods:

Method	Purpose
Item	To get a specific user data entry. Refer to Item on page 58.

Related topics

References

Item

Class

CustomSensorUserData

Syntax

```
CustomSensorUserDataEntry = CustomSensorUserData.Item(Index)
```

Purpose

To get a user data entry of a custom sensor.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	The index of the user data entry.

Return value

The method returns the following parameter:

Type	Description
CustomSensorUserDataEntry[1)]	The user data entry.

[1)] Refer to CustomSensorUserDataEntry on page 58.

Related topics

References

CustomSensorUserDataEntry

Purpose

To specify a user data entry of a custom sensor.

Class Description (CustomSensorUserDataEntry)

Syntax

```
CustomSensorUserDataEntry = CustomSensorUserData.Item(Index)
```

Purpose

To specify a user data entry of a custom sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Name	String	To get/set a name.
Value	Double	To get/set a value.

Methods

–

Related topics

References

CustomSensorUserData.....57

FellowDynamics

Purpose

To specify the dynamic parameters of a fellow.

Class Description (FellowDynamics)

Syntax

```
FellowDynamics = FellowProperties.Dynamics
```

Purpose

To specify the dynamic parameters of a fellow.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Power	Double	To get/set the power.
VehicleMass	Double	To get/set the vehicle mass.
MaximumSpeed	Double	To get/set the maximum speed.
MaximumDeceleration	Double	To get/set the maximum deceleration.

Methods

–

Related topics

References

FellowProperties.....61

FellowGeometry

Purpose

To specify the geometry parameters of a fellow.

Class Description (FellowGeometry)

Syntax

```
FellowGeometry = FellowProperties.Geometry
```

Purpose

To specify the geometry parameters of a fellow.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
WheelRadius	Double	To get/set the wheel radius.
RearTrackWidth	Double	To get/set the rear track width.
FrontTrackWidth	Double	To get/set the front track width.
Wheelbase	Double	To get/set the wheel base.

Methods

–

Related topics

References

FellowProperties.....61

FellowProperties

Purpose

To specify properties of a traffic object so that it can be used as a fellow vehicle.

Class Description (FellowProperties)

Syntax

```
FellowProperties = TrafficObject.Fellow
```

Purpose

To specify properties of a traffic object so that it can be used as a fellow vehicle.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Enabled	Boolean	To get/set the enable state.
Dynamics	FellowDynamics[1)]	To get the dynamic parameters.
Geometry	FellowGeometry[2)]	To get the geometry parameters.

1) Refer to FellowDynamics on page 59.
2) Refer to FellowGeometry on page 60.

Methods

–

Related topics

References

Folder

Purpose

To specify a folder.

Class Description (Folder)

Syntax

```
Folder = FolderList.Item(Index)
Folder = FolderList.Add(Name)
```

Purpose

To specify a folder.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Folders	FolderList[1)]	To get the list of folders.
Name	String	To get/set the name of the folder.
Path	String	To get the path.
TrafficObjects	TrafficObjectList[2)]	To get the list of traffic objects.

1) Refer to FolderList on page 62.
2) Refer to TrafficObjectList on page 89.

Methods

–

Related topics

References

FolderList

Purpose

To manage the folders.

Where to go from here

Information in this section

Class Description (FolderList)

Syntax

```
FolderList = TrafficObjectManager.Folders
```

Purpose

To manage the folders.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of elements.
AvailableElements	String[]	To get all available folders.

Methods

The class contains the following methods:

Method	Purpose
Add	To add a new folder. Refer to Add on page 64.
Item	To get a specific folder. Refer to Item on page 64.
Remove	To remove a folder. Refer to Remove on page 65.

Related topics

Basics

Add

Class

FolderList

Syntax

```
Folder = FolderList.Add(Name)
```

Purpose

To add a folder.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Name	String	The name of the new folder.

Return value

The method returns the following parameter:

Type	Description
Folder[1)]	The new folder.

1) Refer to Folder on page 61.

Related topics

References

Class Description (FolderList)..... 63

Item

Class

FolderList

Syntax

```
Folder = FolderList.Item(Index)
```

Purpose

To get a folder.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The index (integer type) or name (string type) of the folder.

Return value

The method returns the following parameter:

Type	Description
Folder[1)]	The specific folder.

[1)] Refer to Folder on page 61.

Related topics

References

Class Description (FolderList)..... 63

Remove

Class

FolderList

Syntax

```
RetVal = FolderList.Item(Index)
```

Purpose

To remove a folder.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The folder to be removed. To specify the folder, you can use the index (integer type), name (string type), or a reference to the folder object.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (FolderList)..... 63

NCAPReferencePoint

Purpose

To specify an NCAP reference point.

Class Description (NCAPReferencePoint)

Syntax

```
NCAPReferencePoint = NCAPReferencePoints.Item(Index)
NCAPReferencePoint = NCAPReferencePoints.Append()
```

Purpose

To specify an NCAP reference point.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
ID	Integer	To get the ID.
Position	Point[1)]	To get the position.

[1)] Refer to Point on page 71.

Methods

–

Related topics

References

NCAPReferencePoints..... 67

NCAPReferencePoints

Purpose

To manage NCAP reference points.

Where to go from here

Information in this section

Class Description (NCAPReferencePoints)

Syntax

```
NCAPReferencePoints = ObjectPointsSensor.NCAPReferencePoints
```

Purpose

To manage NCAP reference points.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of NCAP reference points.

Methods

The class contains the following methods:

Method	Purpose
Append	To append an NCAP reference point. Refer to Append on page 68.
Item	To get a specific NCAP reference point. Refer to Item on page 68.
Remove	To remove an NCAP reference point. Refer to Remove on page 69.

Related topics

References

ObjectPointsSensor 70

Append

Class

NCAPReferencePoints

Syntax

```
NCAPReferencePoint = NCAPReferencePoints.Append()
```

Purpose

To append an NCAP reference point.

Parameters

–

Return value

The method returns the following parameter:

Type	Description
NCAPReferencePoint[1)]	The new NCAP reference point.

[1)] Refer to NCAPReferencePoint on page 66.

Related topics

References

Class Description (NCAPReferencePoints) 67

Item

Class

NCAPReferencePoints

Syntax

```
NCAPReferencePoint = NCAPReferencePoints.Item(Index)
```

Purpose

To get a specific NCAP reference point.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	The index of the specific NCAP reference point.

Return value

The method returns the following parameter:

Type	Description
NCAPReferencePoint[1)]	The specific NCAP reference point.

[1)] Refer to NCAPReferencePoint on page 66.

Related topics

References

Class Description (NCAPReferencePoints).....67

Remove

Class

NCAPReferencePoints

Syntax

```
RetVal = NCAPReferencePoints.Remove(Index)
```

Purpose

To remove an NCAP reference point.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The NCAP reference point to be removed. To specify the point, you can use the index (integer type) or a reference to the object.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (NCAPReferencePoints)..... 67

ObjectPointsSensor

Purpose

To specify the object points for point sensors.

Class Description (ObjectPointsSensor)

Syntax

```
ObjectPointsSensor = Sensors.ObjectPoints
```

Purpose

To specify the object points for point sensors.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Enabled	Boolean	To get/set the enable state.
CustomPoints	CustomPoints[1)]	To get custom points.
NCAPReflectionsPoints	NCAPReferencePoints[2)]	To get NCAP reflection points.
RadarReflectionsPoints	RadarReflectionPoints[3)]	To get radar reflections points.

1) Refer to CustomPoints on page 53.
2) Refer to NCAPReferencePoints on page 67.
3) Refer to RadarReflectionPoints on page 76.

Methods

–

Related topics

References

Sensors..... 79

Point

Purpose

To specify a point.

Class Description (Point)

Syntax

```
Point = RadarReflectionPoint.Position
Point = NCAPReferencePoint.Position
Point = CustomPoint.Position
Point = PolygonLine.Item(Index)
Point = PolygonLine.Add(X, Y)
```

Purpose

To specify a point.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
X	Double	To get/set the x coordinate of the point.
Y	Double	To get/set the y coordinate of the point.

Methods

–

Related topics

References

PolygonLine

Purpose

To specify a polygon line.

Where to go from here

Information in this section

Class Description (PolygonLine)

Syntax

```
PolygonLine = ContourLineProperties.PolygonLine
```

Purpose

To specify a polygon line.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of points in the polygon line.

Methods

The class contains the following methods:

Method	Purpose
Item	To get a specific point of the polygon line. Refer to Item on page 74.
Add	To add a point to the polygon line. Refer to Add on page 73.
Remove	To remove a point from the polygon line. Refer to Remove on page 74.

Related topics

References

ContourLineProperties.....49

Add

Class

PolygonLine

Syntax

```
Point = PolygonLine.Add(X, Y)
```

Purpose

To add a point to the polygon line.

Parameters

The method uses the following parameters:

Parameter	Type	Description
X	Double	The x coordinate.
Y	Double	The y coordinate.

Return value

The method returns the following parameter:

Type	Description
Point[1)]	The new point.

[1)] Refer to Point on page 71.

Related topics

References

Class Description (PolygonLine).....72

Item

Class

PolygonLine

Syntax

```
Point = PolygonLine.Item(Index)
```

Purpose

To get a specific point of the polygon line.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	The index of the point.

Return value

The method returns the following parameter:

Type	Description
Point[1)]	The specific point.

[1)] Refer to Point on page 71.

Related topics

References

Class Description (PolygonLine) 72

Remove

Class

PolygonLine

Syntax

```
Point = PolygonLine.Remove(Index)
```

Purpose

To remove a point from the polygon line.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	The index of the point.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (PolygonLine)..... 72

RadarReflectionPoint

Purpose

To specify a radar reflection point.

Class Description (RadarReflectionpoint)

Syntax

```
RadarReflectionPoint = RadarReflectionPoints.Item(Index)
RadarReflectionPoint = RadarReflectionPoints.Append()
```

Purpose

To specify a radar reflection point.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
ID	Integer	To get the position of the point in the list of radar reflection points.
OrientationZ	Double	To get/set the orientation in the z direction.
RadarCrossSection	Double	To get/set the radar cross section.

Attributes	Type	Purpose
OpeningAngle	Double	To get/set the opening angle.
Position	Point[1)]	To get the position.

[1)] Refer to Point on page 71.

Methods

–

Related topics

References

RadarReflectionPoints

Purpose

To manage radar reflection points.

Where to go from here

Information in this section

Class Description (RadarReflectionPoints)

Syntax

```
RadarReflectionPoints = ObjectPointsSensor.RadarReflectionPoints
```

Purpose

To manage radar reflection points.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of radar reflection points.

Methods

The class contains the following methods:

Method	Purpose
Append	To append a radar reflection point. Refer to Append on page 77.
Item	To get a specific radar reflection point. Refer to Item on page 78.
Remove	To remove a radar reflection point. Refer to Remove on page 79.

Related topics

References

ObjectPointsSensor 70

Append

Class

Class Description (RadarReflectionPoints)

Syntax

```
RadarReflectionPoint = RadarReflectionPoints.Append()
```

Purpose

To append a radar reflection point.

Parameters

–

Return value

The method returns the following parameter:

Type	Description
RadarReflectionPoint[1)]	The new radar reflection point.

1) Refer to RadarReflectionPoint on page 75.

Related topics

References

Class Description (RadarReflectionPoints)..... 76

Item

Class

Class Description (RadarReflectionPoints)

Syntax

```
RadarReflectionPoint = RadarReflectionPoints.Item(Index)
```

Purpose

To get a specific radar reflection point.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	The index of the radar reflection point.

Return value

The method returns the following parameter:

Type	Description
RadarReflectionPoint[1)]	The specific radar reflection point.

[1)] Refer to RadarReflectionPoint on page 75.

Related topics

References

Class Description (RadarReflectionPoints)..... 76

Remove

Class

Class Description (RadarReflectionPoints)

Syntax

```
RadarReflectionPoint = RadarReflectionPoints.Remove(Index)
```

Purpose

To remove a specific radar reflection point.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The radar reflection point to be removed. To specify the point, you can use the index (integer type) or a reference to the object.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (RadarReflectionPoints) 76

Sensors

Purpose

To manage sensors.

Class Description (Sensors)

Syntax

```
Sensors = TrafficObject.Sensors
```

Purpose

To manage the sensors of a traffic object.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Box	BoxProperties[1)]	To get the box sensor.
ContourLine	ContourLineProperties[2)]	To get the contour line sensor.
Custom	CustomProperties[3)]	To get the custom sensor.
ObjectPoints	ObjectPointsSensor[4)]	To get the object points of the sensor.
TrafficSign	TrafficSignProperties[5)]	To get the traffic sign sensor.
TrafficSignBasic	TrafficSignBasicProperties[6)]	To get the basic traffic sensor.

1) Refer to BoxProperties on page 47.
2) Refer to ContourLineProperties on page 49.
3) Refer to CustomProperties on page 56.
4) Refer to ObjectPointsSensor on page 70.
5) Refer to TrafficSignProperties on page 104.
6) Refer to TrafficSignBasicProperties on page 102.

Methods

–

Related topics

References

StateDependentValue

Purpose

To specify a state-dependent value.

Class Description (StateDependentValue)

Syntax

```
StateDependentValue = StateDependentValues.Item(Index)
```

Purpose

To specify state-dependent values.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Name	String	To get the name.
StateDependentValue	TrafficSignSensorData[1)]	To get the state-dependent value.

[1)] Refer to TrafficSignSensorData on page 106.

Methods

–

Related topics

References

StateDependentValues

Purpose

To manage state-dependent values.

Where to go from here

Information in this section

Class Description (StateDependentValues)

Syntax

```
StateDependentValues = TrafficSignProperties.StateDependentValues
```

Purpose

To specify the state-dependent values.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of state-dependent values.
Item	StateDependentValue[1)]	To get/set the state-dependent value.

[1)] Refer to StateDependentValue on page 80.

Methods

The class contains the following methods:

Method	Purpose
Item	To get a specific state-dependent value. Refer to Item on page 82.

Related topics

References

Item

Class

StateDependentValues

Syntax

```
StateDependentValue = StateDependentValues.Item(Index)
```

Purpose

To get a specific state-dependent value.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The index (integer type) or the name (string type) of the state-dependent value.

Return value

The method returns the following parameter:

Type	Description
StateDependentValue[1)]	The specific state-dependent value.

[1)] Refer to StateDependentValue on page 80.

Related topics

References

Class Description (StateDependentValues)..... 81

SubObjectStates

Purpose

To manage the states of a subobject.

Where to go from here

Information in this section

Class Description (SubObjectStates)..... 83
To describe the class and its attributes.

GetByName..... 84
To get subobject state values by name.

Item..... 85
To get subobject state values.

Class Description (SubObjectStates)

Syntax

```
SubObjectStates = TrafficObject.SubObjectStates
```

Purpose

To manage the subobject states.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of subobject states.

Methods

The class contains the following methods:

Method	Purpose
GetByName	To get subobject state values by name. Refer to GetByName on page 84.

Method	Purpose
Item	To get subobject state values. Refer to Item on page 85.

Related topics

References

TrafficObject.....87

GetByName

Class

SubObjectStates

Syntax

```
SubObjectStateValues = SubObjectStates.GetByName(StateName)
```

Purpose

To get subobject state values by name.

Parameters

The method uses the following parameters:

Parameter	Type	Description
StateName	String	The name of a subobject state value.

Return value

The method returns the following parameter:

Type	Description
SubObjectStateValues[1)]	The subobject state value.

[1)] Refer to SubObjectStateValues on page 85.

Related topics

References

Class Description (SubObjectStates).....83

Item

Class

SubObjectStates

Syntax

```
SubObjectStateValues = SubObjectStates.Item(Index)
```

Purpose

To get subobject state values.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The index (integer type) or the name (string type) of the subobject state values.

Return value

The method returns the following parameter:

Type	Description
SubObjectStateValues[1)]	The subobject state values.

[1)] Refer to SubObjectStateValues on page 85.

Related topics

References

Class Description (SubObjectStates).....83

SubObjectStateValues

Purpose

To specify subobject state values.

Where to go from here

Information in this section

Class Description (SubObjectStateValues).....86
To describe the class and its attributes.

Class Description (SubObjectStateValues)

Syntax

```
SubObjectStateValues = SubObjectStates.Item(Index)
SubObjectStateValues = SubObjectStates.GetByName(StateName)
```

Purpose

To specify the subobject state values.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
ActiveElement	String	To get the active subobject state value.
AvailableElements	String[]	To get all available subobject state values.
Count	Integer	To get the number of subobject state values.
Name	String	To get the name of the subobject state value.

Methods

The class contains the following methods:

Method	Purpose
Activate	To activate a subobject state value. Refer to Activate on page 86.

Related topics

References

Activate

Class

SubObjectStateValues

Syntax

```
RetVal = SubObjectStateValues.Activate(Type)
```

Purpose

To activate a subobject state value.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Type	String	The type of subobject state value to be activated.

Return value

The method returns the following parameter:

Type	Description
String	The activated subobject state value.

Related topics

References

Class Description (SubObjectStateValues)..... 86

TrafficObject

Purpose

To specify a traffic object.

Where to go from here

Information in this section

Class Description (TrafficObject)..... 87
To describe the class and its attributes.

Reload3DObject..... 88
To reload the 3-D object.

Class Description (TrafficObject)

Syntax

```
TrafficObject = TrafficObjectList.Add(Name, Object3D)
TrafficObject = TrafficObjectList.Item(Index)
```

Purpose

To specify a traffic object.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Author	String	To get/set the name of the author.
Comment	String	To get/set a comment.
Dynamic	Boolean	To get/set whether the state of the traffic object is controlled by the simulation.
Fellow	FellowProperties[1)]	To get the fellow properties.
InternalID	String	To get the internal ID.
Name	String	To get/set the name of the traffic object.
ObjectTypeId	Integer	To get/set an ID of the object type.
Orientation	Double	To get the orientation.
Object3D	String	To get the path of the referenced 3-D object.
Path	String	To get the relative path of the traffic object.
Sensors	Sensors[2)]	To get the sensors.
SubobjectStates	SubObjectStates[3)]	To get the subobject states.

1) Refer to FellowProperties on page 61.
2) Refer to Sensors on page 79.
3) Refer to SubObjectStates on page 83.

Methods

The class contains the following methods:

Method	Purpose
Reload3DObject	To reload the 3-D object. Refer to Reload3DObject on page 88.

Related topics

References

Reload3DObject

Class

Traffic Object

Syntax

```
RetVal = TrafficObject.Reload3DObject()
```

Purpose

To reload the 3-D object.

Parameters

–

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful

Related topics

References

Class Description (TrafficObject)..... 87

TrafficObjectList

Purpose

To manage a list of traffic objects.

Where to go from here

Information in this section

Class Description (TrafficObjectList)

Syntax

```
TrafficObjectList = TrafficObjectManager.TrafficObjects
```

Purpose

To manage traffic objects.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Integer	To get the number of traffic objects.
AvailableElements	String[]	To get the available traffic objects.

Methods

The class contains the following methods:

Method	Purpose
Add	To add a traffic object. Refer to Add on page 90.
Item	To get a traffic object. Refer to Item on page 91.
Remove	To remove a traffic object. Refer to Remove on page 92.

Related topics

Basics

TrafficObjectManager 92

Add

Class

TrafficObjectList

Syntax

```
TrafficObject = TrafficObjectList.Add(Name, Object3D)
```

Purpose

To add a traffic object.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Name	String	The name of the traffic object.
Object3D	String	The relative path of the 3-D object used for the traffic object.

Return value

The method returns the following parameter:

Type	Description
TrafficObject[1)]	The traffic object.

[1)] Refer to TrafficObject on page 87.

Related topics

References

Class Description (TrafficObjectList)..... 89

Item

Class

TrafficObjectList

Syntax

```
TrafficObject = TrafficObjectList.Item(Index)
```

Purpose

To get a traffic object.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The index (integer type) or name (string type) of the traffic object.

Return value

The method returns the following parameter:

Type	Description
TrafficObject[1)]	The traffic object.

[1)] Refer to TrafficObject on page 87.

Related topics

References

Class Description (TrafficObjectList)..... 89

Remove

Class

TrafficObjectList

Syntax

```
RetVal = TrafficObjectList.Remove(Index)
```

Purpose

To remove a traffic object.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Object	The traffic object to be removed. To specify the traffic object, you can use the index (integer type), the name (string type), or a reference to the object.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (TrafficObjectList)..... 89

TrafficObjectManager

Purpose

To manage traffic objects.

Where to go from here

Information in this section

Class Description (TrafficObjectManager)..... 93
To describe the class and its attributes.

AddFolder..... 94
To add a folder.

Class Description (TrafficObjectManager)

Syntax

```
TrafficObjectManager = ActiveProject.TrafficObjectManager
```

Purpose

To manage traffic objects.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Folders	FolderList[1)]	To get the list of folders.
TrafficObjects	TrafficObjectList[2)]	To get the list of traffic objects.

1) Refer to FolderList on page 62.
2) Refer to TrafficObjectList on page 89.

Methods

The class contains the following methods:

Method	Purpose
AddFolder	To add a folder. Refer to AddFolder on page 94.
AddTrafficObject	To add a traffic object. Refer to AddTrafficObject on page 95.
GetAvailable3DObjects	To get the traffic objects of a folder. Refer to GetAvailable3DObjects on page 96.
GetFolder	To get a folder. Refer to GetFolder on page 97.
GetTrafficObject	To get a traffic object. Refer to GetTrafficObject on page 97.
MoveFolder	To move a folder. Refer to MoveFolder on page 98.
MoveTrafficObject	To move a traffic object. Refer to MoveTrafficObject on page 99.
RemoveFolder	To remove a folder. Refer to RemoveFolder on page 100.
RemoveTrafficObject	To remove a traffic object. Refer to RemoveTrafficObject on page 100.
Save	To save the contents of the traffic object manager. Refer to Save on page 101.

Related topics

References

Project (ModelDesk Project and Experiment Management 🕮)

AddFolder

Class

TrafficObjectManager

Syntax

```
Folder = TrafficObjectManager.AddFolder(Name)
```

Purpose

To add a folder.

Description

Using the `AddFolder` method of the `TrafficObjectManager` class allows you to create folders and subfolders in one step. For example, to create a folder `Chassis` and a subfolder `Cars`, use the following code:

```
TOM.AddFolder('Chassis\\Cars')
```

Parameters

The method uses the following parameters:

Parameter	Type	Description
Name	String	The name of the folder.

Return value

The method returns the following parameter:

Type	Description
Folder[1)]	The folder object.

[1)] Refer to Folder on page 61.

Related topics

References

Class Description (TrafficObjectManager) 93

AddTrafficObject

Class

TrafficObjectManager

Syntax

```
TrafficObject = TrafficObjectManager.AddTrafficObject(Name, Object3D)
```

Purpose

To add a traffic object.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Name	String	The relative path (optional) and the name of the traffic object. If `Name` contains a relative path, the traffic object is created in the specified folder. For example, `'MyFolder\\MyNewTrafficObject'` creates the new traffic object in `MyFolder`. If `Name` contains only the name of the traffic object, the traffic object is created in the root folder.
Object3D	String	The relative path to the 3-D object that should be assigned to the traffic object.

Return value

The method returns the following parameter:

Type	Description
TrafficObject[1)]	The traffic object.

[1)] Refer to TrafficObject on page 87.

Related topics

References

Class Description (TrafficObjectManager) 93

GetAvailable3DObjects

Class

TrafficObjectManager

Syntax

```
RetVal = TrafficObjectManager.GetAvailable3DObjects(Folder)
```

Purpose

To get all available 3-D objects of a folder.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Folder	String	The folder which contains the 3-D objects.

Return value

The method returns the following parameter:

Type	Description
String[]	An array with the relative paths of the 3-D objects contained in the specified folder.

Related topics

References

Class Description (TrafficObjectManager) 93

GetFolder

Class

TrafficObjectManager

Syntax

```
Folder = TrafficObjectManager.GetFolder(Name)
```

Purpose

To get a folder.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Name	String	The name of the folder.

Return value

The method returns the following parameter:

Type	Description
Folder[1)]	The folder object.

1) Refer to Folder on page 61.

Related topics

References

Class Description (TrafficObjectManager).....93

GetTrafficObject

Class

TrafficObjectManager

Syntax

```
TrafficObject = TrafficObjectManager.GetTrafficObject(ID)
```

Purpose

To get a traffic object.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ID	Object	A reference to the traffic object. To specify the traffic object, you can use the object type ID (integer type) or the relative path (string type).

Return value

The method returns the following parameter:

Type	Description
TrafficObject[1)]	The traffic object.

[1)] Refer to TrafficObject on page 87.

Related topics

References

Class Description (TrafficObjectManager) 93

MoveFolder

Class

TrafficObjectManager

Syntax

```
RetVal = TrafficObjectManager.MoveFolder(SourcePath, DestinationPath)
```

Purpose

To move a folder.

Parameters

The method uses the following parameters:

Parameter	Type	Description
SourcePath	Object	The source path.
DestinationPath	Object	The destination path.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (TrafficObjectManager) 93

MoveTrafficObject

Class

TrafficObjectManager

Syntax

```
RetVal = TrafficObjectManager.MoveTrafficObject(SourcePath,
DestinationPath)
```

Purpose

To move a traffic object.

Parameters

The method uses the following parameters:

Parameter	Type	Description
SourcePath	Object	The source path.
DestinationPath	Object	The destination path.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (TrafficObjectManager) 93

RemoveFolder

Class

TrafficObjectManager

Syntax

```
RetVal = TrafficObjectManager.RemoveFolder(Name)
```

Purpose

To remove a folder.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Name	Object	The folder to be removed. To specify the folder, you can use the name (string type) or a reference to the object.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (TrafficObjectManager).....93

RemoveTrafficObject

Class

TrafficObjectManager

Syntax

```
RetVal = TrafficObjectManager.RemoveTrafficObject(ID)
```

Purpose

To remove a traffic object.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ID	Object	The traffic object to be removed. To specify the traffic object, you can use the object type ID (integer type), the relative path (string type), or a reference to the object.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (TrafficObjectManager) 93

Save

Class

TrafficObjectManager

Syntax

```
RetVal = TrafficObjectManager.Save()
```

Purpose

To save the contents of the traffic object manager.

Parameters

–

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Related topics

References

Class Description (TrafficObjectManager) 93

TrafficSignBasicProperties

Purpose

To specify the properties for a traffic sign basic sensor.

Class Description (TrafficSignBasicProperties)

Syntax

```
TrafficSignBasicProperties = Sensors.TrafficSignBasic
```

Purpose

To specify the properties for a traffic sign basic sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Enabled	Boolean	To get/set the enable state.
Encoding	TrafficSignBasicSensorEncoding[1)]	To get the encoding of the traffic sign.
Category	TrafficSignTypes[2)]	To get/set the category of the traffic sign.
SensorValues	TrafficSignBasicSensorValues[3)]	To get the properties of the traffic sign for the sensor.

1) Refer to TrafficSignBasicSensorEncoding on page 102.
2) Refer to TrafficSignTypes on page 112.
3) Refer to TrafficSignBasicSensorValues on page 103.

Methods

–

Related topics

References

Sensors.....79

TrafficSignBasicSensorEncoding

Purpose

To specify the encoding of a basic traffic sign sensor.

Class Description (TrafficSignBasicSensorEncoding)

Syntax

```
TrafficSignBasicSensorEncoding = TrafficSignBasicProperties.Encoding
```

Purpose

To specify the encoding of a basic traffic sign sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Category	Integer	To get the category.
Identifier	Integer	To get the identifier.
Value	Double	To get the value.
Visibility	Integer	To get the visibility.

Methods

–

Related topics

References

TrafficSignBasicProperties..... 102

TrafficSignBasicSensorValues

Purpose

To specify the values for the sensor of a basic traffic sign.

Class Description (TrafficSignBasicSensorValues)

Syntax

```
TrafficSignBasicSensorValues =
TrafficSignBasicProperties.SensorValues
```

Purpose

To specify the values for the sensor of a basic traffic sign.

Attributes

The class contains the following attributes. The available attribute depends on the selected category of the traffic sign basic sensor:

Attributes	Type	Purpose
TrafficSignSensorPriorityValues	TrafficSignSensorPriorityValues[1)]	To get the properties of a traffic sign of the priority category.
TrafficSignSensorSpeedLimitValues	TrafficSignSensorSpeedLimitValues[2)]	To get the properties of a traffic sign of the speed limit category.
TrafficSignSensorNoOvertakingValues	TrafficSignSensorNoOvertakingValues[3)]	To get the properties of a traffic sign of the no overtaking category.
TrafficSignSensorCustomValues	TrafficSignSensorCustomValues[4)]	To get the properties of a traffic sign of the custom category.

1) Refer to TrafficSignSensorPriorityValues on page 108.
2) Refer to TrafficSignSensorSpeedLimitValues on page 109.
3) Refer to TrafficSignSensorNoOvertakingValues on page 108.
4) Refer to TrafficSignSensorCustomValues on page 105.

Methods

–

Related topics

References

TrafficSignBasicProperties..... 102

TrafficSignProperties

Purpose

To specify the properties for a traffic sign sensor.

Class Description (TrafficSignProperties)

Syntax

```
TrafficSignProperties = Sensors.TrafficSign
```

Purpose

To specify the proerpties for a traffic sign sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Enabled	Boolean	To get/set the enable state.
UseStateDependentValues	Boolean	To get/set the flag that state-dependent values are used.
StateIndependentValue	TrafficSignSensorData[1)]	To get/set the state-independent value.
StateDependentValues	StateDependentValues[2)]	To get/set the state-dependent values.

1) Refer to TrafficSignSensorData on page 106.
2) Refer to StateDependentValues on page 81.

Methods

–

Related topics

References

Sensors.....79

TrafficSignSensorCustomValues

Purpose

To specify custom values of a traffic sign sensor.

Class Description (TrafficSignSensorCustomValues)

Syntax

```
TrafficSignSensorCustomValues =
TrafficSignBasicSensorValues.TrafficSignSensorCustomValues
```

Purpose

To specify custom values of a traffic sign sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Identifier	Integer	To get/set the identifier.
Value	Double	To get/set the value.
Visibility	Integer	To get/set the visibility.

Methods

–

Related topics

References

TrafficSignBasicSensorValues..... 103

TrafficSignSensorData

Purpose

To specify data of a traffic sign sensor.

Class Description (TrafficSignSensorData)

Syntax

```
TrafficSignSensorData = TrafficSignProperties.TrafficSign
```

Purpose

To specify data of a traffic sign sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Sensor	TrafficSignSensorValues[1)]	To get the values of the traffic sign sensor.
Encoding	TrafficSignSensorEncoding[2)]	To get the encoding of the traffic sign sensor.

1) Refer to TrafficSignSensorValues on page 110.
2) Refer to TrafficSignSensorEncoding on page 107.

Methods

–

Related topics

References

TrafficSignProperties..... 104

TrafficSignSensorEncoding

Purpose

To specify the encoding of a traffic sign sensor.

Class Description (TrafficSignSensorEncoding)

Syntax

```
TrafficSignSensorEncoding = TrafficSignSensorData.Encoding
```

Purpose

To specify the encoding of a traffic sign sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Country	Integer	To get the encoding of the country.
State	Integer	To get the encoding of the state.
Visibility	Integer	To get the encoding of the visibility.

Methods

–

Related topics

References

TrafficSignSensorData..... 106

TrafficSignSensorNoOvertakingValues

Purpose

To specify the values of a traffic sign sensor of the no overtaking category.

Class Description (TrafficSignSensorNoOvertakingValues)

Syntax

```
TrafficSignSensorNoOvertakingValues =
TrafficSignBasicSensorValues.TrafficSignNoOvertakingValues
```

Purpose

To specify the values of a traffic sign sensor of the no overtaking category.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Identifier	Integer	To get/set the identifier.
Type	NoOvertakingTypes[1)]	To get/set the type of the traffic sign.
Visibility	Integer	To get/set the visibility.

[1)] Refer to NoOvertakingTypes on page 111.

Methods

–

Related topics

References

TrafficSignBasicSensorValues..... 103

TrafficSignSensorPriorityValues

Purpose

To specify the values of a traffic sign sensor of the priority category.

Class Description (TrafficSignSensorPriorityValues)

Syntax

```
TrafficSignSensorPriorityValues =
TrafficSignBasicSensorValues.TrafficSignSensorPriorityValues
```

Purpose

To specify the values of a traffic sign sensor of the priority category.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Identifier	Integer	To get/set the identifier of the traffic sign.
Type	PriorityTypes[1)]	To get/set the type of the traffic sign.
Visibility	Integer	To get/set the visibility of the traffic sign.

[1)] Refer to PriorityTypes on page 111.

Methods

–

Related topics

References

TrafficSignSensorSpeedLimitValues

Purpose

To specify the values of a traffic sign sensor of the speed limit category.

Class Description (TrafficSignSensorSpeedLimitValues)

Syntax

```
TrafficSignSensorSpeedLimitValues =
TrafficSignBasicSensorValues.TrafficSignSensorSpeedLimitValues
```

Purpose

To specify the values of a traffic sign sensor of the speed limit category.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Identifier	Integer	To get/set the identifier of the traffic sign.
Type	SpeedLimitTypes[1)]	To get/set the type of the traffic sign.
Value	Integer	To get/set the value of the speed limit traffic sign.
Visibility	Integer	To get/set the visibility of the traffic sign.

1) Refer to SpeedLimitTypes on page 112.

Methods

–

Related topics

References

TrafficSignBasicSensorValues..... 103

TrafficSignSensorValues

Purpose

To specify the values of a traffic sign sensor.

Class Description (TrafficSignSensorValues)

Syntax

```
TrafficSignSensorValues = TrafficSignSensorData.Sensor
```

Purpose

To specify the values of a traffic sign sensor.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Country	String	To get/set the country of the traffic sign.
SubType	Integer	To get/set the subtype of the traffic sign.
Text	String	To get/set the text of the traffic sign.
Type	Integer	To get/set the type of the traffic sign.
Value	Double	To get/set the value of the traffic sign.

Attributes	Type	Purpose
Visibility	Integer	To get/set the visibility of the traffic sign.

Methods

–

Related topics

References

TrafficSignSensorData 106

Constants for the Traffic Object Manager

Constants for the Traffic Object Manager

Introduction

You can use predefined constants in the tool automation.

Constants

The following constants exist to automate the Traffic Object Manager.

NoOvertakingTypes The following constants are used to specify the traffic sign of the no overtaking type:

Value	Description
Begin = 0	Begin no overtaking traffic sign.
End = 1	End no overtaking traffic sign.
EndOfAllLimits = 2	End of all limits traffic sign.

PriorityTypes The following constants are used to specify the traffic sign of priority types:

Value	Description
Stop = 0	Stop traffic sign.
Yield = 1	Yield traffic sign.
PriorityRoad = 2	Priority road traffic sign.
EndOfPriorityRoad = 3	End of priority road traffic sign.

SpeedLimitTypes The following constants are used to specify the traffic sign of the speed limit type:

Value	Description
Begin = 0	Begin speed limit traffic sign.
End = 1	End speed limit traffic sign.
EndOfAllLimits = 2	End of all speed limits traffic sign.

TrafficSignTypes The following constants are used to specify the category of the traffic sign:

Value	Description
Priority = 0	Priority traffic sign.
SpeedLimit = 1	Speed limit traffic sign.
NoOvertaking = 2	No overtaking traffic sign.
Custom = 3	Custom traffic sign.

Numerics

B

C

D

L

O

P

S

T