TRAFFIC > MOBILITY > LOGISTICS

ptv vision

VISSIM5.20-05
COM 接口用户手册

traffic mobility logistics.

目录

1 介绍

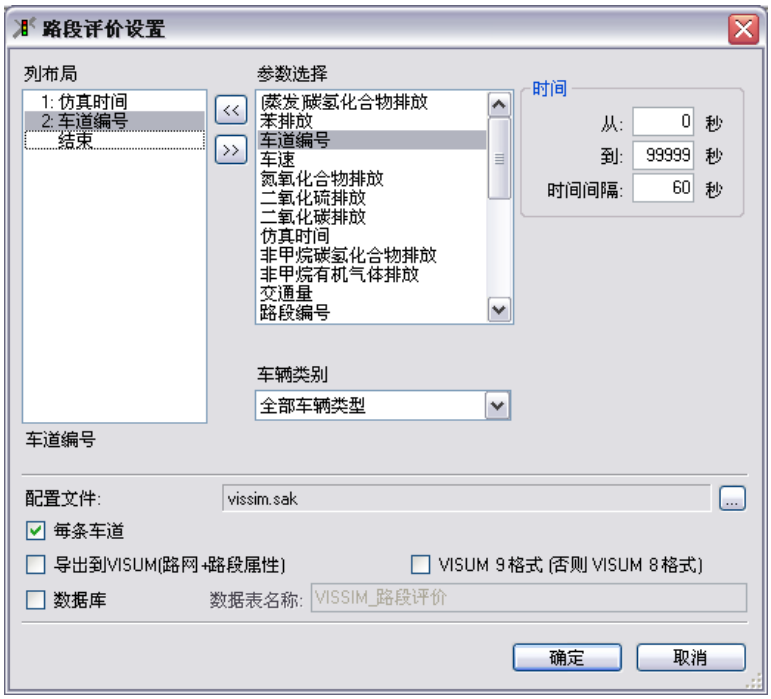
VISSIM 是分析许多交通问题的一个强大的软件工具。有时候，项目要对其本身进行广泛的前期、后期调查或对大量方案进行研究。在这种情况下，VISSIM 应用于其它应用程序，作为一个进行交通规划计算的工具箱。模型的数据和仿真能够通过 COM 界面得到，这时 VISSIM 类似于自动化服务器，能够导出该文件中描述的对象、方法和特征属性。VISSIM COM 界面支持 Microsoft Automation，用户能够应用所有的 RAD(Rapid Application Development)的工具，包括 Visual Basic Script、Java Script 等脚本语言，或 Visual C++，Visual J++等编程环境。这本手册中展示的例子主要是在 Visual Basic 下进行，218 页也有一些相关介绍。在 236 页，用户也能找到关于在 Visual C++下进行操作的简略介绍。

下面我们将通过一个 Visual Basic 的示例来了解这种方法的灵活性。假设我们要用几个不同的随机种子来进行多次仿真，分别得到它们的路段评价。用户可以应用 Excel 和 VBA(Visual Basic for Applications)来加载路网，设置不同的随机种子，然后开始运行。比如：用户想要用路网“fixed_time.inp”，针对四组不同的随机种子：10，20，30 和 42，那么用户可以编辑如下所示的 Excel 表格：

random seed	simulation file
10	G:\PTV\DATA\FIXED_TIME.INP
20	
30	
42	

用户可以插入一个“START”按钮(从工具栏中)，来连接该章节结尾处的 Visual Basic 编码：

用户如要通过运行 Excel 宏来得到路段评价，必须先要在同一目录下创建*.SAK 文件。用户可以用 *VISSIM 路段评价对话框*(或 LinkEvaluation 接口)来做这步操作：



用户同时还需在“评价→文件”对话框中对“路段评价”一栏进行勾选，并将设置保存在*.INI 文件中，例如，在“查看”菜单下点击“保存设置”，将配置文件保存为 LINK_EVAL.INI。

1.1 介绍示例

下面我们将会介绍用于仿真的 Visual Basic 编码，它会运行四次仿真，读取 Excel 表格来得到路网文件名和各自的随机种子。程序中我们将编码被定义为 RandomSeed2VISSIM 的一个程序单元，并设置为在 Excel 表格中 Start 按钮的“OnClick”命令下触发。以引号(')开头的语句是用于解释程序的注释行。

```
SUB RandomSeed2VISSIM()
' Declare VISSIM COM types for a Vissim and a Simulation object
DIM Vissim AS Vissim
DIM Simulation AS Simulation
' Declare further types
DIM SimulationFile AS String ‘ Name of the network file (with the full path)
DIM RandomSeed AS Integer ‘ Current random seed
DIM RunIndex AS Integer ‘ Simulation running index
' Start Vissim and create an instance of a Vissim object and a Simulation object
SET Vissim = CreateObject("VISSIM.Vissim")
SET Simulation = Vissim.Simulation
‘ Get the network file name
Sheets("VISSIM").Select ‘ Select the example sheet named VISSIM
Range("C2").Select ‘ Select the cell with the network file name
SimulationFile = Selection.Value ‘ Get the network file name
' Load the network and the *.ini file
Vissim.LoadNet SimulationFile
Vissim.LoadLayout "link_eval.ini"
' Initialize simulation values
Simulation.Period = 100 ‘ 100 second simulations
Simulation.Resolution = 1 ‘ 1 step per second resolution
RunIndex = 0 ‘ Simulation running index initialization
' Loop of simulation runs
Range("A2").Select ‘ Select the first random seed cell
WHILE Selection <> "" ‘ Run until no more random seeds are available
‘ Get current random seed from the current selected cell
RandomSeed = Selection.Value
‘ Set simulation parameters for next simulation
Simulation.RunIndex = RunIndex ‘ Set the simulation run index
Simulation.Comment = "Random Seed = " & RandomSeed ‘ Set simulation comment
Simulation.RandomSeed = RandomSeed ‘ Set random seed
‘ Run simulation continuously
```

```
Simulation.RunContinuous
‘ Initilize next simulation values
ActiveCell.Offset(1, 0).Select ‘ Select the next random seed cell
RunIndex = RunIndex + 1 ‘ Next simulation running index
WEND
END SUB
```

在创建了一个 VISSIM 对象后，编码演示了如何获得其他 VISSIM 数据对象的相关信息(在这个示例下是一个 Simulation 对象)，怎样得到其属性和方式。

类似的，用户也可以通过 Visual Basic 编程环境创建一个 desktop 应用程序，控制仿真运行，并改变随机种子和其他的一些仿真参数。

VISSIM 在 COM 中自动启动前需要在 Windows 系统下进行注册，这样它可以通过一个 COM client 自动注册。用户可以按照手册 11 页所示手动进行注册，或者只要通过运行 VISSIM 一次，程序就会自动注册。此外，VISSIM COM Library 的引用信息能够在编程环境中通过提前绑定来进行设置，以此来帮助编码的编写和提升性能。在 Excel 的 Visual Basic 编辑器下，可以通过菜单“工具 – 引用”来完成。

如果用户采用提前绑定的方式(有关更多信息参见234页)，VISSIM COM server type library 的相关信息必须要在VB的编程环境下设置。 当用户进行VISSIM的更新版本安装后，由于新版本会自动改变该type library，因而可能会引发一些冲突。重新选择VISSIM COM Server中的指引信息可以强制让VB重新解释type library。请参考262页的附录“提示与技巧”， 得到更多关于VISSIM版本和其COM服务器界面的信息。

1.2 许可和注册

如果要使用 VISSIM COM Server，用户有必要使用正确的 VISSIM 许可。当 VISSIM 安装后，许可将自动激活所有 COM 界面的注册程序，且支持在其他编程环境下实例化和使用 VISSIM COM 的对象，如 Visual Basic 和 Visual C++ 。

用户也可以通过如下的启动命令对 Vissim COM 的服务进行手动注册：

VISSIM -RegServer
VISSIM -UnregServer

在 Vista 操作系统中，用户可以通过 Windows 开始菜单中的“Register COM Server”项目来获取需要的管理员权限。

用以上两个启动参数打开 VISSIM 不会启动 VISSIM 的主窗口。它们会悄悄地注册/取消注册 VISSIM COM Server 和它的接口。

如果“注册/取消注册”失败，将会弹出一个对话框信，显示错误信息(具体的错误信息被写在 VISSIM.ERR 文件)。

用户可以在附录中找到更多关于使用 VISSIM COM Server 的必要注册信息的条目。

1.3 示例

毫无疑问的，用户比较习惯简单启动 VISSIM 来进行仿真。但是在 COM 应用程序下操作的话，VISSIM 的行为将有所改变，具体表现如下所示：

- 如果同时运行一个或多个 VISSIM 例子，那么 COM 的应用会自动与第一个打开的 VISSIM 例子相联。
- 如果此时没有 VISSIM 例子在运行，将自动创建一个新的 COM 对象，激活一个新的 VISSIM 文件(参见 17 页，怎样创建一个新的 *VISSIM* 对象)。
- 在默认模式下，每一个 VISSIM 例子只能作为一个 COM 应用程序的服务器。如果有多个 COM 应用程序打开，那么每个都将打开一个新的 VISSIM 例子，用户可以通过 *Automation* 模式(见下面内容)来保证两个或两个以上的 COM 程序不同时使用同一个 VISSIM 例子。

在某些情况下，允许一个 VISSIM 例子成为多个 COM 应用程序的服务器是非常有用的。用户可以通过如下参数控制 VISSIM 的行为：

- *Embedded*：通过这个参数用户可以启动一个 VISSIM 例子，发生的动作完全符合上面所述的现象(默认行为)。一个 VISSIM 例子只能作为一个 COM 应用程序的服务器，除此以外，当 COM 对象释放后或 COM 客户端关闭后，该 VISSIM 例子都将自动关闭。
- *Automation*：通过这个参数每个 VISSIM 文件可以作为所有 COM 应用程序的服务器，直到该 VISSIM 例子被手动关闭，或者 COM 客户端被手动关闭。

当COM客户端与一个VISSIM事件相连时，用户不能启动其他的COM应用程序。*Automation*服务器只有在用户不同时运行几个COM应用程序下才有用。

当用户的电脑安装了几个版本的 VISSIM 程序后，也可以实例化某个确定版本 VISSIM 对象。要实现这个目的，请参见第 17 页的 VISSIM 对象这一章。

1.4 规定

该说明手册按照统一的格式来描述对象和界面，用户在阅读手册之前应该了解一下的一些规定：

► 对象模型

有两种对象的名称。一种是单数名词，表示的是 VISSIM 中的单个条目或元素，如：Vissim，Net，Simulation，Link，SignalHead,…。另一种是复数名词形式的名称，往往表示一组元素的集合(集合，列表，数组)，比如 Links，SignalHeads，…。

► 属性和方法的描述：

每个接口的属性和方法的完整描述和示例都被清楚无误的表述出来。下面是一个关于接口属性和参数类型说明的范例：

[in] :	input parameter
[out, retval] :	output parameter
unsigned char :	A 8-bit integer type (0 to 255)
long :	A 32-bit integer type (-2,147,483,648 to 2,147,483,647)
double:	A 64-bits floating point type (-1.7E308 to 1.7E308)
BSTR:	A 16-bit string type
VARIANT:	A data type capable of representing various different types.
IObjekt:	The interface of a VISSIM COM object; for example ILink

相关类型(pointer types 或 simple points)都被标上了星号(＊)，一个相关类型就是指某一类型的一个特定的例子。如：long*， ILink*，VARIANT*。

参见IDL(Interface Definition Language)的描述，可得到更多信息。

在接口中不用int 类型，因为在Visual Basic中语言映射过长，而且在不同平台中数据长度会发生改变。

Visual Basic的例子：

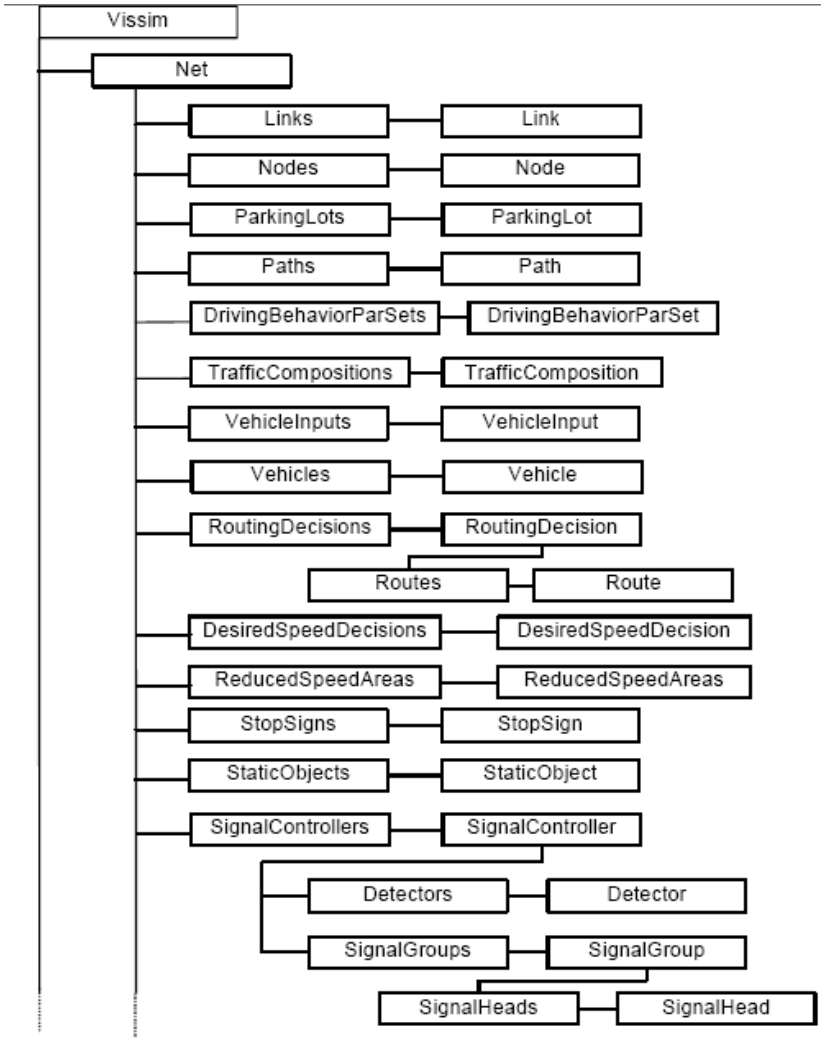
该手册中所有示例都是在 Visual Basic 平台下编写(除了第 236 页的部分基于 Visual C++)。程序的关键词都用大写字母，如：DIM， SET，FOR EACH … NEXT， … . ，见后续章节的程序示例。

2 对象模型

VISSIM COM 对象模型有着一个严格的对象等级。用户如果要进入不同的低等级对象，如：一个 Net 对象中的一个 Link 对象，就必须按照这个等级来进行操作。VISSIM 是最高等级的对象；其他的对象都是它的子对象。通过下面的图，用户可以了解一些对象间的实体相互关系(用户可以在第 15 页看到完整的等级示意图)。

集合是一个特殊的对象类型；它是单个对象们的容器，可用来枚举路网元素。基于这个原则，它们的名称都是以复数形式出现，比如：Links 和 Vehicles 这两个对象。Visual Basic 为集合对象提供了专门的语句要素 For Each…Next，以便在该集合中对对象进行依次的调用。用户可在第 230 页看到更多细节信息。

模型框架一览

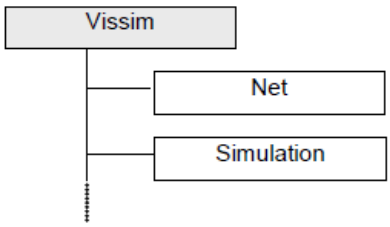

Vissim
Net
Links
Link
Nodes
Node
ParkingLots
ParkingLot
Paths
Path
DrivingBehaviorParSets
DrivingBehaviorParSet
TrafficCompositions
TrafficComposition
VehicleInputs
VehicleInput
Vehicles
Vehicle
RoutingDecisions
RoutingDecision
Routes
Route
DesiredSpeedDecisions
DesiredSpeedDecision
ReducedSpeedAreas
ReducedSpeedAreas
StopSigns
StopSign
StaticObjects
StaticObject
SignalControllers
SignalController
Detectors
Detector
SignalGroups
SignalGroup
SignalHeads
SignalHead

3 语句参考

3.1 基本对象

3.1.1 Vissim

Vissim 是模型中最高等级的对象；其他所有对象都隶属于它，并且只能通过 IVISSIM 接口进行调用。在创建完 Vissim 对象后，COM 应用程序自动进入第一个 VISSIM 实例运行，如果此时没有任何实例，那么系统将会自动创建一个新的 INP 文件(见第 12 页)。



IVISSIM 接口也允许加载路网，创建第二等级的模型对象，如：Net 对象和 Simulation 对象。

举例：

► 创建一个 Vissim 对象：

```
DIM vissim As Vissim
SET vissim = NEW Vissim
```

- 还有另外一种办法，就是使用 CreatObject 函数(VBScript 程序独有)，也可以用来生成一个 Vissim 对象，需要使用的标识编号为“VISSIM.Vissim.520”，可以具体打开 VISSIM 的某一个版本的例子（如果该台计算机上不止安装了一个版本的 Vissim 的话）。大写的 VISSIM 表示的是 VISSIM-COM 服务器，而小写的 Vissim 表示的是 Vissim 这个类型的对象，520 表示的是 VISSIM 的版本：

```
SET vissim=CreateObject (“VISSIM.Vissim.520”)
```

- 32 位或 64 位的版本（如果该计算机上同时安装了两个版本）可以通过分别附加在第二个 Vissim 字段后面的“-32”或“-64”字段来加以识别。

```
SET vissim = CreateObject ("VISSIM.Vissim-32.520”)  or
SET vissim = CreateObject ("VISSIM.Vissim-32")
```

- 如果用户安装和注册的是比 VISSIM4.30 更早的版本，那么用户可以用字符串“VISSIM. Vissim.1”来进行实例化。
- 如果在该过程中用户省略了版本号，那么系统将使用最新注册的版本。

```
SET vissim=CreateObject (“VISSIM.Vissim”)
```

用户可以参考第233页的VisualBasic说明，得到更多关于NEW和CreateObject的区别的说明。

► 删除一个 Vissim 对象

在 Visual Basic 和 VBS 脚本程序中，用户可以通过关键词“Nothing”来达到删除对象的目的。

```
SET Vissim=NOTHING
```

在这种情况下，如果用户使用的是 Automation 模式，那么 VISSIM 应用程序不会自动关闭(与下面的 Exit 方法作对比)。而在 Embedded 模式下，VISSIM 应用程序会自动关闭，和调用 vissim.Exit()效果一致。

VISSIM的工作目录能否设制为系统目录取决于用户的操作环境(如：Visual Basic或Excel)。如果该目录下有一个vissim.ini文件的话，那么它可以用于初始化(如：窗口位置，视图、以及评价的设置)。

IVISSIM 接口的属性

Net([out， retval] INet **ppNet)

实例化一个 Net 路网对象，可以进入到路网的功能(见 21 页)。

参数

[out, retval] INet **ppNet: 返回一个 Net 对象。

例子

```
DIM net As Net
SET net=vissim.Net
```

Simulation([out， retval] ISimulation **ppSimulation)

实例化一个 Simulation 对象，可以进入到仿真的功能(见 169 页)。

参数

[out, retval] ISimulation **ppSimulation: 返回一个 Simulaiton 对象。

例子

```
DIM simulaition As Simulation
SET simulaition = vissim.Simulation
```

Graphics([out， retval] IGraphics **ppGraphics)

实例化一个 Graphic 对象，可以进入到图形选项功能(见 177 页)。

参数

[out, retval]IGraphics**ppGraphics: 返回一个 Graphics 对象。

例子

```
DIM graphics AS Graphics
```

SET graphics=vissim.Graphics

Evaluation([out， retval] IEvaluation **ppEvaluation)

实例化一个 Evaluation 对象，可以进入评价选项功能(见 186 页)。

参数

[out, retval]IEvaluation**ppEvaluation:返回一个 Evaluation 对象

例子

DIM evaluation AS Evaluation

SET evaluation = Vissim.Evaluation

AttValue([in] BSTR Attribute, [out, retval] VARIANT*pValue)

返回一个 VISSIM 的一般选项。可以在下面（本节结束）的表中，了解各个属性与所选语言无关的标识。

参数

[in] BSTR Attribute： 属性的名称（见下）

[out, retval] VARIANT*pValue：返回该属性的值

例子

distance_unit1=vissim.AttValue(“UNITDISTANCE1”)

AttValue([in]BSTR Attribute, [in]VARIANT)

设置一个 VISSIM 的一般选项。可以在下面（本节结束）的表中，了解各个属性与所选语言无关的标识。

参数

[in] BSTR Attribute： 属性的名称（见下）

[in] VARIANT Value： 新的属性值。（类型根据属性）

例子

vissim.AttValue(“UNITDISTNACE1”)=1 ‘设置 Vissim 以英寸为单位

属性览表

R	W	属性	描述
√	√	MENU	启用/不启用主菜单。
√		REVISION	文本格式的 VISSIM revision 编号
√		INPUTFILE	当前加载的输入文件名称
√	√	WORKINGFOLDER	当前的工作目录
√		EXEFOLDER	VISSIM 开始的 EXE 文件夹
√	√	TOOLBAR	启用/不启用除缩放外其他所有工具栏(文件，选择，运行控制，路网元素，动画，测试和

			仿真)。
√		VERSION	文本格式的 VISSIM 版本。
√	√	UNITDISTANCE1	0=[m]， 1=[ft]
√	√	UNITDISTANCE2	0=[km]， 1=[mi]
√	√	UNITSPEED	0=[km/h]，1=[mph]
√	√	UNITACCEL	0=[m/s^2]，1=[ft/s^2]

IVISSIM 接口的方法

New()

创建一个新的空路网。

例子

```
SET vissim=NEW Vissim
vissim.New
```

LoadNet([in, defaultvalue(“”)]BSTR NetPath, [in,defaultvalue(0)] BYTE Additive)

根据字符串 NetPath 的路径信息加载 VISSIM 路网。

参数

[in] BSTR NetPath：将读入路径+文件名（*.inp）的路网文件。该参数是可选的。如果没有输入路径，将会弹出一个文件浏览窗口。

[in] BYTE Additve：如果该值不为 0，则该文件将以附加读取的形式读入 VISSIM 中。该参数是可选的，默认值为 0（即非附加读取的模式）。

例子

```
SET vissim=NEW Vissim
vissim.LoadNet”c:vissim\data\example.inp”
vissim.LoadNet”c:\vissim\data\example_bis.inp,1   ‘附加读取
```

SaveNet()

以相同的文件名保存路网文件。

例子

```
SET vissim=NEW Vissim
vissim.LoadNet ”c:\vissim\data\example.inp”
vissim.SaveNet
```

SaveNetAs([in，defaultvalue(“”)] BSTR NetPath)

根据指定字符串 NetPath 的路径和名字另存该路网文件。

参数

[in] BSTR NetPath：输入另存该路网文件的路径+文件名（*.inp）。如果没有输入或者没有路径信息，将显示出一个文件浏览窗口。

例子

SET vissim=NEW Vissim

Vissim.LoadNet “c:\vissim\data\example.inp”

Vissim.SaveNetAs “c:\vissim\data\example_bis.inp”

ImportANM ([in, defaultvalue(“”)] BSTR NetPath, [in, defaultvalue(“”)] BSTR RoutesPath, [in, defaultvalue(“”)] BSTR InputPath, [in, defaultvalue(ImportInitial)] ImportType, [in, defaultvalue(0xFF)] int importOptions, [in, defaultvalue(600)] int evaluationInterval)

导入一个 ANM 格式的文件，以初始化模式或以附加读取的模式导入到 VISSIM 中。

参数

[in] BSTR NetPath：需要读取的路网文件的路径+文件名（*.anm）。

[in] BSTR RoutesPath：需要读取的路径文件的路径+文件名（*.anmRoutes）。

[in] BSTR InputPath：输入的结果需要以该路径+文件名（*.inp）来保存。如果在未来，需要再额外输入一个文件或者需要创建一个用于动态交通分配的路径信息的话，需要设置该参数。

[in] ImportType：此次输入的形式是初始化输入，还是以 ANM Adaptive 的方式输入。

[in] int InportOptions：更多的输入选项。

[in] int evaluationInterval ：评估区间。如果路径信息需要提供给动态交通分配的话，这个选项是必须要输入的。

例子

SET vissim=NEW Vissim

Vissim.ImportANM “c:\vissim\data\example.anm”, ”c:\vissim\data\example.anmRoutes”, “c:\vissim\data\example.inp”, ImportInitial, ImportForDynAssign, 900

importType属性览表：

属性	描述
ImportInitial	初始化导入
ImportAdditive	在保持原来对象的基础上附加导入
ImportDifference	在删去原来对象的基础上附加导入

importOptions属性览表：

属性	描述
ImportForDynAssign	输入的路径信息提供给动态交通分配
ImportNodeRoutes	输入的路径信息提供给节点路径
ImportForceRoutingImport	强制导入路径信息，即使*.anmRoutes 文件没有更改

LoadLayout([in，defaultvalue(“”)] BSTR LayoutPath)

根据字符串 LayoutPath 的路径信息，读入一个 VISSIM 的配置文件（*.INI）。如果没有输入或者输入了一个空的路径参数，则将会出现文件浏览对话框。

参数

[in] BSTR LayoutPath：需要读入的配置文件的路径+文件名（*.ini）

例子

```
SET vissim= NEW Vissim
vissim.LoadLayout “c:\vissim\data\example.ini”
```

SaveLayout([in, defaultvalue(“”) BSTR LayoutPath])

把当前的 VISSIM 设置文件（*.INI）保存在字符串 LayoutPath 定义的路径里。如果是该参数为空，将会使用默认的文件名“vissim.ini”。如果没有合适的路径，将会出现文件浏览对话框。

参数

[in] BSTR LayoutPath：路径+文件名（*.ini）

例子

```
SET vissim =NEW Vissim
vissim.SaveLayout “c:\vissim\data\example.ini”
```

Exit()

退出 VISSIM 程序，同时删除 Vissim 对象实例（在 Automation 模式下也是一样，参见第 12 页）。

例子

```
SET vissim=NEW Vissim
vissim.Exit
```

ShowMaximized()

VISSIM 的主窗口最大化显示。

ShowMinmized()

VISSIM 的主窗口只显示在任务栏里。

GetWindow([out] VARIANT*Top, [out] VARIANT*Left, [out] VARIANT*Bottom, [out] VARIANT*Right)

提供 VISSIM 主窗口的位置。所给的屏幕坐标的尺寸是相对于屏幕的左上角的位置的。

参数

[out] VARIANT *Top：　　屏幕顶部边界的坐标值

[out] VARIANT *Left：　　屏幕左边界的坐标值

[out] VARIANT *Bottom：　　屏幕底部边界的坐标值

[out] VARIANT *Right：　　屏幕右边界的坐标值

例子

```
SET vissim= NEW Vissim
vissim.GetWindow top, left, bottom, right
```

SetWindow([in] VARIANT Top, [in] VARIANT Left, [in] VARIANT Bottom, [in] VARIANT Right)

设置 VISSIM 主窗口的位置。所给的屏幕坐标的尺寸是相对于屏幕的左上角的位置的。

参数

[in] VARIANT Top：　　窗口顶部边界的屏幕坐标值

[in] VARIANT Left：　　窗口左边界的屏幕坐标值

[in] VARIANT Bottom：　　窗口底部边界的屏幕坐标值

[in] VARIANT Right：　　窗口右边界的屏幕坐标值

例子

```
SET vissim= NEW Vissim
vissim.SetWindow 0, 0, 800, 1000
```

NewWorldPoint([in, defaultvalue(0.0)] double X, [in, defaultvalue(0.0)] double Y,[in, defaultvalue(0.0)] double Z, [out, retval]IWorldPoint ** ppWorldPoint)

创建 WorldPoint 对象的方法。

参数

[out, retval] IWorldPoint ** ppWorldPoint：　　返回 WorldPoint 对象

例子

```
DIM wp AS WorldPoint
SET wp=vissim.NewWorldPoint(100.0, 100.0,100.0)
SET so=vissim.Net.StatictsObjects.GetStaticObjectByCoord(wp)
```

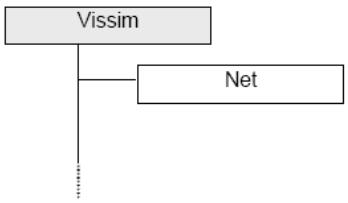
DoEvents()

允许 VISSIM 执行它的后续任务。只有 VISSIM 自己执行脚本时才是有效的。

例子

```
SET vis = CreateObject("VISSIM.Vissim")
Set sim = vis.simulation
For i = 1 to (sim.Period * sim.resolution)
    sim.runsinglestep
    vis.doevents
next
```

3.1.2 Net



Net 这一对象属于 Vissim，可以通过 IVissim 的接口来获得 Net 的特征属性。通过 Net，用户可以接触到路网的对象，比如 links, signal controllers 或者 vehicles。VISSIM 是一个单文档的程序，也就是说，它同时只能对不超过 1 个的路网进行工作。因此，一个 Net 的例子总是指向 VISSIM 例子的当前打开的路网（参见下面的例子）。

例子

```
DIM vissim As Vissim
DIM net1, net2 As Net
SET vissim=NEW Vissim
SET net1=vissim.Net
vissim.LoadNet "c:\vissim\daten\example.inp"
SET net2=vissim.Net
```

在该例子中，“net1”和“net2”指的是同一路网文件，就是“example.inp”。

INet 接口的属性

Name（[out,retval] BSTR *pName）

返回一个仿真的注释。

参数

[out, retval] BSTR*pName：　返回名称

例子

```
Name=net.Name
```

Name（[in] BSTR Name）

设置一个仿真的注释。

参数

[in] BSTR Name：新的名称

例子

net.Name=”Barcelona Example Siml”

Links（[out,retval] ILinks**ppLinks）

创建一个 Links 的对象的实例（参见第 50 页），这样可以单独地获得路网中的路段元素的信息。

参数

[out, retval] ILinks **ppLinks： 返回 Links 对象集。

例子

DIM links AS Links

SET links =net.Links

Nodes（[out,retval] INodes* *ppNodes）

创建一个 Nodes 的对象的实例（参见第 58 页），这样可以单独地获得路网中的节点元素的信息。

参数

[out, retval] INodes **ppNodes：返回 Nodes 对象集。

例子

DIM nodes AS Nodes

SET nodes =net.Nodes

Paths（[out,retval] IPaths* *ppPaths）

创建一个 Paths 的对象的实例（参见第 139 页），这样可以通过 IPaths 的接口单独地获得路网中的路径元素的信息。

参数

[out, retval] IPaths **ppPaths：返回 Paths 对象集。

例子

DIM paths AS Paths

SET paths =net.Paths

Vehicles（[out,retval] IVehicles* *ppVehicles）

创建一个 Vehicles 的对象的实例（参见第 80 页），这样可以单独地获得路网中的车辆元素的信息，包括停下来的车辆。

参数

[out, retval] IVehicles **ppVehicles：返回 Vehicles 这个对象集。

例子

DIM vehicles AS Vehicles

SET vehicles =net.Vehicles

VehicleInputs ([out, retval] IVehicleInputs **ppVehicleInputs)

创建一个 VehicleInputs 对象集的实例（参见第 106 页），这样可以单独的获取路网的车辆输入信息。

参数

[out, retval] IVehicleInputs **ppVehicleInputs : 返回一个车辆输入对象集

例子

DIM inps AS VehicleInputs

SET inps = net.VehicleInputs

RoutingDecisions（[out, retval] IRoutingDecisions **ppRoutingDecisions）

创建一个 RoutingDecisions 的对象的实例（参见第 95 页），这样可以单独地获得路网中的路径决策的信息。

参数

[out, retval] IRoutingDecisions **ppRoutingDecisions：返回 RoutingDecisions 这个对象集。

例子

DIM rds AS RoutingDecisions

SET rds = net.RoutingDecisions

SignalControllers（[out, retval] ISignalControllers **ppSignalControllers）

创建一个 SignalControllers 的对象的实例（参见第 157 页），这样可以单独地获得路网中的信号控制机元素的信息。

参数

[out, retval] ISignalControlle **ppSignalControllers：返回 SignalControlle 这个对象集。

例子

DIM scs AS SignalControllers

SET scs = net.SignalControllers

DataCollections（[out, retval] IDataCollections **ppDataCollections）

创建一个 DataCollections 的对象的实例（参见第 193 页），这样可以单独地获得路网中的定义的数据采集器的信息。

参数

[out, retval] IDataCollections **ppDataCollections：返回 DataCollections 这个对象集。

例子

DIM colls AS DataCollections

SET colls =net.DataCollections

QueueCounters（[out, retval] IQueueCounters **ppQueueCounters）

创建一个 QueueCounters 的对象的实例（参见第 209 页），这样可以单独地获得路网中定义的排队计数器元素的信息。

参数

[out, retval] IQueueCounters **ppQueueCounters：返回 QueueCounters 这个对象集。

例子

```
DIM qcs AS QueueCounters
SET qcs =net. QueueCounters
```

TravelTimes（[out, retval] ITravelTimes **ppTravelTimes）

创建一个 TravelTimes 的对象的实例（参见第 215 页），这样可以单独地获得路网中的定义的出行时间的信息。

参数

[out, retval] ITravelTimes **ppTravelTimes：返回 TravelTimes 这个对象集。

例子

```
DIM tts AS TravelTimes
SET tts =net.TravelTimes
```

Delays（[out, retval] IDelays **ppDelays）

创建一个 Delays 的对象的实例（参见第 200 页），这样可以单独地获得路网中的定义的延误时间的信息。

参数

[out, retval] IDelays **ppDelays：返回 Delays 这个对象集。

例子

```
DIM dels AS Delays
SET dels =net. Delays
```

AttValue（[out] BSTR Attibute, [out, retval] VARAIANT *pValue）

返回路网的一个属性值。请在该节的末尾的表格中了解各个与所选语言无关的特征属性的标识。

参数

[in] BSTR Attribute：	属性的名称（见下）
[out, retval] VARIANT *pValue：	返回该属性的值（类型根据特征属性）

例子

```
Width = net.AttValue("WIDTH")
```

AttValue（[in] BSTR Attibute, [in] VARAIANT Value）

设置一个路网的属性。请在该节的末尾的表格中了解各个特征属性的标识，这部分与内容与语言无关。

参数

[in] BSTR Attribute：　　属性的名称（见下）

[in] VARIANT Value：　　新的属性值（类型根据属性而定）

例子

net.AttValue("NAME")= "my network"

属性览表

R	W	属性	描述
√		ID	没有使用到。
√	√	NAME	名字（目前同等于仿真注释）
√		HEIGHT	路网的竖向尺寸（[m],[ft]）
√		WIDTH	路网的横向尺寸（[m],[ft]）

INet 接口的使用方法

Rotate（[in] double Angle）

逆时针旋转路网，旋转一个特定的角度。

参数

[in] double Angle：　　旋转的角度数

Translate（[in] double X, [in] double Y, [in] double Z）

在当前的单位设置下，根据指定的 X, Y 和 Z 的距离，平移一个路网。

参数

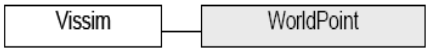
[in] double X：　　当前单位下的 X 坐标值。

[in] double Y：　　当前单位下的 Y 坐标值。

[in] double Z：　　当前单位下的 Z 坐标值。

3.2 COM 的数据对象

3.2.1 WorldPoint



该对象定义了一个非常通用的世界坐标点，它能够作为一个参数来使用，也将作为在一些方法里的一个结果来返回。和其它所有 VISSIM COM 接口的对象一样，当它需要做为一个参数时，该对象的创建必须使用 IVissim 接口（见 NewWorldPoint 方法）。

例子

```
DIM wp AS WorldPoint
SET wp = vissim.NewWorldPoint(100.0, 100.0, 100.0)
SET so = vissim.Net.StatictsObjects.GetStaticObjectByCoord(wp)
DIM veh AS Vehicle
DIM pos AS WorldPoint
SET veh = vissim.Net.Vehicles.GetVehiclesByNumber(1)
SET pos = veh.AttValue(“POINT”)
x = pos.X
y = pos.Y
z = pos.Z
```

IWorldPoint 接口的属性

X ([out, retval] double *pX)

得到 world point（x，y，z）的 X 坐标。

参数

[out, retval] double *pX : 返回 x 坐标.

X ([in] double X)

设置 world point（x，y，z）的 X 坐标。

参数

[in] double X : 新的 x 坐标

Y ([out, retval] double *pY)

得到 world point（x，y，z）的 Y 坐标。

参数

[out, retval] double *pY : 返回 y 坐标.

Y ([in] double Y)

设置 world point（x，y，z）的 Y 坐标

参数

[in] double Y : 新的 y 坐标

Z ([out, retval] double *pZ)

得到 world point（x，y，z）的 Z 坐标

参数

[out, retval] double *pZ : 返回 z 坐标.

Z ([in] double Z)

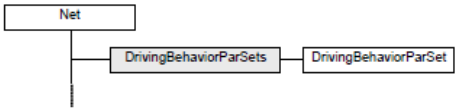
设置 world point（x，y，z）的 Z 坐标

参数

[in] double Z : 新的 z 坐标

3.3 VISSIM 的基础数据

3.3.1 DrivingBehaviorParSets



DrivingBehaviorParSets 对象是 DrivingBehaviorParSet 对象（见第 33 页）的集合。

它隶属于 Net 对象，能够通过 INet 接口的 DrivingBehaviorParSets 属性进入。它包含了路网上所有的驾驶行为参数设置，能够通过循环在一个集合中或者通过具体的编号来获得单个 DrivingBehaviorParSet 对象。（详见第 33 页）

例子

创建一个 DrivingBehaviorParSets 对象实例，并获得所有 DrivingBehaviorParSet 对象：

```
DIM vissim As Vissim
DIM dbpss As DrivingBehaviorParSets
DIM dbps As DrivingBehaviorParSet
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set dbpss = Vissim.Net.DrivingBehaviorParSets
FOR EACH dbps IN dbpss      ‘进入 _NewEnum 创建一个枚举
…
NEXT dbps
‘或者也可以这样：
FOR i = 1 TO dbpss.Count
SET dbps = dbpss(i)   ‘或者 dbpss.Item(i)
…
NEXT i
```

IDrivingBehaviorParSets 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 DrivingBehaviorParSet 对象的集合（或枚举）。一旦集合被创建，运用 GetDrivingBehaviorParSetByNumber 的方法就能返回个别对象，而整个集合可以通过使用 FOR ...TO ... NEXT 的陈述来循环。在 Visual Basic 的环境下用户只需运用 FOR EACH…NEXT 的语句，不需说明_NewEnum，VB 即会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举对象。

Item ([in] VARIANT Index, [out, retval] IDrivingBehaviorParSet **ppDBPS)

返回集合某个位置的一个单个 DrivingBehaviorParSet 对象。只有当所有的驾驶行为参数都可以获得的时候，该属性才有用（见上面的例子）。GetDrivingBehaviorParSetByNumber 则可以通过驾驶行为参数的编号来进行访问（见下）。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

SET dbps = dbpss(1)

SET dbps = dbpss.Item(1)

参数

[in] long Index：	索引值介于 1 和 Count 之间
[out, retval] IDrivingBehaviorParSet **ppDBPS：	返回 DrivingBehaviorParSet 对象。

Count ([out, retval] long *pCount)

在集合中返回 DrivingBehaviorParSet 对象的数量。见上面的例子。

参数

[out, retval] long *pCount ：返回对象的数量。

IDrivingBehaviorParSets 接口的使用方法

GetDrivingBehaviorParSetByNumber ([in] long Number, [out, retval] IDrivingBehaviorParSet **ppDBPS)

用编号返回 DrivingBehaviorParSet 对象。

参数

[in] long Number：	编号
[out, retval] IDrivingBehaviorParSet **ppDBPS：	返回 DrivingBehaviorParSet 对象

例子

SET dbps = dbpss.GetDrivingBehaviorParSetByNumber(2)

IF NOT (dbps IS NOTHING) THEN

name = dbps.AttValue(“NAME”)

END IF

3.3.2 DrivingBehaviorParSet

一 个 DrivingBehaviorParSet

对象代表了驾驶行为参数的设置元素，隶属于 DrivingBehaviorParSets 对象。

它可以通过以下两种方法进入 DrivingBehaviorParSets 对象：

► 通过在一个集合中的循环获得元素

```
DIM dbps As DrivingBehaviorParSet
FOR EACH dbps IN vissim.Net.DrivingBehaviorParSets
List.AddItem dbps.ID
NEXT dbps
```

► 通过具体的编号

```
DIM dbps As DrivingBehaviorParSet
SET dbps = vissim.Net.DrivingBehaviorParSets.GetDrivingBehaviorParSetByNumber(2)
```

DrivingBehaviorParSet 对象能够通过 IDrivingBehaviorParSet 接口进入驾驶行为参数设置的属性。

IDrivingBehaviorParSet 接口的属性

ID ([out, retval] long *pID)

返回驾驶行为参数设置的编号。

参数

[out, retval] long *pID：返回编号。

例子

```
id = dbps.ID
```

Name ([out, retval] BSTR *pName)

返回驾驶行为参数设置的名称。

参数

[out, retval] BSTR *pName： 返回名称。

例子

```
name = dbps.Name
```

Name ([in] BSTR Name)

设置驾驶行为参数设置的名称。最大 255 个字符。

参数

[in] BSTR Name ：新名称。

例子

```
dbps.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个驾驶员行为参数设置的属性值。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识。

参数

[in] BSTR Attribute： 特征属性名称（如下）

[out, retval] VARIANT *pValue ：返回特征属性的值。

例子

name = dbps.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个驾驶员行为参数设置属性。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识。

参数

[in] BSTR Attribute ： 特征属性名称（如下）

[in] BSTR Value： 特征属性的值（类型根据特征属性的定义）

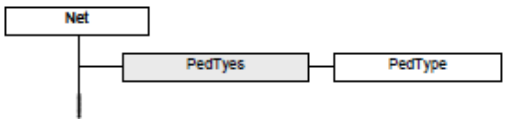
例子

dbps.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称
√	√	CC0	停车间距（采用当前的距离单位设置）
√	√	CC1	车头时距 [s]
√	√	CC2	‘跟车变量（采用当前的距离单位设置）
√	√	CC3	进入跟车状态的阈值[-]
√	√	CC4	消极跟车状态的阈值 [-]
√	√	CC5	积极跟车状态的阈值 [-]
√	√	CC6	车速振动
√	√	CC7	振动加速度 [m/s^2]
√	√	CC8	停车的加速度[m/s^2]
√	√	CC9	车速为 80 km/h 时的加速度 [m/s^2]
√	√	AXADD	停车距离（当前的距离单位）
√	√	BXADD	期望安全距离的附加部分
√	√	BXMULT	期望安全距离的倍数部分

3.3.3 PedTypes



PedTypes 对象是 PedType 对象的集合（详见第 36 页）。

它隶属于 Net 对象，并且可以通过 INet 的接口的 PedTypes 属性来进行访问。它包括了载入路网中所有行人的类型，可以通过枚举的方式或单独访问的方式对其进行调取（可以参照 PedType 的相关章节）。

例子

实例化一个 PedTypes 对象，并且访问所有的 PedType 对象：

```
DIM vissim As Vissim
DIM pTypes As PedTypes
DIM pType As PedType
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set pTypes = Vissim.Net.PedTypes
FOR EACH pType IN pTypes  ‘进入 _NewEnum 创建一个枚举
…
NEXT pType
‘or also:
FOR i = 1 TO pTypes.Count
SET pType = pTypes(i) ‘or pTypes.Item(i)
…
NEXT i
```

IPedTypes 对象的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性为所有的 PedTypes 对象创建了一个集合（或者说是枚举），每当一个集合创建后，每个单独的成员都可以 PedTypeByNumber 方法获取，同时整个集合也可以通过 FOR....TO…NEXTFOR 语句来进行枚举。而在 Visual Basic 的环境下用户只需运用 FOR EACH…NEXT 的语句，不需说明_NewEnum，VB 即会在程序内部调用它（参见上面 PedTypes 的例子）。

Item ([in] VARIANT Index, [out, retval] IPedType **pppType)

返回集合中某个具体位置的单个 PedType 对象。该方法只有当全部的行人类型都可以访问时才能够使用（参见上面 PedTypes 的例子）。若需通过编号来选择一个行人类型则应该通过 PedTypeByNumber 方法（见下）。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET pType = pTypes(1)
SET pType = pTypes.Item(1)
```

参数

[in] long Index :	index 应在 1 至 Count 之间取值
[out, retval] IPedType **pppType:	返回 PedType 对象

Count ([out, retval] long *pCount)

返回 PedType 对象的个数，参见上面 PedTypes 的例子。

参数

[out, retval] long *pCount :	返回对象的个数

IPedTypes 接口的方法

PedTypeByNumber ([in] long Number, [out, retval] IPedType **pppType)

返回 PedType 对象的编号。

参数

[in] long Number :	编号
[out, retval] IPedType **pppType :	返回 PedType 对象

例子

```
SET ptype = pTypes.PedTypeByNumber(2)
IF NOT (pType IS NOTHING) THEN
name = pType.AttValue("NAME")
END IF
```

3.3.4 PedType

PedType 对象属于 PedTypes 对象集，它表示一个行人类型的元素。

它可以通过两种方式从 PedTypes 进行访问。

► 通过在一个集合中的循环获得元素

```
DIM pType As PedType
FOR EACH pType IN vissim.Net.PedTypes
List.AddItem pType.ID
NEXT pType
```

► 通过编号单独访问

```
DIM pType As PedType
```

```
SET pType = vissim.Net.PedTypes.PedTypeByNumber(2)
```

PedType 对象使得用户可以通过 IPedType 接口来访问行人类型的属性。

IPedType 接口的属性

ID ([out, retval] long *pID)

返回行人类型的编号。

参数
[out, retval] long *pID : 返回编号.

例子
id = pType.ID

Name ([out, retval] BSTR *pName)

返回行人类型的名称。

参数
[out, retval] BSTR *pName : 返回名称.

例子
name = pType.Name

Name ([in] BSTR Name)

设置行人类型的名称，最长不超过 255 个字符。

Parameters
[in] BSTR Name : 新名称

Example
pType.Name = „XXX“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个行人类型的属性值。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 属性名称 （见下表）

[out, retval] VARIANT *pValue : 返回该属性值

例子
name = pType.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个行人类型的属性。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下）

[in] BSTR Value : 属性的值.（类型根据特征属性）

例子

pType.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称

3.3.5 PedClasses

PedClasses 对象是一个 PedClass 的集合（参见第 40 页）。

它隶属于 Net 对象并且可以通过 INet 接口访问 PedClasses 属性。它包含了载入路网的所有行人类别对象，可以同时通过循环来访问全部的行人类别对象也可以单独访问（也可参见 PedClass 对象）。

例子

实例化一个 PedClasses 对象并访问所有的 PedClass 对象：

```
DIM vissim As Vissim
DIM pClasses As PedClasses
DIM pClass As PedClass
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set pClasses = Vissim.Net.PedClasses
FOR EACH pClass IN pClassess  ‘进入 _NewEnum，创建一个枚举
…
NEXT pClass
‘or also:
FOR i = 1 TO pClasses.Count
```

```
SET pClass = pClasses(i) ‘or pClasses.Item(i)
…
NEXT
```

IPedClasses 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性创建了一个所有 PedClass 对象的集合（或者枚举）。当这个集合被创建后，每一个成员都可以被 PedClassNumber 方法单独访问。同时整个集合也可以通过 FOR....TO…NEXTFOR 语句来进行枚举。而在 Visual Basic 的环境下用户只需运用 FOR EACH…NEXT 的语句，不需说明_NewEnum，VB 即会在程序内部调用它（参见上面 PedClasses 的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IPedClass **pppClass)

返回集合中某个具体位置的单个 PedClass 对象。该方法只有当全部的行人类别都可以访问时才能够使用（参见上面 PedClasses 的例子）。若需通过编号来选择一个行人类型则应该通过 PedClassByNumber 方法（见下）。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET pClass = pClasses(1)
SET pClass = pClasses.Item(1)
```

参数

[in] long Index :	介于 1 和 Count 之间的索引
[out, retval] IPedClass **pppClass:	返回 PedClass 的对象

Count ([out, retval] long *pCount)

返回集合中所有 PedClass 对象的数量，参见上面 PedClasses 的例子。.

参数

[out, retval] long *pCount : 返回对象的数量.

IPedClasses 接口的方法：

PedClassByNumber ([in] long Number, [out, retval] IPedClass **pppClass)

通过编号来访问 PedClass 对象。

参数
[in] long Number : 编码

[out, retval] IPedClass **pppClass : 返回 PedClass 的对象

例子
```
SET pClass = pClasses.PedClassByNumber(2)
IF NOT (pClass IS NOTHING) THEN
name = pClass.AttValue("NAME")
END IF
```

3.3.6 PedClass

PedClass 对象表示行人类别的基本元素，属于 PedClasses 对象。

有两种方法可以通过 PedClasses 对象对 PedClass 进行访问。

► 通过在一个集合中的循环获得元素

```
DIM pClass As PedClass
FOR EACH pClass IN vissim.Net.PedClasses
List.AddItem pClass.ID
NEXT pClass
NEXT pType
```

► 通过编号单独访问

```
DIM pClass As PedClass
SET pClass = vissim.Net.PedClasses.PedClassByNumber(2)
```

PedClass 对象允许用户通过 IPedClass 接口对行人类别的属性进行访问。

IPedClass 接口的属性

ID ([out, retval] long *pID)

返回行人类别的编号。

参数
[out, retval] long *pID : 返回编号.

例子
```
id = pClass.ID
```

Name ([out, retval] BSTR *pName)

返回行人类别的名称。

参数

[out, retval] BSTR *pName : 返回名称.

例子

name = pClass.Name

Name ([in] BSTR Name)

设置行人类别的名称，最大不超过 255 个字符。

参数

[in] BSTR Name : 新名称

例子

pClass.Name = „XXX“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个行人类别的属性。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下表）

[out, retval] VARIANT *pValue : 返回该属性的值

例子

name = pClass.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个行人类别的属性。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下）

[in] BSTR Value : 属性的值.（类型根据特征属性）

例子

pClass.AttValue(„NAME“)=„XXX“

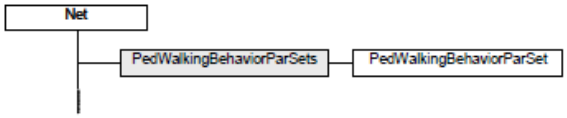
属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称

3.3.7 PedWalkingBehaviorParSets

PedWalkingBehaviorParSets 对象是一个包含 PedWalkingBehaviorParSet 对象的集合（参见第 44 页）。

它隶属于 Net 对象并且可以通过 INet 接口访问



PedWalkingBehaviorParSets 属性。它包含了载入路网的所有的行走行为对象，可以通过循环来访问，也可以单独访问全部的行走行为对象（也可参见 PedWalkingBehaviorParSet 对象）。

例子

实例化 PedWalkingBehaviorParSets 并访问所有的 PedWalkingBehaviorParSet 对象。

```
DIM vissim As Vissim
DIM wbpss As PedWalkingBehaviorParSets
DIM wbps As PedWalkingBehaviorParSet
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set wbpss = Vissim.Net.PedWalkingBehaviorParSets
FOR EACH wbps IN wbpss ‘进入_NewEnum，创建一个枚举
…
NEXT wbps
‘或者也可以写成：
FOR i = 1 TO wbpss.Count
SET wbps = wbpss(i) ‘ 或 wbpss.Item(i)
…
NEXT i
```

IPedWalkingBehaviorParSets 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性创建了一个所有 PedWalkingBehaviorParSet 对象的集合（或者枚举）。当这个集合被创建后，每一个成员都可以被 GetPedWalkingBehaviorParSetByNumber 方法单独访问。同时整个集合也可以通过 FOR....TO…NEXT 语句来进行枚举。而在 Visual Basic 的环境下用户只需运用 FOR EACH…NEXT 的语句，不需说明_NewEnum，VB 即会在程序内部调用它（参见上面 PedWalkingBehaviorParSets 的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IPedWalkingBehaviorParSet **ppWBPS)

返回集合中某个具体位置的单个 PedWalkingBehaviorParSet 对象。该方法只有当全部的行人类别都可以访问时才能够使用（参见上面 PedWalkingBehaviorParSets 的例子）。若需通过编号来选择一个行人类型则应该通过 GetPedWalkingBehaviorParSetByNumber 方法（见下）。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

SET wbps = wbpss(1)

SET wbps = wbpss.Item(1)

参数

[in] long Index : 1 和 Count 之间的索引

[out, retval] IPedWalkingBehaviorParSet **ppWBPS: 返回 PedWalkingBehaviorParSet 对象

Count ([out, retval] long *pCount)

返回集合中所有 PedWalkingBehaviorParSet 对象的数量，参见上面 PedWalkingBehaviorParSets 的例子。.

参数

[out, retval] long *pCount : 返回对象的数量.

IPedWalkingBehaviorParSets 接口的方法

PedWalkingBehaviorParSetByNumber ([in] long Number, [out, retval] IPedWalkingBehaviorParSet **ppWBPS)

通过编号来访问 PedWalkingBehaviorParSet 对象。

参数

[in] long Number : 编码

[out, retval] IPedWalkingBehaviorParSet **ppWBPS :返回 PedWalkingBehaviorParSet 对象

例子

```
SET wbps = wbpss.PedWalkingBehaviorParSetByNumber(2)
IF NOT (wbps IS NOTHING) THEN
name = wbps.AttValue("NAME")
END IF
```

3.3.8 PedWalkingBehaviorParSet

PedWalkingBehaviorParSet 对象表示一个行人行走行为参数元素的组合，并且属于 PedWalkingBehaviorParSets 对象。

有两种方法可以通过 

PedWalkingBehaviorParSets 对象对其进行访问。

► 通过在一个集合中的循环获得元素

```
DIM wbps As PedWalkingBehaviorParSet
FOR EACH wbps IN vissim.Net.PedWalkingBehaviorParSets
List.AddItem wbps.ID
NEXT wbps
```

► 通过编号单独访问

```
DIM pClass As PedClass
SET pClass = vissim.Net.PedClasses.PedClassByNumber(2)
```

PedWalkingBehaviorParSet 对象允许用户通过 IPedWalkingBehaviorParSet 接口对行人行走行为的属性进行访问。

IPedWalkingBehaviorParSet 接口的属性

ID ([out, retval] long *pID)

根据编号返回行人行走行为。

Parameters
[out, retval] long *pID : 返回编号.

Example

```
id = wbps.ID
```

Name ([out, retval] BSTR *pName)

返回行人行走行为的名称。

参数
[out, retval] BSTR *pName : 返回名称.

例子

```
name = wbps.Name
```

Name ([in] BSTR Name)

设置行人行走行为的名称，最多不超过 255 个字符

参数
[in] BSTR Name : 新的名称.

例子

```
wbps.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个行人行走行为的参数。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下表）

[out, retval] VARIANT *pValue : 返回该属性的值

例子

name = wbps.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个行人行走行为的参数。请在该节的末尾的表格中了解与所设语言无关的各个特征属性的标识编号。

[in] BSTR Attribute : 特征属性名称（见下）

[in] BSTR Value : 属性的值.（类型根据特征属性）

例子

wbps.AttValue(„NAME“)=„XXX“

属性览表:

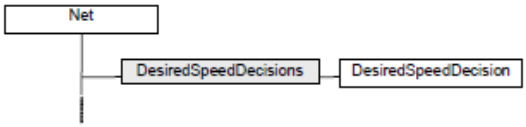
R	W	属性	描述
√		ID	编号
√	√	NAME	名称

3.4 Network

3.4.1 Link-based

3.4.1.1 DesiredSpeedDesicions

DesiredSpeedDesicions 对象是 DesiredSpeedDecision 对象的一个集合（参见第 47 页）。



它属于 Net 对象并可以通过 INet 接口的 DesiredSpeedDecisions 属性来访问。它包含了载入路网的所有的期望速度决策，可以通过循环来访问，也可以单独访问全部的 DesiredSpeedDecision 对象（也可参见 DesiredSpeedDecision 对象）。

例子

实例化一个 DesiredSpeedDescisions 对象并且访问它所有的 DesiredSpeedDescision 对象。

```
DIM vissim As Vissim
DIM decisions As DesiredSpeedDecisions
DIM decision As DesiredSpeedDecision
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET decisions = Vissim.Net.DesiredSpeedDecisions
FOR EACH decision IN decisions ‘进入_NewEnum，创建一个枚举
…
NEXT decision
‘or also:
FOR i = 1 TO decisions.Count
SET decision = decisions(i) ‘or decisions.Item(i)
…
NEXT i
```

IDesiredSpeedDecisions 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性创建了一个所有 DesiredSpeedDecision 对象的集合（或者说枚举）。每个单独的对象都可以通过 GetDesiredSpeedDecisionByNumber 方法来访问，同时整个集合也可以

通过 FOR…TO…NEXT 语句来枚举访问，而在 Visual Basic 的环境下用户只需运用 FOR EACH…NEXT 的语句，不需说明_NewEnum，VB 即会在程序内部调用它（参见上面 DesiredSpeedDecisions 的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IDesiredSpeedDecision **ppDecision)

返回集合中某个具体位置的单个 DesiredSpeedDecision 对象。该方法只有当全部的期望速度决策点都可以访问时才能够使用（参见上面 DesiredSpeedDecisions 的例子）。若需通过编号来选择一个期望速度决策点则应该通过 GetDesiredSpeedDecisionByNumber 方法（见下）。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET decision = decisions(1)
SET decision = decisions.Item(1)
```

参数

[in] long Index:	介于 1 和 Count 之间的索引
[out, retval] IDesiredSpeedDecision **ppDecision:	返回 DesiredSpeedDecision

Count ([out, retval] long *pCount)

返回集合中所有 DesiredSpeedDecision 对象的数量，参见上面 DesiredSpeedDecisions 的例子。.

参数

[out, retval] long *pCount : 返回对象的数量.

IDesiredSpeedDecisions 接口的方法

通过编号来访问 DesiredSpeedDecision 对象。

参数

[in] long Number : 编码

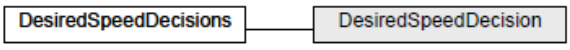
[out, retval] IDesiredSpeedDecision **ppDecision :返回 DesiredSpeedDecision

例子

```
SET decision = decisions.GetDesiredSpeedDecisionByNumber (2)
IF NOT (decision IS NOTHING) THEN
name = decision.AttValue(“NAME”)
END IF
```

3.4.1.2 DesiredSpeedDecision

DesiredSpeedDecision 对象



表示一个期望速度决策点元素，并且属于 DesiredSpeedDecisions 对象。

有两种方法可以通过 DesiredSpeedDecisions 对象对其进行访问。

► 通过在一个集合中的循环获得元素

```
DIM decision As DesiredSpeedDecision
FOR EACH decision IN vissim.Net.DesiredSpeedDecisions
List.AddItem decision.ID
NEXT decision
```

► 通过编号单独访问

```
DIM decision As DesiredSpeedDecision
SET decision = vissim.Net.DesiredSpeedDecisions.
GetDesiredSpeedDecisionByNumber (12)
```

DesiredSpeedDecision 对象允许用户通过 IDesiredSpeedDecision 接口对期望速度决策点的属性进行访问。

IDesiredSpeedDecision 接口的属性

ID ([out, retval] long *pID)

参数

[out, retval] long *pID : 返回编号.

例子

```
id = decision.ID
```

Name ([out, retval] BSTR *pName)

返回期望速度决策点的名称。

参数

[out, retval] BSTR *pName : 返回名称

例子

```
name = decision.Name
```

Name ([in] BSTR Name)

设置决策点的名称，最多不超过 255 个字符

参数

[in] BSTR Name : 新的名称

例子

```
decision.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个期望速度决策点的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下表）

[out, retval] VARIANT *pValue : 返回该属性的值

例子

name = decision.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个期望速度决策点的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

[in] BSTR Attribute : 特征属性名称（见下）

[in] BSTR Value : 属性的值.（类型根据特征属性）

例子

decision.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称
√	√	TIMEFROM	激活的时间区间的开始[s]
√	√	TIMEUNTIL	激活的时间区间的结束[s]
√	√	DESIREDSPEED	第一个车辆类别的期望速度分布编号
√		VEHICLECLASSES	影响的车辆类别（编号的数组）
√		VEHICLETYPES	影响的车辆类型（编号的数组）

i 注意：设置时间间隔时，TIMEUNTIL的值必须要大于等于TIMEFROM的值。如果当前的时间间隔是[100, 1000] ，而期望的设置为[1100, 2000]，用户必须先将TIMEUNTIL设置为2000，然后将TIMEFROM设置为1100，否则程序将提示出错信息。

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

通过参数来返回一个期望速度决策点的一个属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute :	特征属性名称（见下）
[in] VARIANT Parameter :	与属性有关的参数（见下）
[out, retval] VARIANT *pValue :	返回属性的值

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

为一个期望速度决策点的一个属性设置一个参数。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute :	特征属性名称（见下）
[in] VARIANT Parameter :	与属性有关的参数（见下）
[in] VARIANT Value :	属性的值(类型按照下面的属性)

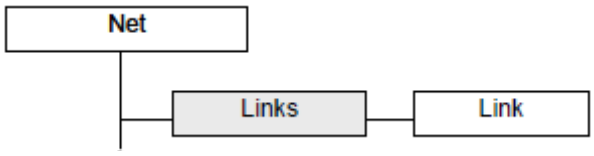
属性览表:

R	W	属性	描述
√	√	DESIREDSPEED	期望速度分布的编号。 参数：车辆类别。
√	√	VEHICLECLASS	如果某车辆类别被该决策点影响，则返回 TRUE。 参数：车辆类别编号。
√		VEHICLETYPE	如果某车辆类型被该决策点影响，则返回 TRUE。 参数：车辆类别编号。

在仿真进行过程中，这些方法可以实时对参数进行改变。

3.4.1.3 Links

Links 对象是 Link 对象的一个集合（参见第 52 页）。它属于 Net 对象并可以通过 INet 接口的 Links 属



性来访问。

它包含了载入路网的所有路段和连接器对象，可以同时通过循环来访问集合中的全部对象也可以单独访问（也可参见 Link 对象）。

例子

实例化一个 Links 对象并访问所有的 Link 对象。

```
Dim vissim As Vissim
Dim links As Links
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set links = Vissim.Net.Links
FOR EACH link IN links ‘进入_NewEnum，创建一个新的枚举
 …
NEXT link
‘or also:
FOR i = 1 TO links.Count
SET link = links(i) ‘or links.Item(i)
…
NEXT i
```

ILinks 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性创建了一个所有 Link 对象的集合（或者枚举）。当这个集合被创建后，每一个成员都可以被 GetLinkByNUmber 方法单独访问。同时整个集合也可以通过 FOR....TO…NEXTFOR 语句来进行枚举。而在 Visual Basic 的环境下用户只需运用 FOR EACH…NEXT 的语句，不需说明 _NewEnum，VB 即会在程序内部调用它（参见上面 Links 的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] ILink **ppLink)

返回集合中某个具体位置的单个 Link 对象。该方法只有当全部的路段都可以访问时才能够使用（参见上面 Links 的例子）。若需通过编号来选择单个路段则应该通过 GetLinkByNumber 方法（见下）。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET link = links(1)
SET link = links.Item(1)
```

参数

[in] long Index:	介于 1 和 Count 的索引号
[out, retval] ILink **ppLink:	返回 Link 对象

Count ([out, retval] long *pCount)

返回集合中所有 Link 对象的数量，参见上面 Links 的例子。

参数

[out, retval] long *pCount : 返回对象的数量.

GetLinkByNumber ([in] long Number, [out, retval] ILink **ppLink)

通过编号返回 Link 对象。

参数

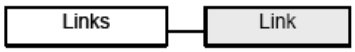
[in] long Number： 路段编号

[out, retval] ILink **ppLink： 返回 Link 对象。

例子

```
SET link = links.GetLinkByNumber(1001)
```

3.4.1.4 Link



该对象代表了一个路段或连接器，并且隶属于 Links 对象。通过 Links 对象，可以通过两种方式访问一个 Link 对象：

► 通过在集合中的循环获得

```
DIM links As Links
DIM link As Link
SET links = Vissim.Net.Links
FOR EACH link IN links
List.AddItem link.ID
NEXT link
```

► 通过具体的编号获得

```
DIM links As Links
DIM link As Link
SET links = Vissim.Net.Links
SET link = links. GetLinkByNumber(1000)
```

通过 ILink 的接口，Link 对象能够访问路段的各个属性。

ILink 接口的属性

ID ([out, retval] long *pID)

返回 Link 标识编号数。若路段对象没有涉及有效的 VISSIM 路段元素，则返回值为 0。

参数

[out, retval] long *pID : 返回标识编号（如果所指无效，则为 0）

例子
DIM link AS Link
id = link.ID

Name ([out, retval] BSTR *pName)

返回一个 Link 名称。

参数
[out, retval] BSTR *pName ：返回名称。

例子
name = link.Name

Name ([in] BSTR Name)

设置一个 Link 名称。最多 255 个字符。

参数
[in] BSTR Name：新名称

例子
link.Name = „Banbury Road up“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个 Link 的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute： 属性名（如下）
[out, retval] VARIANT *pValue：返回属性值。

例子
length = link.AttValue(„LENGTH“)
polyline = link.AttValue(„POINTS“)
FOR i = LBOUND(polyline) TO UBOUND(polyline)
x = polyline (i).X
y = polyline (i).Y
NEXT i

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个 Link 的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute： 属性名（如下）
[in] VARIANT Value： 属性值（类型根据特征属性的定义）

例子
link.AttValue(„NAME“) = “stumpfstr.”

属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称
√	√	BEHAVIORTYPE	路段驾驶行为类型
√		CONNECTOR	若该 Link 为连接器，则 True
√	√	COST	每公里的路段费用
√	√	DISPLAYTYPE	路段的现实类型编号
√	√	EMERGENCYSTOP	连接器紧急停车参数
√		FROMLANE	如果为连接器：即“从路段”的连接的最小车道编号（最右边编号为 1）。否则为 0。
√		FROMLINK	如果为连接器：即原始路段的编号，否则为 0。
√		FROMLINKCOORD	如果路段为连接器：返回连接器在起始路段上开始点的坐标（用当前单位表示）。否则则为 0。
√		GRADIENT	坡度，以%计
√	√	LANECHANGE	连接器的车道变换距离
√		LANEWIDTH	最右边的车道宽度，单位同当前的选项设置
√		LENGTH	路段长度，单位同当前的选项设置
√		NUMLANES	车道数
√		POINTS	通过路段中心的参考线(VARIANTs 的数组)
√	√	SURCHARGE1	敏感的附加费用 1 的值
√	√	SURCHARGE2	不敏感的附加费用 2 的值

R	W	属性	描述
√	√	THICKNESS	3D 显示模式下的厚度，以当前单位表示。
√		TOLANE	如果是连接器：连接的目的地路段的最右边的车道编号。否则为 0。
√		TOLINK	如果为连接器：目的地路段的路段编号，否则为 0
√		TOLINKCOORD	如果为连接器：在目的地路段上的连接器结束位置的坐标（以当前的单位计算），否则为 0
√	√	VISUALIZATION	在仿真过程中该路段上的车辆是否显示

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带参数返回一个 Link 的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下）

[in] VARIANT Parameter： 与属性相关的参数（如下）

[out, retval] VARIANT *pValue：返回属性值

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带参数设置一个 Link 的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下）

[in] VARIANT Parameter： 与属性相关的参数（如下）

[in] VARIANT Value ： 属性值（类型根据属性的定义）

属性览表:

R	W	属性	描述
√	√	CLOSED	连接器关闭。 参数：车辆类别编号
√	√	LANEWIDTH	表示当前单位下车道宽度的数值。 参数：车道编号（最右侧的为1）。

AttValue2 ([in] BSTR Attribute, [in] VARIANT Parameter1, [in] VARIANT Parameter2, [out, retval] VARIANT *pValue)

返回一个 link 的属性，带两个参数。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下）

[in] VARIANT Parameter1： 第一个与属性相关的参数（如下）

[in] VARIANT Parameter2： 第二个与属性相关的参数（如下）

[out, retval] VARIANT *pValue：返回属性值

例子

isclosed = link.AttValue2(„LANECLOSED“, lanenr, vehclass)

AttValue2 ([in] BSTR Attribute, [in] VARIANT Parameter1, [in] VARIANT Parameter2, [in] VARIANT Value)

带两个参数设置一个 Link 的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下）

[in] VARIANT Parameter1： 第一个与属性相关的参数（如下）

[in] VARIANT Parameter2： 第二个与属性相关的参数（如下）

[in] VARIANT Value： 属性值（类型根据属性的定义）

例子

link.AttValue2(„LANECLOSED“, lanenr, vehclass) = TRUE

属性览表:

R	W	属性	描述
√	√	LANECLOSED	车道关闭。 参数：车道编号（最右侧编号为1）及车辆类别编号 。

ILink 接口的使用方法

GetSegmentResult ([in] BSTR Parameter, [in] long VehicleClass, [in] double XCoord, [in, defaultvalue(1)] int Lane, [in, defaultvalue(0)] BYTE Cumulative, [in, retval] VARIANT *pValue)

返回之前配置的路段区段的最后的统计结果（见下面的说明）。该 Link 必须已经选择了路段评价，并且定义了区段的长度。如果评价文件不是根据每个车道给出的话（参见 206 页的 ILinkEvaluation），则会使用车道编号 1。如果要将所有的区段结果对应一个二维的数组返回时，则需要

将路段坐标设置为一个负值。在这个情况下，该数组保存区段的开始坐标，以及对应的结果。下表列出了一个可能提供的参数名称表。

参数

[in] BSTR Parameter： 参数名称（如下）

[in] long VehicleClass： 车辆类别编号；对于所有车辆类型的话，值为 0

[in] double XCoord： link 的坐标（从 0.0 到路段的长度；一个负值（例如-0.1）表示的是对每个区段都是一个二维的数组）

[in] int Lane： 车道编号 (如果评价设置中选中“每条车道”的选项从 1 到 n，否则就为 1。)

[in] BYTE Cumulative： 累计的评估标志 (默认值是 false)

[out, retval] VARIANT *pValue： 返回的数值（real）或者数组值（reals）

例子

```
val = link.GetSegmentResult(“SPEED”, 0, 0.0)        ‘ 车道 1 第一个区段输入结果或者全部车道（如果路段评价没有选择“每条车道”进行统计的话）的第一个区段结果
vals = link.GetSegmentResult(“SPEED”, 0, -1.0)      每个区段的一个结果值数组
FOR i = LBOUND(vals) TO UBOUND(vals)
segindex = vals(i, 0)                               ‘ 区段的索引编号
val = vals(i, 1)                                    ‘区段的结果
NEXT i
```

参数览表:

属性	描述
DENSITY	平均密度（当前单位下）
DELAY	相对的平均损失时间[s/s]
NVEHICLES	车辆数 (流量的累积值)。流量（VOLUME）必须在评价设置中是激活的。
SPEED	平均速度(选择当前的单位设置)
VOLUME	平均流量[veh/h]

所得的结果对应的是最后一次完整的时间间隔里收集到的数据。在该时间间隔的最后一个时间步长运行完毕后，可以立即获得一个时间间隔的数据，而不是仿真周期的最后一个时间步长。

为了得到结果，在计算之前，必须在VISSIM中定义一个路段评价的设置，保存在*.sak文件中。针对路段评价的离线分析选项也需要激活。否则会出现一个错误信息 (“The specified configuration is not defined within VISSIM”)。

GetVehicles ([out, retval] IVehicles ppVehicles)**

返回当前行驶在该条 Link 上的所有的车辆集合。该方法可以在运行期间调用。返回的集合作为该 Link 上车辆的动态指引表，在每一个仿真步长里都会改变（参见第 85 页的说明）。

参数

[out, retval] IVehicles** ppVehicles： 返回 Vehciles 对象

例子

```
vehicles = link.GetVehicles
FOR EACH v IN vehicles
        IF v.AttValue("SPEED") > 100.0 THEN
        v.AttValue("COLOR") = RGB(255, 0, 0)
        END IF
NEXT v
```

RecalculateSpline()

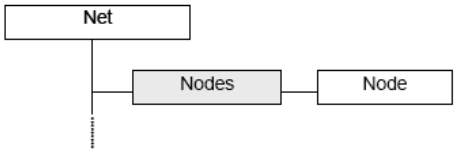
重新计算一个路段或者连接器的中间点。对于路段，只有 z 坐标针对每一个中间点进行更新。对于连接器，x、y、z 坐标都可针对每一个中间点进行更新。使用该方法不能改变中间点的数量。

RecalculateSpline()

重新计算一个路段或者连接器多义线的 z 坐标值。使用该方法不能改变中间点的数量。

3.4.1.5 Nodes

Nodes 对象是对各个 Node 对象的集合。它隶属于 Net 对象，可通过 INet 接口的属性 Nodes 进行调用。



它包含了路网中的所有节点，可以通过循环的方式或者特定的获得一个 Node 对象。

例子

创建 Nodes 实例，并获得其中所有的 Node 对象：

```
Dim vissim As Vissim
Dim nodes As Nodes
SET vissim = NEW Vissim
vissim.LoadNet "c:\vissim\daten\example.inp"
Set nodes = Vissim.Net.Nodes

FOR EACH node IN nodes  '通过 _NewEnum 创建一个枚举
…
```

```
NEXT node
‘or also:
FOR i = 1 TO nodes.Count
SET node = nodes(i)  ‘或者 nodes.Item(i)
…
NEXT i
```

INodes 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个所有 Node 对象的集合（或枚举）。一旦集合被创建，运用 GetNodeByNumber 的方法就能返回单个 Node 对象，而整个集合可以通过使用 FOR ...TO ... NEXT 循环语句来进行访问。用户可以在 Visual Basic 的环境下还可以使用 FOR EACH…NEXT 语句来进行访问，该方法不需说明_NewEnum，VB 会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum： 返回 Enumeration。

Item ([in] VARIANT Index, [out, retval] INode **ppNode)

返回集合某个位置的单个 Node。只有当所有的节点都可以访问的时候，该属性才有用（见上面的例子）。为了选择一个 Node，需要使用 GetNodeByNumber（见下）来进行索引。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET node = nodes(1)
SET node = nodes.Item(1)
```

参数

[in] long Index： 索引介于 1 和 Nodes.Count 之间

[out, retval] INode **ppNode： 返回 Node 对象

Count ([out, retval] long *pCount)

返回集合中的节点的数量，见上面的例子。

参数

[out, retval] long *pCount ： 返回对象的数量。

INodes 接口的使用方法

GetNodeByNumber ([in] long Number, [out, retval] INode **ppNode)

根据编号返回某一个 Node 对象。

参数

[in] long Number： Node 编号

[out, retval] INode **ppNode： 返回 Node 对象

例子

```
DIM node AS Node
SET node =nodes.GetNodeByNumber(101)
```

GetResult ([in] double Time, [in] BSTR Parameter, [in] BSTR Function, [in] long VehicleClass, [out, retval] VARIANT *pValue)

这个方法将返回收集到的所有节点和所需要的参数（见下列参数览表）及车辆类别的结果。返回值是包含特定时间点的时间间隔内的当前收集到的结果。Function 参数目前没有意义。

参数

[in] double Time： 时间点，单位是秒

[in] BSTR Parameter ： 参数名称（参见下面）

[in] BSTR Function： 未使用

[in] long VehicleClass： 车辆类别。如果是所有车辆类别，值取 0

[out, retval] VARIANT *pValue：返回值（实数）

例子

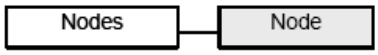
```
nv = nodes.GetResult(600, „NVEHICLES“, “”, 0) ‘ 系统中所有的车辆通过量
```

参数览表:

参数	描述
DELAY	每辆车的平均总延误[s]
PERSONSDELAY	每人的平均总延误 [s]
NPERSONS	乘客通过量（总人数）
NSTOPS	每辆车的平均停车次数
NVEHICLES	车辆通过量 (车辆数)
QUEUELENGTHAVG	平均排队长度(选择当前的单位设置)
QUEUELENGTHMAX	最大排队长度(选择当前的单位设置)
STOPPEDDELAY	每辆车的平均等待时间 (停车延误)[s]

i 为了得到结果，“评价-文件”中的“节点”选项必须激活，否则结果将都是 0.0。

3.4.1.6 Node



一个 Node 对象代表一个节点元素，隶属于 Nodes 对象。通过 Nodes 对象，有两种方式可以访问 Node：

► 在集合中通过循环访问

```
DIM nodes As Nodes
DIM node As Node
SET nodes = Vissim.Net.Nodes
FOR EACH node IN nodes
List.AddItem node.ID
NEXT node
```

► 通过单独的编号获得

```
DIM nodes As Nodes
DIM node As Node
SET nodes = Vissim.Net.Nodes
SET node = nodes. GetNodeByNumber(101)
```

通过 INode 接口，Node 对象能够访问 Node 的属性。

INode 接口的属性

ID ([out, retval] long *pID)

返回节点编号。若 Node 对象没有指向有效的 VISSIM 节点元素，则返回值为 0。

参数

[out, retval] long *pID： 返回编号。

例子

```
id = node.ID
```

Name ([out, retval] BSTR *pName)

返回该节点名称。

参数

[out, retval] BSTR *pName： 返回名称。

例子

```
name = node.Name
```

Name ([in] BSTR Name)

设置该节点的名称。最多 255 个字符。

参数

[in] BSTR Name： 新名称

例子
node.Name = „Durlachertor“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个节点属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute： 属性名（如下）

[out, retval] VARIANT *pValue ： 返回属性值

例子
node = node.AttValue(“NAME”)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个节点属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute： 属性名（如下）

[in] VARIANT Value ： 属性值（类型根据特征属性定义）

例子
node.AttValue(„NAME“) = “Piccadilly Circus”

属性览表:

R	W	属性	描述
√		ID	编号
	√	NAME	名称

INode 接口的使用方法

GetResult ([in] double Time, [in] BSTR Parameter, [in] BSTR Function, [in] long VehicleClass, [out, retval] VARIANT *pValue)

该方法返回该节点所有结果（所有转向关系）和所需参数（见下面的参数表）的值以及车辆类别。返回的值指的是当前已选的评价统计值在包括某一时间区间内特定时间点的的数据。Function 参数目前没有意义。

参数
[in] double Time ： 时间点，单位是秒

[in] BSTR Parameter ： 参数名称（如下）

[in] BSTR Function ： 未使用

[in] long VehicleClass： 车辆类别；如果是所有的车辆类别，值取 0

[out, retval] VARIANT *pValue：返回值（real number）

例子

nv = node.GetResult(600, „NVEHICLES“, “”, 0) ‘车辆的通过量

del = node.GetResult(600, “DELAY”, “”, 1) ‘针对车辆类别 1，每辆车的平均延误

GetMovementResult ([in] double Time, [in] long FromLink, [in] long ToLink [in] BSTR Parameter, [in] BSTR Function, [in] long VehicleClass, [out, retval] VARIANT *pValue)

该方法返回某一个特定转向关系并且对应车辆类别的所需参数（见下面的参数表）的结果。该返回值指的是覆盖某一时间区间内特定时间点的的当前统计的数据。Function 参数目前没有意义。

参数

[in] double Time： 时间点，单位是秒

[in] long FromLink： 进入节点的 link 编号

[in] long ToLink： 离开节点的 link 编号

[in] BSTR Parameter： 参数名称（如下）

[in] BSTR Function： 未使用

[in] long VehicleClass： 车辆类别；如果是所有的车辆类别，值取 0

[out, retval] VARIANT *pValue： 返回值（real number）

例子

nv = n.GetResult(600,11,12, „NVEHICLES“, “”, 0) ‘从 link 11 进入，从 link12 驶出的所有车辆通过量

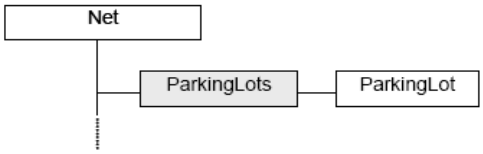
参数览表:

参数	描述
DELAY	每辆车的平均总延误[s]
PERSONSDELAY	每人的平均总延误 [s]
NPERSONS	乘客通过数（总人数）
NSTOPS	每辆车的平均停车次数
NVEHICLES	车辆通过数(车辆数)
QUEUELENGTHAVG	平均排队长度(选择当前的单位)
QUEUELENGTHMAX	最大排队长度(选择当前的单位)
STOPPEDDELAY	每辆车的平均等待时间 (停车延误) [s]

i 为了得到结果，“评价-文件”中的“节点”选项必须激活，否则所得的结果为 0.0。

3.4.1.7 ParkingLots

ParkingLots 对象是对各个 ParkingLot（停车场）对象的集合（见第 65 页）。它隶属于 Net 对象，并通过 INet 接口的属性 ParkingLots 进行调用。



它包含了路网的所有停车场，可以通过循环的方式或者特定的方式获得某个 PathingLot 对象（也可见 ParkingLot 对象）。

例子

创建 ParkingLots 对象的实例，访问所有 ParkingLot 对象：

```
DIM vissim As Vissim
DIM pls As ParkingLots
DIM pl As ParkingLot
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set pls = Vissim.Net.ParkingLots
FOR EACH pl IN pls   ‘通过 _NewEnum 创建一个枚举
…
NEXT pl
‘或者:
FOR i = 1 TO pls.Count
SET pl = pls(i)   ‘或者 pls.Item(i)
…
NEXT i
```

IParkingLots 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 ParkingLot 对象的集合（或枚举）。一旦集合被创建，运用 GetParkingLotsByNumber 的方法就能返回某个单独对象，而整个集合可以通过使用 FOR ...TO ... NEXT 循环语句来访问。用户可以在 Visual Basic 的环境下也可以运用 FOR EACH…NEXT 语句，该方法不需说明_NewEnum，VB 会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum ： 返回 Enumeration

Item ([in] VARIANT Index, [out, retval] IParkingLot **ppParkingLot)

返回集合某个位置的单个 ParkingLot。只有当所有的停车场都可以访问的时候，该属性才有用（见上面的例子）。为了选择某个单独的 ParkingLot，需要使用 GetParkingLotByNumber（见下）方法。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET pl = pls(1)
SET pl = pls.Item(1)
```

参数

[in] long Index：索引号介于 1 和 Count 之间

[out, retval] IParkingLot **ppParkingLots： 返回 ParkingLot 对象

Count ([out, retval] long *pCount)

返回该集合中 ParkingLot 对象的数量。见上面的例子。

参数

[out, retval] long *pCount： 返回对象的数量

IParkingLots 接口的使用方法

GetParkingLotByNumber ([in] long Number, [out, retval] IParkingLot **ppParkingLot)

通过编号返回 ParkingLot 对象。

参数

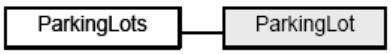
[in] long Number： 编号

[out, retval] IParkingLot **ppParkingLot： 返回 ParkingLot 对象

例子

```
SET pl = pls.GetParkingLotByNumber (2)
IF NOT (pl IS NOTHING) THEN
name = pl.AttValue(“NAME”)
END IF
```

3.4.1.8 ParkingLot



一个 ParkingLot 对象代表了一个停车场元素，隶属于 ParkingLots 对象。通过 ParkingLots 有两种方式访问 ParkingLot 对象：

► 通过在集合中的循环获得

```
DIM pl As ParkingLot
FOR EACH pls IN vissim.Net.ParkingLots
List.AddItem pl.ID
```

```
NEXT pl
```

► 通过具体的编号获得

```
DIM pl As ParkingLot
SET pl = vissim.Net.ParkingLots.GetParkingLotByNumber (12)
```

通过 IParkingLot 接口，ParkingLot 对象能够访问停车场的各个属性。

IParkingLot 接口的属性

ID ([out, retval] long *pID)

返回停车场的编号。

参数

[out, retval] long *pID： 返回编号

例子

```
id = pl.ID
```

Name ([out, retval] BSTR *pName)

返回停车场的名称。

参数

[out, retval] BSTR *pName： 返回名称

例子

```
name = pl.Name
```

Name ([in] BSTR Name)

设置停车场的名称。最大 255 个字符。

参数

[in] BSTR Name： 新名称.

例子

```
pl.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回停车场的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下表）

[out, retval] VARIANT *pValue：返回属性值

例子

```
name = pl.AttValue(„NAME“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置停车场的属性值。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下表）

[in] BSTR Value： 属性值（类型根据属性的定义）

例子

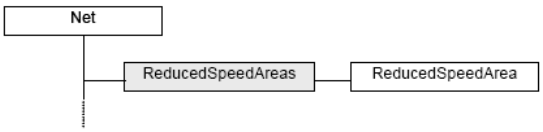
pl.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称
√		LINK	所在路段的编号
√		OCCUPANCY	当前的停车数量（见下面注解）
√		RELATIVEFLOW	占小区总出行需求的百分比
√		NVEHICLES	在停车场停车的且等待发车的实际总车辆数（见下面注解）
√		ZONE	小区编号

特征属性 NVEHICLES 考虑的是该停车场剩下的交通需求总量，与用户定义的停车场初始停车场占有率无关。另一方面，属性 OCCUPANCY 不考虑交通需求，只考虑用户定义的初始停车场占有率减掉要出发的车辆再加上到达的车辆。因此，它可以用来决定一个停车场是否已停满。这两个特征属性，和 Zone 以及 Relatvie flow 一起，仅能够用于作为“小区连接器”或“抽象的停车空间”的停车场，当不能用于实际的停车空间。

3.4.1.9 ReducedSpeedAreas



ReducedSpeed-Areas 对象是 Reduced-SpeedArea 对象（见第 69 页）的集合。它隶属于 Net 对象，并且可以通过 INet 接口进行 ReducedSpeedAreas 属性操作。

它包含了路网中的所有减速区域，并同时允许集合内部的对象进行循环访问，以及对某个内部对象进行单独访问。（同样可参见 ReducedSpeedArea 对象）

例子

创建 ReducedSpeedAreas 对象实例，并对其内部所有的 ReducedSpeedArea 对象进行访问：

```
DIM vissim As Vissim
DIM speedareas As ReducedSpeedAreas
DIM speedarea As ReducedSpeedArea
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET speedareas = Vissim.Net.ReducedSpeedAreas
FOR EACH speedarea IN speedareas ‘进入且创建一个枚举
…
NEXT speedarea
‘or also:
FOR i = 1 TO speedareas.Count
SET speedarea = speedareas(i) ‘or speedareas.Item(i)
…
NEXT i
```

IReducedSpeedAreas 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个包含所有 ReducedSpeedArea 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetReducedSpeedAreaByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 循环语句来进行访问。在 Visual Basic 中，你可以使用 FOR EACH…NEXT 语句来进行访问，该方法不需定义_NewEnum 属性，VB 可以内部读取该属性。（见如上 ReducedSpeedAreas 的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IReducedSpeedArea **ppSpeedArea)

返回集合中选中位置上的 ReducedSpeedArea 对象。这只有在所有减速区域都可以访问的情况下有效（见上文中的 ReducedSpeedAreas 一例）。通过 GetReducedSpeedAreaByNumber 可以根据标识编号对减速

区域进行选择。由于 Item 是集合的默认属性，以下的两个命令代表一样的含义：

SET speedarea = speedareas(1)

SET speedarea = speedareas.Item(1)

参数

[in] long Index: 1 到 Count 之间的索引

[out, retval] IReducedSpeedArea **ppSpeedArea: 返回 ReducedSpeedArea。

Count ([out, retval] long *pCount)

返回集合中 ReducedSpeedArea 对象数量。见上文中 ReducedSpeedAreas 一例。

参数

[out, retval] long *pCount : 返回对象数量

IReducedSpeedAreas 接口的方法

GetReducedSpeedAreaByNumber ([in] long Number, [out, retval] IReducedSpeedArea ** ppSpeedArea)

返回标识编号为 Number 的 ReducedSpeedArea 对象。

参数

[in] long Number : 标识编号

[out, retval] IReducedSpeedArea ** ppSpeedArea: 返回 ReducedSpeedArea。

例子

SET speedarea = speedareas.GetReducedSpeedAreaByNumber (2)

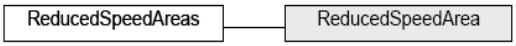
IF NOT (speedarea IS NOTHING) THEN

name = speedarea.AttValue("NAME")

END IF

3.4.1.10 ReducedSpeedArea

一个 ReducedSpeedArea 对象代表了一个减速区域元素，其隶属于 ReducedSpeedAreas 对象。它可以使用以下两种方法通过 ReducedSpeedAreas 来对其进行访问：



► 通过集合进行循环访问

DIM speedarea As ReducedSpeedArea

FOR EACH speedarea IN vissim.Net.ReducedSpeedAreas

List.AddItem speedarea.ID

NEXT speedarea

► 通过标识编号对具体的 ReducedSpeedArea 对象进行访问

```
DIM speedarea As ReducedSpeedArea
SET speedarea = vissim.Net.ReducedSpeedAreas.
GetReducedSpeedAreaByNumber (12)
```

ReducedSpeedArea 对象能够通过 IReducedSpeedArea 接口对减速区属性进行访问。

IReducedSpeedArea 接口的属性

ID ([out, retval] long *pID)

返回减速区的编号。

参数

[out, retval] long *pID : 返回标识编号。

例子

```
id = speedarea.ID
```

Name ([out, retval] BSTR *pName)

返回减速区名称。

参数

[out, retval] BSTR *pName : 返回名称。

例子

```
name = speedarea.Name
```

Name ([in] BSTR Name)

设置减速区名称，最多 255 个字符。

参数

[in] BSTR Name : 新名称

例子

```
speedarea.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个减速区域属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回属性值。

参数

```
name = speedarea.AttValue(„NAME“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个减速区域属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据属性类型）

参数

speedarea.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称
√	√	TIMEFROM	减速区域激活的开始时间[s]
√	√	TIMEUNTIL	减速区域激活的结束时间 [s]
√	√	DESIREDSPEED	第一个车辆类别的期望车速分布编号
√		VEHICLECLASSES	受影响的车辆类别（以数组形式返回其编号）
√		VEHICLETYPES	受影响的车辆类型（以数组形式返回其编号）

在设置时间间隔时，TIMEUNTIL 不能小于 TIMEFROM 的值。假设当前的时间间隔是[100,1000]，而用户需要将其设置为[1100,2000]，此时就需要先将 TIMEUNTIL 设置成 2000，再将 TIMEFROM 设置为 1100，不然会报出错误信息。

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

返回减速区域的属性，同时带一个参数。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[out, retval] VARIANT *pValue : 返回属性值

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

用一个参数设置减速区域的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[in] VARIANT Value : 属性值（根据属性类型）

属性览表:

R	W	属性	描述
√	√	DECELERATION	最大减速度（以当前的单位） 参数：车辆类别编号
√	√	DESIREDSPEED	期望速度分布编号。 参数：车辆类别编号
√	√	VEHICLECLASS	若该车辆类别受影响，为 TRUE 参数：车辆类别编号
√		VEHICLETYPE	若该车辆类型受影响，为 TRUE 参数：车辆类型编号

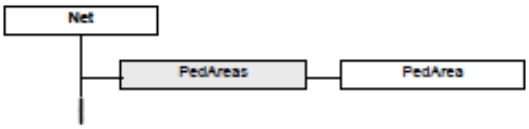
所作的改变在设置完成后，在仿真运行中立即生效。

3.4.2 Area-based（行人模块）

3.4.2.1 PedAreas

PedAreas 对象是 PedArea 对象的一个集合（参见第 74 页）。

它属于 Net 对象并可以通过 INet 接口的 PedAreas 属性来访问。



它包含了路网的所有的行人面域，可以通过循环的方式或者单独访问的方式获得某个 PedArea 对象（同样可参见 PedArea 对象）。

例子

创建 PedAreas 对象的实例，访问所有 PedArea 对象：

```
Dim vissim As Vissim
Dim pedareas As PedAreas
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set pedareas = Vissim.Net.PedAreas
FOR EACH pedarea IN pedareas ‘通过_NewEnum 方法进行枚举访问
…
NEXT pedarea
‘或者也可以:
FOR i = 1 TO pedareas.Count
SET pedarea = pedareas(i) ‘or pedareas.Item(i)
…
NEXT i
```

IPedAreas 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 PedArea 对象的集合（或枚举）。一旦集合被创建，运用 PedAreaByNumber 方法就能返回某个单独对象，而整个集合可以通过使用 FOR ...TO ... NEXT 循环语句来访问。用户可以在 Visual Basic 的环境下也可以运用 FOR EACH…NEXT 语句，该方法不需说明 _NewEnum，VB 会在程序内部调用它（参见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum ： 返回枚举对象

Item ([in] VARIANT Index, [out, retval] IPedArea **ppLink)

返回集合中某个特定位置的单个 PedArea。只有当所有的行人面域都可以访问的时候，该属性才有用（见上面的例子）。为了选择某个单独的 PedArea，需要使用 PedAreaByNumber（见下）方法。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET pedarea = pedareas(1)
SET pedarea = pedareas.Item(1)
```

参数

[in] long Index:	编号应该位于 1 和 PedAreas.Count 之间。
[out, retval] IPedArea **ppPedArea:	返回 PedArea 对象。

Count ([out, retval] long *pCount)

返回该集合中 PedArea 对象的数量。见上面的例子。

Parameters
[out, retval] long *pCount : 返回对象的数量.

IPedAreas 接口的使用方法

PedAreaByNumber ([in] long Number, [out, retval] IPedArea **ppPedArea)

通过编号 Number 返回 PedArea 对象。

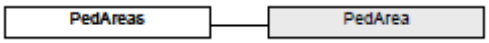
参数

[in] long Number:	行人面域的编号
[out, retval] IPedArea **ppPedArea:	返回 PedArea 对象

例子

```
SET pedarea = pedareas.PedAreaByNumber(1001)
```

3.4.2.2 PedArea



一个 PedArea 对象代表了一个行人面域元素，隶属于 PedAreas 对象。通过 PedAreas 有两种方式访问 PedArea 对象：

► 通过在集合中的循环获得

```
DIM pedareas As PedAreas
DIM pedarea As PedArea
SET pedareas = Vissim.Net.PedAreas
FOR EACH pedarea IN pedareas
List.AddItem pedarea.ID
NEXT pedarea
```

► 通过具体的编号获得

```
DIM pedareas As PedAreas
DIM pedarea As PedArea
SET pedareas = Vissim.Net.PedAreas
SET pedarea = pedareas. PedAreaByNumber(1000)
```

通过 IPedArea 接口，PedArea 对象能够访问行人面域的各个属性。

IPedArea 接口的属性

ID ([out, retval] long *pID)

返回行人面域的编号。

参数

[out, retval] long *pID： 返回编号（如果引用无效则返回 0）

例子

```
DIM pedarea AS PedArea
id = pedarea.ID
```

Name ([out, retval] BSTR *pName)

返回行人面域的名称。

参数

[out, retval] BSTR *pName： 返回名称

例子

```
name = pedarea.Name
```

Name ([in] BSTR Name)

设置行人面域的名称。最大 255 个字符。

参数

[in] BSTR Name： 新名称.

例子

```
pedarea.Name = “Picadilly Circus“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回行人面域的一个属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下表）

[out, retval] VARIANT *pValue： 返回属性值

例子

```
polyline = pedarea.AttValue(„POINTS“)
FOR i = LBOUND(polyline) TO UBOUND(polyline)
x = polyline (i).X
y = polyline
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置行人面域的属性值。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute： 属性名称（如下表）

[in] BSTR Value： 属性值（类型根据属性的定义）

例子
pedarea.AttValue(„NAME“)=”Times Square“

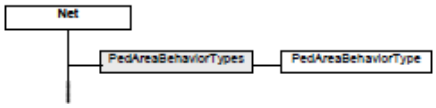
属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称

IPedArea 接口的方法

3.4.3 PedAreaBehaviorTypes

PedAreaBehaviorTypes 对象是 PedAreaBehaviorType 对象的一个集合（参见第 78 页）。它属于 Net 对象并可以通过 INet 接口的 PedAreaBehaviorTypes 属性来访问。



它包含了路网的所有的面域行为类型，可以通过循环的方式或者特定的方式获得某个 PedAreaBehaviorType 对象（同样可参见 PedAreaBehaviorType 对象）。

例子

创建 PedAreaBehaviorTypes 对象的实例，访问所有 PedAreaBehavior 对象：

```
DIM vissim As Vissim
DIM abts As PedAreaBehaviorTypes
DIM abt As PedAreaBehaviorType
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set abts = Vissim.Net.PedAreaBehaviorTypes
FOR EACH abt IN abts ‘进入_NewEnum，创建一个枚举
…
NEXT abt
‘or also:
FOR i = 1 TO abts.Count
SET abt = abts(i) ‘or abts.Item(i)
…
NEXT i
```

IPedAreaBahaviorTypes 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 PedAreaBahaviorType 对象的集合（或枚举）。一旦集合被创建，运用 PedAreaBehaviorTypeByNumber 方法就能返回某个单独对象，而整个集合可以通过使用 FOR ...TO ... NEXT 循环语句来访问。用户可以在 Visual Basic 的环境下也可以运用 FOR EACH…NEXT 语句，该方法不需说明_NewEnum，VB 会在程序内部调用它（参见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举对象

Item ([in] VARIANT Index, [out, retval] IPedAreaBehaviorType **ppABT)

返回集合中某个特定位置的单个 PedAreaBehaviorType。只有当所有的面域行为类型都可以访问的时候，该属性才有用（见上面的例子）。若要选择某个单独的面域行为类型，需要使用 PedAreaBehaviorTypeByNumber（见下）方法。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET abt = abts(1)
SET abt = abts.Item(1)
```

参数

[in] long Index :	编号应在 1 和 Count 之间取值
[out, retval] IPedAreaBehaviorType **ppABT:	返回 PedAreaBehaviorType 对象

Count ([out, retval] long *pCount)

返回该集合中 PedAreaBahaviorType 对象的数量。见上面的例子。

参数

[out, retval] long *pCount : 返回对象的数量.

IPedAreaBehaviorTypes 接口的使用方法

PedAreaBehaviorTypeByNumber ([in] long Number, [out, retval] IPedAreaBehaviorType **ppABT)

通过编号 Number 返回 PedAreaBehavior 对象。

参数

[in] long Number : 对象编号

[out, retval] IPedAreaBehaviorType **ppABT : 返回 PedAreaBehaviorType 对象

例子

```
SET abt = abts.PedAreaBehaviorTypeByNumber(2)
IF NOT (abt IS NOTHING) THEN
```

```
name = abt.AttValue("NAME")
END IF
```

3.4.4 PedAreaBehaviorType

PedAreaBehaviorTypes 对象是 PedAreaBehaviorType 对象的一个集合。它属于 Net 对象并可以通过 INet 接口的 PedAreaBehaviorTypes 属性来访问。

通过 PedAreaBehaviorTypes 有两种方式访问 PedAreaBehaviorType 对象：

► 通过在集合中的循环获得

```
DIM abt As PedAreaBehaviorType
FOR EACH abt IN vissim.Net.PedAreaBehaviorTypes
List.AddItem abt.ID
NEXT abt
```

► 通过具体的编号获得

```
DIM abt As PedAreaBehaviorType
SET abt = vissim.Net.PedAreaBehaviorTypes.PedAreaBehaviorTypeByNumber(2)
```

通过 IPedAreaBehaviorType 接口，PedAreaBehaviorType 对象能够访问行人面域的各个属性。

IPedAreaBehaviorType 接口的属性

ID ([out, retval] long *pID)

返回面域行为类型的编号

参数

[out, retval] long *pID : 返回编号

例子

```
id = abt.ID
```

Name ([out, retval] BSTR *pName)

返回面域行为类型的名称

参数

[out, retval] BSTR *pName : 返回名称.

例子

```
name = abt.Name
```

Name ([in] BSTR Name)

设置面域行为类型的名称，最大不超过 255 个字符。

参数

[in] BSTR Name : 新名称.

例子

abt.Name = „XXX“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回面域行为类型的一个属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下表）

[out, retval] VARIANT *pValue : 返回该属性的值

例子

name = abt.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置面域行为类型的一个属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名称（如下）

[in] BSTR Value : 属性值 (类型与所指属性一致)

例子

abt.AttValue(„NAME“)=„XXX“

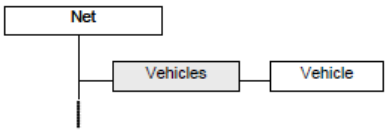
属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称

3.5 Traffic

3.5.1 Vehicles

3.5.1.1 Vehicles



Vehicles 对象是 Vehicle 对象的一个集合（参见第 86 页）。它属于 Net 对象并可以通过 INet 接口的 PedAreas 属性来访问。

它包含了仿真进行时路网中当前在行驶的所有车辆，可以通过循环的方式或者单独访问的方式获得某个 Vehicle 对象。

例子

实例化一个 Vehicles 对象并访问其中每一个 Vehicle 对象：

```
DIM vissim As Vissim
DIM vehicles As Vehicles
DIM vehicle As Vehicle
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set vehicles = Vissim.Net.Vehicles
FOR EACH vehicle IN vehicles ‘访问_NewEnum 属性建立枚举
…
NEXT vehicle
‘or also:
FOR i = 1 TO vehicles.Count
SET vehicle = vehicles(i) ‘或者是 vehicles.Item(i)
…
NEXT i
```

IVehicles 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 Vehicle 对象的集合（或枚举）。一旦集合被创建，运用 GetVehicleByNumber 的方法就能返回其中的具体对象，而整个集合可以通过使用 FOR ...TO ... NEXT 语句循环访问。用户可以在 Visual Basic 的环境下运用 FOR EACH…NEXT 语句，不需定义_NewEnum，因为 VB 会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum： 返回枚举对象

Item ([in] VARIANT Index, [out, retval] IVehicle **ppVehicle)

返回集合某个位置的单个 Vehicle 对象。只有当所有的车辆都可以获得的时候，该属性才有用（见上面的例子）。为了根据定义编号选择一个车辆，需要使用 GetVehicleByNumber（见下）来进行索引。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET vehicle = vehicles(1)
SET vehicle = vehicles.Item(1)
```

参数

[in] long Index：	编号应介于 1 和 vehicles.Count 之间
[out, retval] IVehicle **ppVehicles：	返回 Vehicle 对象

Count ([out, retval] long *pCount)

返回集合中的 Vehicle 对象的数量。详见上例。

参数

[out, retval] long *pCount：返回对象的数量。

IDs ([in, defaultvalue(0)] BSTR Attribute, [in, defaultvalue(0)] VARIANT Value, [out, retval] VARIANT *pIDs)

若指定了某个车辆属性的一个特定值（见第 88 页的属性览表），根据条件“Attribute = Value”返回所有符合条件的车辆编号的数组。否则返回集合中所有车辆编号的数组。

参数

[in] BSTR Attribute：	属性名称（详见 IVehicle 属性览表）。
[in] VARIANT Value ：	特征属性值或数组的值（类型根据特征属性的定义）。
[out, retval] VARIANT *pIDs ：	返回 IDs 数组。

例子

```
ids = vehicles.IDs(„TYPE“, 1)  ‘属于车辆类型为 1 的所有车辆的 ID 数组
```

IVehicles 接口的方法

GetVehicleByNumber ([in] long Number, [out, retval] IVehicle **ppVehicle)

根据编号，返回 Vehicle 对象。

参数

[in] long Number：	编号
[out, retval] IVehicle **ppVehicle：	返回 Vehicle 对象。

例子

```
DIM vehicle AS Vehicle
SET vehicle = vehicles.GetVehicleByNumber (11)
IF NOT (vehicle IS NOTHING) THEN
speed = vehicle.AttValue("SPEED")
END IF
```

GetMultiAttValues ([in] VARIANT IDs, [in] BSTR Attribute, [out] VARIANT *pValues)

输入一组车辆编号 IDs，返回其所对应的属性值来填充数组 pValues（若要返回集合中的所有车辆的相关属性，可以将 IDs 变量设为 Empty 来得到）。请在第 88 页的表中了解各个属性标签的具体内容，这部分与内容与语言无关。

参数

[in] VARIANT IDs：	vehicle 的一组编号数组或者 vehicle 的一个编号
[in] BSTR Attribute：	属性名（见 IVehicle 属性览表)
[out] VARIANT *pValues：	返回数组值

例子

```
vehs.GetMultiAttValues（Array(1, 2, 3, 4, 5), "SPEED", values)
For i = LBound(values) To UBound(values)
val = values (i)
Next i
```

SetMultiAttValues ([in] VARIANT IDs, [in] BSTR Attribute, [in] VARIANT Values)

根据一组车辆编号 IDs，对其相应的属性值进行设置（使用变量 Empty 来对集合中的所有车辆进行设置）。如果设置的值不是一组数组，而是一个值，则该值会赋值给每一个车辆的相应属性。请在第 88 页的表中了解与所设语言无关的各个属性标签的具体内容。

参数

[in] VARIANT IDs：	vehicle 的一组编号或者一个 vehicles 的编号
[in] BSTR Attribute :	属性名（见 IVehicle 属性览表)
[in] VARIANT Values :	数组值或一个属性值（类型根据特征属性的定义）

例子

```
vehs.SetMultiAttValues Empty, "COLOR", RGB(255, 0, 0)
```

AddVehicleInZone ([in] long Type, [in] long ZoneNr, [out, retval] IVehicle **ppVehicle)

该方法在一个特定编号的小区中添加一个 Type 类型的车辆。车辆将会根据相关的流量流向的定义，在该小区的所有停车场中进行选择。如果

所给 Type 不是一个有效的类型或者小区编号有错误，将返回一个错误信息，不会生成车辆。车辆将把当前的仿真时刻作为它的出发时刻（如果没有运行仿真，则赋值为 0），一旦它已经被分配到了一个目的地小区，目的停车场，或者一个路径（参见第 88 页的特征属性"DESTZONE","DESTPARKLOT"和"PATH"），它将立即出发。

参数

[in] long Type： 当前路网存在的车辆类型

[in] long ParkingID： 停车场编号

[out, retval] IVehicle **ppVehicle： 返回 Vehicle 对象

例子

```
DIM vehicle AS Vehicle
vehicle = vehicles.AddVehicleInZone (1, 2)
```

AddVehicleInParkingLot ([in] long Type, [in] long ParkingID, [out, retval] IVehicle **ppVehicle)

该方法添加一个 Type 类型的车辆，出发前往编号为 ParkingID 的停车场中。如果 Type 不是一个有效的类型或者停车场编号有错误，将返回一个错误信息，不会生成车辆。车辆将把当前的仿真时刻作为它的出发时间（如果当前没有仿真在运行中，则赋值为 0），一旦它已经被分配到了一个路径（参见第 88 页的特征属性"PATH"），它将立即出发。

参数

[in] long Type : 当前路网中存在的车辆类型

[in] long ParkingID : 停车场标识编号

[out, retval] IVehicle **ppVehicle : 返回 Vehicle 对象。

例子

```
DIM vehicle AS Vehicle
vehicle = vehicles.AddVehicleInParkingLot (1, 10)
```

AddVehicleAtLinkCoordinate ([in] long Type, [in] double DesiredSpeed, [in] long Link, [in] int Lane, [in] double XCoord, [in, defaultvalue(1)] BYTE Interaction, [out, retval, defaultvalue(0)] IVehicle **ppVeh)

该方法将在特定路段 Link 的特定车道 Lane 上，位于路段坐标 XCoord 处，添加一个 Type 类型的车辆。该方法的标准模式将让新生成的车辆考虑到它的相邻车辆的影响，根据当前的情况和 DesiredSpeed 设定自己的速度。同时，该方法也可以把相互影响设置为否（0），则生成的车辆将不考虑当前的车流情况，直接按照期望速度 DesiredSpeed 行驶。与方法 AddVehicleInZone 或者 AddVehicleParkingLot 不同，以这种方式生成的车辆不按照一个特定的路径行驶，它们将在下一个仿真秒出发。因此，如果在没有仿真运行的情况下调用该方法，将导致错误。

参数

[in] long Type： 当前路网存在的车辆类型

[in] double DesiredSpeed： 期望车速(如果不选择 interaction，就是车辆的速度)

[in] long Link： 路段编号

[in] int Lane： 车道号码（根据路段，从 1 到 n）

[in] double XCoord : 路段的坐标值（从 0.0 到 link length）

[in] BYTE Interaction: 有相互作用的标志(默认状态下是 true)

[out, retval] IVehicle **ppVeh： 返回 Vehicle 对象

例子

```
DIM vehicle AS Vehicle
vehicle = vehicles.AddVehicleAtLinkCoordinate (1, 40.0, 1000, 1, 20.0)
```

AddVehicleInTransitLine ([in] long Type, [in] long TransitLineNr, [out, retval] IVehicle **ppVehicle)

该方法添加一个新的车辆类型到一个特定的公交线路 TransitLine 上去。如果 Type 不是一个有效的类型或者公交线的编号有错误，将返回一个错误信息，不会生成车辆。该车辆将在下一个仿真秒出发。

参数

[in] long Type： 当前路网存在的车辆类型

[in] long TransitLineNr： 公交线路编号

[out, retval] IVehicle **ppVehicle： 返回 Vehicle 对象

例子

```
DIM vehicle AS Vehicle
vehicle = vehicles.AddVehicleInTransitLine(1,10)
```

RemoveVehicle ([in] long Number)

把对应车辆编号为 Number 的车辆从路网中移除，不管它是否处于停车状态。

参数

[in] long Number ： 车辆编号

例子

```
vehicles.RemoveVehicle (112)
```

GetQueued ([out, retval] IVehicles **ppVehicles)

这个方法可以用于返回所有处于排队状态中的车辆，返回值为一个 Vehicle 对象的集合。（根据当前的排队设置条件）。

参数

[out, retval] IVehicles **ppVehicles： 返回 Vehicles 对象

例子

```
DIM queued_vehicles AS Vehicles
queued_vehicles = vehicles.GetQueued
```

GetArrived ([out, retval] IVehicles **ppVehicles)

这个方法返回一个车辆对象的集合，该集合包含所有在最后一个仿真步长里到达目的地停车场的车辆（根据分配的路径）。通过 OD 矩阵生成的车辆，在到达目的停车场后就会消失，因此此方法无法得到这些车辆对象。

参数

[out, retval] IVehicles **ppVehicles : 返回 Vehicles 对象。

例子

```
DIM arrived_vehicles AS Vehicles
arrived_vehicles = vehicles.GetArrived
```

GetParked ([out, retval] IVehicles **ppVehicles)

这个方法将返回一个车辆集合，该集合包含有所有停在停车场中的车辆。（通过 OD 矩阵文件定义的车辆只能在其起始停车场中可见，而通过出行链定义或使用 COM 接口加入路网的车辆在终点停车场中则是可见的。）

参数

[out, retval] IVehicles **ppVehicles : 返回 Vehicles 对象。

例子

```
DIM parked_vehicles AS Vehicles
parked_vehicles = vehicles.GetParked
```

必要知识

Attributes

AttValue 方法中定义的属性与 IVehicle 接口中使用的属性是相同的。（见第 88 页的属性览表）。

Vehicles collections

车辆是动态的路网对象（也就是说在仿真过程中，他们会被创建，移除或更改）。因此，车辆集合也是动态的，并且在仿真过程中也会改变其内部的对象（车辆）以及对象的次序。在基于 COM 编程的环境下，这就意味着车辆的集合变量会根据路网状态进行变化。在下例中，车辆集合的变量 queued_vehicles 在仿真步长的前后就会包含有不同的车辆（即其中的 n_before 和 n_after 就会不同）。

```
DIM queued_vehicles AS Vehicles
queued_vehicles = vehicles.GetQueued
n_before = queued_vehicles.Count
sim.RunSimulationStep
n_after = queued_vehicles.Count
```

AddVehicleAtLinkCoordinate

在向某个路段坐标添加车辆时，该车辆在下一个仿真步长后就会出现在该坐标上。此时可以通过 Igraphics 接口的 Redraw 方法查看这些车辆。

Virtual memory option

动态交通分配选项中的 Virtual memory 复选框，仅针对停车车辆，对于通过 AddVehicleInParkingLot()方法添加的车辆不会造成影响。

3.5.1.2 Vehicle

Vehicle 对象代表单个车辆，隶属于 Vehicles 集合。它可以使用以下两种方法通过 Vehicles 对象来获得：

► 通过集合的迭代

```
DIM vehicle As Vehicle
FOR EACH vehicle IN vissim.Net.Vehicles
List.AddItem vehicle.ID
NEXT vehicle
```

► 通过标识编号获得具体的车辆

```
DIM vehicle As Vehicle
SET vehicle = vissim.Net.Vehicles.GetVehicleByNumber (101)
```

Vehicle 对象可以通过 IVehicle 接口访问车辆的各个属性。但是在执行仿真步骤后，实例化 Vehicle 对象并不一定是 VISSIM 中有效车辆，（请参阅本节末车辆动态方面的说明）。此时就需要使用属性标识编号（如下），检查其有效性。

IVehicle 接口的属性

ID ([out, retval] long *pID)

返回车辆标识编号。若 Vehicle 对象不是指向一个有效的 VISSIM 车辆，则返回值为 0。

参数

[out, retval] long *pID : 返回标识编号（如果没有指向有效车辆，则为 0）。

例子

```
id = vehicle.ID
```

Name ([out, retval] BSTR *pName)

返回车辆名称。

参数
[out, retval] BSTR *pName : 返回名称。

例子
name = vehicle.Name

Name ([in] BSTR Name)

设置车辆名称。

参数
[in] BSTR Name : 新名称

例子
vehicle.Name = “KA-LK 240“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个车辆属性。请在该节结尾的表中，请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回属性值。

例子
speed = vehicle.AttValue(„SPEED“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个车辆属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据属性类型）

例子
vehicle.AttValue(„PATH“)= 1

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称
√	√	BOARDEDPASS	上车乘客数
√	√	COLOR	RGB 格式的色彩
√	√	ALIGHTEDPASS	下车乘客数
√		DESLANECHANGE	期望车道变化（1=左，0=无，-1=右）
√	√	DESIREDSPEED	当前单位选项下的期望车速
√	√	DESSPEEDFRACTIL	随机选择（0，1）中的一个分值，用于期望车速选择
√	√	DESTPARKLOT	终点处停车场编号（若无则为 0）
√	√	DESTZONE	终点处小区编号（若无则为 0）
√	√	DWELLTIME	车辆在公交站点或者停车标志处的停留时间[s]
√		ELAPSEDTIME	路网总时间 [s]
√	√	INTERACTION	车辆在小区见交互的标志（true/false）
√	√	LANE	定位车辆的当前车道
√		LANECHANGE	当前车道变化（1=左，0=无，-1=右）
√		LASTLANECHANGE	自最后一次车道变化开始的时刻[s]
√		LASTNODE	上一个驶过的节点（若无，则为 0）
√	√	LENGTH	实际车辆长度（当前单位设定）
√	√	LINK	定位车辆的当前路段
√	√	LINKCOORD	车辆在当前路段的路段坐标
√		NEXTNODE	下一个驶过的节点（若无，则为 0）
√		ORIGPARKLOT	起始停车场编号（若无，则为 0）
√		ORIGZONE	起始小区编号（若无，则为 0）
√	√	PARKLOT	车辆停放的停车场编号
√	√	PASSENGERS	车辆中乘客数
√	√	PATH	当前所用路径编号（若无，则为 0）
√		POINT	车辆位置的坐标（详见下列标注）
√		PRECEDING	下游的相邻车辆数（不一定相关）
√		QUEUECOUNTER	遇到排队的次数

R	W	属性	描述
√		ROUTE	该车辆的路径编号
√		ROUTINGDECISION	当前路径所属于的路径决策点
√	√	SPEED	当前选项下的车速
√		TOTALDISTANCE	路网中总行驶距离（当前单位设置）
√		TRALING	上游的相邻车辆数（不一定相关）
√	√	TYPE	车辆类型编号
√	√	3DMODELSTATE	三维模型状态（属于行人种类的车辆除外）
√	√	WEIGHT	车辆总重 [mt]

见第 180 页 RGB 色彩格式的标注。

通过IWorldPoint接口可以设置world point的坐标，详见第185页的相关介绍。

IVehicle 接口的方法

MoveToLinkCoordinate ([in] long Link, [in] int Lane, [in] double XCoord)

该方法可以将该车辆对象放置在路段 Link 的 XCoord 路段坐标的 Lane 车道。该方法仅适用于行驶于路网上的车辆，若车辆处于停放状态，则不起作用。

参数

[in] long Link : 路段编号

[in] int Lane : 车道编号（根据路段从 1 到 n）

[in] double XCoord : 路段坐标（从 0.0 到路段长度）

例子

```
DIM vehicle AS Vehicle
SET vehicle = vehicles.GetVehicleByNumber (12)
vehicle.MoveToLinkCoordinate (1000, 1, 20.0)
```

必要知识

属性

LINK: 修改该属性的效果与 MoveToLinkCoordinate 方法在分配的路段、当前车道及路段坐标上的应用相同。

DESTZONE: 修改该属性，可以在动态分配中，根据停车场选择模型，分配给车辆到达目的地停车场的一个动态分配行驶路径。如果没有可选路径，车辆将不会出发。

DESTPARKLOT：修改该属性，可分配给车辆到达指定停车场的动态路径集中的一条路径。如果没有可选路径，车辆将不会出发。

PATH：该属性用于路径重新选择时，可以用新的路径分配覆盖当前路径分配的情况，也可用于车辆行驶在路径重叠区域上更改其路径选择的情况。

LINKCOORD：修改该属性，可以和 MoveToLinkCoordinate 方法一样在当前路段和车道下将路段坐标更改为分配的值。

PARKLOT：修改该属性，可以将停靠的车辆移动至指定的停车场编号。但对于行驶中的车辆，则不起作用。

INTERACTION：关闭 interaction 属性，将使得车辆以恒定的速度行驶，而不受前方车辆的影响。

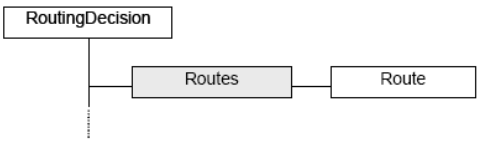
车辆是一个动态路网元素

车辆是一个动态路网元素。（也就是说，他们在仿真运行中被创建、移动与改变）。在 COM 客户端编程的环境中，这意味着车辆变量是指向当前路网状态的。在下面的例子中，速度变化在仿真后未必会保持一致（即之前的速度与滞后的速度可能有所不同 ）：

```
DIM veh AS Vehicle
SET veh = vehicles.GetVehicleByNumber(1)
speed_before = veh.AttValue(“SPEED”)
sim.RunSimulationStep
speed_after = veh.AttValue(“SPEED”)
```

3.5.2 Private Traffic

3.5.2.1 Routes



Routes 对象是 Route 对象（参见第 93 页）的集合。它隶属于 RoutingDecision 对象，可以通过 IRoutingDecision 接口对 Routes 属性进行访问。

它包含了路径决策中的所有路线，同时允许通过枚举来进行访问和单独对其访问（同见 Route 对象）。

例子

创建 Routes 对象实例并访问所有的 Route 对象。

```
DIM vissim As Vissim
DIM routes As Routes
DIM route As Route
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET routes = Vissim.Net.RouteDecisions(1).Routes
FOR EACH route INroutes ‘进入_NewEnum，创建一个枚举
…
NEXT route
‘or also:
FOR i = 1 TO routes.Count
SET route = routes (i) ‘or routes.Item(i)
…
NEXT i
```

IRoutes 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

通过该属性可以创建一个所有路径对象的集合（或枚举）。一旦集合被创建，就可以通过 GetRouteByNumber 方法，使用 FOR ...TO ... NEXT 语句单独访问集合中的对象。在 Visual Basic 中，你不必定义属性 _NewEnum 就可以直接使用 FOR EACH…NEXT 语句，因为 VB 可以内部调用该属性。 (见上文中的例子)。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举对象

Item ([in] VARIANT Index, [out, retval] IRoute **ppRoute)

返回集合中选中位置的某个路径对象。这只有在所有路径可访问的情况下有效（见上文中的 Routes 一例）。通过 GetRouteByNumber 可以根据表示编号对路径进行选择（如下）。由于 Item 是集合的默认属性，因此下述两个命令是一样的：

```
SET route = routes(1)
SET route = routes.Item(1)
```

参数

[in] long Index : 1 和 Count 之间的索引

[out, retval] IRoute **ppRoutes: 返回 Route 对象。

Count ([out, retval] long *pCount)

返回集合中的 Route 对象数量。见上例。

参数

[out, retval] long *pCount : 返回对象数。

IRoutes 接口的方法

GetRouteByNumber ([in] long Number, [out, retval] IRoute **ppRoute)

返回标识编号 Number 对应的 Route 对象。

参数

[in] long Number : 标识编号

[out, retval] IRoute **ppRoute : 返回 Route 对象。

例子

```
SET route = routes.GetRouteByNumber(2)
IF NOT (route IS NOTHING) THEN
rf = route.AttValue("RELATIVEFLOW")
END IF
```

AddRoute ([in] long Link, [in] double Xcoord, [out, retval] long *pNumber)

在指定路段坐标下，增加一个新的路径，默认每段时间间隔的相对流为 1.0。如果成功的话，它将返回分配的编号，否则为 0。

参数

[in] long Link : 目标路段的标识编号

[in] double XCoord : 目标路段坐标

[out, retval] long *pNumber : 返回路径标识编号。

例子

```
id = decisions.AddStaticRoutingDecision(1, 100.0)
SET decision = decisions.GetRoutingDecisionByNumber (id)
IF NOT (decision IS NOTHING) THEN
decision.AddRoute(2, 200.0)
END IF
```

RemoveRoute ([in] long Number)

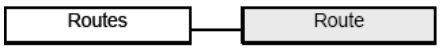
移除特定标识编号对应的路径。

参数

[in] long Number : 标识编号

3.5.2.2 Route

Route 对象代表了路径决策的一个路径元素，隶属于 Routes 对象。可以使用以下两种方法通过 Routes 对象对其进行访问。



► 通过集合的内部循环进行访问

```
DIM routes As Routes
DIM route As Route
SET routes = Vissim.Net.RouteDecisions(1).Routes
FOR EACH route IN routes
List.AddItem route.ID
NEXT route
```

► 通过编号进行单独访问

```
DIM route As Route
SET route = vissim.Net.RouteDecisions(1).Routes.GetRouteByNumber (12)
```

Route 对象能够通过 IRoute 接口对路径的各个属性进行访问。

IRoute 接口的属性

ID ([out, retval] long *pID)

返回路径编号。

参数

[out, retval] long *pID : 返回标识编号。

Example

```
id = route.ID
```

Name ([out, retval] BSTR *pName)

返回路径名称。

参数

[out, retval] BSTR *pName : 返回名称。

Example

```
name = route.Name
```

Name ([in] BSTR Name)

设置路径名称。

参数

[in] BSTR Name : 新名称

Example

```
route.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（见下表）

[out, retval] VARIANT *pValue : 返回属性值。

Example

name = route.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据属性类型）

Example

route.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	标识编号
	√	NAME	名称

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带一个参数返回一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[out, retval] VARIANT *pValue : 返回属性值。

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带一个参数设置一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

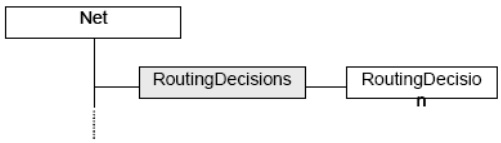
[in] VARIANT Value : 属性值（根据属性类型）

属性览表:

R	W	属性	描述
√	√	RELATIVEFLOW	时间间隔内的相对流量 参数：时间间隔编号 (1..n)

所作的改变会在实行仿真时立即生效。

3.5.2.3 RoutingDecisions



Routing-Decisions 对象是 RoutingDecision 对象（见第 97 页）的集合。

它隶属于 Net 对象，可以通过 INet 接口对 RoutingDecisions 对象进行访问。它包含了路网中所有的路径决策，并且可以通过循环对所有的对象进行访问，同时也可以单独地对具体对象进行访问。（同见 RoutingDecision 对象）。

例子

创建一个 RoutingDecisions 对象的实例，并访问其所有的 RoutingDecision 对象：

```
DIM vissim As Vissim
DIM decisions As RoutingDecisions
DIM decision As RoutingDecision
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET decisions = Vissim.Net.RoutingDecisions
FOR EACH decision IN decisions ‘进入_NewEnum，创建一个枚举
…
NEXT decision
‘or also:
FOR i = 1 TO decisions.Count
SET decision = decisions(i) ‘or decisions.Item(i)
…
NEXT i
```

IRoutingDecisions 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性可以创建所有路径决策对象的集合（枚举）。一旦集合被创建，可以使用 GetRoutingDecisionByNumber 方法，通过 FOR ...TO ... NEXT 语句返回集合中的具体对象。在 Visual Basic 中，用户可以直接使用 FOR ...TO ... NEXT 语句，而不用定义_NewEnum 属性，因为 VB 可以内部调用该属性（见上文中的 RoutingDecisions 一例）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举。

Item ([in] VARIANT Index, [out, retval] IRoutingDecision **ppDecision)

该方法用于返回集合中所选位置的路径决策。这个方法只在所有的路径决策可使用时才有效（见上文中的 RoutingDecisions 一例）。通过 GetRoutingDecisionByNumber 方法可以根据表示编号对路径决策进行选择（如下）。由于 Item 是集合的默认属性，因此以下的语句是同一个意思：

SET decision = decisions(1)

SET decision = decisions.Item(1)

参数

[in] long Index :	介于 1 和 Count 之间的索引
[out, retval] IRoutingDecision **ppDecision:	返回 RoutingDecision 对象。

Count ([out, retval] long *pCount)

返回集合中 RoutingDecision 对象数量。见上例。

参数

[out, retval] long *pCount : 返回对象数。

IRoutingDecisions 接口的方法

GetRoutingDecisionByNumber ([in] long Number, [out, retval] IRoutingDecision **ppDecision)

通过编号返回 RoutingDecision 对象。

参数

[in] long Number :	编号
[out, retval] IRoutingDecision **ppDecision :	返回 RoutingDecision 对象。

例子

SET decision = decisions.GetRoutingDecisionByNumber (2)

IF NOT (decision IS NOTHING) THEN

name = decision.AttValue(“NAME”)

```
END IF
```

AddStaticRoutingDecision ([in] long Link, [in] double Xcoord, [out, retval] long *pNumber)

在指定路段坐标下，增加一个新的静态路径决策点，默认的时间间隔为 0-99999。如果成功的话，它将返回所分配的编号，否则为 0。

参数

[in] long Link : 路段标识编号

[in] double XCoord : 路段坐标

[out, retval] long *pNumber : 返回路径决策的标识编号。

例子

```
id = decisions.AddStaticRoutingDecision(1, 100.0)
SET decision = decisions.GetRoutingDecisionByNumber (id)
IF NOT (decision IS NOTHING) THEN
decision.AddRoute(2, 200.0)
END IF
```

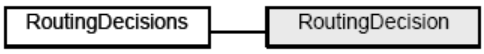
RemoveRoutingDecision ([in] long Number)

移除指定的标识编号对应路径选择。

参数

[in] long Number : 标识编号

3.5.2.4 RoutingDecision



RoutingDecision 对象代表了一个路径决策元素，隶属于 RoutingDecisions 对象。

它可以使用以下两种方法通过 RoutingDecisions 对象来访问：

► 通过集合内部的枚举访问

```
DIM decision As RoutingDecision
FOR EACH decision IN vissim.Net.RoutingDecisions
List.AddItem decision.ID
NEXT decision
```

► 通过标识编号单独访问具体的 RoutingDecision

```
DIM decision As RoutingDecision
SET decision = vissim.Net.RoutingDecisions.GetRoutingDecisionByNumber (12)
```

RoutingDecision 对象能够通过 IRoutingDecision 接口对路径决策的各属性进行访问。

IRoutingDecision 接口的属性

ID ([out, retval] longF *pID)

返回路径决策的标识编号。

参数

[out, retval] long *pID : 返回标识编号。

例子

```
id = decision.ID
```

Name ([out, retval] BSTR *pName)

返回路径决策的名称。

参数

[out, retval] BSTR *pName : 返回名称。

例子

```
name = decision.Name
```

Name ([in] BSTR Name)

设置路径决策的名称。最多为 255 个字符。

参数

[in] BSTR Name : 新名称

例子

```
decision.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（见下表）

[out, retval] VARIANT *pValue : 返回属性值。

例子

```
name = decision.AttValue(„NAME“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据属性类型）

例子

decision.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称
√		VEHICLECLASSES	受影响的车辆类别（数组）
√		VEHICLETYPES	受影响的车辆类型（数组）

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带一个参数返回一个路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[out, retval] VARIANT *pValue : 返回属性值。

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带一个参数设置一个路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[in] VARIANT Value : 属性值（根据属性类型）

属性览表:

R	W	属性	描述
√	√	RELATIVEFLOW	当前时间间隔的相对流量（若无仿真正运行，则为初始的值） 参数：路径决策编号
√	√	TIMEFROM	时间间隔的开始[s]。 参数：时间间隔顺序（1..n）
√	√	TIMEUNTIL	时间间隔的结束[s]。 参数：时间间隔顺序（1..n）
√	√	VEHICLECLASS	若车辆类别受影响，则为 TRUE。 参数：车辆类别的编号（所有类别的话，就为 0）
√		VEHICLETYPE	若车辆类型受影响，则为 TRUE。 参数：车辆类型的编号

仿真过程中所作的更改会即时在路网中产生作用。

在设置时间间隔时，TIMEUNTIL不能小于TIMEFROM 的值。假设当前的时间间隔是[100,1000]，而用户需要将其更改为[1100,2000]，此时就需要先将TIMEUNTIL设置成2000，再将TIMEFROM设置为1100，不然会报出错误信息。

AttValue2 ([in] BSTR Attribute, [in] VARIANT Parameter1, [in] VARIANT Parameter2, [out, retval] VARIANT *pValue)

带两个参数返回路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（见下表）

[in] VARIANT Parameter1 : 第一个与属性相关的参数

[in] VARIANT Parameter2 : 第二个与属性相关的参数

[out, retval] VARIANT *pValue : 返回属性值。

AttValue2 ([in] BSTR Attribute, [in] VARIANT Parameter1, [in] VARIANT Parameter2, [in] VARIANT Value)

带两个参数设置路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（见下表）

[in] VARIANT Parameter1 : 第一个与属性相关的参数

[in] VARIANT Parameter2 : 第二个与属性相关的参数

[in] VARIANT Value : 属性值（根据属性类型）

属性览表:

R	W	属性	描述
√	√	RELATIVEFLOW	对应时间间隔和路径决策编号的相对流量。 参数 1：路径决策编号 参数 2：所需时间间隔中的时间点

i 在仿真过程中，所作的更改会在仿真进行中立即产生作用。

Routes ([out, retval] IRoutes **ppRoutes)

创建一个 Routes 对象（见第 90 页）的实例，这样就可以单独地对路网中所定义的与该路径决策点相关的路径进行访问。

参数

[out, retval] IRoutes **ppRoutes : 返回 Routes 对象。

例子

```
DIM routes AS Routes
DIM decision As RoutingDecision
SET decision = vissim.Net.RoutingDecisions.GetRoutingDecisionByNumber (10)
SET routes = decision.Routes
```

IRoutingDecision 接口的属性

AddTimeInterval ([in] double From, [in] double To, [out, retval] long *pIndex)

该方法用于添加一个新的时间间隔。这个时间间隔只能添加到最后一个已有的时间间隔后面。如果在这个时间间隔和最后一个时间间隔之间存在空档，将会自动创建一个额外的时间间隔，从最后一个时间间隔结束开始，到新创建的时间间隔开始为止。如果添加成功，该方法会返回所分配的列表编号（1…n），否则为 0。

参数

[in] double From : 时间间隔起始时刻

[in] double To : 时间间隔结束时刻

[out, retval] long *pIndex : 返回 列表位置索引

例子

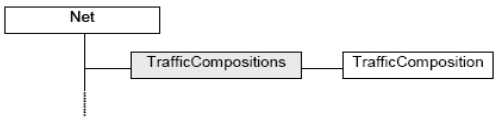
```
id = decisions.AddStaticRoutingDecision(1, 100.0)
```

```
SET decision = decisions.GetRoutingDecisionByNumber (id)
IF NOT (decision IS NOTHING) THEN
decision.AttValue1("TIMEUNTIL", 1) = 1799          '设置默认时间间隔为[0-1799]
index = decision.AddTimeInterval(1800, 3600)
END IF
```

3.5.2.5 TrafficCompositions

TrafficCompositions 对象是 TrafficComposition 对象（参见第 104 页）的集合。

它隶属于 Net 对象，能够通过 INet 接口进入属性 TrafficCompositions。它包含了路网中所有的车辆构成，能够通过循环在集合中或者通过具体的编号来获得每一个 TrafficComposition 对象。（同见 TrafficComposition 对象）。



例子

创建一个 TrafficCompositions 对象实例，并获得所有 TrafficComposition 对象：

```
DIM vissim As Vissim
DIM tcs As TrafficCompositions
DIM tc As TrafficComposition
SET vissim = NEW Vissim
vissim.LoadNet "c:\vissim\daten\example.inp"
Set tcs = Vissim.Net.TrafficCompositions
FOR EACH tc IN tcs   '进入_NewEnum 创建一个枚举
…
NEXT tc
'或者也可以这样：
FOR i = 1 TO tcs.Count
SET tc = tcs(i)   '或者 tcs.Item(i)
…
NEXT i
```

ITrafficCompositions 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 TrafficComposition 对象的集合（或枚举）。一旦集合被创建，运用 GetTrafficCompositionByNumber 的方法就

能返回个别对象，而整个集合可以通过使用 FOR ...TO ... NEXT 语句来循环访问。用户可以在 Visual Basic 的环境下运用 FOR EACH…NEXT 语句，不需说明_NewEnum，VB 会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum ：返回枚举

Item ([in] VARIANT Index, [out, retval] ITrafficComposition **ppTC)

返回集合某个位置的单个 TrafficComposition 对象。只有当所有的车辆构成都可以获得的时候，该属性才有用（见上面的例子）。为了选择一个车辆构成，需要使用 GetTrafficCompositionByNumber（见下）来进行索引。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

SET tc = tcs(1)

SET tc = tcs.Item(1)

参数

[in] long Index ： 索引介于 1 和 Count 之间

[out, retval] ITrafficComposition **ppTC：返回 TrafficComposition 对象

Count ([out, retval] long *pCount)

返回 TrafficComposition 对象的数量。参见上面的例子。

参数

[out, retval] long *pCount :：返回对象的数量。

ITrafficCompositions 接口的使用方法

GetTrafficCompositionByNumber ([in] long Number, [out, retval] ITrafficComposition **ppTC)

用编号返回 TrafficComposition 对象。

参数

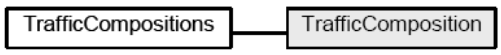
[in] long Number： 编号

[out, retval] ITrafficComposition **ppTC： 返回 TrafficComposition 对象。

例子

```
SET tc = tcs.GetTrafficCompositionByNumber(2)
IF NOT (tc IS NOTHING) THEN
name = tc.AttValue(“NAME”)
END IF
```

3.5.2.6 TrafficComposition



TrafficComposition 对象代表了一个车辆构成元素，隶属于 TrafficCompositions 对象。

它可以通过以下两种方法获得 TrafficCompositions 对象：

► 通过在一个集合中的循环

```
DIM tc As TrafficComposition
FOR EACH tc IN vissim.Net.TrafficCompositions
List.AddItem tc.ID
NEXT tc
```

► 通过具体的编号

```
DIM tc As TrafficComposition
SET tc = vissim.Net.TrafficCompositions.GetTrafficCompositionByNumber(2)
```

TrafficComposition 对象能够通过 ITrafficComposition 接口对车辆构成的属性进行调用。

ITrafficComposition 接口的属性

ID ([out, retval] long *pID)

返回车辆构成的编号。

参数

[out, retval] long *pID : 返回编号。

例子

```
id = tc.ID
```

Name ([out, retval] BSTR *pName)

返回车辆构成的名称。

参数

[out, retval] BSTR *pName： 返回名称。

例子

```
name = tc.Name
```

Name ([in] BSTR Name)

设置车辆构成的名称。最多 255 个字符。

参数

[in] BSTR Name ： 新名称

例子

```
tc.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个车辆构成的属性值。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 特征属性名称（见下表）

[out, retval] VARIANT *pValue： 返回特征属性的值。

例子

name = tc.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个车辆构成的属性值。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 特征属性名称（见下表）

[in] BSTR Value： 特征属性的值（类型根据特征属性的定义）

例子

tc.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	编号
√	√	NAME	名称

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

根据一个参数返回一个车辆构成的属性。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 特征属性的名称（见下）

[in] VARIANT Parameter： 与特征属性有关的参数（见下）

[out, retval] VARIANT *pValue：返回特征属性值

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

根据一个参数设置一个车辆构成的属性。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 特征属性的名称（见下）

[in] VARIANT Parameter： 与特征属性相关的参数（见下）

[in] VARIANT Value：　　特征属性的值（类型根据特征属性的定义）

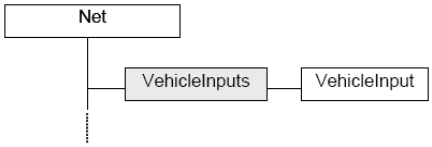
属性览表:

R	W	属性	描述
√	√	RELATIVEFLOW	某一具体车辆类型的相对流量。 参数：车辆类型编号。

如果在仿真过程中调用这些属性，所做的改变将在仿真运行中立刻产生影响。比如，改变相对流量将影响车辆类型的组成，而每一个使用到该车辆构成的车辆输入都将受到影响。

3.5.2.7 VehicleInputs

VehicleInputs 对象是各个 VehicleInput 对象的集合（见第 108 页）。它隶属于 Net 对象，可以通过 INet 接口进入属性 VehicleInputs。



它包含了路网中所有的车辆输入，能够通过循环在一个集合中或者通过具体的编号来获得每一个 VehicleInput 对象（参见 VehicleInput 对象）。

例子

创建一个 VehicleInputs 对象实例，并可获得所有的 VehicleInput 对象：

```
DIM vissim As Vissim
DIM vehins As VehicleInputs
DIM vehin As VehicleInput
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set inputs = Vissim.Net.VehicleInputs
FOR EACH vehin IN vehins   ‘进入 _NewEnum 创建一个枚举
…
NEXT vehin
‘或者也可以：
FOR i = 1 TO vehins.Count
SET vehin = vehins(i)   ‘或者 vehins.Item(i)
…
NEXT i
```

IVehicleInputs 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 VehicleInput 对象的集合（或枚举）。一旦集合被创建，运用 GetVehicleInputByNumber 的方法就能返回单个对象，而整个集合可以通过使用 FOR ...TO ... NEXT 语句来进行循环访问。用户可以在 Visual Basic 的环境下调用 FOR EACH…NEXT 语句，不需说明 _NewEnum，VB 会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum： 返回枚举

Item ([in] VARIANT Index, [out, retval] IVehicleInput **ppVehicleInput)

返回集合某个位置的单个 VehicleInput 对象。只有当所有的车辆输入都可以获得的时候，该属性才有用（见上面的例子）。为了选择一个车辆构成，需要使用 GetVehicleInputByNumber（见下）。因为 Item 是集合的默认属性，因此，下列的命令是一致的：

SET vehin = inputs(1)

SET vehin = inputs.Item(1)

参数

[in] long Index： 索引介于 1 和 Count 之间

[out, retval] IVehicleInput **ppVehicleInputs： 返回 VehicleInput 对象。

Count ([out, retval] long *pCount)

在集合中返回 VehicleInput 对象的数量。详见上例。

参数

[out, retval] long *pCount： 返回对象的数量

IVehicleInputs 接口的使用方法

GetVehicleInputByNumber ([in] long Number, [out, retval] IVehicleInput **ppVehicleInput)

用编号返回 VehicleInput 对象。

参数

[in] long Number： 编号

[out, retval] IVehicleInput **ppVehicleInput： 返回 VehicleInput 对象。

例子

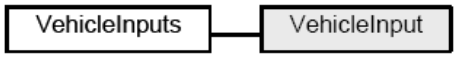
SET vehin = vehins.GetVehicleInputByNumber (2)

IF NOT (vehin IS NOTHING) THEN

name = vehin.AttValue(“NAME”)

```
END IF
```

3.5.2.8 VehicleInput



VehicleInput 对象代表了车辆输入的元素，隶属于 VehicleInputs 对象。

它可以通过以下两种方法进入 VehicleInputs 对象。

► 通过在一个集合里的循环

```
DIM vehin As VehicleInput
FOR EACH vehin IN vissim.Net.VehicleInputs
List.AddItem vehin.ID
NEXT vehin
```

► 通过具体的编号

```
DIM vehin As VehicleInput
SET vehin = vissim.Net.VehicleInputs.GetVehicleInputByNumber (12)
```

VehicleInput 对象能够通过 IVehicleInput 接口进入 VehicleInput 的属性。

IVehicleInput 接口的属性

ID ([out, retval] long *pID)

返回车辆输入的编号。

参数

[out, retval] long *pID：返回编号。

例子

```
id = vehin.ID
```

Name ([out, retval] BSTR *pName)

返回车辆输入的名称。

参数

[out, retval] BSTR *pName： 返回名称。

例子

```
name = vehin.Name
```

Name ([in] BSTR Name)

设置车辆输入的名称。最多 255 个字符。

参数

[in] BSTR Name： 新名称

例子

vehin.Name = „XXX“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个车辆输入的属性。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 特征属性的名称（如下）

[out, retval] VARIANT *pValue： 返回特征属性的值。

例子

name = vehin.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个车辆输入的属性。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 特征属性的名称（如下）

[in] BSTR Value： 特征属性的值（类型根据特征属性的定义）

例子

vehin.AttValue(„NAME“)=„XXX“

属性览表:

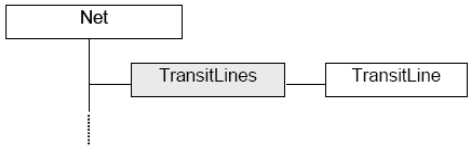
R	W	属性	描述
√		ID	编号
√	√	NAME	名称
√		LINK	路段的编号
√	√	TIMEFROM	时间间隔开始 [s]
√	√	TIMEUNTIL	时间间隔结束 [s]
√	√	TRAFFICCOMPOSITION	车辆构成的编号
√	√	VOLUME	流量 [Veh/h]

如果在仿真过程中调用这些属性，所做的改变将立刻影响仿真运行。比如，当车辆输入设置为“精确流量”的情况下，改变时间间隔（使用特征属性TIMEFROM和TIMEUNTIL）将强制根据当前的特征属性，重新生成车辆的序列。

在设置时间间隔时，TIMEUNTIL必须大于或者等于TIMEFROM。如果当前的时间设置为[100, 1000]，如果用户希望设置为 [1100, 2000]，就必须强制首先设置TIMEUNTIL 为2000，然后TIMEFROM 设置为1100，否则用户将获得一个错误信息。

3.5.3 Transits

3.5.3.1 TransitLines



Transit Lines 对象是 TransitLine（见第 111 页）的集合。

它隶属于 Net 对象，并且能够通过 INet 接口对 TransitLines 的属性进行访问。它包含了路网中所有的公交路线，同时允许集合内部对象的循环操作，和对这些对象的单独访问。

例子

TransitLines 对象的实例并对其所有的 TransitLine 对象进行访问：

```
DIM vissim As Vissim
DIM transitlines As TransitLines
DIM transitline As TransitLine
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET transitlines = Vissim.Net.TransitLines
FOR EACH transitline IN transitlines ‘进入_NewEnum，创建一个枚举
…
NEXT transitline
‘or also:
FOR i = 1 TO transitlines.Count
SET transitline = transitlines(i) ‘or transitlines.Item(i)
…
NEXT i
```

ITransitLines 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性用于创建所有 TransitLine 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetTransitLineByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 语句来进行循环访问。在 Visual Basic 中，你可以使用 FOR EACH…NEXT 语句，而不需定义 _NewEnum 属性，因为 VB 可以内部读取该属性（见上文中的 TransitLines 一例）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举。

Item ([in] VARIANT Index, [out, retval] ITransitLine **ppTransitLine)

返回集合中选中位置上的 TransitLine 对象。这只有在所有公交线路可选的情况下有效（见上文中的 TransitLines 一例）。通过 GetTransitLineByNumber 可以根据标识编号对单独的公交线路进行访问。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

```
SET transitline = transitlines(1)
SET transitline = transitlines.Item(1)
```

参数

[in] long Index: 介于 1 至 Count 之间的编号

[out, retval] ITransitLine **ppTransitLine: 返回 TransitLine

Count ([out, retval] long *pCount)

返回集合中的 TransitLine 对象数量。见上文的 TransitLines 的例子。

参数

[out, retval] long *pCount : 返回对象数量

ITransitLines 接口的使用方法

GetTransitLineByNumber ([in] long Number, [out, retval] ITransitLine **ppTransitLine)

返回标识编号为 Number 的 TransitLine 对象。

参数

[in] long Number : 标识编号

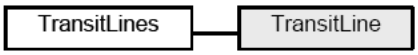
[out, retval] ITransitLine **ppTransitLine : 返回 TransitLine。

例子

```
SET transitline = transitlines.GetTransitLineByNumber (2)
IF NOT (transitline IS NOTHING) THEN
name = transitline.AttValue("NAME")
END IF
```

3.5.3.2 TransitLine

TransitLine对象代表了一个公交线路的元素，隶属于TransitLines对象。



可以通过以下两种方式对TransitLines 对象进行访问：

► 通过集合进行循环访问

```
DIM transitline As TransitLine
FOR EACH transitline IN vissim.Net.TransitLines
```

```
List.AddItem transitline.ID
NEXT transitline
```

► 通过标识编号对具体的 TransitLine 对象进行访问

```
DIM transitline As TransitLine
SET transitline = vissim.Net.TransitLines
GetTransitLineByNumber (12)
```

TransitLine对象能够通过ITransitLine接口对公交路线的属性进行访问。

ITransitLine 接口的属性

ID ([out, retval] long *pID)

返回公交路线的标识编号。

参数

[out, retval] long *pID : 返回标识编号。

例子

```
id = transitline.ID
```

Name ([out, retval] BSTR *pName)

返回公交路线的名称

参数

[out, retval] BSTR *pName : 返回名称。

例子

```
name = transitline.Name
```

Name ([in] BSTR Name)

设置公交路线的名称。最多255个字符。

参数

[in] BSTR Name : 新名称。

例子

```
transitline.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个公交路线的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（见下表）

[out, retval] VARIANT *pValue : 返回属性值。

例子

name = transitline.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个公交路线的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据属性类型）

例子

transitline.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带一个参数返回一个公交路线的属性。请在本节结束的表格中了解与所设语言无关的具体的属性标签说明。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[out, retval] VARIANT *pValue : 返回属性值。

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带一个参数设置一个公交路线的属性。请在本节结束的表格中了解与所设语言无关的具体的属性标签说明。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

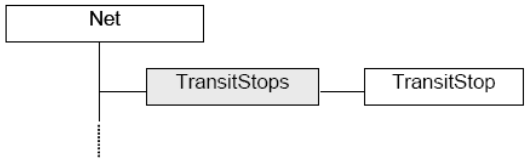
[in] VARIANT Value : 属性值（根据属性类型）

属性览表:

R	W	属性	描述
√	√	ALIGHTINGPROB	乘客在该站下车的百分比
√	√	DWELLTIME	停留时间的分布编号，用来决定停留时间

仿真过程中所作的改变会在设置完成后在仿真进行中立即生效。

3.5.3.3 TransitStops



Transit Stops 对象是 TransitStop 对象（见第 116 页）的集合。

它隶属于 Net 对象，可以通过 INet 接口进行 TransitStops 的属性操作。它包含了路网中所有的公交站点，并同时允许集合内部对象的循环操作，以及对这些对象的单独操作(同见 TransitStop 对象)。

例子

创建 TransitStops 对象实例对其内部所有的 TransitStop 对象进行操作。

```
DIM vissim As Vissim
DIM transitstops As TransitStops
DIM transitstop As TransitStop
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET transitstops = Vissim.Net.TransitStops
FOR EACH transitstop IN transitstops ‘进入_NewEnum，创建一个枚举
…
NEXT transitstop
‘or also:
FOR i = 1 TO transitstops.Count
SET transitstop = transitstops(i) ‘or transitstops.Item(i)
…
NEXT i
```

ITransitStops 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个包含所有 TransitStop 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetTransitStopByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 语句循环来进行访问，而在 Visual Basic 中，你可以使用 FOR EACH…NEXT 语句，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 TransitStops 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举。

Item ([in] VARIANT Index, [out, retval] ITransitStop **ppTransitStop)

返回集合中选中位置上的 TransitStop 对象。这只有在所有公交站点可选的情况下有效（见上文中的 TransitStops 一例）。通过 GetTransitStop ByNumber 可以根据标识编号对公交站点进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

SET transitstop = transitstops(1)

SET transitstop = transitstops.Item(1)

参数

[in] long Index: 介于 1 至 Count 之间的索引

[out, retval] ITransitStop **ppTransitStop: 返回 TransitStop

Count ([out, retval] long *pCount)

在集合中返回 TransitStop 对象数量。见上文中的 TransitStops 一例。

参数

[out, retval] long *pCount : 返回对象数量。

ITransitStops 接口的使用方法

GetTransitStopByNumber ([in] long Number, [out, retval] ITransitStop **ppTransitStop)

返回标识编号为 Number 的 TransitStop 对象。

参数

[in] long Number : 标识编号

[out, retval] ITransitStop **ppTransitStop : 返回 TransitStop

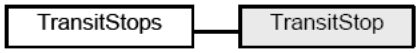
例子

SET transitstop = transitstops.GetTransitStopByNumber (2)

IF NOT (transitstop IS NOTHING) THEN

```
name = transitstop.AttValue("NAME")
END IF
```

3.5.3.4 TransitStop



TransitStop 对象代表了一个公交站点元素，隶属于 TransitStops 对象。

它可以使用以下两种方式通过 TransitStops 对象进行访问：

► 通过集合来进行循环访问

```
DIM transitstop As TransitStop
FOR EACH transitstop IN vissim.Net.TransitStops
List.AddItem transitstop.ID
NEXT transitstop
```

► 通过标识编号对具体的 TransitLine 对象进行访问

```
DIM transitstop As TransitStop
SET transitstop = vissim.Net.TransitStops.
GetTransitStopByNumber (12)
```

TransitStop 对象能够通过 ITransitStop 接口对公交站点的属性进行访问。

ITransitStop 接口的属性

ID ([out, retval] long *pID)

返回公交站点的标识编号。

参数

[out, retval] long *pID : 返回标识编号。

例子

```
id = transitstop.ID
```

Name ([out, retval] BSTR *pName)

返回公交站点的名称。

参数

[out, retval] BSTR *pName : 返回名称。

例子

```
name = transitstop.Name
```

Name ([in] BSTR Name)

设置公交站点的名称。最多 255 个字符。

参数
[in] BSTR Name :新名称

例子
transitstop.Name = „XXX“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个公交站点的属性。请在本节结束的表格中了解与所设语言无关的具体的属性标签说明。

参数
[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回属性值。

例子
name = transitstop.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个公交站点的属性。请在本节结束的表格中了解与所设语言无关的具体的属性标签说明。

参数
[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据属性类型）

例子
transitstop.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称

ITransitStop 接口的使用方法

GetVehicle ([out, retval] IVehicle **ppVehicle)

它将返回第一辆到达公交站点服务的车辆。

参数
[out, retval] IVehicle **ppVehicle : 返回 Vehicle 对象。

例子
DIM veh AS Vehicle

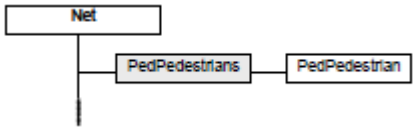
SET veh = transitstop.GetVehicle()

passengers = veh.AttValue(“PASSENGERS”)

3.5.4 Pedestrians

3.5.4.1 PedPedestrians

PedPedestrians 对象是 PedPedestrian 对象的一个集合（参见第 120 页）。它属于 Net 对象并可以通过 INet 接口的 PedPedestrians 属性来访问。



它包含了仿真进行时路网中当前所有正在行走的行人，可以通过循环的方式或者单独访问的方式获得某个 PedPedestrian 对象。

例子

实例化一个 PedPedestrians 对象并访问其中每一个 PedPedstrian 对象：

```
DIM vissim As Vissim
DIM pedestrians As PedPedestrians
DIM pedestrian As PedPedestrian
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set pedestrians = Vissim.Net.PedPedestrians
FOR EACH pedestrian IN pedestrians ‘进入_NewEnum，创建一个枚举
…
NEXT pedestrian
‘或者是
FOR i = 1 TO pedestrians.Count
SET pedestrian = pedestrians(i) ‘或者 pedestrians.Item(i)
…
NEXT i
```

IPedPedestrians 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个所有 PedPedestrian 对象的集合（或枚举）。一旦集合被创建，运用 PedPedestrianByNumber 的方法就能返回其中的具体对象，而整个集合可以通过使用 FOR ...TO ... NEXT 语句循环访问。用户可以在 Visual Basic 的环境下运用 FOR EACH…NEXT 语句，不需定义 _NewEnum，因为 VB 会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IPedPedestrian **ppPedPedestrian)

返回集合某个位置的单 PedPedestrian 对象。只有当所有的行人都可以获得的时候，该属性才有用（见上面的例子）。为了根据定义编号选择一个行人，需要使用 PedPedestrianByNumber（见下）来进行索引。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

SET pedestrian = pedestrians(1)

SET pedestrian = pedestrians.Item(1)

参数

[in] long Index :	编号介于 1 和 edestrians.Count 之间
[out, retval] IPedPedestrian **ppPedPedestrians:	返回 PedPedestrian 对象

Count ([out, retval] long *pCount)

返回集合中 PedPedestrian 对象的数量。参见上面 PedPedestrians 的例子。

参数

[out, retval] long *pCount :	返回对象的数量.

IDs ([in, defaultvalue(0)] BSTR Attribute, [in, defaultvalue(0)] VARIANT Value, [out, retval] VARIANT *pIDs)

返回所有满足条件“Attribute = Value”的行人的 ID，返回形式为一个数组（属性列表参见第 88 页）。若不指定特定的 Value，则返回该集合中一个包含所有行人的编号的数组。

参数

[in] BSTR Attribute :	特征属性名称 （见 IpedPedestrian 属性列表)
[in] VARIANT Value :	属性的值 或者数值的数组（类型根据特征属性）
[out, retval] VARIANT *pIDs :	返回 ID 值的数组.

IPedPedestrians 接口的方法

GetPedPedestrianByNumber ([in] long Number, [out, retval] IPedPedestrian **ppPedPedestrian)

根据特定编号返回 PedPedstrian 对象。

参数

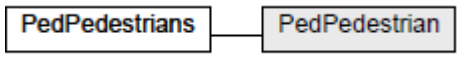
[in] long Number :	标识编号
[out, retval] IPedPedestrian **ppPedPedestrian :	返回 PedPedestrian 对象

例子

DIM pedestrian AS PedPedestrian

SET pedestrian = pedestrians.PedPedestrianByNumber (11)

3.5.4.2 PedPedestrian



PedPedestrian对象代表单个行人，隶属于 PedPedstrians 集合。它可以使用以下两种方法通过PedPedestrians对象来获得：

► 通过集合的迭代

```
DIM pedestrian As PedPedstrian
FOR EACH pedestrian IN vissim.Net.PedPedestrians
List.AddItem pedestrian.ID
NEXT pedestrian
```

► 通过标识编号获得具体的车辆

```
DIM pedestrian As PedPedstrian
SET pedestrian = vissim.Net.PedPedstrian.GetVehicleByNumber (101)
```

PedPedstrian 对象可以通过 IPedPedestrian 接口访问行人的各个属性。但是在执行仿真步骤后，实例化 PedPedestrina 对象并不一定是 VISSIM 中有效的行人单位，（请参阅本节末行人在动态方面的说明）。此时就需要使用属性标识编号（如下），检查其有效性。

IPedPedestrian 接口的属性

ID ([out, retval] long *pID)

返回行人标识编号。若 PedPedestrian 对象不是指向一个有效的 VISSIM 行人，则返回值为 0。

参数

[out, retval] long *pID : 返回编号 (如果指向无效则返回 0)

参数

```
id = pedestrian.ID
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个行人的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名称（见下）

[out, retval] VARIANT *pValue : 返回对应属性的值

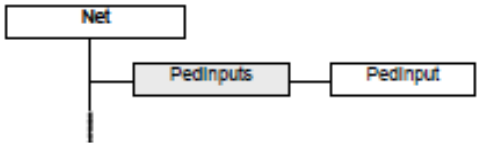
例子

```
type = pedestrian.AttValue(„TYPE“)
```

属性览表:

R	W	属性	描述
√		ID	标识编号
√		TYPE	类型名称

3.5.4.3 PedInputs



PedInputs 对象是 PedInput 对象（参见第 122 页）的集合。它隶属于 Net 对象，可以通过 INet 接口对 PedInputs 的各项属性进行访问。

它包含了路网中所有的行人输入，同时允许通过枚举来进行访问和单独对其访问（同见 PedInput 对象）。

例子

创建 PedInputs 对象实例并访问所有的 PedInput 对象。

```
DIM vissim As Vissim
DIM pedins As PedInputs
DIM pedin As PedInput
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set inputs = Vissim.Net.PedInputs
FOR EACH pedin IN pedins ‘进入_NewEnum，创建一个枚举
…
NEXT pedin
‘or also:
FOR i = 1 TO pedins.Count
SET pedin = pedins(i) ‘or pedins.Item(i)
…
NEXT i
```

IPedInputs 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

通过该属性可以创建一个所有 PedInput 对象的集合（或枚举）。一旦集合被创建，就可以通过 GetPedInputByNumber 方法，使用 FOR ...TO ... NEXT 语句单独访问集合中的对象。在 Visual Basic 中，你

不必定义属性_NewEnum 就可以直接使用 FOR EACH…NEXT 语句，因为 VB 可以内部调用该属性。 (见上文中的例子)。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举对象

Count ([out, retval] long *pCount)

返回集合中的 PedInput 对象的数量。见上例。

参数

[out, retval] long *pCount : 返回对象的数量

IPedInputs 接口的方法

PedInputByNumber ([in] long Number, [out, retval] IPedInput **ppPedInput)

返回标识编码的 PedInput 对象。

参数

[in] long Number : 编码

[out, retval] IPedInput **ppPedInput : 返回 PedInput 对象

例子

```
SET pedin = pedins.GetPedInputByNumber (2)
IF NOT (pedin IS NOTHING) THEN
name = pedin.AttValue("NAME")
END IF
```

3.5.4.4 PedInput

PedInput 对象表示一个行人输入元素，隶属于 PedInputs 对象。可以使用以下两种方法通过 PedInputs 对象对其进行访问。

► 通过集合的内部循环进行访问

```
DIM pedin As PedInput
FOR EACH pedin IN vissim.Net.PedInputs
List.AddItem pedin.ID
NEXT pedin
```

► 通过编号进行单独访问

```
DIM pedin As PedInput
SET pedin = vissim.Net.PedInputs.PedInputByNumber (12)
```

PedInput 对象能够通过 IPedInput 接口对行人输入的各个属性进行访问。

IPedInput 接口的属性

ID ([out, retval] long *pID)

返回行人输入的编号。

参数

[out, retval] long *pID : 返回编号

例子

```
id = pedin.ID
```

Name ([out, retval] BSTR *pName)

返回行人输入的名称。

参数

[out, retval] BSTR *pName : 返回名称

例子

```
name = pedin.Name
```

Name ([in] BSTR Name)

设置行人输入的名称，最长不超过 255 个字符。

参数

[in] BSTR Name : 新名称.

例子

```
pedin.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名称（见下面的表格）

[out, retval] VARIANT *pValue : 对应属性的返回值

例子

```
name = pedin.AttValue(„NAME“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名称（见下面的表格）

[in] BSTR Value : 属性值（返回类型和属性相关）

例子
pedin.AttValue(„NAME“)=„XXX“

属性览表:

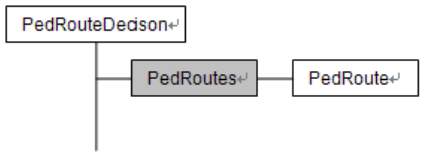
R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称
√	√	TIMEFROM	时间间隔的开始时刻[s]
√	√	TIMEUNTIL	时间间隔的结束时刻[s]
√	√	PEDCOMPOSITION	行人构成的编号
√	√	VOLUME	流量[Ped/h]

在仿真过程中，所作的更改会即时在路网中产生作用。如果勾选了选项“精确流量”的选项，则当时间间隔改变（更改TIMEFROM或TIMEUNTIL）会在当前的属性设置下强制重新生成行人序列。

在设置时间间隔时，TIMEUNTIL不能小于TIMEFROM 的值。假设当前的时间间隔是[100,1000]，而用户需要将其设置为[1100,2000]，此时就需要先将TIMEUNTIL设置成2000，再将TIMEFROM设置为1100，不然会报出错误信息。

3.5.4.5 PedRoutes

PedRoutes 对象是各个 PedRoute 对象的集合（见第 126 页）。它隶属于 PedRouteDecision 对象，可以通过 IPedRouteDecision 接口进入属性 PedRoutes。



它包含了行人路径决策中中所有的行人路径，能够通过循环在一个集合中或者通过具体的编号来获得每一个单独的对象（参见 PedRoute 对象）。

例子

创建 Routes 对象实例并访问所有的 Route 对象。

```
DIM vissim As Vissim
DIM pedroutes As PedRoutes
DIM pedroute As PedRoute
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET pedroutes = Vissim.Net.PedRouteDecisions(1).PedRoutes
```

```
FOR EACH pedroute INpedroutes ‘进入_NewEnum，创建一个枚举
…
NEXT pedroute
‘or also:
FOR i = 1 TO pedroutes.Count
SET pedroute = pedroutes (i) ‘or pedroutes.Item(i)
…
NEXT i
```

IPedRoutes 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

通过该属性可以创建一个所有行人路径对象的集合（或枚举）。一旦集合被创建，就可以通过 GetPedRouteByNumber 方法，使用 FOR ...TO ... NEXT 语句单独访问集合中的对象。在 Visual Basic 中，你不必定义属性_NewEnum 就可以直接使用 FOR EACH…NEXT 语句，因为 VB 可以内部调用该属性。 (见上文中的例子)。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举对象

Item ([in] VARIANT Index, [out, retval] IPedRoute **ppPedRoute)

返回集合中选中位置的某个行人路径。这只有在所有路径可访问的情况下有效（见上文中的 Routes 一例）。通过 GetRouteByNumber 可以根据表示编号对行人路径进行选择（如下）。由于 Item 是集合的默认属性，因此下述两个命令是一样的：

参数

[in] long Index :	介于 1 和 Count 之间的索引
[out, retval] IPedRoute **ppPedRoutes:	返回 PedRoute 对象

Count ([out, retval] long *pCount)

返回集合中的 PedRoute 对象的数量。见上面 PedRoutes 的例子。

参数

[out, retval] long *pCount :	返回对象的数量.

IPedRoutes 接口的方法

PedRouteByNumber ([in] long Number, [out, retval] IPedRoute **ppPedRoute)

返回标识编号对应的 PedRoute 对象。

参数

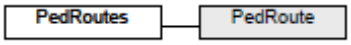
[in] long Number : 编码

[out, retval] IPedRoute **ppPedRoute : 返回 PedRoute 对象

例子

```
SET pedroute = pedroutes.PedRouteByNumber(2)
IF NOT (pedroute IS NOTHING) THEN
rf = pedroute.AttValue("RELATIVEFLOW")
END IF
```

3.5.4.6 PedRoute



一个 PedRoute 对象代表了行人路径决策时的一个行人路径元素，隶属于 PedRoutes 对象。可以使用以下两种方法通过 PedRoutes 对象对其进行访问。

► 通过集合的内部循环进行访问

```
DIM pedroutes As PedRoutes
DIM pedroute As PedRoute
SET pedroutes = Vissim.Net.PedRoutingDecisions(1).PedRoutes
FOR EACH pedroute IN pedroutes
List.AddItem pedroute.ID
NEXT pedroute
```

► 通过编号进行单独访问

```
DIM pedroute As PedRoute
SET pedroute = vissim.Net.PedRoutingDecisions(1).PedRoutes.PedRouteByNumber (12)
```

PedRoute 对象能够通过 IPedRoute 接口对路径的各个属性进行访问。

IPedRoute 接口的属性

ID ([out, retval] long *pID)

返回行人路径的编号。

参数

[out, retval] long *pID : 返回编号

例子

```
id = pedroute.ID
```

Name ([out, retval] BSTR *pName)

返回行人路径的名称。

参数
[out, retval] BSTR *pName : 返回名称.

例子
name = pedroute.Name

Name ([in] BSTR Name)

设置行人路径的名称。

参数
[in] BSTR Name : 新名称.

例子
pedroute.Name = „XXX“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 特征属性名称（见下表）

[out, retval] VARIANT *pValue : 返回该属性的值

例子
name = pedroute.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 特征属性名称（见下）

[in] BSTR Value : 属性的值.（类型根据特征属性）

例子
pedroute.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	标识编号
	√	NAME	名称

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带一个参数返回一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute :	特征属性名称（见下）
[in] VARIANT Parameter :	与特征属性相关的参数（见下）
[out, retval] VARIANT *pValue :	返回该属性的值

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带一个参数设置一个路径属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute :	特征属性名称（见下）
[in] VARIANT Parameter :	与特征属性相关的参数（见下）
[in] VARIANT Value :	属性的值.（类型根据属性定义）

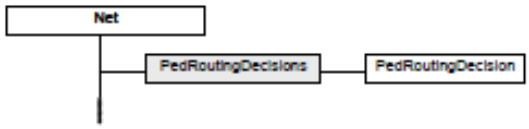
属性览表:

R	W	属性	描述
√	√	RELATIVEFLOW	时间间隔内的相对流量 参数：时间间隔编号 (1..n)

所作的改变会在实行仿真时立即生效。

3.5.4.7 PedRoutingDecisions

PedRouting-Decisions 对象是 PedRoutingDecision 对象（见第 130 页）的集合。



它隶属于 Net 对象，可以通过 INet 接口对 PedRoutingDecisions 对象进行访问。它包含了路网中所有的行人路径决策，并且可以通过循环对所有的对象进行访问，同时也可以单独地对具体对象进行访问。（同见 PedRoutingDecision 对象）。

Example

创建一个 PedRoutingDecisions 对象的实例，进入所有的 PedRoutingDecision 对象：

```
DIM vissim As Vissim
DIM decisions As PedRoutingDecisions
DIM decision As PedRoutingDecision
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET decisions = Vissim.Net.PedRoutingDecisions
FOR EACH decision IN decisions ‘进入_NewEnum，创建一个枚举
…
NEXT decision
‘or also:
FOR i = 1 TO decisions.Count
SET decision = decisions(i) ‘or decisions.Item(i)
…
NEXT i
```

IPedRoutingDecision 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性可以创建所有路径决策对象的集合（枚举）。一旦集合被创建，可以使用 GetPedRoutingDecisionByNumber 方法，通过 FOR ...TO ... NEXT 语句返回集合中的具体对象。在 Visual Basic 中，用户可以直接使用 FOR ...TO ... NEXT 语句，而不用定义_NewEnum 属性，因为 VB 可以内部调用该属性（见上文中的 PedRoutingDecisions 一例）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IPedRoutingDecision **ppDecision)

该方法用于返回集合中所选位置的路径决策。这个方法只在所有的路径决策可使用时才有效（见上文中的 PedRoutingDecisions 一例）。通过 GetPedRoutingDecisionByNumber 方法可以根据表示编号对路径决策进行选择（如下）。由于 Item 是集合的默认属性，因此以下的语句是同一个意思：

```
SET decision = decisions(1)
SET decision = decisions.Item(1)
```

参数

[in] long Index : 介于 1 和 Count 之间的索引

[out, retval] IPedRoutingDecision **ppDecision:返回 PedRoutingDecision 对象

Count ([out, retval] long *pCount)

返回集合中 RoutingDecision 对象数量。见上面 PedRoutingDecisions 的例子。

参数

[out, retval] long *pCount : 返回对象的数量.

IPedRoutingDecision 接口的方法

PedRoutingDecisionByNumber ([in] long Number, [out, retval] IRoutingDecision **ppDecision)

通过编号返回 PedRoutingDecision 对象。

参数

[in] long Number : 编码

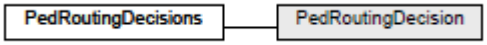
[out, retval] IRoutingDecision **ppDecision : 返回 RoutingDecision 对象

例子

```
SET decision = decisions.GetRoutingDecisionByNumber (2)
IF NOT (decision IS NOTHING) THEN
name = decision.AttValue("NAME")
END IF
```

3.5.4.8 PedRoutingDecision

PedRoutingDecision 对象代表了一个行人路径决策元素，隶属于 PedRoutingDecisions 对象。



它可以使用以下两种方法通过 PedRoutingDecisions 对象来访问：

► 通过集合内部的枚举访问

```
DIM decision As PedRoutingDecision
FOR EACH decision IN vissim.Net.PedRoutingDecisions
List.AddItem decision.ID
NEXT decision
```

► 通过标识编号单独访问具体的 RoutingDecision

```
DIM decision As PedRoutingDecision
SET decision = vissim.Net.PedRoutingDecisions.PedRoutingDecisionByNumber (12)
```

PedRoutingDecision 对象能够通过 IPedRoutingDecision 接口对行人路径选择的各属性进行访问。

IPedRoutingDecision 接口的属性

ID ([out, retval] long *pID)

返回行人路径决策的标识编号。

参数
[out, retval] long *pID : 返回编号.

例子
id = decision.ID

Name ([out, retval] BSTR *pName)

返回行人路径决策的名称。

参数
[out, retval] BSTR *pName : 返回名称.

例子
name = decision.Name

Name ([in] BSTR Name)

设置行人路径决策的名称。最大不超过 255 个字符。

参数
[in] BSTR Name : 新名称.

例子
decision.Name = „XXX“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个行人路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 特征属性名称（见下表）

[out, retval] VARIANT *pValue : 返回该属性的值

例子
name = decision.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个行人路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 特征属性名称（见下）

[in] BSTR Value : 属性的值.（类型根据特征属性）

例子
decision.AttValue(„NAME“)=„XXX“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称
√		PEDCLASSES	受影响的行人类别（编号的数组）
√		PEDTYPES	受影响的行人类型（编号的数组）

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带一个参数返回一个行人路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 特征属性名称（见下）

[in] VARIANT Parameter : 与特征属性相关的参数（见下）

[out, retval] VARIANT *pValue : 返回该属性的值

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带一个参数设置一个行人路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 特征属性名称（见下）

[in] VARIANT Parameter : 与特征属性相关的参数（见下）

[in] VARIANT Value : 属性的值.（类型根据属性定义）

属性览表:

R	W	属性	描述
√	√	RELATIVEFLOW	当前时间间隔的相对流量（若无仿真正运行，则为初始的值） 参数：路径决策编号
√	√	TIMEFROM	时间间隔的开始时刻[s]。 参数：时间间隔顺序（1...n）
√	√	TIMEUNTIL	时间间隔的结束时刻[s]。 参数：时间间隔顺序（1...n）
√	√	PEDCLASS	若该行人类别受影响，则为 TRUE。 参数：行人类别的编号（所有若要输出所有类别则设为 0）
		PEDTYPE	若该行人类型受影响，则为 TRUE。 参数：行人类型的编号

仿真过程中所作的更改会即时在路网中产生作用。

在设置时间间隔时，TIMEUNTIL不能小于TIMEFROM 的值。假设当前的时间间隔是[100,1000]，而用户需要将其更改为[1100,2000]，此时就需要先将TIMEUNTIL设置成2000，再将TIMEFROM设置为1100，不然会报出错误信息。

AttValue2 ([in] BSTR Attribute, [in] VARIANT Parameter1, [in] VARIANT Parameter2, [out, retval] VARIANT *pValue)

带两个参数返回行人路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下表）

[in] VARIANT Parameter1 : 第一个与属性相关的参数

[in] VARIANT Parameter2 : second 与属性相关的参数

[out, retval] VARIANT *pValue : 返回该属性的值

AttValue2 ([in] BSTR Attribute, [in] VARIANT Parameter1, [in] VARIANT Parameter2, [in] VARIANT Value)

带两个参数设置行人路径决策的属性。请在本节结束的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 特征属性名称（见下表）

[in] VARIANT Parameter1 : 第一个与属性相关的参数

[in] VARIANT Parameter2 : 第二个与属性相关的参数

[in] VARIANT Value : 属性的值.（类型根据属性定义）

属性览表:

R	W	属性	描述
√	√	RELATIVEFLOW	对应时间间隔和路径决策编号的相对流量。 参数 1：路径决策编号 参数 2：所需时间间隔中的时间点

仿真过程中所作的更改会即时在路网中产生作用。

Routes ([out, retval] IRoutes **ppRoutes)

创建一个 PedRoutes 对象实例，这样就可以单独地对路网中所定义的与该行人路径决策相关的行人路径进行访问。

参数

[out, retval] IRoutes **ppRoutes : 返回 Routes 对象

例子

```
DIM pedroutes AS Routes
DIM decision As PedRoutingDecision
SET decision = vissim.Net.PedRoutingDecisions.PedRoutingDecisionByNumber (10)
SET pedroutes = decision.Routes
```

IPedRoutingDecision 接口的方法

AddTimeInterval ([in] double From, [in] double To, [out, retval] long *pIndex)

添加一个新的时间间隔。如果成功了，会返回所分配的列表索引号（1..n），否则为 0。

参数

[in] double From : 时间间隔的开始

[in] double To : 时间间隔的结束

[out, retval] long *pIndex : 返回列表索引号

例子

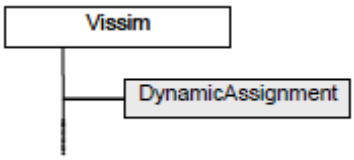
```
id = decisions.AddStaticPedRoutingDecision(1, 100.0)
SET decision = decisions.PedRoutingDecisionByNumber (id)
```

```
IF NOT (decision IS NOTHING) THEN
decision.AttValue1("TIMEUNTIL", 1) = 1799 'sets default time interval to [0-1799]
index = decision.AddTimeInterval(1800, 3600)
END IF
```

3.5.5 Dynamic Assignment

3.5.5.1 DynamicAssignment

DynamicAssignment 对象隶属于 Vissim，可通过 IVissim 接口的 DynamicAssignment 属性来进行访问。它能配置和控制动态分配的选项。一个 DynamicAssignment 例子总是指的是当前打开路网中的动态交通分配选项。



例子

```
DIM vissim AS Vissim
DIM dynassig AS DynamicAssignment
SET vissim = NEW Vissim
SET dynassig = vissim.DynamicAssignment
vissim.LoadNet "example.inp"
dynassig.AttValue("STOREPATHS")
```

IDynamicAssignment 的接口属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个动态分配的属性。请在本节结束的表格中了解与所设语言无关的具体的属性标签说明。

参数

[in] BSTR Attribute：　属性名称（见下）

[out, retval] VARIANT*pValue：返回该属性的值

例子

```
period = dynassig.AttValue(„STOREPATHS ")
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设定一个动态分配的属性。请在本节结束的表格中了解与所设语言无关的具体的属性标签说明。

参数

[in] BSTR Attribute：　　属性名称（见下）

[in] VARIANT Value：　　属性的值（类型根据属性的定义）

例子

dynassig.AttValue(„STOREPATHS“) = TRUE

属性览表:

R	W	属性	描述
√	√	APS	随机通路惩罚，针对其他路径搜寻的选择标志
√	√	APSSPREAD	最大的扩散比例
√	√	APSPASSES	随机通过量
√		CONVERGEALL	如果所有激活的收敛标准都符合，返回 True
√	√	CONVERGENCETTEDGESCHECK	检查通路的出行时间的收敛的选择标志
√	√	CONVERGENCETTEDGESFACTOR	通路的出行时间收敛的系数[%]
√	√	CONVERGENCETTEDGESMAX	当前在新旧通路上的出行时间的最大百分比差值[%]
√	√	CONVERGENCETTPATHSCHECK	检查路径的出行时间的收敛选择标志
√	√	CONVERGENCETTPATHSFACTOR	路径出行时间收敛的系数[%]
√	√	CONVERGENCETTPATHSMAX	当前在新旧路径上的出行时间的最大百分比差值[%]
√	√	CONVERGENCEVOLEDGESCHECK	检查通路流量的熟练的选择标志
√	√	CONVERGENCEVOLEDGESFACTOR	通路流量收敛的系数[Veh]
√	√	CONVERGENCEVOLEDGESMAX	当前在新旧通路流量的最大百分比差值[%]

R	W	属性	描述
√	√	EVENTSMOOTHING	激活/关闭通路平滑操作
√	√	EVENTROUTECHOICEDIST	激活/关闭路径选择分布操作
√	√	KIRCHHOFF	用于路径选择的Kirchhoff指数
√	√	PSALLPAIRS	搜索交通量为0的OD对的选择标志
√	√	PSLIMIT	限制每个停车场的出行路径的数目的选择标志
√	√	PSLIMITNUM	每个停车场的出行路径的允许的数目
√	√	PSREJECTPATHS	丢弃全部费用高于最佳路径全部费用的路径
√	√	PSREJECTPATHSFACTOR	丢弃路径的界限值
√	√	SEARCHPATHS	寻找新的动态交通分配路径的选择标志
√	√	STORECOSTS	保存费用评价BEW文件的选择标志
√	√	STOREPATHS	存贮路径文件*.WEG的选择标志
√	√	VOLUME	缩放总流量的百分比 [%]

作为一个针对动态交通的收敛选项的补充信息，路径出行时间、通路出行时间、以及通路流量的 3 个收敛标准，用户可以根据自己需要进行选择，使用到的属性分别为 CONVERGETTEDGESMAX，CONVERGETTPATHSMAX 和 CONVERGEVOLEDGESMAX。这些结果在仿真期间和仿真之后可以获得。但是，只有从第二次迭代开始的仿真结束时，该结果才有用。

IDynamicAssignment 接口的方法

IDynamicAssignment 对象的事件

EdgeSmoothing ([in] double oldVal, [in] doule newVal, [out, retval] doule retVal)

在仿真期间，创建一个事件，进行通路出行时间的平滑计算，（如果属性 EVENTSMOOTHING 激活的话）。VISSIM 将使用返回的值作为新的平滑后的通路值。

参数

[in] double oldVal：旧的平滑值

[in] double newVal ：新的观测值

[out, retval] double retVal ：新计算得到的平滑值

例子

```
DIM vis AS Vissim
DIM sim AS Simulation
Dim WithEvents dyn as DynamicAssignment
Dim nIteration AS Long
‘实行 DynamicAssignment 事件 EdgeSmoothing
Private Function dyn_EdgeSmoothing(ByVal oldVal As Long, ByVal newVal As Long) As
Double
‘MSA 的方法
dyn_EdgeSmoothing = oldVal * (1# - 1# / nIteration)+ newVal * 1# / nIteration
End Function
SET vis = NEW Vissim
SET sim = vis.Simulation
SET dyn = vis.DynamicAssignment
vis.LoadNet(“example.inp”)
dyn.AttValue(“EVENTSMOOTHING”) = TRUE
FOR nIteration = 1 TO 10
sim.RunContinuous
NEXT nIteration
```

RouteChoiceDistribution ([in] VARIANT Costs, [out] VARIANT *Distribution)

创建一个事件，计算停车场之间的路径选择分布（如果属性 EVENTROUTECHOICEDIST 激活的话）。VISSIM 将使用 Distribution 中的相对百分比来把车辆分配到不同的路径上去。

参数

[in] VARIANT Costs：当前停车场之间的路径费用

[out] VARIANT *Distribution：返回相对百分比

例子

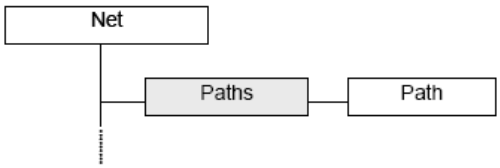
```
DIM vis AS Vissim
DIM sim AS Simulation
Dim WithEvents dyn as DynamicAssignment
‘实行 DynamicAssignment 事件 RouteChoiceDistribution
Private Sub dyn_RouteChoiceDistribution (ByVal Costs As VARIANT,
ByRef Distribution As VARIANT)
‘ Logit function:
```

```
DIM dist() AS Double
s = 0#
FOR i = LBOUND(Costs) To UBOUND(Costs)
s = s + EXP(0.2 / Costs(i))
NEXT i
FOR i = LBOUND(Costs) To UBOUND(Costs)
dist(i) = EXP(0.2 / Costs(i)) / s
NEXT i
Distribution = dist
END SUB

SET vis = NEW Vissim
SET sim = vis.Simulation
SET dyn = vis.DynamicAssignment

vis.LoadNet("example.inp")
dyn.AttValue("EVENTROUTECHOICEDIST") = TRUE
sim.RunContinuous
```

3.5.5.2 Paths



Paths 对象是对各个 Path 对象的集合。它隶属于 Net 对象，并通过 INet 接口的属性 Paths 进行访问。

它包含运用该接口获得的所有的路径（没有动态交通分配的路径，请参阅本章节的结尾处说明）能够通过循环在一个集合中获得每一个 Path 对象，或者通过具体的编号来获得一个 Path 对象。

例子

创建 Paths 对象的实例并获得所有 Path 对象（见第 142 页）：

```
DIM vissim AS Vissim
DIM paths AS Paths
DIM path AS Path
SET vissim = NEW Vissim
vissim.LoadNet "c:\vissim\daten\example.inp"
SET paths = Vissim.Net.Paths
FOR EACH path IN paths  '运用方法 _NewEnum 来创建一个枚举
…
NEXT path
```

```
‘或者，另一种方法：
FOR i = 1 TO paths.Count
SET path = paths(i)  ‘或者 paths.Item(i)
…
NEXT i
```

IPaths 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个所有 Path 对象的集合（或枚举）。一旦集合被创建，运用 GetPathByNumber 的方法就能返回个别对象，而整个集合可以通过使用 FOR ...TO ... NEXT 语句来进行循环访问。用户可以在 Visual Basic 的环境下运用 FOR EACH…NEXT 的语句，不需说明_NewEnum，VB 会在程序内部调用它（见上面的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IPath **ppPath)

返回集合某个位置的单个 Path 对象。只有当所有的路径都可以获得的时候，该属性才有用（见上面的例子）。为了选择一个 Path，可以使用 GetPathByNumber（见下）来进行访问。因为 Item 是一个集合的默认属性，因此，下列的命令是一致的：

```
SET path = paths(1)
SET path = paths.Item(1)
```

参数

[in] long Index：	索引值介于 1 和 Paths.Count 之间
[out, retval] IPath **ppPath：	返回 Path 对象

Count ([out, retval] long *pCount)

在集合中返回 Path 对象的数量。见上例。

参数

[out, retval] long *pCount： 返回对象数量

IPaths 接口的使用方法

GetPathByNumber ([in] long Number, [out, retval] IPath **ppPath)

返回某个编号的 Path 对象。

参数

[in] long Number：	编号
[out, retval] IPath **ppPath：	返回 Path 对象

例子

```
DIM path AS Path
SET path = paths.GetPathByNumber (11)
```

AddPathAsNodeSequence ([in] long Number, [in] VARIANT Nodes, [out, retval] IPath **ppPath)

基于数组中所给的节点的顺序，创建和插入一条新的路径。如果编号为 0，则程序会自动赋给该路径一个路径编号。一个无效的节点顺序将导致生成一个错误的信息（见下面的错误信息以及第 257 页的错误处理）。在第一个节点和第二个节点之间必须要有一个交通出发的停车场。在最后两个节点之间必须要有一个交通目的地的停车场。

参数

[in] long Number： 编号或 0

[in] VARIANT Nodes： 含有节点标志号码的数组

[out, retval] IPath **ppPath： 返回 Path 对象

例子

```
DIM path AS Path
path = paths.AddPathAsNodeSequence (1, Array(10, 20, 30, 40))
```

RemovePath ([in] long Number)

该方法从 COM 接口获得的路径集合中移除某一个编号的路径。如果该路径用于了动态交通分配，它属于动态交通分配的路径集合 (*.weg 文件)，它不会从该集合中被移除。一个路径能够在一个仿真运行期间被移除，在这种状况下如果该方法调用了以后，立即不再把任何车辆分配给该条路径，等到没有车辆再使用这条路径的时候，该条路径就会被移除。

参数

[in] long Number： 编号

例子

```
paths.RemovePath (11)
```

必要知识

Path network element

虽然路径是静态的网络元素，但是可以在仿真过程中的任意一步长时通过 IPaths 接口动态地插入和使用路径。

路径和动态交通分配

使用 IPaths 接口插入的路径，如果它们不是通过最短路径搜索找到的话，就不属于动态交通分配（*.weg 文件）的路径集合中。如果它们是通过最短路径搜索找到的话，它们就属于普通的动态交通分配的路径中的一员。

使用了 IPaths 的界面插入的路径，不考虑任何的关闭设置（不考虑 Link，edge，route 的关闭）。

使用了 IPaths 的界面插入的路径，从不考虑长距离的绕路路径。

只有在仿真开始之前插入的路径，将影响到“丢弃全部费用较高的路径”的选项（菜单：搜索路径 – 丢弃全部费用较高的路径比最优的路径要差）。

在 IPath 接口中使用的路径编号没有必要与路径评价文件（*.WGA）中的路径编号相对应。

可能的错误

“An object with the same identifier already exists.”

当使用一个已有的路径编号来创建一条路径时，将会产生该信息。需要是用正确的号码或者使用自动分配号码选择（0）。

“The object with specified identifier doesn’t exists.”

当使用了不存在的编号时，将会发生该信息。

"No origin parking lot between the first two nodes."

一个 Path 必须开始于一个停车场。必须保证停车场存在与首个两节点之间。

"No destination parking lot between the last two nodes."

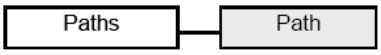
一个 Path 必须结束在一个停车场上。必须保证停车场存在于最末两个节点之间。

“No valid node sequence.”

该信息会在以下情况下发生：

1. 节点序列不能满足任何有效途径。
2. 节点序列对多条路径有效。使用中间节点以解决不明确之处。

3.5.5.3 Path



一个 Path 对象代表了一个路径元素，隶属于 Paths 对象。它可以通过以下两种方法进行调用：

► 通过在集合中的循环获得

```
DIM paths AS Paths
DIM path AS Path
```

```
SET paths = Vissim.Net.Paths
FOR EACH path IN paths
List.AddItem path.ID
NEXT path
```

► 通过具体的编号获得

```
DIM paths AS Paths
DIM path AS Path
SET paths = Vissim.Net.Paths
SET path = paths.GetPathByNumber (11)
```

Path 对象能够通过 IPath 界面进入 Path 的属性。

区别于其他路网元素，通过 IPaths 接口插入的路径与动态交通分配路径并不是完全相同（详见第 141 页的“必要的知识”）。

IPath 接口的属性

ID ([out, retval] long *pID)

返回路径编号。若 Path 对象并没有指向有效的 VISSIM 路径元素，则返回值为 0。

参数

[out, retval] long *pID： 返回编号。

例子

```
id = path.ID
```

Name ([out, retval] BSTR *pName)

返回路径名称。

参数

[out, retval] BSTR *pName： 返回名称。

例子

```
name = path.Name
```

Name ([in] BSTR Name)

设置路径名称。

参数

[in] BSTR Name： 新名称

例子

```
path.Name = „Markplatz_Stumpftsr“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个路径的属性。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 特征属性名（如下）

[out, retval] VARIANT *pValue：返回特征属性的数值。

例子

length = path.AttValue(„LENGTH“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个路径的属性。请在该节结尾的表中了解各个特征属性的标识编号，这部分与内容与语言无关。

参数

[in] BSTR Attribute： 特征属性名（如下）

[in] BSTR Value ： 特征属性的数值。（类型根据特征属性的定义）

例子

path.AttValue(„NAME“) = „Markplatz_Stumpfstrasse“

属性览表:

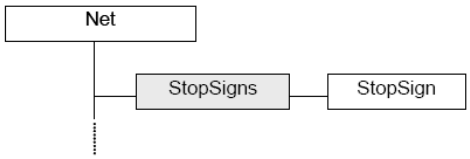
R	W	属性	描述
✓		ID	编号
✓	✓	NAME	名称
✓	✓	LENGTH	长度，单位用当前选项

3.6 Intersection Control

3.6.1 Non-signalized

3.6.1.1 StopSigns

StopSigns 对象是对 StopSign 对象（见第 146 页）的集合。它隶属于 Net 对象，并且可以通过 INet 接口进行 StopSigns 属性操作。



它包含了路网中的所有停车标志，并同时允许集合内部对象的循环操作，以及这些对象的单独操作。（同见 StopSign 对象）。

例子

创建 StopSigns 对象实例对其内部所有的 StopSign 对象进行操作：

```
DIM vissim As Vissim
DIM stops As StopSigns
DIM stop As StopSign
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set stops = Vissim.Net.StopSigns
FOR EACH stop IN stops ‘进入_NewEnum，创建一个枚举
…
NEXT stop
‘or also:
FOR i = 1 TO stops.Count
SET stop = stops(i) ‘or stops.Item(i)
…
NEXT i
```

IStopSigns 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个包含所有 StopSign 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetStopSignByNumber 方法来进行访问，而整个集合可以使用 FOR ...TO ... NEXT 语句来进行循环访问。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义

_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 StopSigns 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回 Enumeration

Item ([in] VARIANT Index, [out, retval] IStopSign **ppStopSign)

返回集合中选中位置上的 StopSign 对象。这只有在所有停车标志可选的情况下有效（见上文中的 StopSigns 一例）。通过 GetStopSignByNumber 可以根据编号对停车标志进行访问。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

SET stop = stops(1)

SET stop = stops.Item(1)

参数

[in] long Index : 1 至 Count 之间的索引

[out, retval] IStopSign **ppStopSigns: 返回 StopSign 对象。

Count ([out, retval] long *pCount)

返回集合中的 StopSign 对象数量。见如上 StopSigns 的例子。

参数

[out, retval] long *pCount : 返回对象数量。

IStopSigns 接口的使用方法

GetStopSignByNumber ([in] long Number, [out, retval] IStopSign **ppStopSign)

返回标识编号 Number 所对应的 StopSign 对象。

参数

[in] long Number : 标识编号

[out, retval] IStopSign **ppStopSign : 返回 StopSign 对象

例子

```
SET stop = stops.GetStopSignByNumber (2)
IF NOT (stop IS NOTHING) THEN
name = stop.AttValue("NAME")
END IF
```

3.6.1.2 StopSign

一个 StopSign 对象代表了一个停车标志元素，隶属于 StopSigns 对象。它可以通过以下两种方式通

过 StopSigns 对象来进行访问：

► 通过集合的循环操作

```
DIM stop As StopSign
FOR EACH stop IN vissim.Net.StopSigns
List.AddItem stop.ID
NEXT stop
```

► 通过标识编号对具体的 StopSign 对象进行操作

```
DIM stop As StopSign
SET stop = vissim.Net.StopSigns.GetStopSignByNumber (12)
```

StopSign 对象能够通过 IStopSign 接口对停车标志属性进行操作。

IStopSign 接口的属性

ID ([out, retval] long *pID)

返回停车标志的标识编号。

参数

[out, retval] long *pID : 返回标识编号。

例子

```
id = stop.ID
```

Name ([out, retval] BSTR *pName)

返回停车标志名称。

参数

[out, retval] BSTR *pName : 返回名称。

例子

```
name = stop.Name
```

Name ([in] BSTR Name)

设置停车标志名称。最多 255 个字符。

参数

[in] BSTR Name : 新名称

例子

```
stop.Name = „XXX“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个停车标志的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数

[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回特征值。

例子
name = stop.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个停车标志的属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Value : 特征值（根据特征类型）

例子
stop.AttValue(„NAME“) = “cashbox1”

属性览表:

R	W	属性	描述
✓		ID	标识编号
✓	✓	NAME	名称

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带一个参数返回停车标志属性。请在本章结尾的表格中了解各个特征属性的标识编号，这部分与内容与语言无关。

参数
[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[out, retval] VARIANT *pValue : 返回特征值。

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带一个参数设置停车标志属性。请在下面的表格中了解与所设语言无关的各个特征属性的标识编号。

参数
[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Parameter : 与属性相关的参数（如下）

[in] VARIANT Value : 特征值（根据特征类型）

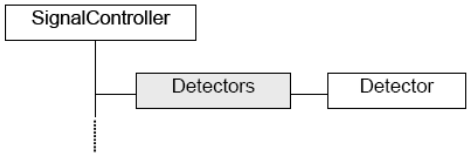
属性览表:

R	W	属性	描述
✓	✓	TIMEDIST	停车时间的时间分布编号（在仿真过程中可进行修改）。参数：车辆类别编号。如果将某一车辆类别的该参数设为 0，就意味着移除该分配。

3.6.2 Signal Control

3.6.2.1 Detectors

Detectors 对象是 Detector 对象（见第 151 页 ）的集合。它隶属于 SignalController 对象，并可以通过 ISignalController 接口进行 Detectors 属性操作。



它包含了所有检测器的转介信号控制，并同时允许集合内部的循环操作，以及内部对象的单独操作。

例子

创建 Detectors 对象实例对其内部所有的 Detector 对象进行操作：

```
DIM vissim AS Vissim
DIM dets AS Detectors
DIM det AS Detector
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET dets = Vissim.Net.SignalControllers(1).Detectors
FOR EACH det IN dets ‘进入_NewEnum，创建一个枚举
…
NEXT det
‘or also:
FOR i = 1 TO dets.Count
SET det = dets(i) ‘or dets.Item(i)
…
NEXT i
```

IDetectors 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 Detector 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetDetectorByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 Detectors 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IDetector **ppDetector)

返回集合中选中位置上的 Detector 对象。这只有在所有检测器可选的情况下有效（见上文中的 Detector s 一例）。通过 GetDetectorByNumber 可以根据标识编号对检测器进行选择（如下）。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

```
SET det = dets(1)
SET det = dets.Item(1)
```

参数

[in] long Index : 介于 1 至 dets.Count 间的索引

[out, retval] IDetector **ppDetector : 返回 Detector 对象。

Count ([out, retval] long *pCount)

返回集合中的 Detector 对象数量。见如上 Detectors 的例子。

参数

[out, retval] long *pCount : 返回对象数。

IDetectors 接口的使用方法

GetDetectorByNumber ([in] long Number, [out, retval] IDetector **ppDetector)

用标识编号数返回 Detector 对象。

参数

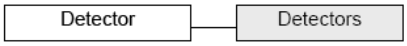
[in] long Number : 标识编号。

[out, retval] IDetector **ppDetector: 返回 Detector 对象。

例子

```
DIM det AS Detector
SET det = dets.GetDetectorByNumber(1)
```

3.6.2.2 Detector



一个 Detector 对象隶属于一个控制器的 Detectors 对象。它可以使用以下两种方法通过 Detector 对象来进行操作：

► 通过集合的循环操作

```
DIM dets As Detectors
DIM det As Detector
SET dets = Vissim.Net.SignalControllers(1).Detectors
FOR EACH det IN dets
List.AddItem det.ID
NEXT det
```

► 通过标识编号对具体的 Detector 对象进行操作

```
DIM dets As Detectors
DIM det As Detector
SET dets = Vissim.Net.SignalControllers(1).Detectors
SET det = dets.GetDetectorByNumber(10)
```

Detector 对象能够通过 IDetector 的接口对检测器的属性进行操作。

IDetector 接口的属性

ID ([out, retval] long *pID)

返回检测器的标识编号。

参数

[out, retval] long pID : 返回标识编号。

例子

```
id = det.ID
```

Name ([out, retval] BSTR *pName)

返回检测器的名称。

参数

[out, retval] BSTR pName : 返回名称。

例子

```
name = det.Name
```

Name ([in] BSTR Name)

设置检测器的名称。最多 255 个字符。

参数

[in] BSTR Name : 新名称.

例子

```
det.Name = „detectomat“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个检测器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回特征值。

例子

cycle = det.AttValue(„CONTROLLER“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个检测器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 特征值（根据特征类型）

例子

det.AttValue(„CONTROLLER“) = 1

属性览表:

R	W	属性	描述
✓		ID	标识编号
✓	✓	NAME	名称
✓		CONTROLLER	信号控制机标识编号
✓	✓	DETECTION	读取：1，如果从前一个信号控制步长开始检测器上有一个车辆，否则为 0。写入：1，产生一个脉冲，表示一辆车在当前时间步长内到达检测器（例如，前端被检测）但在当前的步长里仍未离开；0 就没有创建脉冲。（该值在下一个时间步长开始的时候会重置为 0 。在同一个时间步长里，设置这个就取消了之前 PRESENCE 的设置）

R	W	属性	描述
✓		HEADWAY	时间空挡，以秒为单位。0，表示直到当前信号控制步长结束时，检测器仍旧被占有。
✓		IMPULSE	返回：1，表示脉冲记录为 1(例如，在当前信号控制步长检测到车辆前端通过检测器)，否则为 0。
✓		LINK	路段标识编号
✓		LANE	车道编号
✓		POSITION	在路段上的位置
✓		DETECTORLENGTH	检测器的长度
✓		OCCUPANCY	自从车辆到达检测器起经过的时间，单位是秒。0 表示当前信号控制步长没有检测到车辆。
✓		OCCUPANCYRATE	平滑处理后的占有率
✓	✓	PRESENCE	读取：如果在最后一个仿真的时间步长结束时，一个车辆已经在检测器上了，则为 1，否则为 0。（注意：与 VAP 的 presence() 方法不同！） 写入：1 表示在当前仿真步长结束时把检测器设置为已占有（如果在前一个时间步长结束时检测器没有被占有，则现在一辆车的前部被检测到）。0 就结束占有状态。（在下一个仿真步长开始时，这个 1 的值不会自动重置为 0！在同一步长里该设置取消了之前 DETECTION 的设置）
✓		VEHICLEID	如果在当前仿真步长有车辆通过检测器，就会返回该车辆的编号。否则为 0。如果有不只一辆车被检测到，则返回结果也会改变。
✓		VEHICLELENGTH LENGTH	当前信号控制步长内，最近一次检测到的车辆长度。如果没有车辆，则为 0。
✓		VELOCITY SPEED VEHICLESPEED	当前信号控制步长内，最近一次检测到的车辆长度。如果没有车辆，则为 0。

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [out, retval] VARIANT *pValue)

带一个参数返回一个检测器的特征属性。请查看下表中与所设语言无关的特征属性标识编号。

参数

[in] BSTR Attribute : 属性名称(见下)

[in] VARIANT Parameter : 与属性相关的参数(见下)

[out, retval] VARIANT *pValue : 返回属性值

AttValue1 ([in] BSTR Attribute, [in] VARIANT Parameter, [in] VARIANT Value)

带一个参数设置一个检测器的特征属性。请查看下表中与所设语言无关的特征属性标识编号。

参数

[in] BSTR Attribute : 属性名称(见下)

[in] VARIANT Parameter : 与属性相关的参数(见下)

[in, retval] VARIANT Value : 属性值（类型根据属性）

属性览表:

R	W	属性	描述
✓	✓	VEHICLECLASS	如果车辆类别受影响，则为真。 参数：车辆类别的编号（0 表示的是所有的车辆类型）

例子

在两个车辆类别之间启动检测器：

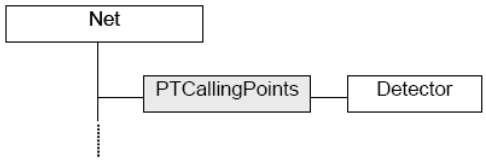
如果一个检测器只对一个车辆类别是激活的，那么用户不能把这个检测器针对该车辆类别设置为非激活。

```
Const ExpressBusses As Integer = 4000
Const Busses As Integer = 5000
DIM det AS Detector
det.AttValue1("VEHICLECLASS",Busses) = True
det.AttValue1("VEHICLECLASS",ExpressBusses) = False
```

因此建议再非激活前先设置激活后者：

```
det.AttValue1("VEHICLECLASS",ExpressBusses) = True
det.AttValue1("VEHICLECLASS",Busses) = False
```

3.6.2.3 PTCallingPoints



PTCallingPoints 对象是 Detector 对象的集合，定义为 public transport calling point（见第 160 页）。它隶属于 NET 对象，并且可以通过 INET 接口对 PTCallingPoints 属性进行操作。

它包含了路网中的 PT calling point（检测器的特别类型），并同时允许集合内部 Detector 对象的循环操作，以及内部 Detector 对象的单独操作。

例子

创建 PTCallingPoints 对象的实例，并对其中所有定义为 PT calling point 的 Detector 对象进行操作：

```
DIM vissim AS Vissim
DIM ptcps AS PTCallingPoints
DIM det AS Detector
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET dets = Vissim.Net.PTCallingPoints
FOR EACH det IN ptcps ‘ 进入_NewEnum，创建一个枚举
...
NEXT det
‘or also：
FOR i = 1 TO ptcps.Count
SET det = ptcps (i) ‘or ptcps.Item(i)
…
NEXT i
```

IPTCallingPoints 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性可以创建所有定义为 PT CallingPoint 的 Detecor 对象的集合（枚举）。该集合一旦创建，集合中的具体对象可以通过 GetDetectorByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见上文中的 Detector 一例）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举。

Item ([in] VARIANT Index, [out, retval] IDetector **ppPTCallingPoint)

返回集合中选中位置上的 Detector 对象。这只有在所有用作发送公共交通信号的检测器处于可选情况时有效（见上文中的 Detector 一例）。通过 GetPTCallingPointByNumber 可以根据标识编号对检测器进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

SET det = ptcps(1)

SET det = ptcps.Item(1)

参数

[in] long Index : 介于 1 至 ptcps.Count 之间的索引

[out, retval] IDetector **ppPTCallingPoint : 返回 Detector 对象。

Count ([out, retval] long *pCount)

返回集合中的 Detector 对象数量，定义为 PT CallingPoint。见上文中 Detectors 的例子。

参数

[out, retval] long *pCount : 返回对象数量。

IDetectors 接口的使用方法

GetPTCallingPointByNumber ([in] long Number, [out, retval] IDetector ** ppPTCallingPoint)

返回集合中标识编号为 Number 且定义为 PT CallingPoint 的 Detector 对象。

参数

[in] long Number : 标识编号

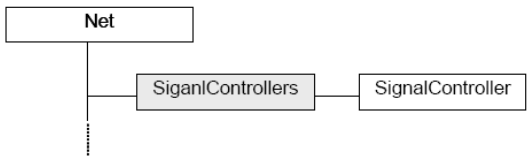
[out, retval] IDetector ** ppPTCallingPoint: 返回 Detector 对象。

例子

DIM det AS Detector

SET det = ptcps.GetPTCallingPointrByNumber(1)

3.6.2.4 SignalControllers



Signal-Controllers 对象是 SignalController 对象（见见第 158 页）的集合。它隶属于 Net 对象，并且可以通过 INet 接口进行 Signal-Controllers 属性操作。

它包括了路网中所有的信号控制机，并同时允许集合内部的循环操作，以及内部对象的单独操作。

例子

创建 SignalControllers 对象实例对其内部所有的 SignalController 对象进行操作：

```
DIM vissim AS Vissim
DIM controllers AS SignalControllers
DIM controller AS SignalController
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET controllers = Vissim.Net.SignalControllers
FOR EACH controller IN controllers ‘进入_NewEnum，创建一个枚举
…
NEXT controller
‘or also:
FOR i = 1 TO controllers.Count
SET controller = controllers(i) ‘or controllers.Item(i)
…
NEXT i
```

ISignalControllers 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 SignalController 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetSignalControllerByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 SignalControllers 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回 Enumeration

Item ([in] VARIANT Index, [out, retval] ISignalController **ppSC)

返回集合中选中位置上的 SignalController 对象。这只有在所有信号控制机可选的情况下有效（见上文中的 SignalControllers 一例）。通过 GetSignalControllerByNumber 可以根据标识编号对信号控制机进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

```
SET sc = controllers(1)
SET sc = controllers.Item(1)
```

参数

[in] long Index :	介于 1 至 controllers.Count 之间的索引
[out, retval] ISignalController **ppSC :	返回 SignalController 对象。

Count ([out, retval] long *pCount)

返回集合中的 SignalController 对象数量。见如上 SignalControllers 的例子。

参数

[out, retval] long *pCount :	返回对象数量。

ISignalControllers 接口的使用方法

GetSignalControllerByNumber ([in] long Number,[out, retval] ISignalController **ppSC)

返回标识编号为 Number 的 SignalContorller 对象。

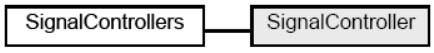
参数

[in] long Number :	标识编号
[out, retval] ISignalController *ppSC :	返回 SignalController 对象。

例子

```
DIM sc AS SignalController
SET sc = controllers.GetSignalControllerByNumber (11)
```

3.6.2.5 SignalController



SignalController 对象隶属于网络中 SignalControllers 对象。它可以使用以下两种方法通过 SignalController 对象来进行操作：

► 通过集合的循环操作

```
DIM controllers As SignalControllers
DIM controller As SignalController
SET controllers = Vissim.Net.SignalControllers
FOR EACH controller IN controllers
```

```
List.AddItem controller.ID
NEXT controller
```

► 通过标识编号对具体的 SignalController 对象进行操作

```
DIM controllers As SignalControllers
DIM controller As SignalController
SET controllers = Vissim.Net.SignalControllers
SET controller = controllers.GetSignalControllerByNumber(1000)
```

SignalController 对象能够通过 ISignalController 接口对信号控制机属性进行操作。

ISignalController 接口的属性

ID ([out, retval] long *pID)

返回信号控制机的标识编号。

参数

[out, retval] long *pID : 返回标识编号。

例子

```
id = sc.ID
```

Name ([out, retval] BSTR *pName)

返回信号控制机的名称。

参数

[out, retval] BSTR *pName : 返回名称。

例子

```
name = sc.Name
```

Name ([in] BSTR Name)

设置信号控制机的名称

参数

[in] BSTR Name : 新名称

例子

```
sc.Name = „CanPanegre“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个信号控制机属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回特征值。

例子
cycle = sc.AttValue(„CYCLETIME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个信号控制机属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 特征值（根据特征类型）

例子
sc.AttValue(„TYPE“) = 1 ‘fixed time

属性览表:

R	W	属性	描述
✓		ID	标识编号
✓	✓	NAME	名称
✓	✓	CYCLETIME	周期时间[s]
✓	~~✓~~	CONTROLLER FREQUENCY	每个仿真秒通过控制机逻辑的次数。 在 VISSIM5.2 中写入功能被移除了。
✓	✓	LOGFILE	记录文件 (*.ldp)的文件名
✓	✓	PROGRAM	激活的 程序编号
✓	✓	OFFSET	偏移时间[s]
✓		SUPPLYFILE1	信号控制供应文件 1
✓		SUPPLYFILE2	信号控制供应文件 2
✓	✓	TYPE	类型(1: Fixed time, 2: SDM, 3: VS_PLUS, 4: TRENDS, 5: VAP, 6: TL, 7: VOS, 8: NEMA, 9: External, 10:EconoliteASC3, 11:SCATS, 12:RBC, 13:LISA, 14:SCOOT)

Detectors ([out, retval] IDetectors **ppDetectors)

创建一个 Detectors 对象的实例（见 第 141 页），这样就可以对路网中的检测器元素进行单独的操作。

参数
[out, retval] IDetectors**ppDetectors : 返回 Detectors 对象。

例子
DIM dets AS Detectors

SET dets = sc.Detectors

SignalGroups ([out, retval] ISignalGroups **ppSignalGroups)

创建一个 SignalGroups 对象的实例（见第 162 页），这样就可以对路网中的信号控制组元素进行单独的操作。

参数

[out, retval] ISignalGroups **ppSignalGroups : 返回 SignalGroups 对象。

例子

DIM sgs AS SignalGroups

SET sgs = sc.SignalGroups

ISignalController 接口的

ReadChannel([in] int channelId, [out, retval] long *pValue)

返回信号控制机特别频道上的值。如果该频道不存在，该函数总是返回-1 。

参数

[in] int channeled: The channel id

[out, retval] long *pValue: The value on the channel, if exists, otherwise -1

WriteChannel([in] int channelId, [in] long value)

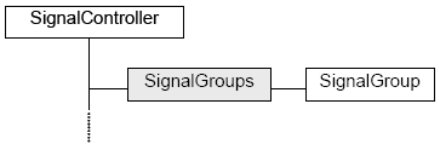
对一个信号控制机特别频道上设置一个值。如果该频道没有连接到另一个信号控制机上的一个频道的话，一个临时的数据链接竟被插入，但是没有信号控制输入信息。

参数

[in] int channeled: The channel id

[in] int value: The new value on the channel

3.6.2.6 SignalGroups



SignalGroups 对象是对 SignalGroup 对象（见第 164 页）的集合。它隶属于 Signal-Controller 对象。可以通过 ISignalController 接口进行 SignalGroups 属性操作。

它包括了所有的相关信号控制机组，并同时允许信号控制组集合内部的循环操作，以及信号控制组对象的单独操作。

例子

创建 SignalGroups 对象实例对其内部所有的 SignalGroup 对象进行操作：

DIM vissim AS Vissim

```
DIM groups AS SignalGroups
DIM group AS SignalGroup
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET groups = Vissim.Net.SignalControllers(1).SignalGroups
FOR EACH group IN groups ‘进入_NewEnum，创建一个枚举
…
NEXT group
‘or also:
FOR i = 1 TO groups.Count
SET group = groups (i) ‘or groups.Item(i)
…
NEXT i
```

ISignalGroups 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 SignalGroup 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetSignalGroupByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 SignalGroups 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举。

Item ([in] VARIANT Index, [out, retval] ISignalGroup **ppSG)

返回集合中选中位置上的 SignalGroup 对象。这只有在所有信号灯组可选的情况下有效（见上文中的 SignalGroups 一例）。通过 GetSignalGroupByNumber（如下）可以根据标识编号对信号控制机进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

```
SET sg = groups(1)
SET sg = groups.Item(1)
```

参数

[in] long Index :	介于 1 至 groups.Count 之间的索引
[out, retval] ISignalGroup **ppSG :	返回 SignalGroup 对象。

Count ([out, retval] long *pCount)

返回集合中的 SignalGroup 对象数量。见如上 SignalGroups 的例子。

参数

[out, retval] long *pCount : 返回对象数量。

ISignalGroups 接口的使用方法

GetSignalGroupByNumber ([in] long Number, [out, retval] ISignalGroup **ppSG)

返回表示符为 Number 的 SignalGroup 对象。

参数

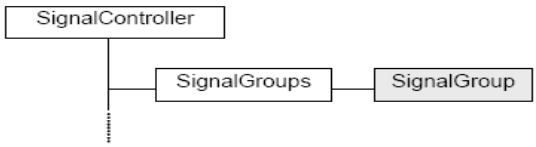
[in] long Number :	标识编号编号
[out, retval] ISignalGroup **ppSG :	返回 SignalGroup 对象。

例子

```
DIM sg AS SignalGroup
SET sg = groups.GetSignalGroupByNumber (1)
```

3.6.2.7 SignalGroup

SignalGroup 对象隶属于一个控制机的 SignalGroups 对象。它可以通过以下两个方法对信号控制机进行操作：



► 通过集合的循环操作

```
DIM groups As SignalGroups
DIM group As SignalGroup
SET groups = Vissim.Net.SignalControllers(1).SignalGroups
FOR EACH group IN groups
List.AddItem group.ID
NEXT group
```

► 通过标识编号对具体的 SignalGroup 对象进行操作

```
DIM groups As SignalGroups
DIM group As SignalGroup
SET groups = Vissim.Net.SignalControllers(1).SignalGroups
SET group = groups.GetSignalGroupByNumber(10)
```

SignalGroup 对象能够通过 ISignalGroup 接口对信号组属性进行操作。

ISignalGroup 接口的属性

ID ([out, retval] long *pID)

返回信号灯组的标识编号。

参数
[out, retval] long pID : 返回标识编号。

例子
id = group.ID

Name ([out, retval] BSTR *pName)

返回信号灯组名称。

参数
[out, retval] BSTR pName : 返回名称。

例子
name = group.Name

Name ([in] BSTR Name)

设置信号灯组名称。最多 255 个字符。

参数
[in] BSTR Name : 新名称。

例子
group.Name = „phaseone“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个信号灯组的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回特征值。

例子
cycle = group.AttValue(„TYPE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个信号灯组的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 特征值（根据特征类型）

例子
group.AttValue(„TYPE“) = 1

属性览表:

R	W	属性	描述
✓		ID	标识编号
✓	✓	NAME	名称
✓	✓	AMBER	黄灯时间 [s] (0 为关闭)
✓	✓	CONTROLLER	信号控制机标识编号
✓		GREENEND	该信号灯组当前使用的绿灯结束 [s]。（该函数只能返回一个有效值，如果该控制机支持这个功能的话）
✓		REDEND	该信号灯组当前使用的红灯结束 [s]。（该函数只能返回一个有效值，如果该控制机支持这个功能的话）
✓	✓	REDAMBER	红/黄灯时间 [s] （0 为关闭)
✓		GREENEND2	该信号灯组当前使用的第二个绿灯结束 [s]。（该函数只能返回一个有效值，如果该控制机支持这个功能的话）
✓		REDEND2	该信号灯组当前使用的第二个红灯结束 [s]。（该函数只能返回一个有效值，如果该控制机支持这个功能的话）
✓	✓	STATE	0=默认 (表示的是：使用外部信号控制机的状态),1 = 红灯, 2 = 红灯/黄灯 3 = 绿灯, 4 = 黄灯, 5 = 关闭, 6 = 未定, 7 = 闪黄灯, 8 = 闪红灯, 9 = 闪绿灯, 10 = 闪红绿灯, 11 = 绿灯/黄灯, 12 = 红灯结束
✓	✓	TYPE	1 = 周期, 2 = 持续绿灯, 3 = 持续红灯

State ([out, retval] SignalState *pState)

根据以下 *SignalGroupState* 类型表返回当前信号灯组的状态。

属性	描述
缺省	=缺省值（意味着：使用外部信号控制机的状态）
Red	红灯
Redamber	红灯/黄灯
Green	绿灯
Amber	黄灯
Off	关闭,没有信号灯
Undefined	不确定
Amber_f	闪黄灯
Red_f	闪红灯
Green_f	闪绿灯
Redgreen_f	闪红绿灯
Greenamber	绿灯/黄灯
Off_red	关闭, 为红灯

参数

[out, retval] SignalSate *pState : 返回状态。

例子

```
DIM state AS SignalState
state = group.State
```

SignalHeads ([out, retval] ISignalHeads **ppSignalHeads)

创建一个 SignalHeads 对象（见第 167 页）并对路网中的信号灯组元素进行单独操作。

参数

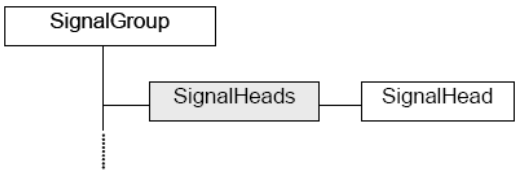
[out, retval] ISignalHeads **ppSignalHeads : 返回 SignalHeads 对象

例子

```
DIM shs AS SignalHeads
SET shs = group.SignalHeads
```

3.6.2.8 SignalHeads

SignalHeads 对象是 SignalHead 对象（见第 168 页）的集合。它隶属于 SignalGroup 对象并能够通过 ISignalGroup 接口对



SignalHeads 的属性进行操作。

它包含了信号灯组里所有的信号灯，并同时允许 SignalHead 对象集合内部的循环操作，以及 SignalHead 对象的单独操作。

例子

创建 SignalHeads 对象的实例并对其所有的 SignalHead 对象进行操作：

```
DIM vissim AS Vissim
DIM heads AS SignalHeads
DIM head AS SignalHead
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET heads = Vissim.Net.SignalControllers(1).SignalGroups(1).SignalHeads
FOR EACH head IN heads ‘进入_NewEnum，创建一个枚举
…
NEXT head
‘or also:
FOR i = 1 TO heads.Count
SET head = heads(i) ‘or heads.Item(i)
…
NEXT i
```

ISignalHeads 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 SignalHead 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetSignalHeadByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见上文中的 SignalHeads 一例）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举。

Item ([in] VARIANT Index, [out, retval] ISignalHead **ppSH)

返回集合中选中位置上的 SignalHead 对象。这只有在所有信号灯可选的情况下有效（见上文中的 SignalHeads 一例）。通过 GetSignalGroupByNumber 可以根据标识编号对信号灯进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

```
SET sh = heads(1)
SET sh = heads.Item(1)
```

参数

[in] long Index : 介于 1 至 heads.Count 之间的索引

[out, retval] ISignalController *ppSH : 返回 SignalHead 对象

Count ([out, retval] long *pCount)

在集合中返回 SignalHead 对象数量。见上文中 SignalHeads 一例。

参数

[out, retval] long *pCount : 返回对象数

ISignalHeads 接口的使用方法

GetSignalHeadByNumber ([in] long Number, [out, retval] ISignalHead **ppSH)

返回标识编号为 Number 的 SignalHead 对象。

参数

[in] long Number : 标识编号

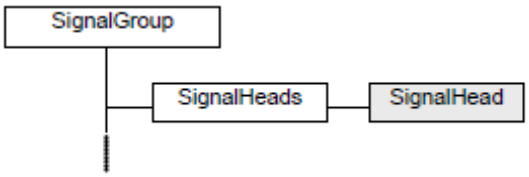
[out, retval] ISignalHead **ppSH : 返回 SignalHead 对象。

例子

```
DIM sh AS SignalHead
SET sh = heads.GetSignalHeadByNumber (1)
```

3.6.2.9 SignalHead

SignalHead 对象隶属于信号组的 SignalHeads 对象，可以通过以下两种方式对其进行操作：



► 通过集合的循环操作

```
DIM heads As SignalHeads
DIM head As SignalHead
SET heads = Vissim.Net.SignalControllers(1).SignalGroups(1).SignalHeads
FOR EACH head IN heads
List.AddItem head.ID
NEXT heads
```

► 通过标识编号对具体的对象进行操作

```
DIM heads As SignalHeads
DIM head As SignalHeads
SET heads = Vissim.Net.SignalControllers(1).SignalGroups(1).SignalHeads
SET head = heads.GetSignalHeadByNumber(10)
```

SignaHead 对象可以通过 ISignalHead 接口进入信号灯的属性。

ISignalHead 接口的属性

ID ([out, retval] long *pID)

返回信号灯的标识编号。

参数

[out, retval] long *pID : 返回标识编号。

例子

id = head.ID

Name ([out, retval] BSTR *pName)

返回信号灯的名称。

参数

[out, retval] BSTR *pName : 返回名称。

例子

name = head.Name

Name ([in] BSTR Name)

设置一个信号灯的名称。最多 255 个字符。

参数

[in] BSTR Name : 新名称.

例子

head.Name = „turnright“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个信号灯的名称。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回特征值。

例子

or = head.AttValue(„ORSIGNALGROUP“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个信号灯的名称。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 特征值（根据特征类型）

例子
head.AttValue(„ORSIGNALGROUP “) = 5

属性览表:

R	W	属性	描述
✓		ID	标识编号
✓	✓	NAME	名称
✓	✓	CONTROLLER	信号控制机标识编号
✓	✓	GROUP	信号灯组数
✓	✓	LINK	路段标识编号
✓	✓	LANE	车道数编号
✓	✓	ORGROUP	可选信号灯组
✓	✓	LINKCOORD	坐标（以当前坐标为单位）

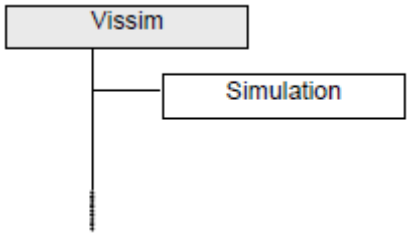
i 仿真过程中所作的改变会在设置完成后立即生效。

i 信号控制机和信号灯组必须是相辅相成的。在对信号控制机进行设置时，首先需要确定所设置的控制机与路网中存在的信号控制机组相连。换句话说，在建立新的信号控制机组时，要注意重新建立其与相应信号控制机的连接。

3.7 Simulation&Test

3.7.1 Simulation

Simulation 对象隶属于 Vissim，并可通过 IVissim 接口的 Simulation 属性来获得。它能配置以及控制仿真的运行。VISSIM 是一个单一的项目计划，也就是说，它只可以同时进行一个路网的仿真。因此，仿真实例总是指的当前的仿真，参数也是针对当前打开的路网。



例子

```
DIM vissim AS Vissim
DIM simulation AS Simulation
SET vissim = NEW Vissim
SET simulation = vissim.Simulation
vissim.LoadNet “example.inp”
simulation.RunContinuous    ‘将同步运行整个一个仿真
```

ISimulation 接口的属性

Comment ([out, retval] BSTR *pComment)

返回仿真参数的说明。

参数

[out, retval] BSTR *pComment：返回注释

例子

```
comment = simulation.Comment
```

Comment ([in] BSTR Comment)

设定仿真参数的说明。最大为 199 个字符。

参数

[in] BSTR Comment：新的注释

例子

```
simulation.Comment = „Karlsruhe-Durlach Test11“
```

Period ([out, retval] double *pPeriod)

返回仿真的时间区间。

参数

[out, retval] double *pPeriod：返回仿真的时间区间，以仿真秒为单位

例子

```
period = simulation.Period
```

Period ([in] double Period)

设置仿真的时间区间长度。

参数

[in] double Period ：新的仿真区间，以仿真秒为单位。

例子

```
simulation.Period = 3600
```

StartTime ([out, retval] BSTR *pStartTime)

返回仿真的开始时间。

参数

[out, retval] BSTR *pStartTime：返回开始时间，格式为 hh:mm:ss

例子

```
starttime = simulation.StartTime
```

StartTime ([in] BSTR StartTime)

设置仿真的开始时间。

参数

[in] BSTR StartTime：新的开始时间，格式为 hh:mm:ss

例子

```
simulation.StartTime = „10:30:00“
```

Speed ([out, retval] double *pSpeed)

返回仿真运行的速度（每个实际的秒对应运行的仿真秒大小）。若选择最大值，则它将返回一个负值。

参数

[out, retval] double *pSpeed : 返回仿真运行速度，单位是仿真秒 /秒。(如果是最大值，则返回≤ 0 的值)

例子

```
speed = simulation.Speed
```

Speed ([in] double Speed)

设置仿真运行速度（每个实际的秒对应运行的仿真秒大小）。一个负值或者 0，将切换到最大运行速度。但是如果该值小于 0 的话，该值的绝对值将作为新的仿真运行速度保存起来。

参数

[in] double Speed : 新的仿真运行速度，单位是仿真秒/秒。（≤0 的话表示的是最大速度）

例子

```
simulation.Speed = 10.0
simulation.Speed = 0        ‘最大速度，没有设置具体的速度值
simulation.Speed = -10.0    ‘设置了速度值为 10 仿真秒/秒，但是使用了最大速度进行仿真
```

Resolution ([out, retval] int *pResolution)

返回仿真精度。

参数

[out, retval] int *pResolution：返回仿真精度，单位是时间步长/仿真秒

例子

```
resolution = simulation.Resolution
```

Resolution ([in] int Resolution)

设置仿真精度。

参数

[in] int Resolution：新的仿真精度，单位是时间步长/仿真秒

例子

```
simulation.Resolution = 5    ‘ 时间步长/仿真秒
```

~~ControllerFrequency~~ ([out, retval] int *pControllerFrequency)

从 VISSIM5.20-00 开始移除该函数。

~~ControllerFrequency~~ ([in] int ControllerFrequency)

从 VISSIM5.20-00 开始移除该函数。

RandomSeed ([out, retval] long *pRandomSeed)

返回仿真的随机种子。

参数

[out, retval] long *pRandomSeed：返回仿真的随机种子

例子

```
randomseed = simulation.RandomSeed
```

RandomSeed ([in] long RandomSeed)

设置仿真的随机种子。

参数

[in] long RandomSeed：新的仿真随机种子

例子

```
simulation.RandomSeed = 13
```

BreakAt ([out, retval] double *pBreakAt)

返回仿真应该停止的那个仿真秒，然后仿真切换到单步仿真运行模式。

参数

[out, retval] double *pBreakAt：返回中断时刻对应的仿真秒的值

例子

```
breakat = simulation.BreakAt
```

BreakAt ([in] double BreakAt)

设置仿真应该停止的那个仿真秒，然后仿真切换到单步仿真运行模式。

参数

[in] double BreakAt ：新的中断时刻对应的仿真秒的值

例子

```
simulation.BreakAt = 1000  ‘当运行到 100 秒，停止仿真运行
```


由于属性BreakAt，连续仿真运行后将停止，继续仿真可以通过RunSingleStep来进行单步仿真运行，或者通过RunContinuous来进行连续仿真。如果需要启动一个整的新的仿真，在调用Run之类函数之前必须首先调用Stop函数。

LeftSideTraffic ([out, retval] unsigned char *pLeftSideTraffic)

如果交通规则设置为“左行”的话，则返回 true，否则为 false。

参数

[out, retval] unsigned char *pLeftSideTraffic ：左行规则的话为 1，右行规则的话则为 0。

例子

```
is_leftsidetraffic = simulation.LeftSideTraffic
```

LeftSideTraffic ([in] unsigned char LeftSideTraffic)

将左行交通打开/关闭。

参数

[in] unsigned char LeftSideTraffic：0 表示左行交通关闭，否则为开

例子

```
simulation.LeftSideTraffic= 1  ‘交通规则设置为左行交通
```

RunIndex ([out, retval] long *pIndex)

返回内部的仿真运行索引值，当有几个仿真相继运行时，需要用到该索引。这个索引只能由该特性的 put version 来改变（见下文）。

参数

[out, retval] long *pIndex：返回索引

例子
index = simulation.RunIndex

RunIndex ([in] long Index)

设置仿真运行索引值。它对一些评价文件（如路段及路径评价）有影响，当相继运行仿真时有作用。当 Microsoft Access 数据库用来记录评估的结果时，该值必须初始化为 0，每一次迭代增量为 1。

参数
[in] long Index： 新的索引

例子
simulation.RunIndex = 2

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个仿真的属性。请从本节末的表格中获得与语言无关的属性标签。

参数
[in] BSTR Attribute： 属性名称（见下）

[out, retval] VARIANT*pValue： 返回属性值

例子
period = simulation.AttValue(„PERIOD“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个仿真属性。请从本节末的表格中获得与语言无关的属性标签。

参数
[in] BSTR Attribute： 属性名称（见下）

[in] VARIANT Value： 属性值（类型根据属性的定义）

例子
simulation.AttValue(„PERIOD“) = 3600

属性览表:

R	W	属性	描述
✓	✓	ARCHIVEFILES	路径和费用文件 归档（ * .weg, *.bes ）(True/false)
✓	✓	BREAKAT	在所给的仿真秒处中断
✓	✓	COMMENT	字符串的说明
~~✓~~	~~✓~~	~~CONTROLLER FREQUENCY~~	在 VISSIM5.20-00 中已移除该功能
✓	✓	DIRECTORY	多次运行模式的输出目录

R	W	属性	描述
✓		ELAPSEDTIME	从仿真开始后的仿真时间（秒）
✓		ISRUNNING	仿真在运行（True/False）
✓	✓	LEFTSIDETRAFFIC	0 为右行交通 , ≠ 0 为左行交通
✓	✓	NUMRUNS	多运行模式中仿真运行总次数
✓	✓	PERIOD	仿真时间区间，以仿真秒为单位
✓	✓	RANDOMSEED	随机种子
✓	✓	RANDOMSEEDINC	多运行模式中的随机种子的增量
✓	✓	RESOLUTION	仿真精度，单位时间步长/仿真秒
✓	✓	RUNINDEX	相继仿真运行的运行索引值
✓	✓	SEARCHPATHS	搜索新的动态分配路径的选择标志
✓	✓	SPEED	仿真速度，单位：仿真秒/秒，≤0 是选择了最大速度。
✓	✓	STARTTIME	开始时间，字符串类型，格式”hh:mm:ss”
✓	✓	STATUSLINE	仿真时，允许/禁用状态栏
✓	✓	STORECOSTS	在 BEW 文件中储存动态分配的费用的选择标志
✓	✓	STOREPATHS	在 WEG 文件中储存动态分配的路径的选择标志
✓	✓	THREADS	仿真使用的核数。在仿真开始前设定
✓		TIME	仿真时间 [hh:mm:ss]
✓	✓	VOLUME	多运行模式中动态分配量的增量

ISimulation 接口的使用方法

RunContinuous

开始或继续（如果从单步仿真模式中切换出来）连续仿真（详见在预定的仿真时间内用于切换到单步模式的 BreakAt 属性）。如果当前正在运行一个仿真，则先要停止该运行。

例子

simulation.RunContinuous

RunSingleStep

执行一个仿真时间的步长的仿真运行（若没有仿真运行，则开始一个仿真运行）。如果当前正在运行一个仿真，则先要停止该运行。

例子

simulation.RunSingleStep

RunMulti ([in, defaultvalue(10)] long NumRuns)

开始一个由<NumRuns>构成的多次仿真运行。

例子

simulation.AttValue("RANDOMSEEDINC") = 3

simulation.AttValue("DIRECTORY") = 'C:\example\output1'

simulation.RunMulti(5)

Stop

终止仿真运行。如果仿真没有运行，此命令可被忽略。

例子

simulation.Stop

LoadSnapshot ([in, defaultvalue("")] BSTR SnapshotPath)

读取之前保存的仿真状态快照文件，开始单步仿真模式。

参数

[in] BSTR SnapshotPath：需要读取的 snapshot 文件的路径+文件名(*.snp)

例子

SET vissim = NEW Vissim

vissim. Simulation.LoadSnapshot "c:\vissim\data\example.snp"

SaveSnapshot ([in, defaultvalue("")] BSTR SnapshotPath)

保存当前的仿真状态为一个 snapshot 文件(*.snp)。

参数

[in] BSTR SnapshotPath：需要保存的 snapshot 文件的路径+文件名(*.snp)

例子

SET vissim = NEW Vissim

vissim.Simulation.SaveSnapshot "c:\vissim\data\example.snp"

所有仿真的方法（包括 RunContinuous 在内）都是同步执行。不能处理其他命令，直至目前的一个命令执行完毕才可以。按照 Visual Basic 程序的语句，它是指用户将等待，直到 VISSIM COM Server 已经执行完命令。详见在预定的仿真时间内、连续模式下用于中断仿真的属性 BreakAt。

例子

- 首先前 1000 仿真秒以连续仿真的形式运行，然后切换为单步仿真运行模式：

```
simulation.BreakAt = 1000
simulation.RunContinuous
simulation.RunSingleStep
```

- 运行一个仿真，手动可以停止：

```
FOR i = 0 TO simulation.Period      ‘如果仿真精度是 1 时间步长/仿真秒
simulation.RunSingleStep

DOEVENTS    ‘允许 Visual Basic 检测到新的事件
IF stop_pushed THEN GOTO the_end
NEXT I

the_end:
simulation.Stop
```

- 连续运行 10 次仿真，使用不同的运行索引和随机种子：

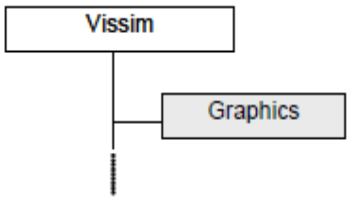
```
simulation.RunIndex = 0
FOR i = 0 to 10
simulation.RunContinuous
simulation.RunIndex = simulation.RunIndex + 1
simulation. RandomSeed = simulation.RandomSeed + 1
NEXT i
```

如果用户使用数据库评价，索引应从 0 开始，否则将不会创建数据库，且出现错误信息。

3.8 Graphics & Presentation

3.8.1 Graphics

Graphics 对象属于 Vissim，可通过 IVissim 接口的属性 Grahpics 来获取。它提供路网的图形选项。这些选择可应用在于全局范围内 Vissim 的实例，并对所有路网和仿真产生影响。VISSIM 内部设定的默认图形选项是用于创建一个新的实例。使用 IVissim 的方法 LoadLayout 可以读取一个已保存的图形设置。



例子

```
DIM vissim AS Vissim
DIM graphics AS Graphics
SET vissim = NEW Vissim
SET graphics = vissim.Graphics
```

IGraphics 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个图形属性。请从本节末的表格中获得与语言无关的属性标签。

参数

[in] BSTR Attribute： 属性的名称（见下表）

[out, retval] VARIANT *pValue： 返回该属性值

例子

```
type = graphics.AttValue(„VISUALIZATION“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设定一个图形的属性。请从本节末的表格中获得与语言无关的属性标签。

参数

[in] BSTR Attribute： 属性名称（见下表）

[in] VARIANT Value： 属性值（类型根据属性的定义）

例子

```
graphics.AttValue(„VISUALIZATION“) = 0 ‘不激活
```

属性览表:

R	W	Attribute	Description
√	√	3D	0 = 2D 模式, 1 = 3D 模式
√	√	DISPLAY	0 = 普通, 1 = 中心线, 2 =不可见, 3 =alternative
√	√	LAND	土地的颜色，数据类型为 Long，以 RGB 格式
√	√	LINKS	路段的颜色，数据类型为 Long，以 RGB 格式
√	√	SKY	天空的颜色，数据类型为 Long，以 RGB 格式
√	√	VISUALIZATION	单个车辆/集聚值可视化开/关(true/false)
√	√	PEDVISUALIZATION	行人的显示 开/关（1/0）

注：颜色值从 VISSIM 与 VISSIM 之间的传递和返回必须是以 RGB 格式。正常 RGB 颜色的有效范围为 0 到 16,777,215 (&HFFFFFF)。每种颜色设置（属性或分辨率）为一个 4 字节整数。高字节的数在这个范围内等于 0。低三字节，从最小至最大的字节分别确定红，绿，蓝的份额。红，绿，蓝的组成成分分别用一个介于 0 和 255 (&HFF)的数字代表。在 Visual Basic 中，用户可以用 RGB(red, green, blue)函数来获取得相应的整数。

如果用户所工作的应用程序对字节顺序有要求，则用户可以在 VB 环境下使用下列方法：

```
color = CLng(blue + (green*&H100) + (red*&H10000))
rgb = RGB(&HFF And (color\&H10000), &HFF And (color\&H100), &HFF And color)
```

IGraphics 接口的使用方法

Redraw ()

以当前图形选项，更新路网中的所有元素。

例子

```
Dim vissim As Vissim
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
vissim.Graphics.Redraw
```

GetWindow ([in] BSTR WinName, [out] VARIANT *Top, [out] VARIANT *Left, [out] VARIANT *Bottom, [out] VARIANT *Right, [in, defaultvalue(0)] long ID)

把视图窗口的位置退回到属性 WinName 和 可选的 ID 编号对应的窗口。屏幕的坐标值是相对于屏幕的左上角的地方的。

参数

[in] BSTR Attribute： 窗口名称（见下表）

[out] VARIANT *Top： 屏幕的上边界坐标

[out] VARIANT *Left： 屏幕的左边界坐标

[out] VARIANT *Bottom ： 屏幕的下边界坐标

[out] VARIANT *Right : ： 屏幕的右边界坐标

[in] long ID ： 可选的窗口 ID 编号（见下表）

例子

SET gr = vissim.Graphics

gr.GetWindow “SIGNALTT”, top, left, bottom, right, 1

SetWindow ([in] BSTR WinName, [in] VARIANT Top, [in] VARIANT Left, [in] VARIANT Bottom, [in] VARIANT Right, ,m[in, defaultvalue(0)] long ID)

设置视图窗口的位置到属性 WinName 和 可选的 ID 编号对应的窗口。屏幕的坐标值是相对于屏幕的左上角的地方的。

参数

[in] BSTR Attribute： 窗口名称（见下表）

[in] VARIANT Top ： 屏幕的上边界坐标

[in] VARIANT Left ： 屏幕的左边界坐标

[in] VARIANT Bottom ： 屏幕的下边界坐标

[in] VARIANT Right ： 屏幕的右边界坐标

[in] long ID ： 可选的窗口 ID 编号（见下表）

例子

SET gr = vissim.Graphics

gr.SetWindow “SIGNALTT”, 800, 1000, 900, 1100, 1

WinName	ID	描述
NETWORK	-	目前的主窗口
SCDETRECORD	信号控制机 SC 标识编号	ID 信号控制机的信号控制检测器记录窗口
SIGNALTT	信号控制机 SC 标识编号	ID 信号控制机的信号时刻表窗口

Screenshot ([in] BSTR FileName, [in, defaultvalue(1)] double sizeFactor)

它根据扩展创建格式化 VISSIM 主窗口的图形文件：PNG， TIFF，GIF，JPG，JPEG 或 BMP。若扩展不能被识别，将被写入一个 BMP 文件中。

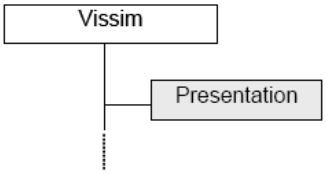
参数

[in] BSTR FileName： 文件的名称，含扩展名

[in, defaultvalue(1)] double sizeFactor： 相对的尺寸系数

3.8.2 Presentation

Presentation 对象属于 Vissim，可通过 IVissim 接口的属性 Presentation 获取。它允许记录及控制演示。



例子

```
DIM vissim AS Vissim
DIM present AS Presentation
SET vissim = NEW Vissim
SET present = vissim.Presentation
```

IPresentation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个演示的属性。请从本节末的表格中获得与语言无关的属性标签。

参数

[in] BSTR Attribute： 属性名称（见下表）

[out, retval] VARIANT *pValue ： 返回属性值

例子

```
backwards = present.AttValue(„BACKWARDS“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设定一个演示的属性。请从本节末的表格中获得与语言无关的属性标签。

参数

[in] BSTR Attribute： 属性名称（见下表）

[in] VARIANT Value ： 返回属性值（类型根据属性的定义）

例子

```
present.AttValue(„RECORDING“) = TRUE
```

属性览表:

R	W	Attribute	Description
√	√	BACKWARDS	向后运行的动画
√	√	FORWARDS	向前运行的动画
√		NINTERVALS	返回记录时间区间的个数
√	√	RECORDING	在仿真时段内记录该动画
√		TIMEFROM	时间间隔开始 [s]。 参数: 时间间隔顺序(1..n)
√		TIMEUNTIL	时间间隔结束 [s]。 参数: 时间间隔顺序(1..n)

目前动画文件(*.ani)与*.inp的文件同名，当记录时，如果该文件已存在，则原文件将被覆盖。同时，在仿真开始前，需设定记录属性。如果启动，整个路网的动画都会记录下来，直到仿真结束。

IPresentation 接口的使用方法

RunContinuous

在连续演示模式下，开始或者继续（如果从单步仿真模式中切换过来）一个动画，使用*.inp 文件的名字来定义*.ani 文件。如果一个演示当前在运行，它将事先停下来。

例子

present.RunContinuous

RunSingleStep

执行一个演示步长（如果没有演示在运行将打开一个演示）。*.ani 文件和*.inp 文件同名。如果一个演示在当前运行，需要将它事先停止。

例子

present.RunSingleStep

Stop

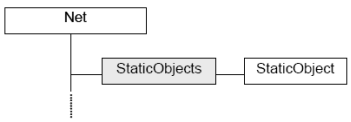
停止一个演示。如果没有演示正在运行，那么这个命令可被忽略。

例子

present.Stop

3.8.3 StaticObjects

StaticObjects 对象是 StaticObject 对象（见第 185 页）的集合。 它隶属于 Net 对象，并且可以



通过 INet 接口进行 StaticObjects 属性操作。

它包括了网络中所有的 3D 静态对象，并同时允许集合内部的循环操作，以及内部对象的单独操作（同见 StaticObject 对象）。

例子

创建 StaticObjects 对象实例对其内部所有的 StaticObject 对象进行操作：

```
DIM vissim As Vissim
DIM stobjs As StaticObjects
DIM stobj As StaticObject
SET vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
SET stobjs = Vissim.Net.StaticObjects
FOR EACH stobj IN stobjs ‘进入_NewEnum，创建一个枚举
…
NEXT stobj
‘or also:
FOR i = 1 TO stobjs.Count
SET stobj = stobjs(i) ‘or stobjs.Item(i)
…
NEXT i
```

IStaticObjects 接口的属性

_NewEnum ([out, retval] LPUNKOWN *ppEnum)

该属性能够创建一个包含所有 StaticObject 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetStaticObjectByNumbe 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 StaticObjects 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回 Enumeration

Item ([in] VARIANT Index, [out, retval] IStaticObject **ppStaticObject)

返回集合中选中位置上的 StaticObject 对象。这只有在所有静态对象可选的情况下有效（见上文中的 StaticObjects 一例）。通过 GetStaticObjectByNumber 可以根据标识编号对静态对象进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义。

```
SET stobj = stobjs(1)
SET stobj = stobjs.Item(1)
```

参数

[in] long Index : 在 1 至 Count 之间的索引

[out, retval] IStaticObject **ppStaticObjects:　返回 StaticObject 对象。

Count ([out, retval] long *pCount)

返回集合中的 StaticObject 对象数量。见如上 StaticObjects 的例子。

参数

[out, retval] long *pCount : 返回对象数。

IStaticObjects 接口的使用方法

GetStaticObjectByName ([in] BSTR Name, [in, defaultvalue(0)] IWorldPoint *pWorldPoint, [out, retval] IStaticObject **ppStaticObject)

返回名称为 Name，且 world 坐标为 pWorldPoint 的 StaticObject 对象。静态对象的名称是该 3D 模型（*.v3d）的文件名。如果找不到指定的 world 坐标，方法会返回搜索到的第一个符合名称为 Name 条件的对象。

参数

[in] BSTR Name : 名称（*.v3d 文件的名称）

[in] IWorldPoint *pWorldPoint : world 坐标系中可选的位置

[out, retval] IStaticObject **ppStaticObject : 返回 StaticObject 对象

例子

```
SET stobj = stobjs.GetStaticObjectByName(“Biz-Man_suit.v3d”)
IF NOT (stobj IS NOTHING) THEN
state = stobj.AttValue(“STATE”)
END IF
```

GetStaticObjectByCoord ([in] IWorldPoint *pWorldPoint, [in, defaultvalue(“”)]BSTR Name, [out, retval] IStaticObject **ppStaticObject)

返回 world 坐标为 pWorldPoint，且名称为 Name 的 StaticObject 对象。静态对象的名称为其三维模型（ *. v3d ）的文件名 。如果找不到指定的名称 Name，方法会返回搜索到的第一个符合坐标为 pWorldPoint 条件的对象。

参数

[in] IWorldPoint *pWorldPoint : 对象所在位置的 world 坐标

[in] BSTR Name : 可选名称（ *.v3d 文件的名称）

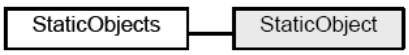
[out, retval] IStaticObject **ppStaticObject : 返回 StaticObject 对象。

例子

```
DIM wp AS WorldPoint
SET wp = vissim.NewWorldPoint(100, 100, 100)
SET stobj = stobjs.GetStaticObjectByCoord(wp)
IF NOT (stobj IS NOTHING) THEN
```

```
state = stobj.AttValue("STATE")
END IF
```

3.8.4 StaticObject



一个 StaticObject 对象代表了一个静态对象元素，隶属于 StaticObjects 对象。它可以使用以下两种方法通过 StaticObjects 对象来进行操作：

► 通过集合的循环操作

```
DIM stobj As StaticObject
FOR EACH stobj IN vissim.Net.StaticObjects
List.AddItem stobj.ID
NEXT stobj
```

► 通过标识编号对具体的 StaticObject 对象进行操作

```
DIM stobj As StaticObject
SET stobj = vissim.Net.StaticObjects.GetStaticObjectByName ("Biz-Man_suit.v3d")
```

StaticObject 对象能够通过 IStaticObject 接口对静态对象属性进行操作。

IStaticObject 接口的属性

ID ([out, retval] long *pID)

返回静态对象的标识编号。标识编号是内部生成的，并可能在路网改动时发生变化。

参数

[out, retval] long *pID : 返回标识编号。

例子

```
id = stobj.ID
```

Name ([out, retval] BSTR *pName)

返回静态对象的名称。静态对象的名称即为*.v3d 文件的文件名。

参数

[out, retval] BSTR *pName : 返回名称。

例子

```
name = stobj.Name
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个静态对象的属性。请在下表中选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回特征值。

例子

state = stobj.AttValue(„STATE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置静态对象的属性。请在下表中选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[in] VARIANT Value : 特征值（根据特征类型）

例子

stobj.AttValue(„STATE“) = 2

属性览表:

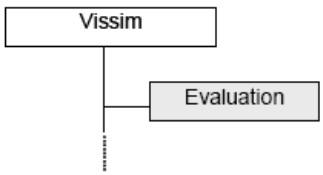
R	W	属性	描述
√		ID	标识编号
√		NAME	名称
√		NSTATES	该对象的一些可能的三维模型状态总数
√		POSITION	网络中对象的坐标
√	√	STATE	当前对象的三维模型状态

仿真过程中所作的改变会在设置完成后立即生效。例子：

```
DIM sim AS Simulation
DIM stobj AS StaticObject
SET sim = vissim.Simulation
SET stobj = vissim.Net. StaticObjects.GetStaticObjectByName (“Biz-Man_suit.v3d”)
n_states = stobj.AttValue(„NSTATES“)
FOR i = 1 TO sim.Period * sim.Resolution
sim.RunSingleStep
stobj.AttValue(„STATE“) = (stobj.AttValue(„STATE“) + 1) MOD n_states
NEXT si
```

3.9 Results

3.9.1 Evaluation



Evaluation 对象隶属于 Vissim，并能通过 IVISSIM 接口的属性 Evaluation 进行调用。它提供了仿真评价选项。这些选项能够全局地运用于 Vissim 的实例，并对所有的路网及仿真产生影响。没有一套默认的评价选项能够用于一个新实例的创建。使用 IVissim 方法 LoadLayout 载入已存的一套评价。

例子

```
DIM vissim AS Vissim
DIM eval AS Evaluation
SET vissim = NEW Vissim
SET eval = vissim.Evaluation
```

IEvaluation 接口的属性

LinkEvaluation ([out, retval] ILinkEvaluation **ppLinkEvaluation)

创建一个 LinkEvaluation 对象的实例（见 204 页），使其能够进入评价的个性化配置。

参数

[out, retval] ILinkEvaluation **ppLinkEvaluation：返回对象

例子

```
DIM linkeval AS LinkEvaluation
SET linkeval = eval.LinkEvaluation
```

DataCollectionEvaluation ([out, retval] IDataCollectionEvaluation **ppEval)

创建一个 DataCollectionEvaluation 对象的实例（见 198 页），使其能够进入评价的个性化配置。

参数

[out, retval] IDataCollectionEvaluation **ppEval： 返回对象。

例子

```
DIM dceval AS DataCollectionEvaluation
SET dceval = eval.DataCollectionEvaluation
```

QueueCounterEvaluation ([out, retval] IQueueCounterEvaluation **ppEval)

创建一个 QueueCounterEvaluation 对象的实例（见 212 页），使其能够进入评价的个性化配置。

参数

[out, retval] IQueueCounterEvaluation **ppEval ： 返回对象。

例子

```
DIM qceval AS QueueCounterEvaluation
SET qceval = eval.QueueCounterEvaluation
```

TravelTimeEvaluation ([out, retval] ITravelTimeEvaluation **ppEval)

创建一个 TravelTimeEvaluation 对象的实例（见 217 页），使其能够进入评价的个性化配置。

参数

[out, retval] ITravelTimeEvaluation **ppEval ：返回对象。

例子

```
DIM tteval AS TravelTimeEvaluation
SET tteval = eval.TravelTimeEvaluation
```

DelayEvaluation ([out, retval] IDelayEvaluation **ppEval)

创建一个 DelayEvaluation 对象的实例（见 203 页），使其能够进入评价的个性化配置。

参数

[out, retval] IDelayEvaluation **ppEval：返回对象。

例子

```
DIM deval AS DelayEvaluation
SET deval = eval.DelayEvaluation
```

NodeEvaluation ([out, retval] INodeEvaluation **ppEval)

创建一个 NodeEvaluation 对象的实例（见 207 页），使其能够进入评价的个性化配置。

参数

[out, retval] INodeEvaluation **ppEval : 返回对象。

例子

```
DIM nodeeval AS NodeEvaluation
SET nodeeval = eval.NodeEvaluation
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个评价属性。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute ： 属性名称(见下)

[out, retval] VARIANT *pValue： 返回属性值

例子

enabled = eval.AttValue(„VEHICLERECORD“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个评价属性。请在该节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute： 属性名称(见下)

[in] VARIANT Value ： 属性值（类型根据属性的定义）

例子

eval.AttValue(„VEHICLERECORD“) = True

属性览表:

R	W	属性	描述
√	√	ANALYZER	分析器数据库(True/False)
√	√	CONVERGENCE	收敛(True/False)
√	√	DATACOLLECTION	数据采集(True/False)
√	√	DELAY	延误(True/False)
√	√	EXPORT	输出(True/False)
√	√	GREENTIMES	绿灯时间分布(True/False)
√	√	LANECHANGE	车道变换(True/False)
√	√	LINK	路段评价(True/False)
√	√	MANAGEDLANES	收费车道评估(True/False)
√	√	NETPERFORMANCE	路网性能(True/False)
√	√	NODE	节点(True/False)
√	√	PATHS	路径（动态交通分配）(True/False)
√	√	PEDDATACOLLECTION	行人面域的评估(True/False)
√	√	PEDPROTOCOL	行人记录(True/False)
√	√	PEDQUEUEEVALUATION	行人排队评估(True/False)
√	√	PEDTRAVELTIME	行人的出行时间(True/False)
√	√	PUBLICWAITTIME	公交/轨道车辆等待时间(True/False)
√	√	OBSERVER	观察者评估(True/False)
√	√	QUEUECOUNTER	排队长度(True/False)
√	√	SCDETRECORD	信号控制机/检测器记录(True/False)
√	√	SIGNALCHANGES	信号变化(True/False)
√	√	SPECIAL	特殊评价(True/False)
√	√	TRAVELTIME	行程时间(True/False)
√	√	VEHICLEINPUT	车辆输入(True/False)
√	√	VEHICLERECORD	车辆记录(True/False)

3.9.2 AnalyzerEvaluation

该对象允许通过 IAnalyzerEvaluation 接口的方法和属性来配置分析器评价。

离线的评价可以通过设置适当的标志（请看 IEvaluation 接口的 ANALYZER 属性，第 190 页）。

IAnalyzerEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[out, retval] VARIANT *pValue : 返回属性值

例子

writtingfile = anaeval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[in] BSTR Value : 属性值. （根据特征类型）

例子

anaeval.AttValue(„FILE“)= FALSE

属性览表:

R	W	属性	描述
√	√	FILE	写入评价文件标志（True/false）

i 设置FILE的属性为FALSE可以避免在在线收集评价数据时（如果分析器评价已经激活）改写*.MDB数据库评价文件。该属性指定给COM接口并是VISSIM程序的全局属性（例如，它将从打开的*.INP文件中独立的保持自身的状态）。

IAnalyzerEvaluation 接口的使用方法

LoadReport ([in] BSTR ReportPath)

打开一个分析器报告数据库（*.mdb）。未检查路径和扩展名。这将由用户来确定文件格式是否合适。如果没有给出路径，一个数据库的浏览器将打开。

参数

[in] BSTR ReportPath : 要打开的报告数据库的路径

例子

anaeval.LoadReport(“example.mdb”)

3.9.3 DataCollections

DataCollections 对象是 DataCollection 对象（见第 195 页）的集合。它隶属于 Net 对象，并且能够通过 INet 接口对 DataCollections 属性进行操作。

它包含了路网中所有目前定义的数据采集评估，并同时允许集合内部对象的循环操作，以及 DataCollection 内部对象的单独操作。

例子

创建 DataCollections 对象实例对其内部所有的 DataCollection 对象进行操作：

```
Dim vissim As Vissim
Dim datacollections As DataCollections
Dim datacollection As DataCollection
Set vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set datacollections = Vissim.Net.DataCollections
FOR EACH datacollection IN datacollections ‘access to _NewEnum
…
NEXT datacollection
‘or also:
FOR i = 1 TO datacollections.Count
SET datacollection = datacollections(i) ‘or datacollections.Item(i)
…
NEXT i
```

IDataCollections 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 DataCollection 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetDataCollectionByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 DataCollections 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举。

Item ([in] VARIANT Index, [out, retval] IDataCollection **ppDC)

返回集合中选中位置上的 DataCollection 对象。这只有在所有需数据检测点可选的情况下有效（见上文中的 DataCollections 一例）。通过 Get DataCollectionByNumber 可以根据标识编号对数据收集进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

```
SET dc = DataCollections(1)
SET dc = DataCollections.Item(1)
```

参数

[in] long Index : 介于 1 和 DataCollections.Count 之间的索引号

[out, retval] IDataCollection **ppDC : 返回 DataCollection 对象。

Count ([out, retval] long *pCount)

返回集合中的 DataCollection 对象数量。见上文中 DataCollections 一例。

参数

[out, retval] long *pCount : 返回对象数量。

IDataCollections 接口的使用方法

GetDataCollectionByNumber ([in] long Number, [out, retval] IDataCollection **ppDC)

用标识号码返回 DataCollection 对象。

参数

[in] long Number : 标识编号

[out, retval] IDataCollection **ppDC : 返回 DataCollection 对象。

例子

```
DIM dc AS DataCollection
SET dc = DataCollections.GetDataCollectionByNumber(101)
```

GetIDs ([out] VARIANT *pIDs, [in, defaultvalue("")] BSTR Attribute, [in, optional] VARIANT Value, [in, defaultvalue(0)] short Compare)

返回一个数据收集标识编号的数组，把 DataCollection(Attribute) 与 Value 进行比较。该比较可以得到三个不同的值：-1 为“小于”，0 为“等于”和 1 为“大于”。若给定的特征为空，或无特征，则集合中的数组将以所有的标识编号返回。

参数

[out] VARIANT *pIDs: 返回标识编号数组（VARIANTs 数组）

[in] BSTR Attribute: 对属性加以比较。如果为空，则条件将始终为真。

[in] VARIANT Value : 需要加以比较的数值。

[in] short Compare : -1，0，1，分别代表“小于”、“等于”、“大于”。

例子

```
DIM ids() AS LONG
dcs.GetIDs ids                    ‘ ids 将包含说有的 IDs
```

GetMultiAttValues ([in] VARIANT IDs, [in] BSTR Attribute, [out] VARIANT *pValues)

该方法可以根据输入的变量 IDs，返回相应的数据收集及其所输入的属性 Attribute 的值 pValues，并按输入的 IDs 的顺序对这些返回值进行排列。如果没有输入具体标识编号（即输入的变量 IDs 为缺省值），则随后返回的数组将包含所有数据集合在非特定顺序下的值。如果输入的 IDs 是一个标识编号，而非一个数组，则该方法将根据输入的 ID 返回特定的特征值。

参数

[in] VARIANT IDs : 所要求的标识编号（VARIANTs 数组）

[in] BSTR Attribute : 属性名（见 IDataCollection 属性览表）

[out] VARIANT *pValues :返回数组值。

例子

```
dcs.GetMultiAttValues Empty, “NAME”, vals          ‘ vals 将包含所有的名称
```

SetMultiAttValues ([in] VARIANT IDs, [in] BSTR Attribute, [in] VARIANT Values)

该方法可以根据输入的变量 IDs，设置相应的数据收集的属性 Attribute 的值 Values，并按输入的 IDs 的顺序对这些返回值进行排列。如果在使用该方法时没有输入足够的属性值，则该方法会使用最末输入的属性值进行剩余对象属性的设置。超出有效范围的值也不能进行输入。如果没有输入具体标识编号号码（即输入的变量 IDs 为缺省值），则所有的数据收集器都会进行更新设置。同样也可以只输入一个属性值，这种情况下，所有的更新使用的都是这个值。

参数

[in] VARIANT IDs : 所需的标识编号（VARIANTs 数组）

[in] BSTR Attribute : 属性名（见 IDataCollection 属性览表）

[in] VARIANT Values : 值或数组的值

例子

dcs.SetMultiAttValues Empty, “NAME”, “” ‘ 删除所有名称

3.9.4 Datacollection

DataCollection 对象代表了一个数据检测器（通过离线分析的 Data collection 配置来定义），并属于 DataCollections 对象。它可以通过 DataCollections 对象的 2 种方法来操作。

- 通过集合的循环操作

```
DIM datacollection As DataCollection
FOR EACH datacollection IN vissim.Net.DataCollections
List.AddItem datacollection.ID
NEXT datacollection
```

- 通过标识编号对具体的对象进行操作

```
DIM datacollection As DataCollection
SET datacollection = vissim.Net.DataCollections.GetDataCollectionByNumber
(101)
```

DataCollection 对象能够通过 IDataCollection 接口对数据检测器的属性进行编辑。

IDataCollection 接口的属性

ID ([out, retval] long *pID)

返回数据检测器的标识编号。如果 DataCollection 对象并不是指一个有效的 VISSIM 数据检测器元素，返回值为 0 。

参数

[out, retval] long *pID : 返回标识编号

例子

id = datacollection.ID

Name ([out, retval] BSTR *pName)

返回一个数据检测器的名称。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[out, retval] BSTR *pName : 返回名称

例子
name = datacollection.Name

Name ([in] BSTR Name)

设置一个数据检测器的名称。

参数
[in] BSTR Name : 新名称.

例子
datacollection.Name = „Friedhofeingang“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个数据检测器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回特征值。

例子
name = datacollection.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个数据检测器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 特征值（根据特征类型）

例子
datacollection.AttValue(„NAME“)=„Friedhofeingang“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称
√		VEHICLEIDS	车辆标识编号

VEHICLEIDS对应的总是最后一个完整时间间隔内收集到的数据。

IDataCollection 接口的使用方法

GetResult ([in] BSTR Parameter, [in] BSTR Function, [in] long VehicleClass [out, retval] VARIANT *pValue)

该方法将返回先前设置好参数的函数和车辆类别的采集结果（见下面的注释）。如果使用函数“FREQUENCIES”，返回结果是两维数组。在这种情况下，该数组将分别保存类别的上下限和结果。

参数

[in] BSTR Parameter : 参数名（如下）

[in] BSTR Function : 函数名 （如下）

[in] long VehicleClass: 车辆类别编号. 0 代表所有车辆类型

[out, retval] VARIANT *pValue : 返回值或数组值（real 类型）

例子

```
maxspeed = datacollection.GetResult(„SPEED“, „MAX“)
minspeed = datacollection.GetResult(„SPEED“, „MIN“)
freqs = datacollection.GetResult(„SPEED“, „FREQUENCIES“, 0)
FOR i = LBOUND(freqs) TO UPBOUND(freqs)
List.AddItem “From “+ CStr(freqs(i,0)) + “, To “+ CStr(freqs(i,1)) + “ : “+ CStr(freqs(i,2))
NEXT i
```

参数览表:

参数	描述
ACCELERATION	加速度 [m/s2] [ft/s2]. 最小值, 最大值, 平均值, 频率
LENGTH	车辆长度 [m] [ft]. 最小值, 最大值, 平均值, 频率
MOTORTEMPERATURE	冷却水温度 [°C]. 最小值, 最大值, 平均值, 频率
NVEHICLES	车辆数. 总数
NPERSONS	人群数. 最小值, 最大值, 平均值,总数, 频率
OCCUPANCYRATE	占有率 [%]. 总和
QUEUEDELTIME	总的排队延误时间 [s]. 最小值, 最大值, 平均值, 总数,频率
SPEED	车速 [km/h] [mph]. 最小值, 最大值, 平均值, 频率
TACHO	路网中总的行驶距离 [m] [ft]. 最小值, 最大值, 平均值, 频率

函数览表

Function	Description
MIN	最大值
MAX	最小值
MEAN	平均值
SUM	总和
FREQUENCIES	在一个数组里所有配置的频率

说明：参数“NVEHICLES”和“OCCUPANCYRATE”能够和函数“SUM”一起使用。例如
result = datacollection.GetResult(„NVEHICLES“, „SUM“)

说明：结果对应于最后完整的时间间隔收集的数据。对应一个时间间隔的数据可以在仿真步长完成后立即使用，但是不能在仿真期间调用。

说明：为了从数据检测器获得数据，必须在需要使用前在 VISSIM 中定义相关配置并将其保存在一个后缀名为*.qmk 的文件中。数据检测器的离线分析选项必须激活。否则将返回一个错误信息（“The specified configuration is not defined within VISSIM”）

3.9.5 DataCollectionEvaluation

该对象允许通过 IDataCollectionEvaluation 接口的方法和属性来配置数据检测点。

离线的评价可以通过设置适当的标志（请看 IEvaluation 接口的 DATACOLLECTION 属性第 190 页），在线的评价通过 IEvaluation 接口的 GetResult()方法。

IDataCollectionEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名 （见下表）

[out, retval] VARIANT *pValue : 返回属性值

例子
writtingfile = dceval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[in] BSTR Value : 属性值.（根据特征类型）

例子

dceval.AttValue(„FILE“)= FALSE

属性览表

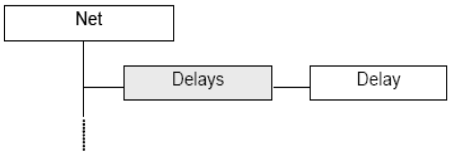
R	W	属性	描述
√	√	COMPILED	写入标志，针对编译输出（True/false）
√	√	FILE	写入评价文件标志（True/false）
√	√	RAW	写入标志，针对原始数输出（True/false）

设置FILE的属性为FALSE可以避免在在线收集评价数据时（如果数据检测评价已经激活）改写*.MES 和 *.MER评价文件。该属性指定给COM接口并是VISSIM程序的全局属性（例如，它将从打开的*.INP文件中独立的保持自身的状态）。

IDataCollectionEvaluation 接口的方法

3.9.6 Delays

Delays 对象是 Delay 对象（见第 201 页）的集合。它隶属于 Net 对象，并且能够通过 INet 接口对 Delays 属性进行操作。



它包含了路网中所有目前定义的延误时间测算（仿真运行期间包含为节点评价创建的临时行程时间），并同时允许集合内部对象的循环操作，以及 Delay 内部对象的单独操作。

例子

创建 Delays 对象实例对其内部所有的 Delay 对象进行操作：

```
Dim vissim As Vissim
Dim delays As Delays
Dim delay As Delay
Set vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set delays = Vissim.Net.Delays
FOR EACH delay IN delays ‘access to _NewEnum
…
NEXT delay
‘or also:
FOR i = 1 TO delays.Count
SET delay = delays (i) ‘or delays.Item(i)
…
NEXT i
```

IDelays 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 Delay 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetDelayByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 TravelTimes 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IDelay **ppTT)

返回集合中选中位置上的 Delays 对象。这只有在所有延误时间检测器可以调用的情况下有效（见上文中的 Delays 一例）。通过 GetDelaysByNumber （见下）可以根据标识编号对数据收集进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

```
SET delay = Delays(1)
SET delay = Delays.Item(1)
```

参数

[in] long Index : 介于 1 和 Delays.Count 的索引

[out, retval] IDelay **ppDelay : 返回 Delay 对象

Count ([out, retval] long *pCount)

返回集合中的 Delay 对象数量。见上文中 Delays 一例。

参数

[out, retval] long *pCount : 返回对象数.

IDelays 接口的使用方法

GetDelayByNumber ([in] long Number, [out, retval] IDelay **ppDelay)

用编号返回 Delay 对象。

参数

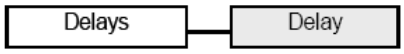
[in] long Number : 编号

[out, retval] IDelay **ppDelay : 返回 Delay 对象

例子

```
DIM delay AS Delay
SET delay = Delays.GetDelayByNumber (123)
```

3.9.7 Delay



一个 Delay 对象代表了一个行程时间，隶属于 Delays 对象。它可以通过 Delays 对象的 2 种方法来操作：

► 通过集合的循环操作

```
DIM Delay As Delay
FOR EACH Delay IN vissim.Net.Delays
List.AddItem Delay.ID
NEXT Delay
```

► 通过标识编号对具体对象进行操作

```
DIM Delay As Delay
SET Delay = vissim.Net. Delays.GetDelayByNumber (123)
```

Delay 对象能够通过 IDelay 接口对延误的属性进行编辑。

IDelay 接口的属性

ID ([out, retval] long *pID)

返回延误的标识编号。如果 Delay 对象并不是指一个有效的 VISSIM 延误测定元素，返回值为 0 。

参数

[out, retval] long *pID : 返回标识编号

例子

```
id = Delay.ID
```

Name ([out, retval] BSTR *pName)

返回一个延误的名称。如果没有名称指定，针对该延误的第一个行程时间检测器的名称将被使用。

参数
[out, retval] BSTR *pName : 返回名称

例子
name = Delay.Name

Name ([in] BSTR Name)

设置一个延误检测器的名称。

参数
[in] BSTR Name : 新名称.

例子
Delay.Name = „xxx“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个延误的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回属性值。

例子
name = Delay.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个延误属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数
[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据特征类型）

例子
Delay.AttValue(„NAME“)=„xxx“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称

IDelay 接口的使用方法

GetResult ([in] double Time, [in] BSTR Parameter, [in] BSTR Function, [in] long VehicleClass, [out, retval] VARIANT *pValue)

该方法将返回指定参数（见下面的参数表）和车辆类别的采集结果。返回值为此时刻的数据采集值，因为时间间隔关闭了特殊的时间点（它只包含已经驶过行程时间检测区段终点的车辆的行程时间）。函数参数当前无意义。

参数

[in] double Time : 时间点

[in] BSTR Parameter : 参数名 (见下表)

[in] BSTR Function : 未使用

[in] long VehicleClass: 车辆类别编号 0 代表所有类型

[out, retval] VARIANT *pValue : 返回值 (real 型)

例子

```
delay = delay.GetResult(600, „DELAY“,””, 0) ‘每辆车平均总延误
delay = delay.GetResult(600, „DELAY“,””, 1) ‘车辆类别 1 的平均总延误
```

参数览表:

参数	描述
DELAY	每车辆平均总延误[s]
PERSONS	每人平均总延误[s]
NPERSONS	通过行人
NVEHICLES	通过车辆
NSTOPS	每车辆的平均延误停车数 [s]
STOPPEDDELAY	每辆车的平均停车时间 [s]

为了取得结果，延误的离线分析选项必须激活。否则结果为0.0

需要按照车辆类别分类的数据，即使车辆类别在延误时间检测其中没有选择。

3.9.8 DelayEvaluation

该对象允许通过 IDelayEvaluation 接口的方法和

属性来配置延误时间评价。

离线的评价可以通过设置适当的标志（请看 IEvaluation 接口的 DELAY 属性第 192 页）。

IDelayEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[out, retval] VARIANT *pValue : 返回属性值

例子

writtingfile = deval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个属性值。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[in] BSTR Value : 属性值.（根据特征类型）

例子

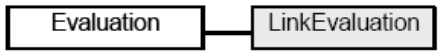
deval.AttValue(„FILE“)= FALSE

属性览表:

R	W	属性	描述
√	√	COMPILED	针对编译输出的写入标志（True/false）
√	√	FILE	写入评价文件标志（True/false）
√	√	RAW	为原始数输出写入评价文件的标志（True/false）

设置FILE的属性为FALSE可以避免在在线收集评价数据时（如果延误评价已经激活）改写*.VLZ 和 *.VLR评价文件。该属性指定给COM接口并是VISSIM程序的全局属性（例如，它将从打开的*.INP文件中独立的保持自身的状态）。

3.9.9 LinkEvaluation



该对象允许通过 ILinkEvaluation 的方法和属性来配置路段评价。

离线的评价可以通过设置适当的标志（请看 IEvaluation 接口的路段属性 第 186 页），在线的评价通过 IEvaluation 接口的 GetSegmentResult()方法（见 49 页）。

例子

打开现有的路段评价配置文件(*.sak)：

```
DIM linkeval As LinkEvaluation
SET linkeval = vissim.Evaluation.LinkEvaluation
linkeval.LoadConfiguration("c:\vissim\daten\example.sak")
```

ILinkEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[out, retval] VARIANT *pValue : 返回属性值

例子

```
interval = linkeval.AttValue(„INTERVAL“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[in] BSTR Value : 属性值.（根据特征类型）

例子

```
linkeval.AttValue(„UNTIL“)= 3600
```

属性览表:

R	W	属性	描述
√	√	DATABASE	数据库标志
√	√	EXPORT	导出到 VISUM
√	√	FILE	写入评价文件的标志（True/false）
√	√	FILENAME	配置文件的路径和文件名
√	√	FROM	第一秒
√	√	INTERVAL	数据采集的时间间隔
√	√	PERLANE	每条车道独立的数据采集器
√	√	TABELNAME	数据库名称
√	√	UNTIL	最后一秒

设置 FILE 的属性为 FALSE 可以避免在在线采集评价数据时（如果路段评价区段已经激活）改写*.STR 评价文件。该属性指定给 COM 接口并是 VISSIM 程序的全局属性（例如，它将从打开的*.INP 文件中独立的保持自身的状态）。

ILinkEvaluation 接口的使用方法

LoadConfiguration ([in] BSTR ConfigurationPath)

打开一个路段评价的配置文件(*.sak)。未检查扩展名和路径。这将由用户来决定文件的格式是否合适。如果没有给定文件的路径，一个文件浏览器将打开。

参数

[in] BSTR ConfigurationPath : 要打开的配置文件的路径

例子

linkeval.LoadConfiguration(“example.sak”)

SaveConfiguration ([in] BSTR ConfigurationPath)

以 VISSIM 格式保存当前选择的路段评价配置文件(*.sak)。未检查扩展名和路径。

参数

[in] BSTR ConfigurationPath : 要保存的配置文件的路径

例子

linkeval.SaveConfiguration(“example.sak”)

在使用LoadConfiguration()函数打开或保存一个配置文件时，当前打开路网配置文件的文件名不会更改。可以使用“FILENAME”属性来修改。如果当前路网使用了另外的一条路径，仿真开始时将报错（如果路段评价被激活）。

AddParameter ([in] BSTR Parameter, [in] long VehicleClass)

为评价添加新的属性，它受到给定车辆类别或全部车辆类别（此时车辆类别为 0）的限制。未检查是否重复。

参数

[in] BSTR Parameter : 参数名 (见下表)

[in] long VehicleClass : 车辆类别编号或所有类别

例子

linkeval.AddParameter(“SPEED”, 0) ‘ 所有车辆类型的区段平均速度

RemoveParameter ([in] BSTR Parameter, [in] long VehicleClass)

从评价参数列表中删除给定类别车辆的指定参数（0 代表全部车辆类型）。如果没有匹配的参数输入或车辆类别组合，就无法进行任何操作。如果有多个匹配，则只有第一个会被删除。

参数

[in] BSTR Parameter : 参数名称 (见下表)

[in] long VehicleClass : 车辆类别号码

例子

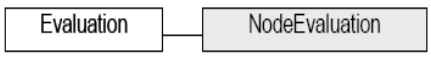
linkeval.RemoveParameter(“SPEED”, 0) ‘segment average speed for all vehicle types

参数览表:

R	W	属性	描述
√	√	TIMESTEP	仿真时间步长 [s]
√	√	SPEED	平均速度 [km/h] 或 [mph]
√	√	VOLUME	平均流量 [veh/h]
√	√	NVEHICLES	累积的车辆数
√	√	DENSITY	平均密度 [veh/km]
√	√	DELAY	相对损失时间 [s/s]

3.9.10 NodeEvaluation

该对象允许通过 INodeEvaluation 接口的方法和属



性来配置节点评价。

离线的评价可以通过设置适当的标志（请看 IEvaluation 接口的 NODE 属性第 186 页）。

INodeEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[out, retval] VARIANT *pValue : 返回属性值

例子

writtingfile = nodeeval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[in] BSTR Value : 属性值.（根据特征类型）

例子

nodeeval.AttValue(„FILE“)= FALSE

属性览表:

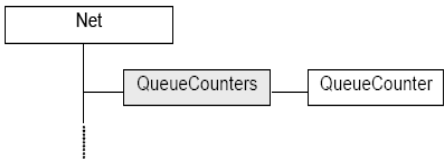
R	W	属性	描述
√	√	FILE	写入评价文件标志（True/false）

设置FILE的属性为FALSE可以避免在在线收集评价数据时（如果节点评价已经激活）改写*.KNA评价文件。该属性指定给COM接口并是VISSIM程序的全局属性（例如，它将从打开的*.INP文件中独立的保持自身的状态）。

INodeEvaluation 接口的使用方法

3.9.11 QueueCounters

QueueCounters 对象是 QueueCounter（见第 209 页）的集合。它隶属于 Net 对象，并且能够通过 INet 接口对 QueueCounters 的属性进行操作。



它包含了路网中所有目前定义的排队计数测算（仿真运行期间包含为节点评价创建的临时排队计数器），并同时允许集合内部对象的循环操作，以及 QueueCounter 内部对象的单独操作。

例子

创建 QueueCounters 对象实例对其内部所有的 QueueCounter 对象进行操作：

```
Dim vissim As Vissim
Dim queuecounters As QueueCounters
Dim queuecounter As QueueCounter
Set vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set queuecounters = Vissim.Net.QueueCounters
FOR EACH queuecounter IN queuecounters ‘access to _NewEnum
…
NEXT queuecounter
‘or also:
FOR i = 1 TO queuecounters.Count
SET queuecounter = queuecounters (i) ‘or queuecounters.Item(i)
…
NEXT i
```

IQueueCounters 的接口属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 QueueCounter 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetQueueCounterByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义_NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 QueueCounters 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] IQueueCounter **ppQC)

返回集合中选中位置上的 QueueCounter 对象。这只有在所有排队计数器可以调用的情况下有效（见上文中的 QueueCounters 一例）。通过

GetQueueCounterByNumber 可以根据标识编号对数据收集进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

SET qc = QueueCounters(1)

SET qc = QueueCounters.Item(1)

参数

[in] long Index : 介于 1 和 QueueCounters.Count 之间的索引

[out, retval] IQueueCounter **ppQC : 返回 QueueCounter 对象

Count ([out, retval] long *pCount)

返回集合中的 QueueCounter 对象数量。见上文中 QueueCounters 的例子。

参数

[out, retval] long *pCount : 返回对象数量.

IQueueCounters 接口的使用方法

GetQueueCounterByNumber ([in] long Number, [out, retval] IQueueCounter **ppQC)

用编号返回 QueueCounter 对象。

参数

[in] long Number : 编号

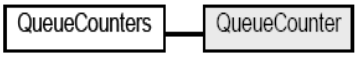
[out, retval] IQueueCounter **ppQC :返回 QueueCounter 对象

例子

DIM qc AS QueueCounter

SET qc = QueueCounters.GetQueueCounterByNumber (123)

3.9.12 QueueCounter



QueueCounter 对象代表了一个排队计数器元素，隶属于 QueueCounter 对象。它可以通过 QueueCounters 对象用 2 种方法来操作：

- 通过集合的循环操作

```
DIM queuecounter As QueueCounter
FOR EACH queuecounter IN vissim.Net.QueueCounters
List.AddItem queuecounter.ID
NEXT queuecounter
```

- 通过标识编号对具体对象进行操作

```
DIM queuecounter As QueueCounter
SET queuecounter = vissim.Net.QueueCounters.GetQueueCounterByNumber(123)
```

QueueCounter 对象能够通过 IQueueCounter 接口对排队计数器的属性进行编辑。

IQueueCounter 接口的属性

ID ([out, retval] long *pID)

返回排队计数器的标识编号。如果 QueueCount 对象并不是指一个有效的 VISSIM 排队计数器元素，返回值为 0 。

参数

[out, retval] long *pID : 返回标识编号

例子

id = queuecount.ID

Name ([out, retval] BSTR *pName)

返回一个排队计数器的名称。

参数

[out, retval] BSTR *pName : 返回名称

例子

name = queuecount.Name

Name ([in] BSTR Name)

设置一个排队计数器的名称。

参数

[in] BSTR Name : 新名称.

例子

queuecount.Name = „xxx“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个排队计数器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[out, retval] VARIANT *pValue : 返回属性值。

例子

name = queuecounter.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个排队计数器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据特征类型）

例子

queuecounter.AttValue(„NAME“)=„xxx“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称

IQueueCounter 接口的使用方法

GetResult ([in] double Time, [in] BSTR Parameter, [out, retval] VARIANT *pValue)

该方法将返回指定参数（见下面的参数表）的最后收集结果（见下面的注释）。返回值为此时刻的数据收集值，不包含特殊的时间点。

参数

[in] double Time : 时间点

[in] BSTR Parameter : 参数名 (见下表)

[out, retval] VARIANT *pValue : 返回值 (real 型)

例子

mean = queuecounter.GetResult(600, „MEAN“)

max = queuecounter.GetResult(600, “MAX”)

stops = queuecounter.GetResult(600, “STOPS”)

参数览表:

参数	描述
MEAN	平均排队长度 ([m] 或 [ft], 取决于当前单位选择)
MAX	最大排队长度 ([m] 或 [ft], 取决于当前单位选择)
NSTOPS	排队区域的停车次数

为了取得结果，排队计数器的离线分析选项必须激活。否则结果为 0.0

3.9.13 QueueCounterEvaluation

该对象允许通过IQueueCounterEvaluation接口的方法和属性来配置排队计数器。

离线的评价可以通过设置适当的标志（请看 IEvaluation 接口的 QueueCounter 属性第 186 页），在线的评价通过 IQueueCounter 接口的 GetResult()方法。

IQueueCounterEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[out, retval] VARIANT *pValue : 返回属性值

例子

writtingfile = qceval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[in] BSTR Value : 属性值.（根据特征类型）

例子

qceval.AttValue(„FILE“)= FALSE

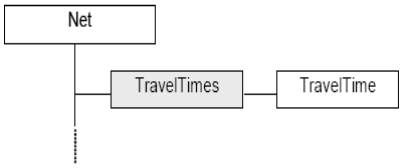
属性览表:

R	W	属性	描述
√	√	FILE	写入评价文件标志（True/false）

设置FILE的属性为FALSE可以避免在在线收集评价数据时（如果排队计数器评价已经激活）改写*.STZ评价文件。该属性指定给COM接口并是VISSIM程序的全局属性（例如，它将从打开的*.INP文件中独立的保持自身的状态）。

IQueueCounterEvaluation 接口的方法

3.9.14 TravelTimes



TravelTimes 对象是 TravelTime 对象（见第 214 页）的集合。它隶属于 Net 对象，并且能够通过 INet 接口对 TravelTimes 属性进行操作。

它包含了路网中所有目前定义的行程时间测算（仿真运行期间包含为节点评价创建的临时行程时间），并同时允许集合内部对象的循环操作，以及对 TravelTime 对象的单独操作。

例子

创建 TravelTimes 对象实例对其内部所有的 TravelTime 对象进行操作：

```
Dim vissim As Vissim
Dim traveltimes As TravelTimes
Dim traveltime As TravelTime
Set vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set traveltimes = Vissim.Net.TravelTimes
FOR EACH traveltime IN traveltimes ‘access to _NewEnum
…
NEXT traveltime
‘or also:
FOR i = 1 TO traveltimes.Count
SET traveltime = traveltimes (i) ‘or traveltimes.Item(i)
…
NEXT i
```

ITravelTimes 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

该属性能够创建一个包含所有 TravelTime 对象的集合（或枚举）。一旦该集合创建后，集合中的具体对象可以通过 GetTravelTimeByNumber 方法返回，而整个集合可以使用 FOR ...TO ... NEXT 来进行迭代。在 Visual Basic 中，你可以使用 FOR EACH…NEXT，而不需定义 _NewEnum 属性，因为 VB 可以内部读取该属性。（见如上 TravelTimes 的例子）

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举

Item ([in] VARIANT Index, [out, retval] ITravelTime **ppTT)

返回集合中选中位置上的 TravelTime 对象。这只有在所有行程时间检测器可以调用的情况下有效（见上文中的 TravelTime 一例）。通过 GetTravelTimeByNumber 可以根据标识编号对数据收集进行选择。由于 Item 是集合的默认属性，以下的两个命令其实代表的是一样的含义：

SET tt = TravelTimes(1)

SET tt = TravelTimes.Item(1)

参数

[in] long Index : 介于 1 和 TravelTimes.Count 之间的索引

[out, retval] ITravelTime **ppTT :返回 TravelTime 对象

Count ([out, retval] long *pCount)

返回集合中的 TravelTime 对象数量。见上文中 TravelTimes 一例。

参数

[out, retval] long *pCount : 返回对象数量.

ITravelTimes 接口的使用方法

GetTravelTimeByNumber ([in] long Number, [out, retval] ITravelTime **ppTT)

用编号返回 TravelTime 对象。

参数

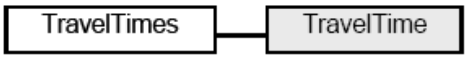
[in] long Number : 编号

[out, retval] ITravelTime **ppTT :返回 TravelTime 对象

例子

```
DIM tt AS TravelTime
SET tt = TravelTimes.GetTravelTimeByNumber (123)
```

3.9.15 TravelTime



一个 TravelTime 对象代表了一个行程时间，隶属于 TravelTimes 对象。它可以通过 TravelTimes 对象采用 2 种方法来操作：

► 通过集合的循环操作

```
DIM traveltime As TravelTime
FOR EACH traveltime IN vissim.Net.TravelTimes
List.AddItem traveltime.ID
NEXT traveltime
```

► 通过标识编号对具体对象进行操作

```
DIM traveltime As TravelTime
SET traveltime = vissim.Net. TravelTimes.GetTravelTimeByNumber (123)
```

TravelTime 对象能够通过 ITravelTime 接口对排队计数器的属性进行编辑。

ITravelTime 接口的属性

ID ([out, retval] long *pID)

返回行程时间检测器的标识编号。如果 TravelTime 对象并不是指一个有效的 VISSIM 行程时间元素，返回值为 0 。

参数

[out, retval] long *pID : 返回标识编号

例子

```
id = traveltime.ID
```

Name ([out, retval] BSTR *pName)

返回一个行程时间检测器的名称。

参数

[out, retval] BSTR *pName : 返回名称

例子

```
name = traveltime.Name
```

Name ([in] BSTR Name)

设置一个行程时间的名称。最大 255 的字符

参数

[in] BSTR Name : 新名称.

例子

```
traveltime.Name = „xxx“
```

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个行程时间检测器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute :	属性名（如下）
[out, retval] VARIANT *pValue :	返回属性值。

例子

```
name = traveltime.AttValue(„NAME“)
```

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个行程时间检测器的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名（如下）

[in] BSTR Value : 属性值（根据特征类型）

例子

traveltime.AttValue(„NAME“)=„xxx“

属性览表:

R	W	属性	描述
√		ID	标识编号
√	√	NAME	名称

ITravelTime 接口的使用方法

GetResult ([in] double Time, [in] BSTR Parameter, [in] BSTR Function, [in] long VehicleClass, [out, retval] VARIANT *pValue)

该方法将返回指定参数（见下面的参数表）和车辆类别的收集结果。返回值为此时刻的数据收集值，因为时间间隔关闭了特殊的时间点（它只包含已经驶过行程时间检测区段终点的车辆的行程时间）。函数参数当前无意义。

参数

[in] double Time : 时间点

[in] BSTR Parameter : 参数名 (见下表)

[in] BSTR Function : 未使用

[in] long VehicleClass: 车辆类别编号 0 代表所有类型

[out, retval] VARIANT *pValue : 返回值 (real 型)

例子

nv = traveltime.GetResult(600, „NVEHICLES“, “”, 0) ‘通过车辆

tt = traveltime.GetResult(600, “TRAVELTIME”, “”, 1) ‘车辆类别 1 的行程时间

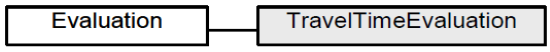
参数览表:

参数	描述
NVEHICLES	通过的车辆 (检测的车辆数)
TRAVELTIME	行程时间[s]

为了取得结果，行程时间检测器的离线分析选项必须激活。否则结果为 0.0

需要按照车辆类别分类的数据，即使车辆类别在行程时间检测器中没有选择。

3.9.16 TravelTimeEvaluation



该对象允许通过 ITravelTimeEvaluation 接口的方法和属性来配置行程时间评价。

离线的评价可以通过设置适当的标志（请见 IEvaluation 接口的 TravelTime 属性第 186 页）。

ITravelTimeEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[out, retval] VARIANT *pValue : 返回属性值

例子

writtingfile = tteval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置一个配置的属性。请在本节结尾的表中，选择一个与语言无关的属性标签。

参数

[in] BSTR Attribute : 属性名 （见下表）

[in] BSTR Value : 属性值. （根据特征类型）

例子

tteval.AttValue(„FILE“)= FALSE

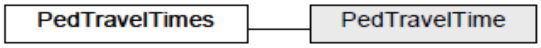
属性览表:

R	W	属性	描述
√	√	COMPILED	写入标志，针对编译输出（True/false）
√	√	FILE	写入评价文件标志（True/false）
√	√	RAW	写入标志，针对原始数据输出（True/false）

设置 FILE 的属性为 FALSE 可以避免在在线收集评价数据时（如果行程时间评价已经激活）改写*.RSZ 和 *.RSR 评价文件。该属性指定给 COM 接口并是 VISSIM 程序的全局属性（例如，它将从打开的*.INP 文件中独立的保持自身的状态）。

ITravelTimeEvaluation 接口的使用方法

3.9.17 PedTravelTimes



PedTraveltimes 对象是 PedTravelTime（见第 220 页）对象的集合。它属于 Net 的对象，可以通过 INet 接口的 PedTravelTimes 属性来访问。

它包含所有当前定义的在已经加载网络上的行人出行时间（仿真运行期间，包括为节点评价创建的临时元素），可以通过集合进行迭代或单独访问 PedTravelTime 对象。

例子

PedTravelTimes 对象和访问所有 PedTravelTime 对象的实例。

```
Dim vissim As Vissim
Dim traveltimes As PedTravelTimes
Dim traveltime As PedTravelTime
Set vissim = NEW Vissim
vissim.LoadNet “c:\vissim\daten\example.inp”
Set traveltimes = Vissim.Net.PedTravelTimes
FOR EACH traveltime IN traveltimes ‘access to _NewEnum
…
NEXT traveltime
‘or also:
FOR i = 1 TO traveltimes.Count
SET traveltime = traveltimes (i) ‘or traveltimes.Item(i)
```

…
NEXT i

IPedTravelTimes 接口的属性

_NewEnum ([out, retval] LPUNKNOWN *ppEnum)

此属性创建一个集合（或枚举）所有 PedTravelTime 对象。一旦创建了一个集合，每个元素能通过使用 PedTravelTimeByNumber 方法进行返回；对于整个集合，则可用 FOR ...TO ... NEXT 语句迭代。在 Visual Basic 中，可以不用 _NewEnum 的属性而使用 FOR EACH…NEXT 语句（见上面 PedTravelTimes 的例子）。

参数

[out, retval] LPUNKNOWN *ppEnum : 返回枚举值

Item ([in] VARIANT Index, [out, retval] IPedTravelTime **ppTT)

按位置选择的方式返回集合中单一元素 PedTravelTime。这只有当所有的行人出行时间测量都能获得时才是有用的（见上文 PedTravelTimes 的例子）。通过识别码进行数据收集，可以使用 PedTravelTimeByNumber（见下）的方法。由于项目是一个集合的默认属性，所以下面两个命令都是等价的。

SET tt = PedTravelTimes(1)

SET tt = PedTravelTimes.Item(1)

参数

[in] long Index : 指数介于 1 和 PedTravelTimes.Count 之间

[out, retval] IPedTravelTime **ppTT : 返回 PedTravelTime 对象

Count ([out, retval] long *pCount)

Count ([out, retval] long *pCount)

返回集合中 PedTravelTime 对象的数量。请见上面 PedTravelTimes 例子。

参数

[out, retval] long *pCount : 返回对象的个数

IPedTravelTimes 接口的使用方法

PedTravelTimeByNumber ([in] long Number, [out, retval] IPedTravelTime **ppTT)

根据编码返回 PedTravelTime 对象。

参数

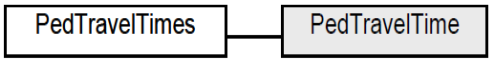
[in] long Number : 编码

[out, retval] IPedTravelTime **ppTT : 返回 PedTravelTime 对象

例子

```
DIM tt AS PedTravelTime
SET tt = PedTravelTimes.PedTravelTimeByNumber (123)
```

3.9.18 PedTravelTime



一个 PedTravelTime 对象代表一个行人出行时间测量元素，并属于 PedTravelTimes 对象集合。它可以通过 PedTravelTimes 对象用两个方式进行访问：

► 通过集合的迭代

```
DIM traveltime As PedTravelTime
FOR EACH traveltime IN vissim.Net.PedTravelTimes
List.AddItem traveltime.ID
NEXT traveltime
```

► 通过标识码

```
DIM traveltime As PedTravelTime
SET traveltime = vissim.Net. PedTravelTimes.PedTravelTimeByNumber (123)
```

该 PedTravelTime 对象通过 IPedTravelTime 接口来访问行人出行时间测量的属性。

IPedTravelTime 接口的属性

ID ([out, retval] long *pID)

返回行人出行时间测量的标识码。如果 PedTravelTime 对象不指向一个有效的 VISSIM 的行人出行时间测量元素，那么返回值为 0。

参数

[out, retval] long *pID : 返回标识码

例子

```
id = traveltime.ID
```

Name ([out, retval] BSTR *pName)

返回行人出行时间测量的名称。

参数

[out, retval] BSTR *pName : 返回名称

例子

```
name = traveltime.Name
```

Name ([in] BSTR Name)

设置的行人出行时间测量的名称。最大为 255 个字符。

参数

[in] BSTR Name : 新名称

例子

traveltime.Name= „xxx“

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回行人出行时间测量的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[out, retval] VARIANT *pValue : 返回属性值

例子

name = traveltime.AttValue(„NAME“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置行人出行时间度量值的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[in] BSTR Value : 属性值 (根据属性决定类型)

例子

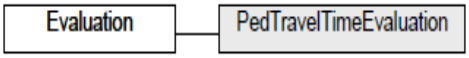
traveltime.AttValue(„NAME“)=„xxx“

属性览表

R	W	属性	描述
√		ID	标识码
√	√	NAME	名称

IPedTravelTime 接口的使用方法

3.9.19 PedTravelTimeEvaluation



该对象允许通过 IPedTravelTimeEvaluation 接口的属性和评价方法进行出行时间的配置。只要设置了适当的标记，也能进行脱机评价。（请见 186 页的 IEvaluation 接口的 TRAVELTIME 属性）

IPedTravelTimeEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回配置的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[out, retval] VARIANT *pValue : 返回配置值

例子

writtingfile = tteval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置配置的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[in] BSTR Value : 属性值 (根据属性决定类型)

例子

tteval.AttValue(„FILE“)= FALSE

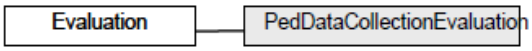
表格览表

R	W	属性	描述
√	√	COMPILED	写入标志，针对编译输出（True/false）
√	√	FILE	写入评价文件标志（True/false）
√	√	RAW	写入标志，针对原始数据输出（True/false）

设置 FILE 的属性为 FALSE 可以避免在在线收集评价数据时（如果行程时间评价已经激活）改写*.RSZ 和 *.RSR 评价文件。该属性指定给 COM 接口并是 VISSIM 程序的全局属性（例如，它将从打开的*.INP 文件中独立的保持自身的状态）。

IPedTravelTimeEvaluation 接口的使用方法

3.9.20 PedDataCollectionEvaluation



该对象允许通过 IpedDataCollectionEvaluation 接口的属性和方法来进行采集数据评估的配置。只要设置了适当的标记，也能进行离线评价。（请见 186 页的 IEvaluation 接口的 DATACOLLECTION 属性）

IpedDataCollectionEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回配置的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[out, retval] VARIANT *pValue : 返回属性值

例子

writtingfile = tteval.AttValue(„FILE“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置配置的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[in] BSTR Value : 属性值 (根据属性决定类型)

例子

tteval.AttValue(„FILE“)= FALSE

表格览表

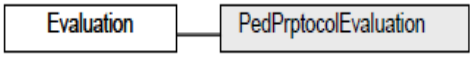
R	W	属性	描述
√	√	COMPILED	为编译输出写入标志（True/false）
√	√	FILE	写入评价文件标志（True/false）

√	√	RAW	为原始数输出写入评价文件的标志（True/false）

设置 FILE 的属性为 FALSE 可以避免在在线收集评价数据时（如果数据采集评价已经激活）改写*.MEZP 和 *.MERP 评价文件。该属性指定给 COM 接口并是 VISSIM 程序的全局属性（例如，它将从打开的*.INP 文件中独立的保持自身的状态）。

IpedDataCollectionEvaluation 接口的评价方法

3.9.21 PedProtocolEvaluation



该对象允许通过 IPedProtocolEvaluation 接口的属性和调用方法来进行数据收集评估的配置。只要设置了适当的标记，也能进行离线评价。（请见 186 页的 IEvaluation 接口的 PROTOCOL 属性）

IpedProtocolEvaluation 接口的属性

AttValue ([in] BSTR Attribute, [out, retval] VARIANT *pValue)

返回配置的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[out, retval] VARIANT *pValue : 返回属性值

例子

tFrom = ppeval.AttValue(„TIMEFROM“)

AttValue ([in] BSTR Attribute, [in] VARIANT Value)

设置配置的属性。请从本章节最后的表格中获得与语言无关的属性标记。

参数

[in] BSTR Attribute : 属性名称(见下表)

[in] BSTR Value : 属性值 (根据属性决定类型)

例子

ppeval.AttValue(„TIMEFROM“)= 42

表格览表

R	W	属性	描述
√	√	TIMEFROM	写入标志，定义一个评估的开始时间

√	√	TIMEUNTIL	写入标志，定义一个评估的结束时间
√	√	RESOLUTION	写入标志，设置评估的仿真精度

设置 FILE 的属性为 FALSE 可以避免在在线收集评价数据时（如果 protocol 评估已经激活）改写*.PP 评价文件。该属性指定给 COM 接口并是 VISSIM 程序的全局属性（例如，它将从打开的*.INP 文件中独立的保持自身的状态）。

IpedProtocolEvaluation 接口的评价方法

3.10 Triggered Scripting

在某些用户定义的脚本附加情况下，VISSIM 提供内部“挂钩”。访问是通过 TriggeredScripting 接口进行的。

3.10.1 TriggeredScripting

TriggeredScripting 对象属于 VISSIM，并可以通过 IVissim 接口的 TriggeredScripting 的属性进行访问。它可以访问配置和触发脚本的详细信息的接口。

例子（TriggeredScripting 对象的实例）：

```
DIM vissim AS Vissim
DIM trigscripting AS TriggeredScripting
SET vissim = NEW Vissim
SET trigscripting = vissimm.TriggeredScripting
```

ItriggeredScripting 接口的属性

ManagedLanesTollCalculation([out,retval] IManagedLanesTollCalculation ppManagedLanesTollCalc);**

创建一个 ManagedLanesTollCalculation 对象的实例（参见 226 页），它。

参数

[out, retval] IManagedLanesTollCalculation ** ppManagedLanesTollCalc: 返回对象

例子

```
DIM managlaneTollCalc AS ManagedLanesTollCalculation
SET managlaneTollCalc = trigscripting.ManagedLanesTollCalculation
```

3.10.2 ManagedLanesTollCalculation

ManagedLanesTollCalculation 属于 TriggeredScripting 对象，可以通过 ITriggeredScripting 接口属性对 MnagedLanesTollCalculation 访问。针对收费车道的费用计算，它可以访问 triggered scripting 的单独属性。

例子（ManagedLanesTollCalculation 对象的实例）

```
DIM managlaneTollCalc AS ManagedLanesTollCalculation
SET managlaneTollCalc = trigscript.ManagedLanesTollCalculation
```

ImanagedLanesTollCalculation 接口的属性

CurrentFacilityNo([out, retval] long* pNo)

返回与触发脚本相关的收费车道设施的编号，例如：该脚本控制哪个收费车道设施进行收费计算。

参数

[out, retval] long* pNo : 返回设施的编号

CurrentUserClass([out, retval] long* pUserClass)

返回脚本要针对哪个 User class 需要计算收费。

User class 1 = SOV (占有率为 1 人)

User class 2 = HOV (占有率为 2 人)

User class 3 = HOV3+ (占有率为 3 人或 3 人以上)

参数

[out, retval] long* pUserClass : 返回 user class

CurrentToll([in] double toll)

设置为当前 user class 服务的现有收费车道设置的收费的值。

参数

[in] double toll : 费用值

4 COM Access

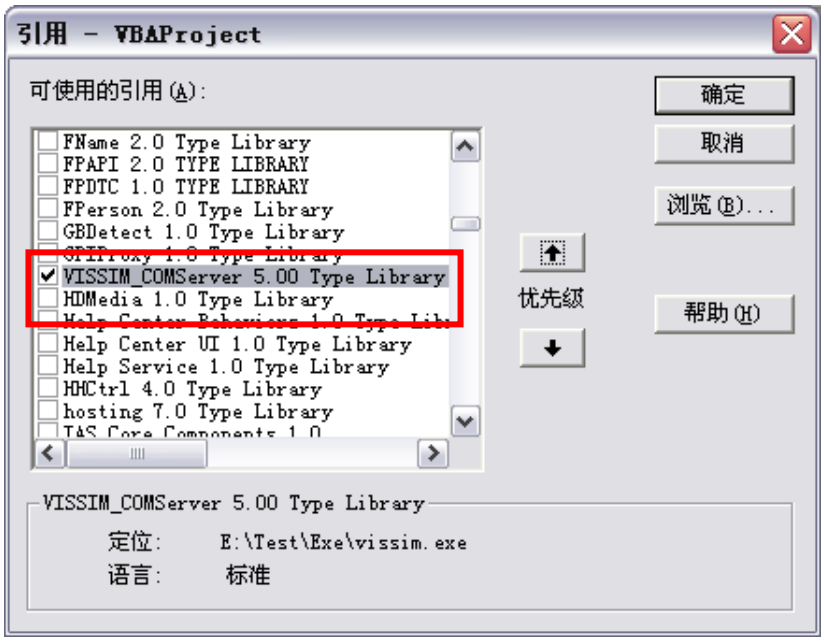
所有的 VISSIM 的 COM 接口都支持自动化，它们是基于 Microsoft standard interface IDispatch 基础上的。自动化的接口公开了 COM 对象，允许使用一般的编程环境，比如 Visual Basic，或者脚本语言如 Visual Basic Script(VBScript)，或 Java Script（Jscript）。

4.1 Visual Basic

本章将介绍使用 Visual Basic 来创建客户端应用程序。

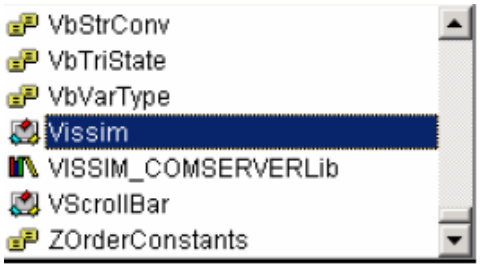
4.1.1 创建一个Visual Basic Client

为了使得 Visual Basic 编码可以调用 VISSIM 对象，要设置 VISSIM COM Server 类型库的引用。



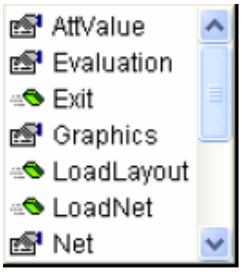
在激活了 COM 类型信息的引用后，Visual Basic 可以列出所有可能的对象的说明以及通过编码调用的方法：

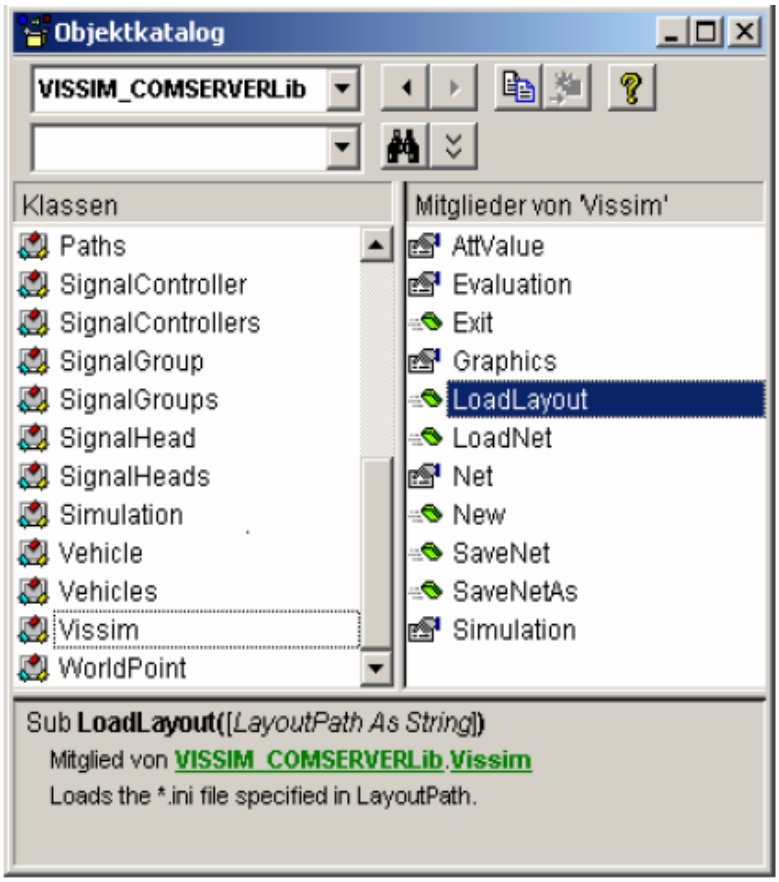
```
Dim vissim As Vissim
Dim net As Net
```



确认的类别名称，也会提供一个自动功能的列表。每一次用户输入对象的变量名称，会得到一个下拉菜单表，里面列出了该对象包含的属性和方法。

此外，对象的目录可以使得用户浏览所有可用的对象，接口的属性以及 VISSIM COM Server 库的方法。它也可以对每一个方法以及一个 VISSIM COM server 接口作简短的描述：





4.1.2 集合（枚举的不同的方法）

Visual Basic 有一个特殊的语言构造器来处理集合的问题：

```
FOR EACH element IN collection
‘ do something with the element
```

```
NEXT element
```

为了使用这个类型的语法来对集合里每一个元素进行循环，必须提供一个特别的 COM 标准接口（IEnumVARIANT）。这中集合类的 VISSIM 对象是以复数名词来命名的（比如 Nodes，Paths，Vehicles，…），可以执行该功能语句。这些对象能够被作为枚举的对象，允许有序地处理它们内部含有的对象：比如，下列的 Visual Basic 编码将在集合 Links 中的所有 link 对象进行逐一循环计算：

```
DIM links AS Links
DIM link AS Link
SET links=vissim.Net.Links
FOR EACH link IN links
WITH link
nr    =  .ID
name       =.Name
length     =.AttValue(“LENGTH”)
END WITH
NEXT link
```

一个 VISSIM 的对象集合所含元素的个数可以通过命令 Count 来获得：

```
Dim number As Long
number = links.Count
```

4.1.3 数组

另一个对象序列的语言元素是数组。在 VISSIM COM server 的环境中，一个数组是固定为 n-维的数据的序列。这些序列可以是 VISSIM 的对象，或者是 Visual Basic 的类型，属于一个 VRIANT 类型。返回的数组类型在一些 VISSIM 对象的自动化方法中会使用到，允许用户使用 Visual Basic 的数组功能如 LBOUND 和 UBOUND 来确保在数组上下界内安全获得数据。如果想知道哪些方法会返回或者需要一个数组作为参数的话，请参考本手册的相关内容。比如：为了得到 Link 几何特性的属性，可以使用到名为”POINTS”的函数。该属性返回一个世界坐标系的数组。

```
DIM polyline()                          数组变量
Polyine = link.AttValue(“POINTS”)
FOR i=LBOUND(polyline) TO UBOUND(polyline)
    x= polyline(i).X
    y=polyline(i).Y
NEXT i
```

从一个数据集合点（比如速度或者加速度）得到的结果返回为一个二维的数组：

```
res = datacollection.GetResult(“ACCELERATION”, “FREQUENCIES”) ‘2 维数组
```

```
FOR i = LBOUND(res) TO UBOUND(res)
    from = res(i,0)
    to   = res(i,1)
    val  = res(i,2)
NEXT i
```

像 AddPathAsNodeSequence 这样的方法需要通过 IPaths 接口，因此需要一个数组作为参数。在这个例子中，建议使用 Visual Basic 的功能数组，这将根据所给的参数序列，创建一个 VARIANT 的数组。

```
DIM path AS Path
SET path=paths.AddPathAsNodeSequence(1, Array(10,20,30,40))
```

4.1.4 错误的处理

当一个接口返回错误信息时，将会在全局范围内，生成一个内在的对象"Err"，它可以用在 Visual Basic 的例外处理机制内：On Error Goto。如果用户不使用这个 On Error GoTo 语句，任何运行时间错误都是致命的；也就是说，一个错误信息显示出来，随即就停止执行任务。下面的例子说明了处理错误的一个方法：

```
DIM net AS Net
DIM links AS Links
SUB Load_Click()
    SET net = vissim.Net
    SET links = net.Nodes
    ON ERROR GOTO exception   ‘创建了一个 Err 对象，在例外中继续

    NodesListbox.Clear
    FOR EACH link IN links
    NodesListbox.AddItem link.ID
    NEXT link

    EXIT SUB ‘子程序到此结束

    exception:
    MsgBox Err.Description ‘在这儿使用了生成的 Err 对象
END SUB
```

错误处理流程 – 注有注释的一部分编码，这儿“exception” – 在其他错误发生之前或者在调用一个可能出现错误的程序之前，应该进行测试或者保存 Err 对象相关的特征属性值。Err 对象的特征属性值仅仅反映了最近发生的错

误。该错误信息与 Err.Number 的属性相联系，包含在 Err.Description 里。请参见附录第 255 页的 VISSIM 所有错误信息汇总。

另外一些处理 Visual Basic 错误的可能方法是 On Error Resume Next 和 On Error Goto 0。如想得到详细的相关介绍请查看 VB 的手册。

4.1.5 一个 Visual Basic Client 的例子

下面就是一个 VB Client 的简单例子，作用是如何在不同的随机种子下运行多次仿真。

```
‘说明一些 VISSIM COM 的类型
 DIM vissim AS Vissim
DIM simulation AS Simulation
Sub Main()
     ON ERROR GOTO exception
     ‘启动 VISSIM，创建一个 Vissim 的对象例子
     SET vissim = NEW Vissim
     ‘ 读入一个路网文件
     Vissim.LoadNet App.Path+”\3path.inp”
     ‘ 运行一个仿真
     DIM seeds AS VRIANT
     Seeds = Array(22,34,56,11)
     SET simulation = vissim.Simulation
     FOR I = LBOUND(seeds) TO UBOUND(seeds)
     Simulation.RandonSeed = seeds(i)
     Simulation.RunContinuous
     NEXT i
     vissim.Exit
     EXIT SUB
     exception;
     MsgBox Err.Description    ‘这儿使用了生成的 Err 对象
END SUB
```

如果想收集每一次仿真的结果是可能的，比如运用 ISimulation 接口的 RunIndex 方法得到的路径评估。该方法允许用户把每次运行的评估文件通过索引进行编号。建议在*.ini 文件中进行期望选项的设置，然后通过 IVissim 接口的方法 LoadLayout 在启动迭代程序前把该配置文件读入进来。

4.1.6 运用 Visual Basic 的一些高级事项

创建 Vissim 的类别对象 class object

在程序内部，VB 有两个方法来创建对象：VB 可以使用它自己的对象生成服务，或者它可以使用 COM 的相关服务。根据类别（class）所在的位置以及用于生成对象的编码，VB 选择采用的服务。共有 3 个编码的方法来创建一个 Vissim 对象。用户可以明确地使用 SET 通过 NEW 这个关键词来创建，就如之前的例子中所写的那样。用户也可以使用 NEW 这个关键词来隐式进行说明，或者用户可以使用 CreateObject 的命令：

```
‘ 1) 方法一，明确通过 VB 内部的生成命令来创建：
DIM vissim AS Vissim
SET vissim = NEW Vissim

‘ 2) 方法二，隐式通过 VB 内部的生成命令来创建
DIM vissim AS NEW Vissim
Vissim.LoadNet(“example.inp”)

‘ 3) 方法三，通过注册（无需在 VB 内做类型的指引操作）来创建
DIM v AS Object
SET v=CreateObject (“VISSIM.Vissim”)
```

当用户在说明语句或者 SET 语句中使用了 NEW 关键词，VB 使用了它自己的对象服务（以及指引类型库的信息）。当用户使用了 CreateObject 函数（此时无需建立一个 VISSIM COM server 类型库的指引）时，VB 使用了 COM 的服务。这个区别会在用户如何拓展自己的运用以及如果执行程序的时候体现出来。

此外，远程地使用 CreateObject 的可选参数，创建一个 Vissim 对象类别的例子是可能的：

```
‘ 3) 通过远程的机器的注册信息来创建
DIM v AS Object
SET v=CreateObject (“VISSIM.Vissim”,”MyServer”)  ‘名称或者 IP 地址
```

有关远程登陆章节的详细信息，请参阅第 5 章。

Early-Bound 以及 Late-Bound

当用户调用一个对象的属性或者方法时，VB 必须查找这些属性或者方法，把他们放在存储栈中，然后激活这些属性或者方法，返回。早期联系指的是 VB 在编译的时候，VB 就可以寻找一个对象的属性或者方法。这样就可以把属性或者方法的内部位置保存起来。当 VB 在运行的时候调用一个方法时，

它可以直接运行该方法。后期联系是指 VB 在运行结束前不能够查看一个对象的属性或者方法，这种寻找指的是另一种交叉调用，也就是内部两次调用。

用户如何定义对象，而不是用户如何创建对象，将决定对象的属性或者方法是早期联系还是后期联系。如果要早期联系， 用户需要做如下的声明：

```
DIM vissim AS Vissim
```

这样的话，VB 可以在编译的时候执行连接。如果要后期联系，用户可以这样定义一个对象：

```
DIM vissim AS Object
```

这样的话，在运行的时候，该对象才会确认。VB 针对该特殊对象，必须要寻找每一个属性和方法。这就意味着增加了出现运行错误的机会，因为 VB 不能事先确定该对象包含的属性表和方法表，直到运行开始时才行。

当使用早期联系来指引一个 VISSIM COM Server 类型库时，必须在 VB 的编程环境中设置。当安装 VISSIM 新的版本时，这可能导致一些冲突，可能会修改类型库。取消选择和重新选择 VISSIM COM Server 的指引，将迫使 VB 重新翻译类型库文件。请查阅第 6.3 节的附件“技巧和说明”，了解有关 VISSIM 版本和它的 COM server 接口的详细信息。

4.2 Visual C++

根据平台，编译器以及库的不同，在 C++中进行 COM client 的编程可能会有所改变。在这儿所提到的为使用 COM 的功能和组成成分而进行的初始化和定义说明，主要集中在 Microsoft Visual C++ 编译器和它的标准 COM 库（ole32.lib）上。其他特别的选择也会在本章里提到。

4.2.1 创建一个 VC++ Client

要应用 Cilent，在它们可以调用 COM 库函数之前，必须初始化 COM 库。可以通过函数 CoInitialize() 或者 CoInitializeEx()来完成。初始化的结尾，必须使用函数 CoUninitialize()来关闭应用，它将自动释放所有生成的 COM 对象。预处理器命令_WIN32_DCOM 必须被定义，必须包含头文件<objbase.h>，以便能够使用这些函数。

```
#define _WIN32_DCOM
#include <iostream>
using namespace std;
#include <objbase.h>
int main()
{
cout << "Client: Calling CoInitialize()" << endl;
CoInitializeEx(NULL, COINIT_MULTITHREADED);
// do something
cout << "Client: Calling CoUninitialize()" << endl;
CoUninitialize ();
return 0;
}
```

通过 Visual C++编译器，用户可以使用 #import 预处理命令来使得 Visual C++编译器可以获得 VISSIM 对象：

```
// modify path to your needs
#import "c:\Programs\ptv-vision\VISSIM400\exe\vissim.exe"
using namespace VISSIM_COMSERVERLib;
```

用户可以使用该命令选项的 no_namespace，如果用户不想使用一个 namespace 给 VISSIM COM server 的话。或者用户可以输入所提供的. TLB 库 VISSIM_COMServer.tlb，可达到同样的效果。在对 COM 对象进行操作时，使用该命令允许直接调用 Microsoft specific smart pointers (关于详细的解释，请参见 compiler-generated Primary Type Library Header File, *.tlh)。Smart pointers 提供一个直线获得 VISSIM COM server 对象的方式；下面的编码行创建了一个 VISSIM 对象的例子：

```
IVissimPtr spVissim(__uuidof(Vissim));
```

该 smart-pointer IVissimPtr 是在生成的*.tlh 文件中定义的，使用了模版 _com_ptr_t。该种应用密封了 pointers 的接口，消除了通过调用 AddRef() and Release() 对索引计数进行跟踪的必要性。此外，它还隐藏了 CoCreateInstance()的调用。

另一个可能性，就是使用 CreateInstance()的方法：

```
HRESULT hr;
IVissimPtr spVissim;
hr = spVissim.CreateInstance("VISSIM.Vissim", NULL, CLSCTX_LOCAL_SERVER);
```

该直接的命令#import 也可以使用方法的接口和属性，只要对每个接口进行 wrapping inline member methods 的合适的说明就可以了（关于详细的解释，请参见 compiler-generated Secondary Type Library Header File, *.tli）。一旦生成一个 VISSIM 的例子，对象 Net 和 Simulation 就可以通过 VISSIM 的接口获得：

```
INetPtr spNet;
spNet = spVissim->GetNet();
ISimulationPtr spSim;
spSim = spVissim->GetSimulation();
double period = spSim->GetPeriod();
```

当不使用 Microsoft Visual C++，而使用其他的编译器时，必须采用一个更加标准的方法来调用 COM 的组成部分。所提供的 VISSIM_COMServer.h 文件和 VISSIM_COMServer_i.c 文件包含了针对 VISSIM server 接口的标准的 C++编码说明。包含了这些的文件和 VC++ 的#import 预处理命令的效果是相似的：

```
#include "VISSIM_COMServer.h"
#include "VISSIM_COMServer_i.c"
```

在这种情况下，有必要通过 C++ pointers 来运用 COM 技术的可用函数。在 Microsoft 中， objbase.h 文件必须包含，使用 CoCreateInstance 允许 Vissim 对象的第一个实例：

```
HRESULT hr;
IVissim* pVissim;
hr = CoCreateInstance(CLSID_Vissim, NULL,
CLSCTX_LOCAL_SERVER, IID_IVissim, (void**)&pVissim);
if (FAILED(hr)) exit;
```

如果需要获得这些命令和属性，必须在头文件: VISSIM_COMServer.h 中使用函数的定义：

```
HRESULT hr;
INet* pNet;
hr = pVissim->get_Net(&pNet);
if (FAILED(hr)) exit;
ISimulation* pSim;
hr = pVissim->get_Simulation(&pSim);
```

```
if (FAILED(hr)) exit;
double period;
hr = pSim->get_Period(&period);
if (FAILED(hr)) exit
```

4.2.2 集合（不同的方法来进行枚举）

在 Visual Basic 中，可以通过不同的方法在一个集合里进行循环：1）使用 IEnumVARIANT 接口，它支持所有 VISSIM COM server 枚举对象，2）使用 Item()方法。虽然在 VB 中可以使用隐含的_NewEnum() 方法，但是在 C++ 中，必须显式地调用。由于和模版类别的组合，允许 cast smart pointers 而不是调用 QueryInterface()，该编码占据一行：

```
IEnumVARIANTPtr spEnum(spLinks->Get_NewEnum());
```

该 smart-pointer spEnum 指的是集合中枚举的对象，提供了方法 Next()，来进行每个元素的循环：

```
long id;
unsigned long ulFetched;
_variant_t var;
spEnum->Next(1, &var, &ulFetched);
while (ulFetched == 1) {
id = ((ILinkPtr)var.pdispVal)->GetID();
spEnum->Next(1, &var, &ulFetched);
}
```

一个更有效的形式是为一个数组调用一次 Next()：

```
long id, count = spLinks->GetCount();
unsigned long ulFetched;
_variant_t* avar= new _variant_t[count];
spEnum->Next(count, avar, &ulFetched);
for (int i = 0; i < count; i++) {
id = ((ILinkPtr)avar[i].pdispVal)->GetID();
}
```

也可以像 VB，使用方法 Item()：

```
long jCount = spLinks->GetCount();
_variant_t jvar = 1L;
for (long j = 1; j <= jCount; j++, jvar = j) {
id = spLinks->GetItem(jvar)->GetID();
}
```

4.2.3 数组

数组的自动类型是 SAFEARRAY。一个 SAFEARRAY 是一个自描述的，多维的向量类型。它最初是从 Visual Basic 中引进来的，因此该类型与 Visual Basic 的数组类型相同。在 C++中，使用它需要一些帮助。我们在这儿说明在 C++ 中使用 SAFEARRAYS 的两种可能性：1）使用 API 功能，以及 2）使用 Active Template Library ATL wrappers (从 version ATL 7.0 开始)。

使用 API 功能 SafeArrayCreate()有时会更繁冗，需要使用更多行的编码。比如创建一个数组，就要包括

```
long ids[] = {1, 2, 5, 11, 3, 7};
VARIANT var;
VariantInit(&var);
SAFEARRAY *psa;
SAFEARRAYBOUND rgsabound[1];
long rgindice[1];
rgsabound[0].lLbound = 0;
rgsabound[0].cElements = sizeof ids / sizeof *ids;
psa = SafeArrayCreate(VT_VARIANT, 1, rgsabound);
for (long i = 0; i < sizeof ids / sizeof *ids; i++) {
rgindice[0] = i;
var.lVal = ids[i];
var.vt = VT_I4;
SafeArrayPutElement(psa, rgindice, &var);
}
var.parray = psa;
var.vt = VT_ARRAY | VT_VARIANT;
spPaths->AddPathAsNodeSequence(1, var);
```

在 ATL7 CComSafeArray<T>中介绍的可用的 SAFEARRAY wrapper classe 简化了构造和使用：

```
CComSafeArray<VARIANT> array(sizeof ids / sizeof *ids);
for (long i = 0; i < sizeof ids / sizeof *ids; i++) {
array.SetAt(i, CComVariant(ids[i]));
}
spPaths->AddPathAsNodeSequence(1, CComVariant(array));
```

4.2.4 错误处理

在 Secondary Type Library Header File (*.tli) 中定义的 wrapping functions 会在调用 virtual functions 之后检查是否有错误，把错误转换为一个 _com_error (一个密封 COM error 对象的一个类别)，然后扔掉它。因此，当

使用#import 预处理命令时，错误处理就限制在 catch of _com_error exceptions：

```
try {
IVissimPtr spVissim(__uuidof(Vissim));
ISimulationPtr pSim;
pSim = pVissim->GetSimulation();
}
catch (_com_error &error) {
cout << error.Description() << endl;
}
```

Microsoft 特别的错误处理打包类别_com_error 密封 HRESULT，提供更多准确错误信息的可能性（针对所有的 VISSIM VOM server 对象，如果一个 ErrorInfo 对象与错误相连接）。

当用户不使用 #import 命令，用户必须检查返回的 HRESULT，手工进入 IErrorInfo：

```
HRESULT hr;
INet* pNet;
hr = pVissim->get_Net(&pNet);
If (FAILED(hr)) {
BSTR description = 0;
IErrorInfo* pEI;
GetErrorInfo (0, & pEI);
pEI->GetDescription (&description);
}
```

4.2.5 一个 Visual C++ Client 例子

下面就是一个简单的 VC++ client，显示了如何通过不同的随机种子进行多次仿真运行：

```
#define _WIN32_DCOM
#include <iostream>
using namespace std;
// import of all VISSIM COM server interfaces // modify path to your needs
#import "G:\VISSIM\Exes\371#\vissim.exe"
using namespace VISSIM_COMSERVERLib;
int main()
{
HRESULT hr;
cout << "Client: Calling CoInitialize()" << endl;
```

```
hr = CoInitialize(NULL);
if (FAILED(hr)) return 1;
try {
// Create the Vissim object (connection to ISSIM COM server)
cout << "Client: Creating a Vissim instance with the smartpointer IVissimPtr" << endl;
IVissimPtr spVissim(__uuidof(Vissim));
// Create a Simulation object
cout << "Client: Calling GetSimulation()" << endl;
ISimulationPtr spSim;
spSim = spVissim->GetSimulation();
// read network and layout
char buf[255] = "";
GetCurrentDirectory (sizeof(buf), buf);
_bstr_t path(buf);
spVissim->LoadNet (path + "\\fixed_time.inp", 0);
spVissim->LoadLayout (path + "\\link_eval.ini");
// initialize simulations
cout << "Client: Setting simulations" << endl;
long seeds[4] = {10, 20, 30, 42};
spSim->PutPeriod(100);
spSim->PutResolution(1);
spSim->PutSpeed(5);
// run simulations
cout << "Client: Starting simulations:" << endl;
for (int index = 0; index < 4; index++) {
cout << " Client: Initializating simulation " << index+1 << endl;
spSim->PutRunIndex(index);
spSim->PutRandomSeed(seeds[index]);
spSim->PutComment("Random seed =" + _bstr_t(ltoa(seeds[index], buf, 10)));
cout << " Client: Running simulation " << index+1 << " ..." << endl;
spSim->RunContinuous();
}
}
catch (_com_error &error) {
cout << (char*)(error.Description()) << endl;
}
cout << "Client: Calling CoUninitialize()" << endl;
CoUninitialize ();
return 0;
}
```

4.3 .NET

有两个方法可以在 COM 和.NET 技术之间进行互通。正因为如此，从一个基于.NET client 中使用 VISSIM COM Server 差不多都是自动的。之所以说差不多，是因为有必要生成一个组合体（assembly）来适应 VISSIM COM Server 类型库的元数据。

这儿有四种方法得到相关的组合体：

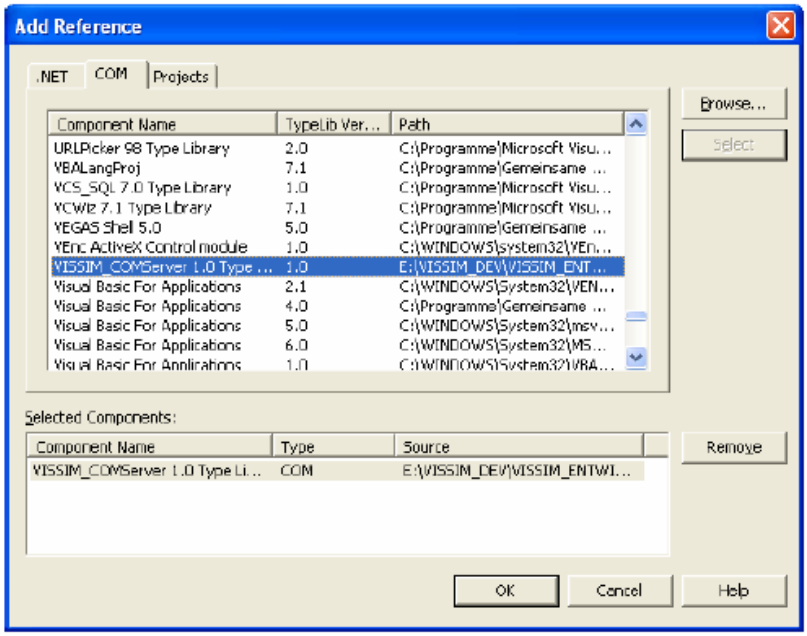
使用 Visual Studio .NET	自动把在一个类型库中的 COM 类型转换为在一个组合体的元数据。
使用 Importer tool 的类型库 (tlbimp.exe)	提供命令行，把对应的元数据转换到结果文件中，输入类型根据一个已有的类型库，生成一个组合体和一个 namespace。
使用 TypeLibraryConverter class	暴露方法，执行转换相关的操作。
编一个用户自定义的 wrapper.	作为一个差一些的选项，用户可以从 scratch 创建类型的定义。

如果用户使用 Visual Studio.NET 来创建一个.NET client，我们建议使用第一种方法。如果不是，使用.NET Framework 的工具 TLBIMP.EXE。其他的两种可能性在使用 VISSIM CM Server 时需要用到。

4.3.1 使用 Visual Studio.NET 来创建一个 Client

当使用 Visual Studio.NET 时，必须通过以下方式添加一个对 VISSIM COM Server 的索引，独立于所选择的编程语言：

1. 从 Preject 菜单中，选择 References。
2. 选择 COM 的子窗口。
3. 从索引列表中选择 VISSIMCOM_Server 库。
4. 点击 OK。



这将自动创建 wapper 组合体 Interop.VISSIM_COMSERVERLib.dll (Interop 提供了 COM 和.NET 的互通性)，允许通过默认的 namespace VISSIM_COMSERVERLib 来获得 VISSIM COM Server 的对象和接口。

接口，通常的预定义是以一个字母“I”作为前缀，如果没有预定义则获得一个额外的名称（比如，VISSIM_COMSERVERLib::ISimulation 和 VISSIM_COMSERVERLib:: Simulation，都是允许进入 ISimulation 的接口的。）

COM 对象词尾是“Class”。在这里，仅与 Vissim 对象相关，必须引用为 VISSIM_COMSERVERLib::VissimClass。

运用 Visual Basic .NET 的例子

```
Dim vis As VISSIM_COMSERVERLib.Vissim
Dim sim As VISSIM_COMSERVERLib.Simulation
vis = New VISSIM_COMSERVERLib.VissimClass
vis.LoadNet(“example.inp")
sim = vis.Simulation
sim.RunContinuous()
```

请参阅章节 MSDN 的章节“Upgrading from Visual Basic 6.0”，里面有关于 Visual Basic 和 Visual Basic.NET 的区别说明。

运用 Visual C++.NET 的例子

```
#using <mscorlib.dll>
using namespace VISSIM_COMSERVERLib;
int main()
{
Vissim *vis = new VissimClass;
vis->LoadNet("example.inp", false);
Simulation *sim = vis->Simulation;
sim->RunContinuous();
return 0;
}
```

运用 Visual C#的例子

```
using System;
using VISSIM_COMSERVERLib;
public class Client
{
public static void Main()
{
Vissim vis = new VissimClass();
vis.LoadNet("E:\\VISSIM\\COM\\VBA\\GROSSFKT.INP", 0);
Simulation sim = vis.Simulation;
sim.RunContinuous();
System.Console.WriteLine("Hello, World!");
}
}
```

运用 Visual J#的例子

```
package VissimClient;
import VISSIM_COMSERVERLib.*;
public class Client
{
public static void main()
{
Vissim vis = new VissimClass();
vis.LoadNet("E:\\VISSIM\\COM\\VBA\\GROSSFKT.INP", (ubyte)0);
Simulation sim = vis.get_Simulation();
sim.RunContinuous();
}
```

```
}
```


请参阅章节 MSDN 的章节“Upgrading from Visual J++6.0”，里面有关于 Visual J++和 Visual J#的区别说明。

4.3.2 数组

相应的.NET Framework built-in 类型对于 COM 自动类型 VARIANT 是 Sytem.Object 类型。因为 VISSIM 里的数组作为 VARIANT 传输，.NET Framework 的被管理的数组是 System.Object，我们可以直接使用被管理的数组操作符_gc[]来建立 VISSIM 的数组。比如 AddPathAsNodeSequence()的方法，需从 IPaths 接口进入，不仅需要 VARIANTs，而且要排列这些量，因此我们需要把这些值密封到 System.Object。为了这个目的，我们使用关键词 _box：

```
System::Object* array __gc[] = {__box(1), __box(2), __box(5), __box(7)};
pPaths->AddPathAsNodeSequence(1, array);
```

这儿的变量数组是一个被管理的 System.Objects 数组，它将会以一个对象本身来被传输，默认情况下编组到 VARIANT。

在 Visual Basic 中简单的语句可能是这样：

```
Dim array() As Object = {1, 2, 5, 11, 3, 7}
paths.AddPathAsNodeSequence(1, array)
```

4.3.3 Events

一些 VISSIM COM Server 对象（参见第 133 页 DynamicAssignment 的描述，作为例子）把事件发给 client。为了在一个 client 应用中消化一个事件，用户必须提供一个事件的处理器(an event-handling method)来处理对应事件的逻辑程序和事件资源一起登记的事件处理器。WithEvents 的说明以及处理子句提供了一个辨别事件处理的说明方式。从一个对象那儿发生的一个事件通过 WithEvents 关键词的说明可以应用到任何一个程序中被处理，只要有那个事件的处理说明。下面的 Visual Basic .NET 编码以 DynamicAssignment 对象以及它的事件 EdgeSmoothing 为例子，来显示的是如何做这件事情：

```
Dim WithEvents dyn As DynamicAssignemnt
Private Function dyn_EventHandler(ByVal a As Double, ByVal b As Double) As Double
Handles dyn.EdgeSmoothing
Return (1.0# - 1.0# / nIt) * a + 1.0# / nIt * b
End Function
```

从 dyn 对象那儿发展的 EdgeSmoothing 事件将可以被函数 dyn_EventHandler 处理。

4.3.4 错误处理

COM 方法通过返回 HRESULTs 报告错误；.NET 方法通过扔掉 exceptions 来报告错误。这个运行时间自动对应 HRESULT，从 COM interop 到更多的特别 exceptions。比如，E_OUTOFMEMORY 变成了 OutOfMemoryException。如果 HRESULT 是一个用户的结果或者如果它对于运行时间是未知的，运行时间把一个一般的 COMException 传递到 client。COMException 的错误信息属性包含 HRESULT 值。在 C++ 中，写法如下：

```
try {
vis->LoadNet("example.inp", false);
}
catch (System::Exception* pEx) {
System::Console::WriteLine(S"Generic Exception Handler: {0}", pEx->Message);
}
```

在 Visual Basic 的.NET 中，写法如下：

```
Try
vis.LoadNet("example.inp")
Catch ex As Exception
MsgBox(ex.Message)
End Try
```

4.4 Java

与.NET 的用法类似，Java 的使用是 homogeneous。通过生成包含类型库中所用实体的软件包，Java 客户端能够访问 VISSIM COM 服务器类型库的对象和接口。这个软件包可以使用任何 Java 代码，并且通常使用类的缩写形式进行语句输入。

4.4.1 生成一个 COM Wrapper

用于创建 Java-COM 联系桥梁的可以列出一串工具表，有商业的以及非商业的 (Jawin, Jace, JACOB, R-JAX, jacoZoom, jactivex, J-Integra, JunC++ion, JCOM, ...)。取决于具体的平台以及软件环境，可能一种工具优于另外一些工具。对于 COM Automation types 的支持必须得到保证。比如，一个从 COM VARIANT type 转换到一个 java type，如一个. java.lang.Object type 和包含一个数组的 VARIANT 转换成一个 Java 数组。如果用户寻找一个 Java-COM 桥梁，它可以在任何一个平台上为任何一个 Java VM 工作，包括 UNIX 的话，请查阅 J-Integra 产品。

如果使用的是 Microsoft Java Virtual Machine Support (MSJVM) 以及 jactivex 工具，请考虑在网页上 http://www.microsoft.com/mscorp/java/的说明。这个 MSJVM 已经在 2007 年 12 月 31 日结束了它的使用生命。

一个另外的选择就是使用连接 Java 和.NET 的软件解决方案。要考虑在 COM 以及.NET 技术之间的互通性（参见第 242 页通过.NET 使用 COM）以及 VISSIMCOMServer assembly 的使用。

4.4.2 生成一个 Java Client

一旦 COM wrapper classes 创建出来，Java 可以准确地使用 VISSIM 对象和接口，就如同它们本身就是 Java 对象和接口一样。这些 wrapper classes 与 VISSIM COM Server 组织有着同样的名字：

```
import VISSIM.VISSIM_COMSERVER.*;
class VissimSim
{
private static IVissim vis;
private static ISimulation sim;
public static void main ( String[] args )
{
try {
vis = new Vissim();
sim = vis.getSimulation();
sim.runContinuous();
```

```
}
catch (Exception e) {
e.print 存储栈 Trace();
}
}
}
```

关于使用 Automations types，请参阅使用的 Java-COM 联系相关技术文档。

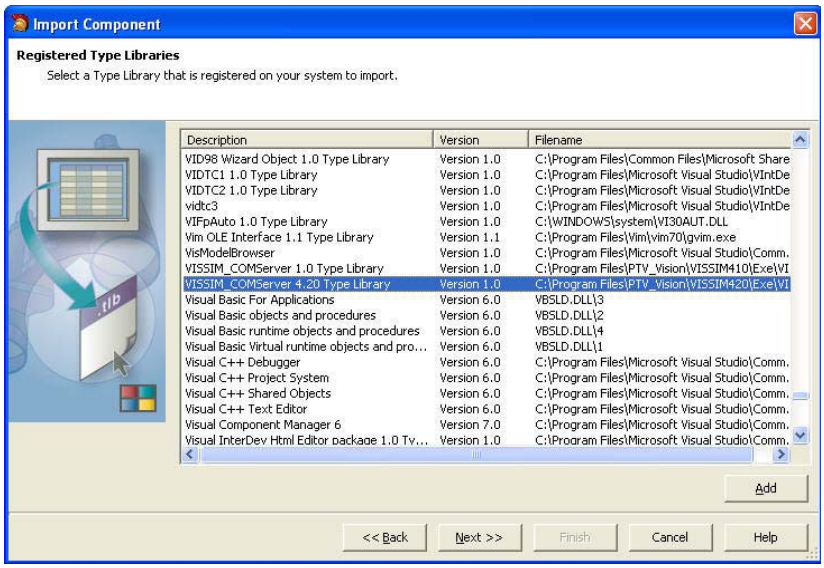
4.5 Delphi

与 Microsoft Visual Studio 类似，Borland 研发机构 Delphi 也提供一个导入类型数据库的可能。在这章里，将简要地介绍运用 Delphi2006 来调用 VISSIM COM 接口，以及一个使用 Object Pascal 的例子。

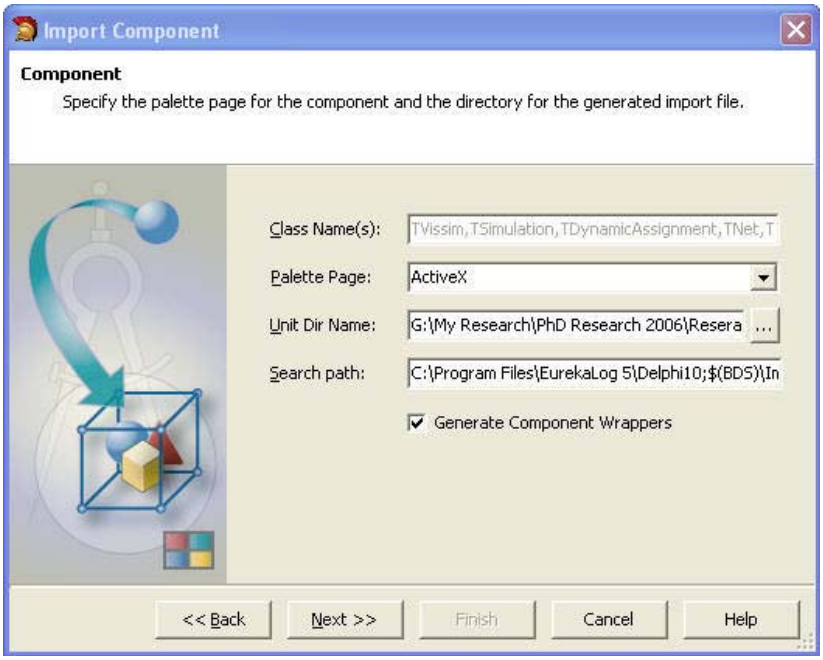
4.5.1 使用 Delphi 2006 生成一个 Client

首先有必要添加一个对于 VISSIM COM Server 的指引，可以通过以下步骤：

1. 从 Delphi IDE 的配置菜单中选择导入 Import Component 子窗口。
2. 选择“Import a Type Library”。
3. 从所注册的 Type Libraries 表中选择 VISSIMCOM_Server library，并且点击”Next>>”。



4. 切换到“Generate Component Wrappers”。这将生成必要的. wrapper classes。同时，在“Unit or Dir Name”中，创建具体的对于 unit 文件的路径。这个文件命名为 VISSIM_COMSERVERLib_TLB.pas。
5. 选择“Create Unit”，并且点击“Finish”。



例子

下面的编码显示的是在 Object Pascal 中使用 COM 接口。注意的是，单位“VISSIM_COMSERVERLib_TLB”在 *uses* 表中包含了：

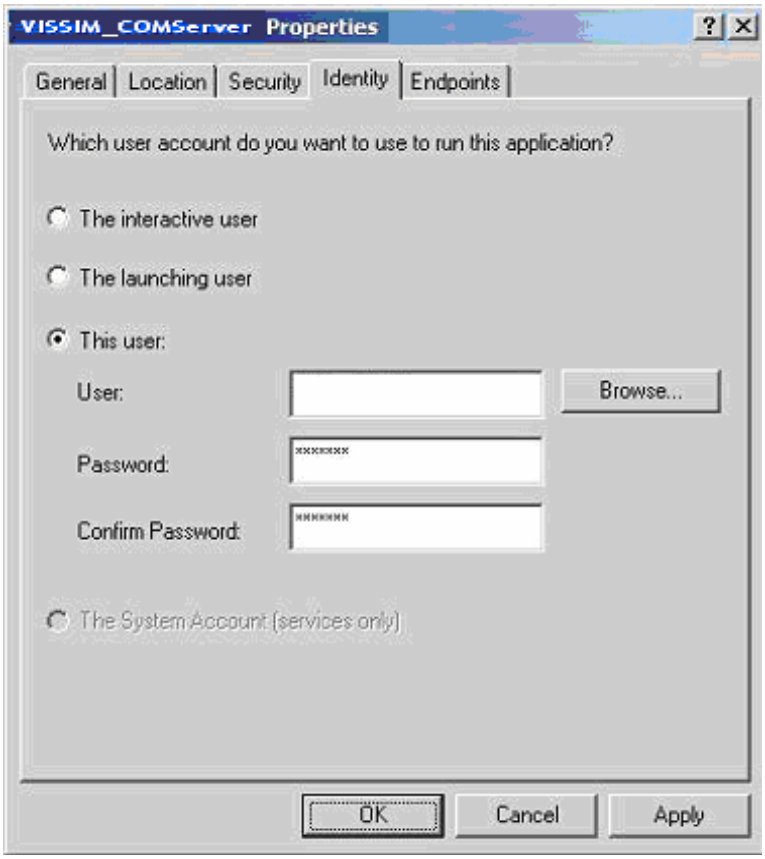
```
unit VISSIMCOM_Delphi;
interface
uses
Windows, Messages , SysUtils, Variants, Classes, VISSIMCOMSERVERLib_TLB;
implementation
procedure start;
var
intfVISSIM : TVissim;
intfSIMULATION : TSimulation
begin
intfVISSIM := TVissim.Create(Self);
intfSIMULATION := intfVISSIM.Simulation;
intfSIMULATION.RunContinuous;
intfVISSIM.Exit;
intfVISSIM.Free;
end;
```

end.

5 远程调用COM

当从一个远程机器上通过一个 client 进入 VISSIM COM Server 的时候，必须考虑以下一些问题。首先，VISSIM 必须要如通常一样（第 9 页）注册到服务器计算机上。然后，必须设置合适的发射和进入许可，以便于允许客户端的确认能够运行，以及使用注册过的 VISSIM COM Server。

为了能够设置安全内容，建议运行 DCOMCNFG.EXE（请查阅使用该工具的 Windows 说明文件）。该工具主要允许在身份验证层面上设置合适而必要的许可标签，发射许可以及获得许可。它也允许用户对于 server process 选择一个身份（或者 RunAs register）：1) interactive user， 2) launching user 以及 3) a specific user：



如果用户选择 interactive user，用户指令 COM 去布置一个 server process，当一个发射出现时，在 server console – the"interactive user"里，用户身份进行登录。交换式的对话选择只有一个是否允许正常通过它的 GUI

启动 VISSIM 的选择。但是如果没有人登录在这个 server 上也就是说，没有交互的使用者的话，它不能被发射。

要进行发射的用户简单地告诉 COM，来分配 server process，在执行该发射过程时进行了远程 client 的身份确认过程。这使得就算没有交互的用户，也能完成发射的过程，这意味着 VISSIM 将在 underground 模式下在没有 GUI 的情况下进行工作。

这个“This user”选项为每一个服务器建立了一个特别的用户帐户，来运行 under。如果一个 COM server 被配置的用来作为一个 VISSIM Object 来运行，如果 VISSIM Object 是一个服务器上有效的帐户，然后当发射时，远端的 server process 被分配了一个 VISSIM 对象的确认。它不管谁来登录服务器（或者说无论谁来登录服务器），或者谁来发射这个过程。

6 附录

6.1 错误信息提示

在运用 VISSIM COM Server 的时候，所有可能发生的错误被归并为四大类：

1. Windows 错误：系统创建出了超出最小范围的值（错误的内存处理，除以 0，等等）。当这些错误发生时，它不可能恢复 COM Server。这些错误必须被报告，需要 VISSIM 开发团队查找固定错误发生的位置。请联系 VISSIM 的 Hotline 服务。

2. COM 错误：通过一个 COM 支持的 classes 和 libraries（比如 COM 对象 VISSIM 不能用例子来说明）产生的错误。当安装了一个新的 VISSIM 版本的时候，使用一个前面的工作 client，会出现一个常见的 COM 错误。请参见第 262 页解决该问题的技巧和提示。其他这类错误可能发生的原因是注册 VISSIM 或者使用系统权限。请确保 VISSIM 正确地安装，此外要拥有管理员权限，所有的注册输入都是可以选择的（参见附录中关于注册信息的详细列表）。如果错误仍然继续或者用户不清楚系统的注册函数，请联系 VISSIM 的 Hotline 服务。

3. VISSIM 错误：通过 VISSIM 的内部程序产生的错误（比如一个仿真运行不能够记录所需的结果）。这类型的错误可能有着非常不同的原因，取决于上下文和使用到的具体界面和方法。请参阅下面的信息表或者适当的错误工作区。另外一种可能是通过一个没有预料到的但是又是有效的方式使用了一个接口，这将会在未来的新版本中加以改进。请联系 VISSIM 的 Hotline 服务，报告具体的例子。

4. User 错误：通过错误地运用 COM 接口（比如在创建一个新的路径的时候输入了一个错误的节点顺序）而产生的错误。另外，这一类型的错误也是有着非常不同的原因，取决于上下文和使用到的具体界面和方法。请请参阅下面的信息表或者适当的错误工作区。

COM 错误信息	描述
An unspecified error has occurred in COM.	如果没有更具体的信息提供，这就是提供的一般出错信息。
An error has occurred while initializing the VISSIM COM server. Error <number> returned.	COM server 不能初始化。VISSIM 将在不支持 COM 功能的情况下进行工作。这个错误不应该出现，如果出现了请联系 VISSIM Hotline。
Access denied while registering the COM server.	VISSIM 不能建立进入 COM server 接口的注册通道，因为系统缺乏进入权限。VISSIM 将在不支持 COM 功能的情况下进行工作。请确认你有进入系统注册器的权限（读与写的权利）

COM 错误信息	描述
An error has occurred while registering the VISSIM COM server. Error <number> returned.	由于不可知的原因， VISSIM 不能建立进入 COM server 接口的注册通道。VISSIM 将在不支持 COM 功能的情况下进行工作。请确认你有进入系统注册器的权限（读与写的权利）。
Access denied while unregistering the COM server.	由于系统进入的权限的原因，VISSIM 不能建立移除进入 COM server 接口的注册通道。
An error has occurred while unregistering the VISSIM COM server. Error <number> returned.	由于不可知的原因，VISSIM 不能建立移除进入 COM server 接口的注册通道。
An error has occurred while creating an object.	它不可能创建一个所需对象的例子。一个可能的原因是在安装了一个新的 VISSIM 版本后，使用了之前的用户工作账户。(参见 6.3 节的技巧与提示)
An error has occurred while binding an object.	内部错误。所需的对象不能连接 VISSIM 数据。这个错误的原因是内部的特性没有被用户解决或者根本没可能有一个工作区。请把该问题报告给 VISSIM hotline。
Not implemented method called.	界面的方法不被支持。可能该方法仍旧没有实施或者相关的对象例子不支持它。

VISSIM 错误信息	描述
An unspecified error has occurred in VISSIM.	如果没有更具体的信息提供，这就是提供的一般出错信息。
An error has occurred dispatching the VISSIM event <number>.	VISSIM不能从一个COM call中为一个所需事件服务。对于这个错误，有任何可能的具体的原因，请联系VISSIM hotline.
The object link to the	在用户这边的对象不再有效。

VISSIM 错误信息	描述
VISSIM data is no longer valid.	在 VISSIM。发生该错误最通常的例子是 Vehicle 对象，这个只有在仿真间的 time frame 期间才可用。一旦前面它们被调用过，并且成功地进入了，那么在尝试从 time frame 外部获得，会产生这个错误。请参阅本手册上关于 IVehicle 界面的更详细的介绍。另外一个可能性是当数据（路网元素，比如路段元素，数据采集…）已经被手动地从路网编辑器中移除的时候。
The file couldn't be loaded.	它不可能打开或者读取所需的文件。请确认该文件是否存在，它的格式以及读取它的权限是否具备。
The file couldn't be saved.	它不可能写入所需的文件。请确认你是否有该权限以及硬盘的空间是否已经满了。
No valid time text.	传递的字符格式不能被解释为一个有效的时间。有效的格式是：[hh:mm:ss] 或者 [h:mm:ss]， [hh:mm.ss] 或者 [h:mm.ss]，[hh:mm] 或者 [h:mm]。
The simulation couldn't be initialized	开始一个仿真不可能。引起的原因可能有很多种。请手动从路网编辑器中开始仿真，此外确保所有必需的为所需仿真准备的数据都是可以获得的（比如：对于结果的配置文件，信号灯控制文件等）。请尽量保存工作的仿真所对应的设置文件（*.ini 文件），通过 IVISSIM 的界面 LoadLayout()来读取该配置文件。

User 错误信息	描述
An unspecified error has occurred.	如果没有具体的错误信息可以提供，将显示这个一般错误信息。请联系 VISSIM hotline。
A wrong data type has been passed as a parameter.	没有有效的数据类型可以传输给一个方法。请在本手册内查阅该接口的详细说明。
No valid directory.	传过去的路径字符不匹配硬盘上的一个有效路径。
No valid file name	The passed name string doesn’t represent any existing file.

User 错误信息	描述
Method call only possible during a simulation run.	由于内部的 VISSIM 仿真初始化，一些方法只有在联系仿真运行的时候才有意义。使用它们是有可能的，使用 ISimulation 界面的 ISimulationStep()方法。当在仿真运行外部调用它们时，会出现这个错误信息。
The object with the specified identifier number doesn't exist.	当标识编码作为一个方法执行的参数时，如果这些标识编码其中之一不能代表一个有效的元素的话，该错误就会发生。比如，把一个无效的节点编号传送给方法 AddPathAsNodeSequence()，该方法属于 IPaths 接口。
identifier number overflow.	由于 Automation 的一致性，使用 COM 接口最大的确认编码是 2,147,483,647。否则，VISSIM 允许使用标识编码最大可到 4,294,967,295。当使用方法来交换大于 2,147,483,647 的标识编码时，一个溢出的错误就会发生。
Too large value.	传送了过大的数值。请参阅该手册里关于该界面方法的具体介绍。
Too low value.	传送了太小的数值。请参阅该手册里关于该界面方法的具体介绍。
An object with the same identifier number already exists.	传送的标识编码已经使用过了。请使用一个不同的编码。
The specified attribute < attribute name > is unknown.	传送的字符名称与任何有效的特征属性都不匹配。
The specified attribute <attribute name> is not allowed.	传递的特征属性对于方法无效。请阅读本手册关于该界面方法的特征属性列表说明。
The specified attribute < attribute name > is not readable	传递的特征属性对于获得数据无效。请阅读本手册关于该界面方法的特征属性列表说明。
The specified attribute < attribute name > is not writable.	传递的特征属性对于设置数据无效。请阅读本手册关于该界面方法的特征属性列表说明。

User 错误信息	描述
The specified attribute < attribute name > requires one parameter.	对于所传递的特征属性需要一个额外的参数。请阅读本手册关于该界面方法的特征属性列表说明。
The specified attribute < attribute name > requires two parameters.	对于所传递的特征属性需要两个额外的参数。请阅读本手册关于该界面方法的特征属性列表说明。
First passed parameter is not valid.	首先传送的参数不能被使用。比如，使用 ILink 接口的特征属性 LANECLOSED 没有有效的车道编号。
Second passed parameter is not valid.	第二个传送的参数不能被使用。比如，使用 ILink 接口的特征属性 LANECLOSED 没有有效的车辆类别编号。
The passed value is not valid.	传送的数据值不能被用于设置所需的数据。比如使用 IVehicle 接口的特征属性 DESSPEEDFRACTIL 没有有效期望速度 fractil。
The specified parameter is unknown.	传送的字符名字与任何有效的名字参数不符。
The specified parameter is not allowed.	传送的名字参数对于该方法而言无效。请查看该用户手册中关于该接口方法中的可能参数列表。（通常是针对评价结果的）。
The specified function is unknown.	传送的名字字符与任何有效的名字函数都不匹配。
The specified function is not allowed.	传送的名字函数对于该方法无效。请查看该用户手册中关于该接口方法中的可能参数列表。（通常是针对评价结果的）。
The specified configuration is not defined within VISSIM.	一些针对评价结果的方法，对于数据采集需要一个之前定义好的配置设置。当所需的结果之前没有设置好，就会出现这个错误。比如，使用在 ILink 接口的 ()方法来要求得到密度结果，如果该密度之前在配置文件中没有要求得到的话，将会出现该错误。

User 错误信息	描述
The specified vehicle class doesn't exist.	如果传送了一个错误的标识，需要获得车辆类别编号的方法就可能发生这个错误。请确保这个车辆类别编号存在于所用到的这个项目中。
A path requires at less 3 nodes.	当传送一个小于 3 个节点编号的数组给 IPaths 接口的方法 AddPathAsNodeSequence()时，出现的错误信息。
No valid node sequence.	它不可能通过 IPath 接口的 AddPathAsNodeSequence()方法从一个从所传送的节点顺序中生成一条路径。请确认所有传送的节点都存在，而且都与路段或者连接器相连，实际的顺序与所给的节点顺序一致。
No origin parking lot between the first two nodes.	一条路径必须开始与一个停车场。最开始两个节点间的停车场必须确保存在。
No destination parking lot between the last two nodes.	一条路径必须结束于一个停车场。最后的两个节点间的停车场必须确保存在。

6.2 Warning 信息提示

Warning 信息写入在 VISSIM 的 trace 文件（文件名字同输入文件，文件后缀名为*.err）。

Warning信息	描述
The <network element> with the specified identifier number <number> doesn't exist.	当使用对象集合的 MultiAttValues()方法，标识编码必须作为一个参数来进行传送。针对每一个不存在的元素，该警告会写入 trace 文件。
The path <p>, starting from parking lot <plo>, can't be assigned to vehicle <v> parked in parking lot <pl>.	一个路径 p 试图被分配到一个停在不同停车场 pl 的车辆上，而不是原先路径的出发地停车场 pl。

6.3 技巧与提示

使用 Excel 作为 client，获得警告“Microsoft Excel is waiting for another application to complete an OLE action”

一个针对 VISSIM COM interface 的调用，像 IVissim 中的 RunContinuous，发生的次数会比想象的还要多。Excel 显示它在等待 VISSIM 来结束程序。为了不显示该警告，用户可以把 Excel 的属性 DisplayAlerts of the Application object 为 false。

VISSIM COM server 接口和 VISSIM 的版本

因为 VISSIM COM server 的当前发展状态，接口不断地得到改进。一个直接的接口变化结果就是先前的 working clients 的类型库信息已经和现在的不一致了。因此，建议如果可能的话或者使用 late binding 来 code client，当一个新的 VISSIM 版本安装后，重新编译 client（或者重新设置对于 VISSIM type library 的引用）。在一个更加完整的 VISSIM COM server 状态下，该 client 的实现将不再需要。

6.4 注册

Windows 注册在创建和使用 COM 对象时，起着一个中心的作用。VISSIM COM Server 必须在一个 client 可以使用一些它的接口之前(参见第 1.2 节中关于如何做这个注册)必须进行合适的注册。

不同信息的类型为了使用 VISSIM COM Server 要在系统中进行注册。下列的进入详细列表如果有必要的话，可以用于检查注册的正确性（为了书写的方便，HKEY_CLASSES_ROOT 被缩写成了 HKCR）。

ProgID

[HKEY_CLASSES_ROOT\VISSIM.Vissim]

@="Vissim Class"

[HKEY_CLASSES_ROOT\VISSIM.Vissim\CLSID]

@="{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}"

[HKEY_CLASSES_ROOT\VISSIM.Vissim\CurVer]

@="VISSIM.Vissim.500"

[HKEY_CLASSES_ROOT\VISSIM.Vissim.500]

@="Vissim Class"

[HKEY_CLASSES_ROOT\VISSIM.Vissim.500\CLSID]

@="{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}"

AppID

[HKEY_CLASSES_ROOT\AppID\{79C4AA74-E6DB-4727-B5A9-2E6D16917263}]

@="VISSIM_COMServer"

[HKEY_CLASSES_ROOT\AppID\VISSIM_COMServer.EXE]

"AppID"="{79C4AA74-E6DB-4727-B5A9-2E6D16917263}"

Type library ID

[HKEY_CLASSES_ROOT\TypeLib\{0B9A4ADC-F3E4-4513-A129-9F0983C149E9}]

[HKEY_CLASSES_ROOT\TypeLib\{0B9A4ADC-F3E4-4513-A129-9F0983C149E9}\1.0]

@="VISSIM_COMServer 1.0 Type Library"

[HKEY_CLASSES_ROOT\TypeLib\{0B9A4ADC-F3E4-4513-A129-9F0983C149E9}\1.0\0]

[HKEY_CLASSES_ROOT\TypeLib\{0B9A4ADC-F3E4-4513-A129-9F0983C149E9}\1.0\0\win32]

@="C:\Programme\PTV_Vision\VISSIM\Exe\VISSIM.exe"

[HKEY_CLASSES_ROOT\TypeLib\{0B9A4ADC-F3E4-4513-A129-9F0983C149E9}\1.0\FLAGS]

@="0"

[HKEY_CLASSES_ROOT\TypeLib\{0B9A4ADC-F3E4-4513-A129-9F0983C149E9}\1.0\HELPDIR]

@="C:\Programme\PTV_Vision\VISSIM\Exe\VISSIM.exe"

VISSIM class object ID

[HKCR\CLSID\{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}]

@="Vissim Class"
"AppID"="{79C4AA74-E6DB-4727-B5A9-2E6D16917263}"
[HKCR\CLSID\{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}\LocalServer32]
@="C:\Programme\PTV_Vision\VISSIM\Exe\VISSIM.exe"
[HKCR\CLSID\{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}\ProgID]
@="VISSIM.Vissim.500"
[HKCR\CLSID\{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}\Programmable]
[HKCR\CLSID\{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}\TypeLib]
@="{0B9A4ADC-F3E4-4513-A129-9F0983C149E9}"
[HKCR\CLSID\{2FFFE6E8-47C1-4ad4-97ED-C8E284C9B806}\VersionIndependentProgID]
@="VISSIM.Vissim"

PTV AG
Geschäftsfeld Traffic
Traffic Customerservice
Stumpfstr. 1
D - 76131 Karlsruhe
Tel. +49 721 9651-300
Fax +49 721 9651-562
E-Mail： info.vision@ptv.de
www.ptvag.com
www.ptv-vision.com

辟途威交通科技（上海）有限公司
技术支持部
上海市人民路 885 号
淮海中华大厦 901-902 室
邮编：200010
电话：+86-21-63288260
传真：+86-21-63288236
电邮：hotline@ptvchina.cn
www.ptvchina.cn

traffic mobility logistics.