PTV GROUP

PTV 비심 2023

COM API 소개

날인

PTV Planung Transport Verkehr GmbH

주소:

하이드 운트 노이슈트라세 15

76131 Karlsruhe, Germany 관리위원회:

Christian U. Haas (Vors.),

Johannes Klutz 연락처: 전화: +49 (0)721 9651-0 팩스: +49

(0)721-9651-699 이메일:

info@ptvgroup.com

www.ptvgroup.com 상업 등록부 기

재:

만하임 지방 법원 HRB 743055

판매세 ID:

§ 27 a Umsatzsteuergesetz에 따른 판매세 식별 번호: DE 812 666 053

내용물

전문

이 문서에는 PTV Vissim COM 인터페이스의 소개와 개념적인 부분, 그리고 몇 가지 예제가 포함되어 있습니다. COM 인터페이스에 대한 전체 참조는 이 문서에 포함되어 있지 않지만, Vissim 온라인 도움말 시스템(Vissim GUI에서 HELP - COM HELP를 통해 접근 가능)의 일부로 HTML 형식으로 제공됩니다. 이 문서에는 모든 COM 객체와 그 메서드, 속성, 특성 및 관계가 나열되어 있습니다.

COM 예제

다양한 COM 애플리케이션 예제 외에도, COM을 통해 PTV Vissim을 제어하는 기본 구문을 보여주는 다양한 프로그래밍 언어에 대한 예제도 제공됩니다. 이러한 예제는 다음 프로그래밍 언어에 대한 …\EXAMPLES TRAINING\COM\BASIC COMMANDS\ 에 있습니다.

• VBA • 파일: COM Basic Commands.bas

Python, 파일: COM Basic Commands.py

• C++, 파 파일: COM 기본 명령.cpp

• C# • 일: COM Basic Commands.cs

Matlab 파일: COM 기본 명령.m

• Java 자세 파일: COM 기본 명령.java, COM 기본 명령 - Utils.java

한 내용은 설명 파일 'COM Basic Commands Desc_ENG.pdf' 또는 해당 파일의 주석을 참조하십시오. 모든 소스 파일은 일반 명령문을 사용하여 열 수 있습니다.
텍스트 편집기.

웹 세미나 "COM 인터페이스를 사용한 PTV Vissim 스크립팅"

PTV Vissim에서 COM 스크립팅에 대한 웨비나에 오신 것을 환영합니다. 이 웨비나에서는 PTV Vissim에서 COM 스크립트를 사용하는 방법을 설명합니다. 이 웨비나는 COM 스크립팅 초보자를 위한 것으로, COM 인터페이스나 스크립팅에 대한 사전 지식이 필요하지 않습니다. 또한 COM 스크립트의 활용 방법도 시연합니다.

모든 웨비나는 웨비나 아카이브를 통해 시청할 수 있습니다: https://company.ptvgroup.com/en/resources/resource-library

1 COM 프로그래밍 소개

1.1 COM이란 무엇인가요?

컴포넌트 객체 모델은 서로 다른 프로그램의 바이너리 컴포넌트들이 어떻게 협업하는지 설명합니다. COM은 다른 프로그램에 포함된 데이터와 함수에 대한 액세스를 제공합니다.
Vissim에 포함된 데이터와 객체는 Vissim을 사용하여 COM 인터페이스를 통해 액세스할 수 있습니다.
자동화 서버로 사용됩니다. Vissim COM 인터페이스는 Vissim 소프트웨어 설치 시 자동으로 포함되며(데모 버전 제외), 설치 과정에서 해당 옵션을 활성화하면 COM 서버로 설정됩니다.

COM은 특정 프로그래밍 언어에 의존하지 않습니다. COM 객체는 VBA, VBS, Python, C, C++, C#, Delphi, MATLAB 등 다양한 프로그래밍 및 스크립팅 언어에서 사용할 수 있습니다. 다음 예시에서는 VBA가 가장 많이 사용됩니다.
Python 프로그래밍 언어를 사용하는 예외는 명시적으로 표시됩니다.

1.2 Visual Basic 방언(VBA, VBS)

Visual Basic for Applications(VBA)는 표, 데이터 및 다이어그램 작업을 용이하게 하는 프로그래밍 언어입니다. VBA는 워크플로에서 반복되는 단계의 자동 처
리 및 기록을 지원합니다.

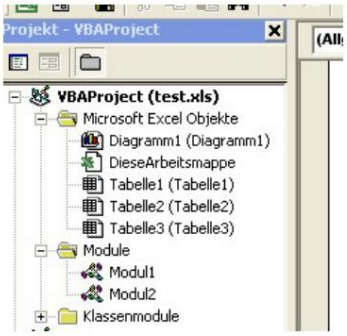
- 사용자가 Vissim, Microsoft Excel 등의 애플리케이션에 자체 기능을 추가할 수 있도록 허용
  또는 액세스.
- 사용자가 Microsoft Office 제품(예: Excel 또는 Access)에서 Vissim과 같은 애플리케이션을 시작하고 제어할 수 있습니다.

VBA는 "진짜" 프로그래밍 언어이기 때문에 VBA 프로그래밍에 능숙해지려면 일정 시간과 지속적인 훈련이 필요합니다. 강력한 프로그램은 데이터 처리 시간 절약과 안전성 확보가 필수적이기 때문에, 궁극적으로는 노력할 가치가 있습니다. VBA는 Microsoft Excel과 함께 제공되므로 많은 개인용 컴퓨터에서 사용할 수 있습니다.

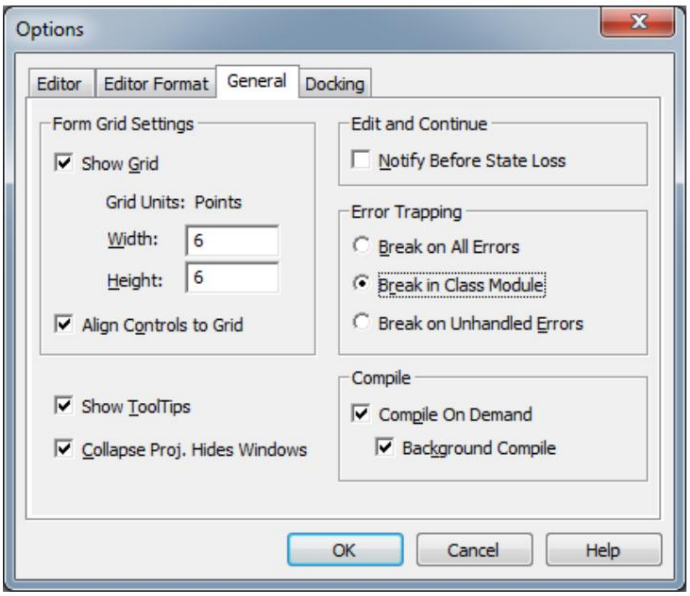
이 문서에서는 VBA 프로그램의 기본 구조와 특정 요구 사항에 대한 추가 옵션을 다양한 간단한 예를 통해 설명합니다.

Visual Basic Script(VBS)는 VBA와 밀접한 관련이 있는 스크립트 언어입니다. 이러한 스크립트는 Windows 컴퓨터에서 Windows Script Host를 사용하여 실행됩니다. VBA와 달리 VBS는 형식이 지정되지 않은 언어입니다. 즉, 모든 종류의 데이터를 저장할 수 있는 Variant라는 단일 데이터 형식만 있습니다. VBA에 비해 기능 범위가 제한적입니다. "Sub ...()"와 같이 주 프로그램을 나타내는 것은 허용되지 않으며, 필요하지도 않습니다.

1.3 VBA 프로그램을 위한 컨테이너로서의 Microsoft Excel



COM을 통해 Vissim을 활성화하는 VBA 프로그램의 컨테이너로 Microsoft Excel을 사용하는 경우, 해당 코드는 워크시트 또는 별도의 모듈에 통합될 수 있습니다. 코드가 워크시트의 일부인 경우 오류 메시지가 충분히 자세하게 표시되지 않을 수 있습니다. 이러한 동작은 VBA 편집기의 "도구 - 옵션 - 일반" 설정에 따라 결정됩니다.



클래스 모듈에서 오류 트래핑을 중단 으로 설정하는 것이 좋습니다 . 런타임 오류가 발생하면 일반적으로 특정 오류 메시지가 표시되고 VBA 디버거는 오류를 일으킨 코드 줄을 가리킵니다.

오류 트래핑이 처리되지 않은 오류 발생 시 중단 으로 설정된 경우 , 보다 일반적인 오류 메시지가 표시될 수 있습니다. 예를 들어, 존재하지 않는 네트워크 파일을 로드하려고 하면 VBA에서 "자동화 오류"라는 오류 메시지가 표시됩니다. 동일한 코드가 모듈에 있는 경우, VBA는 "CVissim::LoadNet 실패! 파일..."이라는 올바른 오류 메시지를 생성합니다.

Microsoft Excel에서 매우 긴 COM 문을 호출하면 "Microsoft Office Excel이 다른 응용 프로그램이 OLE 작업을 완료하기를 기다리고 있습니다."라는 경고가 발생할 수 있습니다. 이 경고 메시지를 방지하려면 코드에 다음 줄을 추가하세요. Application.DisplayAlerts = False

이 옵션을 사용하면 유용한 경고가 표시되지 않을 수 있습니다.

기본적으로 VBA 배열의 첨자/인덱스는 0부터 시작합니다(이를 배열의 하한값이라고 합니다). 배열 인덱스 번호를 1부터 시작하려면 "Option Base 1" 문을 사용할 수 있습니다. 그러나 일부 COM 함수(예: "GetMultiAttValues" 또는 "GetMultipleAttributes")에서 반환된 배열은 이 설정을 지원하지 않으며 항상 인덱스 0부터 시작합니다.

1.4 파이썬

Python은 Vissim 컨텍스트에서 직접 실행할 수 있는 매우 강력한 두 번째 프로그래밍 언어를 제공합니다. 사용자는 Vissim 스크립트 메뉴에서 Python 스크립트를 실행하거나 Python 인터프리터에서 Python 스크립트를 수동으로 시작할 수 있습니다. 이 스크립트는 Vissim 인스턴스를 생성하고 COM을 통해 연결할 수 있습니다.

Excel 설치 시 포함된 VBA와 달리 Python은 대부분의 경우 직접 설치해야 합니다. PTV Vision Python 설치 패키지는 설치 웹페이지(https://cgi.ptvgroup.com/php/vision-setups/)에서 제공됩니다.

패키지는 C:\Program Files\Python이 아닌 C:\Python에 설치하는 것이 좋습니다. 인터프리터와 라이브러리 외에도 여러 개발 환경이 있지만, 모든 텍스트 편집기를 사용하여 Python 스크립트를 작성할 수 있습니다. Python에 대한 자세한 내용은 http://www.python.org/를 방문하세요.

PTV Vissim 2020부터 PTV Vissim은 버전 3.7의 Python을 지원합니다.
현재 지원되는 버전은 3.9입니다. • Vissim은
Python 배포판 Anaconda를 지원하지 않습니다. • Vissim은 Wrapper wxPython을
지원하지 않습니다.

파이썬은 숫자 값과 문자열 등 다양한 기본 데이터 유형을 제공합니다. 프로그래밍 언어 자체의 특성 외에도, 파이썬은 다양한 응용 프로그램을 포괄하는 방대한 라이브러리를 제공합니다.

Python은 속성의 Set 변형이 정확히 하나의 매개변수를 받는 경우에만 속성을 지원합니다. 다른 모든 경우에는 속성이 Get 및 Set 메서드로 처리되며, Set 메서드에는 "Set" 접두사가 추가됩니다. 가장 대표적인 예는 Python에서 두 개의 별도 메서드로 제공되는 AttValue 속성입니다.

- 값 읽기: number = node.AttValue(“NO”)
- 값 설정: node.SetAttValue(“NO”, number)

기본 구문에 대해서는 7.2장과 다음에서 Python COM 예제를 참조하세요.
…\예제 교육\COM\기본 명령\

1.5 객체 인터페이스 식별자

COM 인터페이스의 이름은 Vissim 클래스 이름에 맨 앞의 "I"를 더한 것과 같습니다. 객체(예: ILink)의 맨 앞 "I"는 Interface를 의미합니다. Vissim 객체를 통해 객체에 접근할 때는 식별자에서 맨 앞의 "I"가 삭제됩니다. 형식 선언의 경우 "I"가 필요합니다.

예를 들어 Vissim 링크를 주소 지정하는 경우,

```
Vissim.Net.Links
```

사용되는 반면, 링크 객체 선언은 다음과 같이 수행됩니다.

```
Dim linkObj를 ILink로
```

일부 프로그래밍 언어(VBA 포함)는 소위 개체 카탈로그에서 올바른 식별자를 찾는 기능을 제공합니다.

2 Vissim 내에서 스크립트 실행

VBS로 작성된 스크립트를 Vissim 내에서 직접 실행하는 데는 두 가지 옵션이 있습니다.
(Visual Basic Script), Java-Script 또는 Python:
• 스크립트 파일의 직접적인 단일 호출: SCRIPTS - RUN SCRIPT FILE...
• 이벤트 기반 스크립트: 스크립트 - 이벤트 기반 스크립트 (아래의 전용 섹션 참조)

VBS 스크립트의 경우 추가 소프트웨어 설치가 필요하지 않지만 Python 스크립트의 경우 별도로 Python을 설치해야 합니다.

PTV Vision Python에 대한 별도의 설치 프로그램을 여기에서 다운로드할 수 있습니다: https://cgi.ptvgroup.com/php/vision-setups/. 이 설치를 수행하면 PTV Vissim에서 Python 스크립트를 실행하는 데 필요한 모든 단계가 설정됩니다.

PTV Vissim 2020부터 PTV Vissim은 Python 버전 2.7 및 Python 3.7을 지원합니다.
두 가지 Python 버전 간에는 사용자 환경 설정(편집 > 사용자 환경 설정 > 작업 환경 > 스크립트 파일)에서 전환할 수 있습니다.

Python을 수동으로 설치하려는 경우 Python 스크립트를 실행하는 데 필요한 단계는 Python 2.7 또는 3.7을 설치하는 것입니다 (https://www.python.org/downloads/) python.exe를 어떤 디렉터리에서든 실행하려면 python.exe 디렉터리(예: C:\Python)를 Windows 경로에 추가해야 합니다. 이 작업은 Windows의 "환경 변수" 설정에서 수행할 수 있습니다. 또한, pyWin 빌드 218 이상의 호환 버전을 설치해야 합니다 (http://sourceforge.net/projects/pywin32/files/pywin32/).

Python 설치가 사용자 계정 컨트롤(UAC) 아래의 폴더에 있는 경우(예:
C:\Program Files 또는 C:\Windows), PTV Vissim을 관리자 권한으로 실행하여 Python 폴더에 Python 캐시 파일을 생성해야 할 수 있습니다. 따라서 UAC가 아닌 폴더(예: C:\Python)에 Python을 설치하는 것이 좋습니다.

Vissim 객체에 대한 액세스

스크립트는 COM 서버로 추가 Vissim 인스턴스를 시작할 필요가 없습니다. Vissim 객체에 액세스하려면 미리 정의된 전역 변수 Vissim을 사용할 수 있습니다. 이 변수는 스크립트가 현재 실행 중인 Vissim 인스턴스를 가리킵니다. 이를 통해 현재 로드된 네트워크에 액세스하고 해당 네트워크에서 반복적인 작업을 실행할 수 있습니다. 예를 들어, 여러 시뮬레이션을 실행하거나, 각 실행 전에 매개변수를 변경하는 등의 작업이 가능합니다.

이벤트 기반 스크립트

스크립트는 시뮬레이션 실행 전, 실행 중, 실행 후 특정 시간에 자동으로 시작될 수도 있습니다. 이러한 스크립트와 그 속성은 Vissim의 "이벤트 기반 스크립트" 목록에 정의되어 있으며, 이 목록은 "스크립트 - 이벤트 기반 스크립트"를 통해 액세스할 수 있습니다. 자세한 내용은 Vissim 온라인 도움말이나 매뉴얼을 참조하십시오.

이벤트 기반 스크립트의 경우, 스크립트가 실행되는 동안 Vissim 외에도 두 가지 사전 정의된 변수를 더 사용할 수 있습니다.

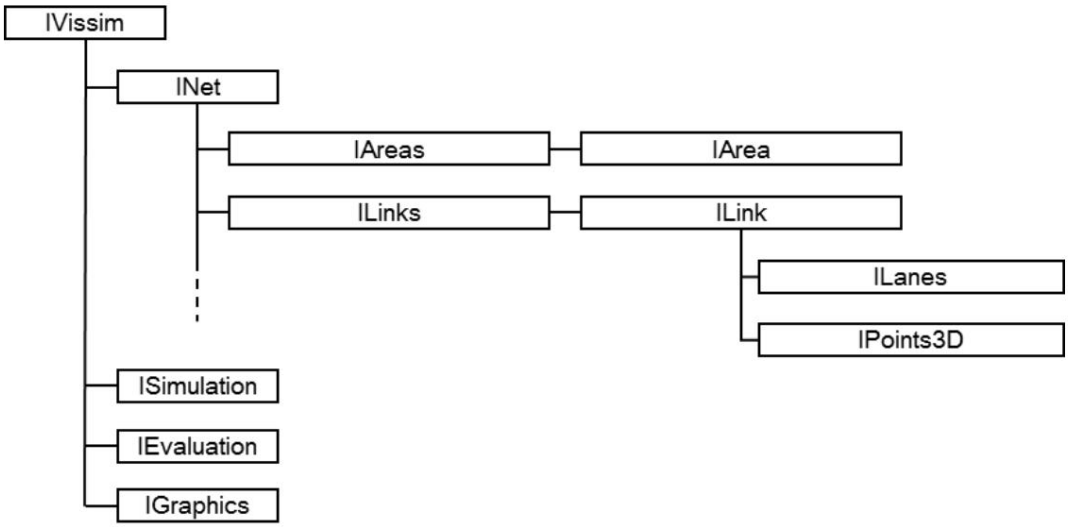
• CurrentScript는 현재 실행 중인 스크립트의 Vissim 객체를 반환합니다. 스크립트의 속성(사용자 정의 속성 포함)에 직접 액세스할 수 있습니다.
름을 반환합니다. 예(VBS): CurrentScript.AttValue("Period"). • CurrentScriptFile은 스크립트의 파일 이

예

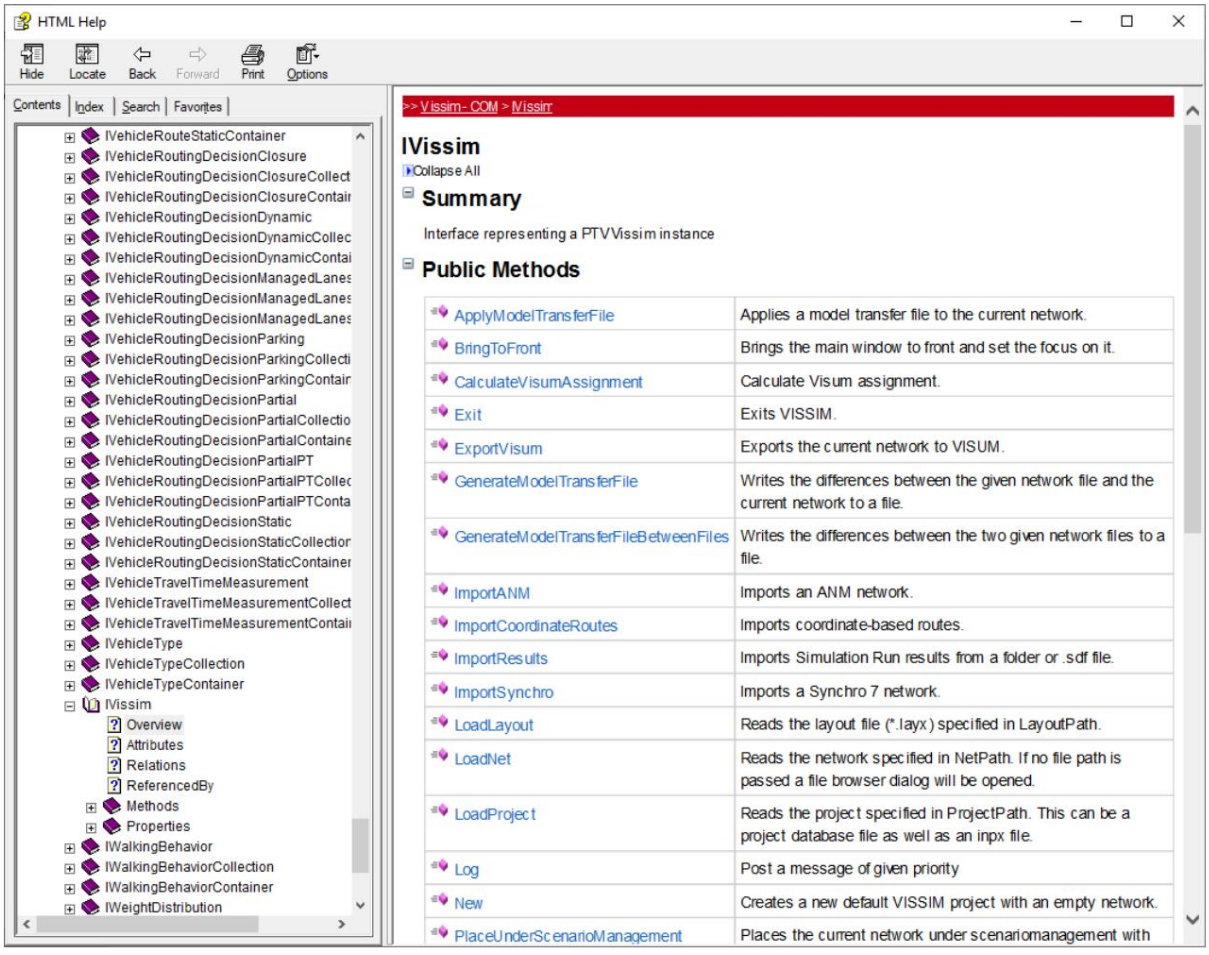
이벤트 기반 스크립트가 포함된 예제는 VBS 및 Python 언어로 Vissim 설치와 함께 제공됩니다 . \EXAMPLES TRAINING\COM\DROP-OFF ZONE

3 Vissim 객체 모델

Vissim COM 모델은 엄격한 객체 계층 구조를 따릅니다(아래 그림 참조). IVissim은 최상위 객체입니다. 네트워크 링크와 같은 하위 객체에 접근하려면 계층 구조를 따라야 합니다.



다양한 객체와 해당 메서드 및 속성에 대한 자세한 내용은 Vissim의 COM 인터페이스 참조(COM 도움말)를 참조하세요.

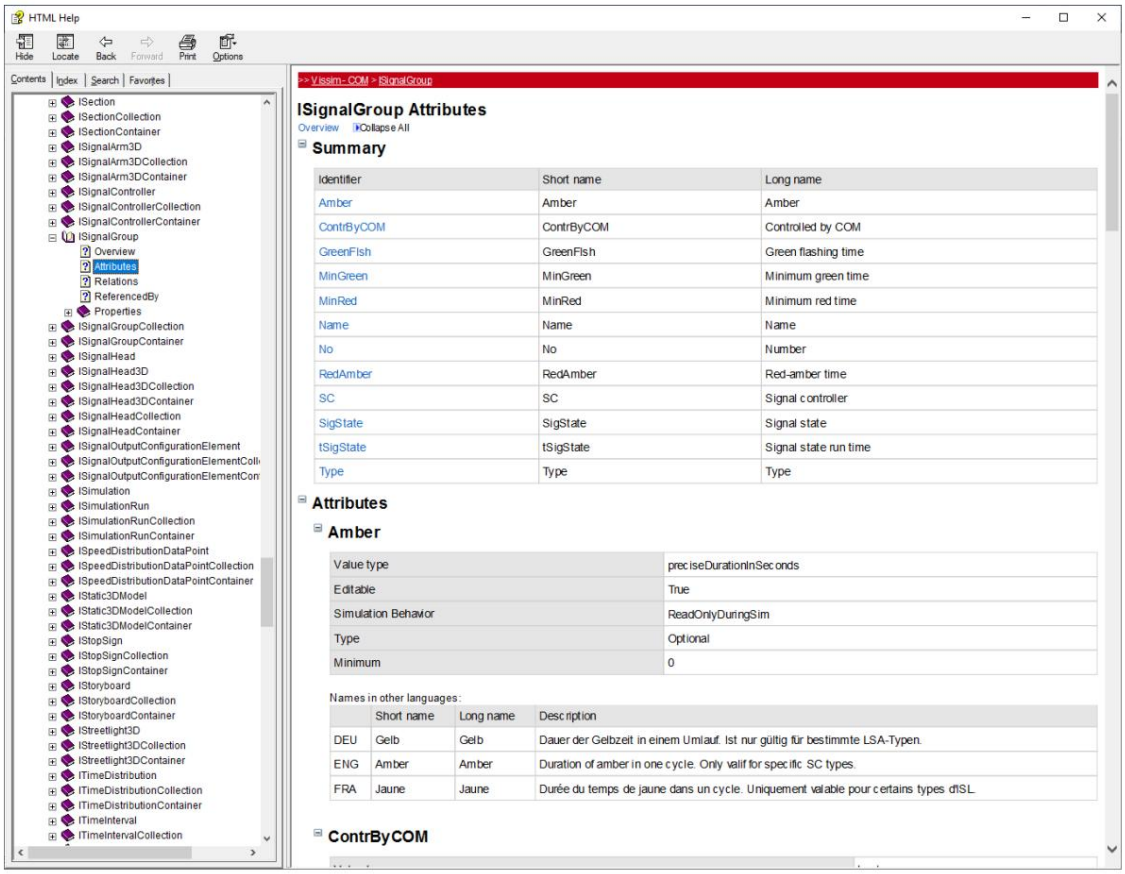


개체의 모든 속성은 COM 도움말 참조의 해당 속성 페이지에 나열되어 있습니다. 각 속성에 대해 다음 정보가 표시됩니다.

- 식별자(COM 코드에서 사용해야 함)
- 짧은 이름과 긴 이름(독일어, 영어 및 프랑스어)
- 값 유형
- COM을 통해 속성을 편집할 수 있는 경우
- 시뮬레이션 중 속성의 동작:
  - EditableDuringSim (항상 편집 가능)
  - ReadOnlyDuringSim(시뮬레이션 중에는 편집 불가)
  - SimAttr(속성은 시뮬레이션 중에만 존재함) – Editable = True인 경우 편집 가능
- 가치 소스("유형"):
  - 필수(입력 속성),
  - 선택 사항(입력 속성, 비어 있을 수 있음),
  - 계산됨(다른 속성에서 파생됨),
  - 평가(시뮬레이션 실행 중 활성화된 평가에 의해 결정됨)

값 유형 "color"는 알파(불투명도), 빨간색, 초록색, 파란색을 나타내는 네 자리 16진수 [00...FF]로 구성된 문자열을 사용합니다. 예를 들어, 초록색에 불투명도가 100%인 경우 문자열은 다음과 같습니다. "FF00FF00": 알파 FF, 빨간색 00, 초록색 FF, 파란색 00".

- 최대 3개의 하위 속성
- 최소값, 최대값 및 기본값(비어 있지 않은 경우)
- 속성에 액세스하는 데 필요한 추가 모듈(비어 있지 않은 경우)



4 첫걸음

4.1 Vissim 버전 및 파일에 대한 액세스

Vissim 인스턴스

Vissim을 수동으로 작업하는 경우 Vissim의 여러 인스턴스를 시작할 수 있습니다.
여러 다른 버전(예: Vissim 2022 및 Vissim 2023)을 시작할 수도 있습니다. COM 작업 시에도 동일하게 적용됩니다. 버전 번호가 지정되지 않은 경우(예:
CreateObject("Vissim.Vissim")를 호출하면 가장 최근에 COM 서버로 등록된 Vissim 버전이 시작됩니다. 호출 중에 버전 번호를 지정하면 특정 버전의 Vissim을 시작할 수 있습니다. Vissim 2022의 경우 CreateObject("Vissim.Vissim.22"), Vissim 2023의 경우 CreateObject("Vissim.Vissim.23")를 사용할 수 있습니다. COM을 통해 시작된 Vissim 인스턴스는 모든 COM 객체가 '없음'으로 설정되는 즉시 자동으로 종료됩니다.

COM 프로그램이 이미 시작된 Vissim 인스턴스에 연결될 것으로 예상되는 경우 이 Vissim 인스턴스는 명령줄 매개변수 –Automation을 사용하여 시작되어야 합니다.
자동화 서버로 사용됩니다. 이 Vissim 인스턴스는 Vissim 개체에 액세스하는 모든 COM 프로그램에서 사용되며, 인스턴스를 수동으로 닫을 때까지 사용됩니다.

기본 예제

이 예제는 Vissim 객체를 생성합니다. 화면에 Vissim GUI 창이 나타납니다. 네트워크 파일 D:\Data\example.inpx가 로드됩니다. 시뮬레이션 실행이 시작됩니다. 그 후
실행이 완료되면 Vissim 객체가 삭제됩니다.

예제 파일: RunSimulation.xls

```
'이 예제는 네트워크를 로드하고 시뮬레이션을 실행하는 방법을 보여줍니다.
Sub FirstVissimExample()

    '변수 선언
    Dim Vissim을 객체로 사용                    ' 가변 비심

    'Vissim-Object를 생성합니다
      '(변수 Vissim을 소프트웨어 Vissim 2023에 연결)
    Vissim = CreateObject("Vissim.Vissim.23") 설정

    '네트워크를 로드하다
    Vissim.LoadNet ("D:\Date\example.inpx")

    '시뮬레이션을 실행하다
    Vissim.Simulation.RunContinuous

    'Vissim 개체를 삭제하여 Vissim 소프트웨어를 닫습니다.
    Vissim = 아무것도 설정하지 않음
끝 서브
```

4.2 객체 및 속성에 대한 액세스

4.2.1 소개

단일 Vissim 객체에 액세스하려면 객체 계층 구조(그림 참조: 객체 모델)가 있어야 합니다.
IVissim은 모델에서 가장 높은 순위의 객체입니다. IVissim은 하위 객체인 INet(네트워크)과 ISimulation(시뮬레이션) 및 그보다 낮은 순위의 객체에 접근할 수 있도록 합니다.

INet 객체에서 ILinks(링크), INodes(노드), IAreas(보행자 구역) 등의 하위 객체에 접근할 수 있습니다. 이러한 객체는 소위 컬렉션 객체로, 동일한 객체 유형의 여러 객체를 나타냅니다. 일반적으로 Vissim COM 모델 내의 컬렉션 객체는 복수형으로 명명됩니다. IAreas 객체는 모든 보행자 구역을 포함하고, INodes 객체는 모든 노드를 포함하며, ILinks 객체는 네트워크의 모든 링크를 포함합니다. (HTML COM 참조의 "내용" 탭에 있는 "객체" 목록에 나열된 객체 유형은 일반적으로 접미사 "컨테이너"가 붙습니다(예: "ILinksContainer Collection").)

여러 네트워크 객체에는 컬렉션 과 컨테이너가 모두 존재합니다. 컨테이너는 실제 객체를 포함하는 반면, 컬렉션은 객체에 대한 참조만 포함하고 객체 자체는 포함하지 않습니다. 이러한 구분은 객체가 서로 연결되어 있기 때문에 필요합니다.

> 예: 정적 차량 경로의 링크 시퀀스는 링크 집합을 사용합니다. 경로 경로를 변경하면 링크 시퀀스와 링크 집합이 변경됩니다(일부 링크는 추가되고 일부는 삭제될 수 있음). 이 집합의 링크 변경은 네트워크의 링크에는 영향을 미치지 않으며, 동일한 링크가 네트워크에 계속 존재합니다. 경로의 링크 시퀀스에서 변경된 것은 링크에 대한 참조뿐입니다.

컬렉션을 통해 특정 유형의 모든 객체에 다양한 방식으로 접근할 수 있습니다. 루프, 반복자, 여러 속성을 동시에 조작하는 메서드 등이 있습니다. 무작위
특정 객체에 대한 접근은 "ItemByKey"를 사용하는 컨테이너를 통해서만 가능합니다(아래 섹션 참조).

다음 시뮬레이션 시간 단계 이후에는 컬렉션과 컨테이너가 유효하지 않게 될 수 있습니다(예: 차량이 네트워크를 이탈하는 경우). 따라서 반복자가 더 이상 작동하지 않을 수 있습니다. 따라서 컨테이너나 컬렉션은 사용 직전에 Vissim에서 가져와야 하며, 그 사이에는 시뮬레이션 시간 단계가 없어야 합니다.

객체(예: IVissim)의 맨 앞 "I"는 Interface를 의미합니다. COM 인터페이스의 이름은 Vissim 클래스 이름에 맨 앞 "I"를 더한 것과 같습니다.

COM 메서드 Vissim.SuspendUpdateGUI() (속성 수정 전) 및 Vissim.ResumeUpdateGUI() (마지막 수정 후) 를 사용하면 전체 Vissim 작업 공간(네트워크 편집기, 목록, 차트 및 신호 시간표 창)의 업데이트를 각각 중지하고 중지할 수 있습니다. GUI 업데이트를 일시 중지하면 COM 스크립트 실행 중(창이 열려 있는 경우) 시간을 크게 절약할 수 있습니다. 그렇지 않으면 정적 네트워크 객체의 속성을 수정할 때마다 네트워크가 완전히 다시 그려지고 해당 창들이 업데이트되기 때문입니다(빠른 모드가 활성화되어 있더라도).

4.2.2 필터

컨테이너와 컬렉션에 필터를 사용하여 주어진 조건을 충족하는 객체만 반환할 수 있습니다. 두 가지 옵션이 있으며, 두 옵션 모두 주어진 수식이 true를 반환하는 모든 객체의 컬렉션을 반환합니다. 하지만 반환되는 컬렉션의 변동성(volume)은 서로 다릅니다.

- GetFilteredSet: 반환된 컬렉션은 고정된 객체 집합을 포함합니다. 즉, 객체는 나중에 필터 조건과 더 이상 일치하지 않더라도 필터 컬렉션에 남아 있습니다. • FilteredBy: 반환된 컬렉션은 변경될 수 있습니다. 처음에 필터 조건이 true를 반환했던 객체가 변경되어 필터

조건이 더 이상 true가 아닌 경우, 해당 객체는 필터 컬렉션에서 제외되어 예기치 않은 동작이 발생할 수 있습니다.

필터 조건을 정의하는 수식은 '수식' 유형의 사용자 정의 속성(UDA)에 정의된 수식과 유사한 문자열이어야 합니다.

예:

```
필터링된 차량 설정 = Vissim.Net.Vehicles.GetFilteredSet("[속도]<10")
```

4.2.3 키를 사용하여 특정 객체에 대한 임의 액세스

ItemByKey를 통한 액세스

ItemByKey 메서드는 단일 Vissim 객체에 대한 명확한 액세스를 보장합니다. 특정 Vissim 객체에 액세스하려면 해당 객체의 외부 키(컬렉션에서 고유한 숫자)를 사용합니다. 그러면 해당 Vissim 객체의 속성(특성)에 액세스할 수 있습니다. 네트워크 객체에 대한 액세스는 네트워크 객체 유형에 따라 달라집니다.

링크 #10:

```
Obj = Vissim.Net.Links.ItemByKey(10) 설정
길이 = Obj.AttValue("Length2D")
```

신호 컨트롤러 #42의 신호 그룹 #1:

```
Obj = Vissim.Net.SignalControllers.ItemByKey(42).SGs.ItemByKey(1) 설정
상태 = Obj.AttValue("SigState")
```

시간 간격의 경우, ItemByKey 메서드는 열거형과 함께 사용됩니다. 특정 유형의 시간 간격 컨테이너에 액세스하려면 해당 유형의 내부 번호가 필요합니다.

필수입니다. 모든 시간 간격 유형의 열거형 목록은 COM 온라인 도움말의 "TimeIntervalSetNo 열거형" 페이지에서 확인할 수 있습니다.

차량 입력의 두 번째 시간 간격(열거형 1):

```
TimeIntVehInput = Vissim.Net.TimeIntervalSets.ItemByKey(1).TimeInts로 설정합니다.
TimeIntNo2 = TimeIntVehInput.ItemByKey(2) 설정
TimeIntNo2.AttValue("시작") = 500
```

차량 경로 정적의 세 번째 시간 간격(열거 2):

```
TimeIntVehStaRoute = Vissim.Net.TimeIntervalSets.ItemByKey(2).TimeInts를 설정합니다.
TimeIntNo3 = TimeIntVehStaRoute.ItemByKey(3) 설정
TimeIntNo3.AttValue("시작") = 600
```

IDE에서 Early Binding을 사용하면(6장 참조) 열거형 객체를 직접 사용할 수 있습니다.

4.2.4 참조를 사용한 객체의 단순화된 처리

참조는 연관된 객체를 식별하기 위해 소위 키를 사용합니다. 키는 특정 유형의 네트워크 객체를 고유하게 식별하는 속성 또는 속성 조합입니다. 모든 유형의 네트워크 객체에서 키를 구성하는 속성은 다를 수 있습니다. 거의 모든 상황에서 네트워크 객체 유형의 키는 "아니요" 속성입니다. 다른 네트워크 객체에 의존하는 네트워크 객체의 경우, 차선의 경우처럼 키 조합이 필요할 수 있습니다. 차선을 참조하려면 먼저 링크의 키를 지정한 다음 차선 번호를 지정해야 합니다. 결과 키는 번호가 1인 링크와 번호가 2인 차선의 경우 "1-2"와 같습니다. 일반적으로 Vissim에서 참조를 설정하려는 네트워크 객체 목록을 참조할 수 있습니다. 예를 들어, 정지 신호 목록을 열어 차선 속성을 확인할 수 있으며, 값은 "1-2"일 수 있습니다. COM을 사용하는 경우 공백만 생략하면 동일한 키를 사용할 수 있습니다.

두 개 이상의 네트워크 객체를 참조하는 참조는 단지 쉼표로 구분된 키 목록일 뿐입니다.

객체 대신 참조를 사용할 수 있습니다. 그러나 AttValue를 사용하여 값 유형이 "참조"인 속성을 가져오면 항상 문자열이 반환되고 객체 자체는 반환되지 않습니다.

예:

```
'
Vissim.Net.VehicleInputs.ItemByKey(1).AttValue(“Link”) = Vissim.Net.Links.ItemByKey(2) 객체를 사용합니다.
```

이는 다음과 같습니다.

```
'참조(키)를 사용하여
Vissim.Net.VehicleInputs.ItemByKey(1).AttValue("링크") = 2
```

4.2.5 컬렉션의 모든 객체에 대한 액세스

여러 객체의 속성을 가져오거나 설정하려면 GetMulti/SetMulti 기능을 사용하는 것이 좋습니다. 자세한 내용은 6.2.4장 "여러 네트워크 객체의 속성 읽기 및 저장"을 참조하십시오.

GetAll을 사용하면 컬렉션의 모든 객체를 목록으로 가져올 수 있습니다. 결과는 배열이며, 객체 자체는 배열 인덱스를 사용하여 액세스할 수 있습니다. 배열은 먼저 동적 길이로 선언해야 합니다. 클라이언트 코드는 배열의 경계가 준수되도록 해야 하며, 그렇지 않으면 오류가 발생합니다.

원칙적으로 GetAll을 사용하여 모든 객체를 배열에 넣은 다음 배열을 반복하면 루프를 생성할 수 있습니다(아래의 Item을 사용한 루프 와 For-Each를 사용한 루프 참조). 하지만 간단한 루프를 프로그래밍할 때는 더 빠르고 메모리를 절약하는 방법이 제공됩니다.

4.2.6 컬렉션의 모든 객체에 대한 루프

객체 열거를 사용한 루프

이 방법은 컬렉션의 모든 객체에 대한 루프를 프로그래밍하는 가장 편리한 방법으로, 속도가 빠르고 메모리를 절약할 수 있습니다. 이 방법은 새 스크립트에서만 사용하는 것이 좋습니다.

VBA:

```
vissim.net.nodes의 각 노드에 대해
    node.AttValue("...") = 다음 노드 ...
```

파이썬:

```
vissim.net.nodes의 노드에 대해:
    노드.SetAttValue("...", ...)
```

항목을 사용하여 루프

이 방법을 사용하면 루프 프로그래밍도 가능합니다.

VBA:

```
AllNodes = Vissim.Net.Nodes.GetAll
i = 0인 경우 UBound(AllNodes)로
    currentNode = AllNodes(i)로 설정합니다.
    currentNode.AttValue("...") = 다음 ...
```

이 메서드는 GetAll을 사용하여 컬렉션의 모든 객체를 목록으로 변환합니다. Item을 통해 단일 객체(여기서는 INode)에 액세스할 때, 해당 객체는 컬렉션 객체(여기서는 INodes)의 인덱스로 선택됩니다. Vissim 네트워크의 변경으로 인해 객체의 인덱스가 변경될 수 있으므로, 인덱스 번호만으로 객체를 명확하게 판단할 수는 없습니다. 따라서 이러한 유형의 액세스는 인덱스 변경이 발생하기 전에 완료된 루프에서만 사용해야 합니다.

일반적으로 객체 열거를 사용하는 방법이 선호되지만, 컬렉션의 두 번째 객체마다 반복하는 것과 같은 일부 특수 요구 사항은 Item을 사용해서만 실현할 수 있습니다.

For…Each를 사용한 루프

For…Each 루프는 컬렉션 객체에 대한 반복을 허용하여 컬렉션의 모든 객체에 접근할 수 있도록 합니다. For…Each 루프는 Item 메서드 대신 사용할 수도 있습니다.

VBA:

```
AllNodes = Vissim.Net.Nodes.GetAll
AllNodes의 각 노드에 대해
    node.AttValue("...") = 다음 노드 ...
```

VBA에서는 For Each 루프의 제어 변수를 Variant 또는 Object 로 선언해야 합니다 .

반복자 객체를 사용하여 액세스

각 컬렉션 객체는 원래 컬렉션의 첫 번째 객체(Reset 이후)를 가리키는 반복자 객체를 제공합니다. 반복자는 다음 객체를 가리키도록 증가할 수 있습니다(다음 객체가 더 이상 없고 반복자가 유효하지 않을 때까지). 컬렉션의 모든 객체를 배열에 저장하고 Item 또는 For...Each 메서드를 사용하기 전에 사용해야 하는 GetAll 속성과 달리, 반복자 객체의 사용은 해당 배열의 크기 제한에 의존하지 않습니다. 그러나 반복자 객체 사용은 객체 열거형 메서드보다 느립니다.

VBA:

```
nodeIter = Vissim.Net.Nodes.Iterator 설정
nodeIter.Valid 동안
    curNode = nodeIter.Item을 설정합니다.
    curNode.AttValue("...") = nodeIter.Next ...

향하게 하다
```

4.2.7 속성 값 읽기 및 쓰기

모든 속성에 대해 읽기 권한이 부여되는 반면(차량과 같은 일부 동적 객체의 경우 시뮬레이션 실행 중에만), 쓰기 권한은 일부 속성으로 제한됩니다(때로는 시뮬레이션 실행 외부 또는 시뮬레이션 실행 중에만). 어떤 객체에 어떤 속성을 사용할 수 있는지, 그리고 어떤 종류의 액세스가 가능한지는 HTML COM 참조(도움말 > COM 도움말)에 설명되어 있습니다.

속성은 항상 AttValue 메서드를 통해 액세스됩니다.
VBA:

```
숫자 = Node.AttValue("아니요")
Node.AttValue("NO") = 숫자 + 1
```

속성에 액세스하려면 COM 도움말에 표시된 (영어) 식별자만 필요합니다.
참조를 사용할 수 있으며(언어별 단축 이름 및 긴 이름은 제외), 대소문자를 구분하지 않습니다. Vissim의 GUI 언어가 영어로 설정된 경우, 목록 창의 열 머리글에는 식별자와 동일한 영어 단축 이름이 표시됩니다.

AttValue 메서드는 Double, Integer, String의 세 가지 데이터 유형만 반환합니다.
Vissim의 데이터 유형이 다르면(예: 부울), 값은 이 세 가지 유형 중 하나로 변환됩니다.

단위의 백분율[%] 속성은 COM을 통해 0~1 사이의 값으로 처리되지만(예: 0.45) Vissim에서는 실제 백분율(45%)로 표시됩니다.

하위 속성은 속성에 괄호로 묶어 추가합니다. 예를 들어, 차량 입력의 첫 번째 시간 간격의 볼륨을 나타내는 "Volume(1)"과 같습니다. 하위 속성이 여러 개인 경우, 쉼표로 구분합니다. 예를 들어, 차량 구성에서 원하는 속도 분포 번호가 4인 차량 유형 번호 1의 상대적인 흐름을 나타내는 "RelFlow(1,4)"와 같습니다. 각 속성의 하위 속성과 값은 COM 도움말 참조에도 나와 있습니다.

Vissim의 목록 창에 있는 열 머리글에서 확인할 수 있습니다(물론 속성 선택에 따라 다릅니다). 시간 간격의 경우, 목록 창의 열 머리글에는 시간 간격의 시작이 표시되는 반면, COM 인터페이스는 인덱스(1부터 시작)를 요구합니다.

계산된 수치적 속성의 값이 정의되지 않은 경우(예: 해당 시간 간격이 아직 시뮬레이션되지 않은 경우) AttValue 함수는 쓸모없는 값을 반환할 수 있습니다.

결과 속성은 여러 시뮬레이션 실행, 시간 간격 및 다양한 차량 등급에 대해 저장할 수 있습니다. COM을 통해 저장된 모든 결과 값에 접근할 수 있습니다. 이러한 값에 접근하려면 세 가지 하위 속성이 필요합니다.

하위 속성 # 가능한 입력 설명		
1 시뮬레이션 실행 1,2,3, ... 또는 현재		시뮬레이션 실행 수 속성에 따른 숫자(시뮬레이션 실행 목록 참조) 또는 현재 실행 중인 시뮬레이션 실행의 경우 현재
	평균, 표준 편차, 최소, 최대	모든 시뮬레이션 실행의 집계된 값
2 TimeInterval 1,2,3 ... 또는 마지막		특정 시간 간격의 인덱스(시간 간격의 인덱스는 항상 첫 번째 시간 간격을 나타내는 1에서 시작함) 또는 마지막으로 완료된 시간을 나타내는 마지막 인덱스 간격.
	평균, 표준 편차, 최소, 최대	한 시뮬레이션의 모든 시간 간격의 집계된 값
	총	한 시뮬레이션의 모든 시간 간격에 대한 합계
3 차량등급 10,20, ...	또는 모두	하나 이상의 차량 등급 번호(차량 등급 수 속성에 따라) 또는 전체 . 여기에 정의된 차량 등급에 대한 데이터만 표시됩니다.

일부 속성에는 열거형이 사용됩니다. 한 가지 예로 신호화 상태가 있습니다.
각 상태는 이름과 값을 갖습니다. 예를 들어 SignalizationStateGreen의 값은 3입니다.

시그널 그룹의 시그널 상태 속성을 변경하려면 값, 열거형 문자열 또는 열거형 객체를 사용할 수 있습니다. 참고: 열거형 객체를 사용하려면 초기 바인딩이 필요합니다(6장 참조). 모든 열거형 유형은 COM 도움말에 나열되어 있습니다.

예:

```
'
SignalGroup.AttValue("SigState") = 3 값을 사용합니다 .
```

이는 다음과 같습니다.

```
'열거형 문자열 SignalGroup.AttValue("SigState")
= "GREEN" 사용
```

그리고 다음과 같습니다:

```
' SignalGroup.AttValue("SigState")
= SignalizationState.SignalizationStateGreen 열거형 사용
```

참고: PTV Vissim은 해석하기 쉬운 문자열 값을 주로 반환합니다.

결과 속성에 액세스하는 방법에 대한 몇 가지 예는 기본 명령 예제에서 확인할 수 있습니다.

…\예제 교육\COM\기본 명령\

4.2.8 시뮬레이션 실행

COM 객체 Vissim.Simulation은 여러 가지 방식으로 시뮬레이션을 실행할 수 있도록 합니다. 시뮬레이션은 연속적으로 실행될 수 있는데, 이는 시뮬레이션 실행이 완료되기 전에 다른 작업을 수행할 수 없다는 것을 의미합니다. 또 다른 옵션은 시뮬레이션을 단계별로 실행하는 것입니다. 이는 각 시뮬레이션 단계 후에 다음 시간 단계가 시뮬레이션되기 전에 다른 작업을 수행할 수 있다는 것을 의미합니다. 시뮬레이션 중지는 이러한 작업 중 하나입니다. 예제는 6.3장을 참조하십시오.

빠른 시뮬레이션을 실행하려면 모든 동적 요소의 표시를 비활성화하세요.
따라서 간단히 빠른 모드를 활성화하세요.
Vissim.Graphics.CurrentNetworkWindow.AttValue("빠른 모드") = 1.

5 다른 프로그래밍 언어의 COM

5.1 C#에서의 COM

C# 프로그래밍 언어 내에서 큰 비용 없이 COM 함수를 사용할 수 있습니다. 사용하는 개발 환경에서 Vissim에 대한 참조를 설정하세요.

Vissim COM 객체를 사용하기 위해 실행 파일(현재 VISSIM230.EXE) 이 설치되었습니다 . 그러면 Vissim COM 객체 모델의 객체와 메서드를 C# 프로젝트에서 사용할 수 있습니다. Microsoft Visual C# .NET 개발 프레임워크를 사용하는 경우, VBA의 객체 카탈로그와 유사한 "객체 브라우저"를 사용하여 객체 모델을 탐색할 수 있습니다.

Vissim 객체를 사용하는 클래스에서는 Vissim에 액세스하기 위해 "VISSIMLIB 사용" 절을 설정합니다. COM 개체 모델. Vissim 엔터티는 "Vissim Vissim = new Vissim()";이라는 표현을 사용하여 생성자를 호출하여 생성됩니다.

"AttValue" 속성은 입력 또는 출력 매개변수를 사용하는 시그니처에 따라 두 번 발생합니다. 두 변형은 "get_AttValue"와 "set_AttValue"이며, 마지막 변형은 속성 식별자와 새 값을 매개변수로 사용합니다. 다른 식별자의 예로는 "get_ItemByKey"와 "get_Count"가 있습니다.

또한 C#에서는 선택적 매개변수를 명시적으로 지정해야 할 수 있는데, 이는 선택적 매개변수가 없다는 것을 의미합니다. 예를 들어 입력 파일을 로드할 때 (VBA에서는 선택 사항인) 입력 매개변수 "Additive"를 지정해야 할 수 있습니다. 이는 개발 환경에 따라 다릅니다.

C#은 암시적 형변환을 허용하지 않으므로 필요한 경우 명시적 형변환을 사용해야 합니다. Vissim COM 인터페이스의 대부분 메서드는 (C#-)형 System.Object의 객체를 반환합니다. 이러한 객체는 해당 형에 맞는 속성과 메서드가 필요한 경우 정적 형변환을 통해 "실제" 형(예: ILink)으로 변환해야 합니다.

C# 예제:

```
시스템 사용
VISSIMLIB를 사용하여

네임스페이스 ConsoleApplication1
{
    클래스 NameReader
    {
        /// <요약>
        /// 애플리케이션의 주요 진입점.
        /// <요약>

        정적 void Main(문자열[] 인수)
        {
            비심 비심 = 새로운 비심();
            Vissim.LoadNet(@"D:\Date\example.inpx");
            foreach(Vissim.Net.Links의 ILink 링크)
            {
                Int32 링크 번호 = (Int32) 링크.AttValue["아니요"];
                문자열 링크 이름 = (문자열) 링크.AttValue["이름"];
                Console.WriteLine("링크 " + linkNumber +                    " (" + 링크 이름 + ")");
            }
            Vissim.Exit();
        }
    }
}
```

이 예제는 입력 파일을 로드하고 모든 링크를 반복하며, 링크 번호와 이름을 콘솔에 기록합니다. 두 개의 링크로만 구성된 네트워크에 대한 출력 예시는 다음과 같습니다.

C# 예제 출력:

```
링크 1(북쪽 메인 유입)
링크 2(북쪽 메인 출구)
```

기본 구문은 7장과 다음에서 C# COM 예제를 참조하세요.
…\예제 교육\COM\기본 명령\

5.2 JAVA의 COM

Microsoft 전용 개발 프레임워크를 사용하지 않고 일반적인 JAVA 가상 머신에서 실행되는 순수 JAVA 애플리케이션의 경우 COM 인터페이스 개념은 사용할 수 없습니다. 하지만 COM 브리지를 사용하면 모든 단일 COM 서버에 JAVA 애플리케이션에 접근할 수 있습니다. 이를 위해서는 Vissim COM 인터페이스에서 제공하는 객체에 해당하는 래퍼 클래스를 오프라인 프로세스로 생성해야 합니다. 이러한 클래스는 JAVA 애플리케이션에 통합되어야 합니다. 이 프로세스가 완료되면 Vissim을 사용할 수 있습니다.

JAVA 애플리케이션에서 COM 객체를 사용하는 것은 다른 프로그래밍 언어에서와 마찬가지로 쉽습니다.

여러 가지 JAVA COM 브릿지가 존재합니다. 이러한 브릿지 중에는 Bridge4Java(IBM)와 같은 상용 제품과 JaCoB (https://sourceforge.net/projects/jacob-project/)와 같은 오픈 소스 소프트웨어가 있습니다.

앞서 설명한 어려움 때문에 순수 JAVA 환경에서 COM을 사용하려면 이러한 소프트웨어 개발 기술에 대한 깊은 지식이 필요합니다.

기본 구문에 대해서는 7장과 Java COM 예제를 참조하세요: …\Examples Training\COM\Basic Commands\

이 폴더에는 Eclipse에서 JaCoB COM 브리지를 설정하는 방법에 대한 매뉴얼이 제공됩니다(COM 예제 - Java.pdf).

5.3 C++에서의 COM

개념은 비슷하지만 C++에서 COM을 통해 Vissim 객체를 사용하는 것은 C#에 비해 더 복잡합니다. 먼저, "#import..." 지시어는 설치된 실행 파일을 프로젝트에 포함합니다. 이 예에서는 "VISSIMLIB" 네임스페이스가 할당됩니다. Microsoft Visual C++ 개발 환경에서 "CoInitialize(NULL)"을 호출하여 트리거되는 COM 인터페이스를 초기화한 후 Vissim 객체를 생성할 수 있습니다.

이제 Vissim COM 객체 모델의 객체를 사용할 수 있으며, 이를 사용하여 변수를 선언할 수 있습니다. 인터페이스를 통한 통신은 이 문서에 포함된 이름에 "Ptr"을 추가하여 생성된 포인터를 사용합니다.

이러한 포인터는 Vissim 객체에 명시적으로 연결되어야 합니다. 이를 통해 Vissim 데이터 모델에 접근할 수 있습니다. 사용 후에는 이러한 연결을 명시적으로 해제해야 합니다.

여기 보여드리는 예는 Microsoft Visual C++ 개발 환경을 사용하여 만들어졌습니다.

C++ 예제:

```
#include <iostream>
// COM 인터페이스를 제공하는 *.exe 파일을 가져옵니다.
// 여기에 예시와 같이 Vissim 설치 경로를 사용하세요.
#import "C:\\Program Files\\PTV Vision\\PTV Vissim 2023\\Exe\\Vissim230.exe" rename_namespace ("VISSIMLIB")


네임스페이스 std를 사용함;

템플릿<typename T> 인라인 void QueryAttach(T& cp, IUnknown *pUnknown) {

    T::인터페이스                    *pT;
    결과                             시간;


    hr = pUnknown -> QueryInterface(__uuidof(T::인터페이스), reinterpret_cast<void **>(&pT));
    만약 (hr != S_OK)
        _com_error(hr)을 throw합니다.
    또 다른
        cp.첨부(pT);
}

int main(int argc, char* argv[])
{
    // COM 초기화
    CoInitialize(NULL);

    // Vissim 객체 생성
    VISSIMLIB::IVissimPtr pVissim;
    HRESULT hr = pVissim.CreateInstance("Vissim.Vissim");
    만약 (hr != S_OK)
    {
        cout << "Vissim에 COM 연결을 설정할 수 없습니다." << endl;
    }

    노력하다 {
        pVissim->LoadNet(bstr_t("D:\Date\example.inpx "), false);
        VISSIMLIB::INetPtr pNet;
        VISSIMLIB::ILinkContainerPtr pLinkContainer;
        VISSIMLIB::IIteratorPtr pIter;
        VISSIMLIB::ILinkPtr pLink;
        int 숫자;
```

```
            bstr_t 이름;
            // Vissim 객체에 포인터를 연결합니다.
            QueryAttach(pNet, pVissim -> GetNet());
            QueryAttach(pLinkContainer, pNet->GetLinks());
            QueryAttach(pIter, pLinkContainer->GetIterator());
            while (pIter -> GetValid()) {
                QueryAttach(pLink, pIter->GetItem());
                숫자 = pLink->GetAttValue("아니요");
                이름 = pLink->GetAttValue("이름");
                cout << "링크 " << 숫자 << pIter -> Next();          " (" << 이름 << ")" << endl;

            }

            // 포인터 해제
            if (pLink.GetInterfacePtr() != NULL)
                pLink.Detach()->Release();

            if (pIter.GetInterfacePtr() != NULL)
                pIter.Detach() -> Release();

            if (pLinkContainer.GetInterfacePtr() != NULL)
                pLinkContainer.Detach() -> Release();

            if (pNet.GetInterfacePtr() != NULL)
                pNet.Detach() -> Release();

        } 잡다 (...) {
            cout << "COM 연결 중 오류가 발생했습니다." << endl;
        }

        // Vissim 객체 해제
        if (pVissim.GetInterfacePtr() != NULL)
            pVissim.Detach()->Release();

        // COM 초기화 해제
        CoUninitialize();
        0을 반환합니다.
}
```

이 예제는 입력 파일을 로드하고 모든 링크를 반복하며 링크 번호와 이름을 콘솔에 기록합니다. 이는 C# 예제와 정확히 동일합니다. 따라서 가능한 출력은 동일합니다.

C++ 예제 출력:

```
링크 1(북쪽 메인 유입)
링크 2(북쪽 메인 출구)
```

기본 구문은 7장과 C++ COM 예제를 참조하세요.
…\예제 교육\COM\기본 명령\

5.4 MATLAB에서의 COM

MATLAB는 프로그래밍 언어이자 수치 연산 환경입니다. MATLAB를 사용하면 VBA나 Python처럼 Vissim COM 객체를 쉽게 제어할 수 있습니다. MATLAB에서 Vissim을 시작하려면 "Vissim = actxserver('Vissim.Vissim')" 표현식을 사용하세요.
Vissim이 시작되면 모든 메서드, 속성 및 개체 모델이 "Vissim" 변수를 사용하여 표시됩니다.

MATLAB 예제:

```
Vissim = actxserver('Vissim.Vissim'); % Vissim 시작
Vissim.LoadNet('D:\Date\example.inpx');
LinkNo = Vissim.Net.Links.GetMultiAttValues('아니요');
링크 이름 = Vissim.Net.Links.GetMultiAttValues('이름');
i = 1인 경우: Vissim.Net.Links.Count,
disp(['Link',32,num2str(LinkNo(i)),32,'(',LinkName{i},')']) % char(32)는 공백입니다.
Vissim.release; % Vissim 종료
```

이 예제는 입력 파일을 로드하고 명령 창에 연결된 모든 링크의 번호와 이름을 표시합니다. 이는 C# 예제와 정확히 동일합니다. 따라서 가능한 출력은 동일합니다.

기본적으로 MATLAB는 2차원 배열을 COM에 전달합니다. PTV Vissim은 항상 1차원 배열을 예상합니다. (SafeArray[0,0], SafeArray[0,1], … 대신 SafeArray[0], SafeArray[1], …). MATLAB에서 다음 명령을 사용하여 동작을 변경할 수 있습니다: feature('COM_SafeArraySingleDim', 1)

기본 구문에 대해서는 7.5장과 MATLAB COM 예제를 참조하세요.
…\예제 교육\COM\기본 명령\

6 VBA의 Vissim 예제

예시에 관한 예비 참고 사항

대부분의 예제에서 객체 변수는 소위 "조기 바인딩"을 사용하여 선언됩니다.
조기 바인딩은 VBA에서 객체를 단순히 "객체"로 선언하는 것이 아니라, 정확한 객체 유형(클래스라고 함)을 명시한다는 것을 의미합니다. 장점은 다음과 같습니다.

- 사용자가 프로그램을 읽고 이해하기가 더 쉽습니다.
- 프로그램 실행은 일반적으로 더 빠릅니다.
- 프로그램 작성 시 개발 환경의 "명령문 작성 도구"를 사용할 수 있습니다. 객체의 사용 가능한 속성과 메서드가 자동으로 표시되며 프로그래머가 선택할 수 있습니다.

초기 바인딩은 VBA 환경이 Vissim 개체 라이브러리를 인식한 후에만 사용할 수 있습니다. 이는 "옵션" - "참조" 메뉴에서 Vissim 개체 라이브러리에 대한 참조를 설정하여 수행됩니다. 개체 라이브러리는 Vissim 프로그램의 일부이며 Vissim 설치 후 자동으로 사용할 수 있습니다.

조기 바인딩의 가장 큰 단점은 특정 상황에서 프로그램 실행이 특정 컴퓨터에만 바인딩된다는 것입니다. Vissim이 다른 컴퓨터의 다른 경로에 설치된 경우, 참조가 아무 곳도 가리키지 않습니다. 그러면 VBA는 실행을 거부합니다. 참조가 전혀 설정되지 않으면 Vissim 인스턴스에 접근할 수 있는 모든 컴퓨터에서 프로그램이 실행됩니다. 프로그램을 공유하여 여러 컴퓨터에서 사용하려는 경우 조기 바인딩을 피하는 것이 더 편리합니다. 이러한 이유로 저희가 제공하는 예제 파일은 조기 바인딩을 사용하지 않습니다. 따라서 프로그램 코드가 인쇄된 버전과 약간 다릅니다.

6.1 데이터 읽기 및 저장

6.1.1 Vissim 객체 생성 및 입력 파일 읽기

이 예제에서는 Vissim 객체를 생성하고 입력 파일을 로드합니다. 입력 파일의 이름은
Excel 워크시트에서 읽습니다. Vissim 창에 네트워크가 표시됩니다. 또한,
Vissim 창이 전경에 표시됩니다.

예: LoadInputFile

```
'이 예제에서는 Vissim 객체를 생성하고 워크시트 셀에서 입력 파일을 로드하는 방법을 보여줍니다.
Sub LoadInputFileExample()

    'Vissim 변수의 변수 선언
    Dim Vissim을 객체로 사용

    'Vissim 객체 생성(Vissim 변수를 Vissim 소프트웨어에 연결)
    Vissim = CreateObject("Vissim.Vissim") 설정

    '입력 파일 로드(셀 B1의 파일 이름)
    Vissim.LoadNet 셀(1, 2)

    Vissim이 입력 파일을 로드했음을 보여주는 메시지를 표시합니다.
    MsgBox "입력 파일 로딩이 완료되었습니다"

    'Vissim 창을 앞으로 가져와서 초점을 설정합니다.
    Vissim.BringToFront

    'Vissim 개체를 삭제하여 Vissim 소프트웨어를 닫습니다.
    Vissim = 아무것도 설정하지 않음
끝 서브
```

6.1.2 입력 파일 읽기 및 저장

이 예제에서는 Vissim 객체를 생성하고 입력 파일을 로드합니다. 입력 파일 이름은 Excel 워크시트에서 읽습니다. 너비 속성이 설정된 모든 차선의 너비를 수정합니다. 그런 다음 결과를 새 입력 파일에 저장합니다.

예: SaveInputFile

```
'이 예제에서는 Vissim 객체를 만들고 '워크시트 셀'에 정의된 입력 파일을 로드합니다.

'각 차선의 너비 속성이 수정되고 결과는 '새로운 입력 파일'에 저장됩니다.


Sub SaveInputFileExample()

    'Vissim 객체를 선언합니다
    Dim Vissim을 객체로 사용

    '다른 변수를 선언합니다
    변형으로서의 차수 너비

    'Vissim 객체 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Vissim = CreateObject("Vissim.Vissim") 설정

    '입력 파일 로드(셀 B1의 파일 이름)
    Vissim.LoadNet 셀(1, 2)

    '모든 링크를 반복합니다
    Vissim.Net.Links의 각 링크에 대해
        ' 연관된 모든 차선을 반복합니다.
        Link.Lanes의 각 차선에 대해
            '차선 너비를 구하다
            너비 = Lane.AttValue("너비")
            '차선의 너비가 있는지 확인
            너비가 비어 있지 않으면
                '새로운 너비를 계산하고 설정
                너비 = 너비 * 1.1
                Lane.AttValue("너비") = 너비
            종료
        다음 차선
    다음 링크

    '입력 파일 저장(셀 B2의 파일 이름)
    Vissim.SaveNetAs 셀(2, 2)

    'Vissim 소프트웨어를 닫으려면 모든 객체를 삭제하세요.
    Vissim = 아무것도 설정하지 않음

끝 서브
```

6.2 객체 및 속성에 대한 액세스

6.2.1 VBA에서 개체 및 속성에 액세스하는 다양한 방법

예제에서는 VBA를 사용하여 Vissim 객체와 속성에 액세스하는 다섯 가지 방법을 보여줍니다. 프레임워크는 Vissim을 열고 Vissim 네트워크를 로드한 다음 다섯 가지 서브루틴 중 하나를 호출합니다. 각 서브루틴은 다음 6.2.1.1 장에서 설명합니다.

6.2.1.5까지. 코드를 실행하려면 VBA 모듈에서 프레임워크와 모든 서브루틴을 하나씩 복사하세요.

뼈대:

```
'변수 선언
객체로서의 공공 Vissim                                    ' 가변 비심
변형으로서의 공공 차량 입력

공개 하위 프레임워크()
    Dim No_programm을 정수로 변환

    '워크시트의 12번부터 끝까지 모든 행을 지웁니다.
    행("13:65536").ClearContents

    'Vissim 객체 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Set Vissim = CreateObject("Vissim.Vissim") '입력 파일 로드(셀 B1의 파
    일 이름)
    Vissim.LoadNet 셀(1, 2)

    '-------------------------

    No_programm = Cells(3, 2)                    ' 프로그램 번호는 B3 셀에서 선택되었습니다.
    Select Case No_programm Case 1 '6.2.1.1
        ItemByKey
            를 사용하여 차량 입력 찾기 VehicleInputAttributGetItemByKeyExample Case 2 '6.2.1.2 객체 열
            거형을 사용한 루프 VehicleInputAttributeEnumerationExample Case 3




            '6.2.1.3 VehicleInputAttributeItemExample 항목
            을 사용한 루프 사례 4 '6.2.1.4 For...Each

        VehicleInputAttributeForEachExample을 통한 루프

        사례 5
            '6.2.1.5 반복자 객체를 사용한 루프 VehicleInputAttributeIteratorExample
        End Select


    '-------------------------
    'Vissim 소프트웨어를 닫으려면 모든 객체를 삭제하세요.
    VehicleInput = Nothing 설정
    Vissim = 아무것도 설정하지 않음

끝 서브
```

6.2.1.1 ItemByKey를 사용하여 차량 입력 찾기

아래 예는 단일 노드에 대한 액세스를 보여줍니다. 필요한 노드 번호는 Excel 워크시트의 지정된 셀에서 읽습니다.

예: VehicleInputAttributGetItemByKeyExample

```
'이 예제는 ItemByKey Public Sub VehicleInputAttributGetItemByKeyExample()을 사용하여 네트워크의 단일 차
량 입력에 액세스하는 방법을
보여줍니다. '숫자(키)로 차량 입력에 액세스합니다. 숫자는 셀 C1에서 읽습니다. Set
        VehicleInput = Vissim.net.VehicleInputs.ItemByKey(Cells(1, 3))


        '차량 입력 속성을 읽고 셀에 씁니다. Cells(13, 1) = VehicleInput.AttValue("No")

       셀(13, 2) = VehicleInput.AttValue("이름")
       Cells(13, 3) = VehicleInput.AttValue("링크")
끝 서브
```

6.2.1.2 객체 열거를 사용한 루프

이 예제는 객체 열거형을 사용하여 루프 내에서 차량 입력에 접근하는 방법을 보여줍니다. 모든 객체를 반복하며 일부 속성 값을 Excel 시트에 기록합니다.

예: VehicleInputAttributeEnumerationExample

```
'이 예제는 열거형을 사용하여 네트워크의 단일 차량 입력에 액세스하는 방법을 보여줍니다.

공개 Sub VehicleInputAttributeEnumerationExample()

       Vissim.net.VehicleInputs의 각 VehicleInput에 대해 행 = 13
              '차량 입력 속성을 읽고 셀에 씁니다. Cells(Row, 1) = VehicleInput.AttValue("No")

              셀(행, 2) = VehicleInput.AttValue("이름")
              Cells(행, 3) = VehicleInput.AttValue("링크")
              행 = 행 + 1
       다음 차량 입력
끝 서브
```

6.2.1.3 항목을 사용한 루프

이 예제는 Item을 사용하여 루프 내에서 차량 입력에 접근하는 방법을 보여줍니다. 차량 입력 참조 목록이 AllVehicleInputs에 할당되면, 해당 목록에 대한 루프가 시작되어 일부 속성 내용을 Excel 시트에 출력합니다.

예: VehicleInputAttributeItemExample

```
'이 예제는 Item을 사용하여 네트워크의 단일 차량 입력에 액세스하는 방법을 보여줍니다.
'주의: 항목은 컨테이너의 인덱스로 차량 입력을 식별합니다.
'차량 입력은 인덱스를 통해 고유하게 식별될 수 없으며, Vissim 네트워크가 변경되면 인덱스가 변경될 수 있습니다.'

'항목별 접근은 루프(즉, 사용 가능한 모든 '차량 입력' 처리)와 함께만 사용해야 합니다.

공개 하위 VehicleInputAttributeItemExample()
        행 = 13
        AllVehicleInputs = Vissim.net.VehicleInputs.GetAll For i = 0 To UBound(AllVehicleInputs)

                VehicleInput = AllVehicleInputs(i) 설정

                 '차량 입력 속성을 읽고 셀에 씁니다. Cells(Row, 1) = VehicleInput.AttValue("No")

                셀(행, 2) = VehicleInput.AttValue("이름")
                Cells(행, 3) = VehicleInput.AttValue("링크")
                행 = 행 + 1 다음 끝 하위
```

6.2.1.4 For...Each를 사용한 루프

이 예제는 For...Each 루프를 사용하여 접근하는 방법을 보여줍니다. 차량 입력 컬렉션을 AllVehicleInputs에 할당한 후, 해당 목록에 대한 루프를 시작하여 일부 속성 내용을 Excel 시트에 출력합니다.

예: VehicleInputAttributeForEachExample

```
'이 예제는 For-Each-Loop를 사용하여 단일 차량 입력에 액세스하는 방법을 보여줍니다.
공개 Sub VehicleInputAttributeForEachExample()
            행 = 13
            AllVehicleInputs = Vissim.net.VehicleInputs.GetAll For Each VehicleInput In
            AllVehicleInputs '차량 입력 속성을 읽고 셀에 쓰기 Cells(Row, 1) =
                 VehicleInput.AttValue("No")

                셀(행, 2) = VehicleInput.AttValue("이름")
                Cells(행, 3) = VehicleInput.AttValue("링크")
                행 = 행 + 1 다음 끝 하위
```

6.2.1.5 반복자 객체를 사용한 루프

이 예제는 IVehicleInputContainer 객체의 Iterator 객체를 사용하여 설정된 루프를 보여줍니다. 이터레이터는 VehicleInputsContainer 객체에서 수신됩니다. 그런 다음 루프가 시작됩니다. 매 라운드마다 이터레이터의 현재 항목을 수신하고, 마지막에 이터레이터를 증가시킵니다. 이터레이터에 대한 변수는

Variant 유형이 아닌 Object 유형으로 선언해야 합니다.

예: VehicleInputAttributeIteratorExample

```
'이 예제는 Iterator를 사용하여 단일 차량 입력에 액세스하는 방법을 보여줍니다.
공개 Sub VehicleInputAttributeIteratorExample()
    행 = 13
    VehicleInputIterator = Vissim.net.VehicleInputs.Iterator 설정
    VehicleInputIterator.Valid인 동안
        VehicleInput = VehicleInputIterator.Item으로 설정합니다.

        '차량 입력 속성을 읽고 셀에 씁니다.
        Cells(행, 1) = VehicleInput.AttValue("아니요")
        셀(행, 2) = VehicleInput.AttValue("이름")
        Cells(행, 3) = VehicleInput.AttValue("링크")

        행 = 행 + 1
        VehicleInputIterator.Next
    향하게 하다
끝 서브
```

6.2.2 개별 속성 읽기 및 저장

다음 예제는 속성을 읽고 쓰는 방법을 보여줍니다. 먼저 차량 입력의 이름을 변경하여 차량 번호를 포함합니다. "Name" 속성은 AttValue 속성을 통해 수신 및 기록됩니다. 그런 다음 첫 번째 시간 간격 동안 차량 입력의 볼륨을 10% 감소시킵니다. 시간 간격은 속성 이름 "Volume"과 함께 지정해야 합니다. 속성 이름 뒤에 쉼표로 구분된 하위 속성 목록을 괄호 안에 작성하여 이를 수행합니다. 필요한 하위 속성 목록은 Vissim 온라인 도움말의 COM 인터페이스 참조를 참조하십시오.

일부 속성의 경우 GUI에 표시된 값과 다른 값을 전달해야 할 수 있습니다. 예를 들어 시간 간격의 경우, 첫 번째 시간 간격을 나타내려면 "Volume(1)"과 같이 시간 대신 인덱스를 제공해야 합니다. 시간 간격의 시작 시간을 하위 속성으로 사용하는 것은 효과적이지 않습니다. 자세한 내용은 4.2.7장을 참조하십시오.

예: VehicleInputNameAndVolumeExample

```
하위 차량 입력 이름 및 볼륨 예()

    '변수 선언
    Dim Vissim을 객체로 사용                                    ' 가변 비심
    변형으로 희미한 차량 입력

    'Vissim 객체 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Vissim = CreateObject("Vissim.Vissim") 설정
    '입력 파일 로드(셀 B1의 파일 이름)
    Vissim.LoadNet 셀(1, 2)

    Vissim.net.VehicleInputs의 각 VehicleInput에 대해
        '차량 이름 변경 입력
        newName = "입력 번호" + Str(VehicleInput.AttValue("아니요"))
        VehicleInput.AttValue("이름") = newName

        '첫 번째 시간 간격의 차량 입력 볼륨을 10% 감소시킵니다.
        볼륨 = VehicleInput.AttValue("볼륨(1)")
        부피 = 0.9 * 부피
        VehicleInput.AttValue("볼륨(1)") = 볼륨
    다음 차량 입력

    'Vissim 소프트웨어를 닫으려면 모든 객체를 삭제하세요.
    VehicleInput = Nothing 설정
    Vissim = 아무것도 설정하지 않음

끝 서브
```

6.2.3 결합된 키를 사용하여 속성 읽기 및 저장

네트워크 객체를 참조하는 속성에는 키가 포함되어 있습니다. 키는 단일 키 또는 결합된 키일 수 있습니다. 이러한 결합된 키는 신호 그룹을 참조할 수 있는 신호 헤드에서 찾을 수 있습니다. 신호 그룹은 신호 컨트롤러의 일부이므로 신호 컨트롤러를 통해서만 액세스할 수 있습니다. 따라서 Vissim의 신호 헤드 목록에 있는 키는 "5-1"처럼 보입니다. 여기서 5는 신호 컨트롤러의 번호를 나타내고 1은 신호 그룹의 번호를 나타냅니다. COM에서 이 속성을 사용하려면 결합된 키도 지정해야 합니다. 위에 언급된 목록의 문자열과 동일한 결과를 얻으려면 공백 없이 "5-1"을 작성해야 합니다. 다음 예제는 주어진 신호 헤드의 신호 그룹을 첫 번째 신호 컨트롤러의 첫 번째 신호 그룹으로 변경합니다.

예: SignalHeadsExample

```
하위 SignalHeadsExample()

    '변수 선언
    Dim Vissim을 객체로 사용                                    ' 가변 비심
    Dim SignalHead 변형
    Dim SignalController를 변형으로 사용
    Dim SignalGroup을 변형으로 사용

    'Vissim 객체 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Set Vissim = CreateObject("Vissim.Vissim") '입력 파일 로드(셀 B1의 파
    일 이름)
    Vissim.LoadNet 셀(1, 2)

    '신호 헤드 가져오기 Set SignalHead
    = Vissim.net.SignalHeads.ItemByKey(Cells(1, 3))

    'Vissim.net.SignalControllers의 각 SignalController에 대해 사
    용할 첫 번째 신호 그룹을 검색합니다. SignalController.SGs.Count > 0이면 SignalGroup =
        SignalController.SGs.Iterator.Item을 설정합니다.

            종료
        종료
    다음

    '새로운 신호 그룹 SCNo =
    Str(SignalController.AttValue("No"))을 설정합니다.
    SGNo = Str(SignalGroup.AttValue("아니요"))
    SignalHead.AttValue("SG") = SCNo + "-" + SGNo

    '모든 객체를 삭제하여 Vissim 소프트웨어를 닫습니다.
    SignalHead = Nothing으로 설정
    SignalController = Nothing으로 설정
    SignalGroup = Nothing 설정

    Vissim = 아무것도 설정하지 않음

끝 서브
```

6.2.4 여러 네트워크 객체의 속성 읽기 및 저장

여러 네트워크 객체의 속성을 읽고 저장하는 데에는 여러 가지 방법이 있습니다. AttValue()를 사용하여 여러 객체에 접근하려면 더 복잡한 코딩이 필요하고 속도가 느립니다. 여러 속성에 접근하려면 다음 방법을 사용하는 것이 좋습니다.

GetMultiAttValues

한 네트워크 개체 유형의 모든 네트워크 개체의 속성을 가져옵니다.

예: 모든 링크의 이름 가져오기:

```
Vissim.Net.Links.GetMultiAttValues("이름")
```

SetMultiAttValues

하나의 네트워크 객체 유형의 여러 네트워크 객체의 속성을 설정합니다.
예: 링크 번호 1과 3의 이름 설정:

```
Dim newNames(1, 1)을 Variant로 지정
newNames(0, 0) = 1                          '링크의 참조(키)
newNames(0, 1) = "newLink1name" '링크 #1의 새 이름
newNames(1, 0) = 4                          '링크의 참조(키)
newNames(1, 1) = "newLink4name" '링크 #2의 새 이름
Vissim.Net.Links.SetMultiAttValues "이름", newNames
```

매개변수 1: 속성 이름, 여기서는 "이름"
매개변수 2: 새 값, 첫 번째 열은 네트워크 객체를 참조하고 두 번째 열은 새 값(여기서는 이름)을 포함하는 배열

여러 속성 가져오기

한 네트워크 객체 유형의 모든 네트워크 객체의 여러 속성을 가져옵니다.
예: 모든 링크의 이름과 길이 가져오기:

```
Vissim.Net.Links.GetMultipleAttributes(배열("이름", "길이2D"))
```

여러 속성 설정

한 네트워크 객체 유형의 모든 네트워크 객체의 여러 속성을 설정합니다.
예: 모든 링크의 "이름"과 "km당 비용" 속성을 설정합니다.

```
Dim 값(3, 1)을 변형으로 사용
값(0, 0) = "name1"
값(1, 0) = "name2"
값(2, 0) = "name3"
값(3, 0) = "name4"
값(0, 1) = 12
값(1, 1) = 7
값(2, 1) = 5
값(3, 1) = 3
Vissim.Net.Links.SetMultipleAttributes 배열("이름", "km당 비용"), 값
```

매개변수 1: 속성 이름 (여기서는 "Name", "CostPerKm")
매개변수 2: 새 값, 매개변수 1에 주어진 속성 이름과 같은 순서로 속성의 새 값을 포함하는 배열입니다.

이 메서드에서는 네트워크 객체의 번호를 입력하지 않습니다. 매개변수 2의 첫 번째 값(여기서는 "name1"과 12) 은 가장 낮은 번호의 네트워크 객체로 설정됩니다. 그 다음 값들은 번호가 증가하는 네트워크 객체로 설정됩니다.

예: VehicleInputNameAndVolumeExample(VBA)

이 프로그램은 GetMultipleAttributes를 사용하여 모든 차량 입력에 대한 "No", "Name" 및 "Volume(1)" 속성 값을 읽습니다. 이러한 속성은 Variant로 입력된 목록 형태로 이 함수에 전달됩니다. 그런 다음 새로운 이름과 볼륨이 계산됩니다. 새로운 속성 값은 SetMultipleAttributes를 통해 설정됩니다. 이 작업의 결과는 6.2.2와 동일합니다. 이러한 함수는 속성 값을 행렬에 저장합니다.

Result(i, j)를 사용하여 접근할 수 있습니다. 여기서 i는 행렬에서 차량 입력의 인덱스이고 j는 속성의 인덱스입니다. 이 예에서 Result(0, 1)은 첫 번째 차량 입력의 이름을 나타냅니다.

```
하위 차량 입력 이름 및 볼륨 예()

    '변수 선언
    Dim Vissim을 객체로 사용                    ' 가변 비심
    Dim 결과를 변형으로

    'Vissim 객체 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Set Vissim = CreateObject("Vissim.Vissim") '입력 파일 로드(셀 B1
    의 파일 이름)
    'Vissim.LoadNet 셀(1, 2)

    '관련 속성을 사용하여 변형을 만듭니다.
    Dim relevantAttributes(0~2) As Variant relevantAttributes(0)
    = "아니요" relevantAttributes(1) = "이름"
    relevantAttributes(2) = "볼륨(1)"


    '속성을 행렬로 가져옵니다. 결과 =
    Vissim.net.VehicleInputs.GetMultipleAttributes(relevantAttributes)

    '모든 차량 입력의 속성을 반복합니다. For i = 0 To UBound(Result) '차량 입력의 이
    름을 변경합니다. Result(i, 1) = "Input #" +
            Str(Result(i, 0)) '차량 입력 볼륨을 10% 감소시킵니
        다. Result(i, 2) = 0.9 * Result(i, 2)


    다음 나

    '수정된 속성을 차량 입력 Vissim.net.VehicleInputs.SetMultipleAttributes
    relevantAttributes, Result에 적용합니다.

    'Vissim 소프트웨어를 닫으려면 모든 객체를 삭제하세요.
    Vissim = 아무것도 설정하지 않음

끝 서브
```

6.3 시뮬레이션 실행

네트워크에서 작업하는 것 외에도 시뮬레이션을 구성하고 실행할 수 있습니다. Vissim에서는 시뮬레이션을 실행하는 두 가지 옵션이 있으며, 다음 섹션에서 설명합니다.

6.3.1 시뮬레이션 실행 구성

Vissim의 "시뮬레이션" 속성을 통해 시뮬레이션 실행을 구성할 수 있습니다. 다음 예에서는 초기 난수 시드를 21로 설정합니다. 시뮬레이션은 난수 시드를 3씩 증가시키면서 네 번 실행됩니다. 따라서 시뮬레이션은 난수 시드를 21, 24, 27, 30으로 설정하여 네 번 실행됩니다. 더 많은 구성 옵션은 COM 인터페이스 참조를 참조하십시오.

예: ConfigureSimulationRun

```
하위 ConfigureSimulationRun()

    '변수 선언
    Dim Vissim을 객체로 사용                    ' 가변 비심
    Dim 결과를 변형으로

    'Vissim 객체 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Set Vissim = CreateObject("Vissim.Vissim") '입력 파일 로드(셀 B1
    의 파일 이름)
    Vissim.LoadNet 셀(1, 2)
```

```
    '시뮬레이션 구성
    Vissim.Simulation.AttValue("RandSeed") = 21
    Vissim.Simulation.AttValue("RandSeedIncr") = 3
    Vissim.Simulation.AttValue("NumRuns") = 4

    '시뮬레이션을 실행하다
    Vissim.Simulation.RunContinuous

    'Vissim 소프트웨어를 닫으려면 모든 객체를 삭제하세요.
    Vissim = 아무것도 설정하지 않음

끝 서브
```

6.3.2 연속 모드에서 시뮬레이션 실행

연속 모드로 시뮬레이션을 실행한다는 것은 시뮬레이션을 시작하고 시뮬레이션 실행 중에는 COM을 통해 다른 작업을 실행할 수 없음을 의미합니다. 시뮬레이션이 완료되면 시뮬레이션 실행 함수 호출 이후의 명령이 처리됩니다. 이 예에서는 대기열 카운터의 계산을 시뮬레이션 후 셀에 읽고 씁니다. 표시된 것처럼 하위 속성 "SimulationRun"과 "TimeInterval"을 속성 이름과 함께 키로 지정해야 합니다.

아래에.

예제 파일: StartAssignment.xls

```
Sub RunSimulationContinuous()

    '변수 선언
    Dim Vissim을 객체로 사용                                    ' 가변 비심
    Dim QueueCounter를 변형으로 사용
    Dim Row As Long

    '워크시트에서 12번부터 끝까지 모든 행을 지웁니다 . Rows("13:65536").ClearContents


    'Vissim 객체 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Set Vissim = CreateObject("Vissim.Vissim") '입력 파일 로드(셀 B1의 파일 이
    름)
    Vissim.LoadNet 셀(1, 2)

    '평가 구성
    Vissim.Evaluation.AttValue("QueuesCollectData") = True

    '시뮬레이션
    Vissim.Simulation.RunContinuous 실행

    '평가를 읽어보세요
    행 = 13
    Vissim.net.QueueCounters의 각 QueueCounter에 대해
        '큐 카운터 속성을 읽고 셀에 씁니다. Cells(Row, 1) = QueueCounter.AttValue("No")

        셀(행, 2) = QueueCounter.AttValue("QLen(1,1)")
        셀(행, 3) = QueueCounter.AttValue("QLenMax(1,1)")
        행 = 행 + 1 다음


    'Vissim 소프트웨어를 닫으려면 모든 객체를 삭제하세요.
    Vissim = 아무것도 설정하지 않음

끝 서브
```

6.3.3 단일 단계로 시뮬레이션 실행

시뮬레이션을 단일 단계로 실행하면 각 시뮬레이션 단계 후에 네트워크 또는 시뮬레이션 작업을 수행할 수 있습니다. 또한, 각 시뮬레이션 단계 후에 시뮬레이션을 중지할 수도 있습니다. 시뮬레이션 중에는 일부 작업이 허용되지 않습니다. "EditableDuringSim" 또는 "SimAttr"로 표시되고 "Editable"이 True인 속성만 시뮬레이션 실행 중에 수정할 수 있습니다. 자세한 내용은 COM 인터페이스 참조를 참조하십시오. 다음 예제는 시뮬레이션을 3600단계로 실행하며, 이는 시뮬레이션 해상도 1의 경우 3600초에 해당합니다. C1 셀의 번호를 가진 차량이 네트워크에 진입했다가 나가면 시뮬레이션이 더 일찍 중지됩니다.

예: RunSimulationSingleStep

```
Sub RunSimulationSingleStep()

    '변수 선언
    Dim Vissim을 객체로 사용                                    ' 가변 비심
    Dim Vehicle As Variant
    vehicleWasInNet = False
    vehicleFound = False

    'Vissim-Object 생성(변수 Vissim을 소프트웨어 Vissim에 연결)
    Set Vissim = CreateObject("Vissim.Vissim") '입력 파일 로드(셀 B1의 파일 이
    름)
    'Vissim.LoadNet 셀(1, 2)

    '시뮬레이션
    Vissim.Simulation.AttValue("SimRes") = 1 구성

    '시뮬레이션을 실행하다
    i = 0 ~ 3600
        ' Vissim.Simulation.RunSingleStep
        으로 한 단계 시뮬레이션 수행

        '원하는 번호의 차량을 찾아라
        차량 발견 = 거짓
        Vissim.net.Vehicles의 각 차량에 대해
            'C1 셀에서 차량 번호를 찾습니다. Vehicle.AttValue("No") = Cells(1, 3)
            이면
                차량WasInNet = 참
                차량 발견 = 참
                종료
            종료
        다음

        차량이 네트워크를 이탈하면 종료
        만약 vehicleWasInNet이고 VehicleFound가 없다면
            종료
        종료
    다음 나

    ' Vissim.Simulation.Stop 시뮬
    레이션을 중지합니다.

    '모든 객체를 삭제하여 Vissim 소프트웨어를 닫습니다.
    Vissim = 아무것도 설정하지 않음

끝 서브
```

7 다양한 프로그래밍 언어의 기본 명령

이 장에서는 COM 인터페이스를 통해 Vissim을 제어하기 위한 몇 가지 기본 명령을 제공합니다.
VBA, Python, MATLAB®, C++, C# 및 JAVA용입니다. 각 프로그램 언어에 대해 다음 명령이 제공됩니다. • Vissim 시작

- 네트워크 파일을 엽니다
- 네트워크 파일 저장
- 시뮬레이션을 실행하다
- 네트워크 객체에 액세스
- 속성 읽기 및 설정
- Vissim 닫기

COM 예제

COM을 사용하여 PTV Vissim을 제어하는 기본 구문을 보여주는 다양한 프로그래밍 언어에 대한 예제가 제공됩니다. 예제는 다음 위치에 있습니다.
…\예제 교육\COM\기본 명령\.

다음 프로그래밍 언어에 대한 COM 예제가 있습니다.
- VBA 파일: COM Basic Commands.bas
- Python, 파일: COM Basic Commands.py
- C++, 파일: COM Basic Commands.cpp
- C# 파일: COM Basic Commands.cs
- Matlab 파일: COM Basic Commands.m
- 자바 파일: COM 기본 명령.java, COM 기본 명령 - Utils.java

자세한 내용은 설명 파일 COM Basic Commands Desc_ENG.pdf 또는 해당 파일의 주석을 참조하십시오. 간단한 텍스트 편집기를 사용하여 모든 파일을 읽을 수 있습니다.

7.1 VBA

행동	VBA 명령
COM 서버 생성	Vissim = CreateObject("Vissim.Vissim") 설정
특정 Vissim 버전으로 COM 서버 생성	Vissim = CreateObject("Vissim.Vissim.23") 설정
부하 네트워크	Vissim.LoadNet(파일 이름)
네트워크 저장	Vissim.SaveNetAs 파일 이름
Vissim.Simulation.RunSingleStep으로 단일 단계 시뮬레이션을 실행합니다.	
시뮬레이션을 지속적으로 실행	Vissim.Simulation.RunContinuous
특정 객체에 접근	ObjLink10 = Vissim.Net.Links.ItemByKey(10) 설정
속성 읽기	… = Obj.AttValue("...")
속성 설정	객체.AttValue("...") = …
COM 서버 닫기	Vissim = 아무것도 설정하지 않음

7.2 파이썬

행동	파이썬 명령어
COM 서버 생성	win32com.client를 com으로 가져오기
특정 Vissim 버전으로 COM 서버 생성	Vissim = com.gencache.EnsureDispatch("Vissim.Vissim") Vissim = com.gencache.EnsureDispatch("Vissim.Vissim.23")
부하 네트워크	Vissim.LoadNet(파일 이름)
네트워크 저장	Vissim.SaveNetAs(파일 이름)
Vissim.Simulation.RunSingleStep()에서 단일 단계 시뮬레이션을 실행합니다.	
Vissim.Simulation.RunContinuous()를 사용하여 시뮬레이션을 지속적으로 실행합니다.	
특정 객체에 접근	ObjLink10 = Vissim.Net.Links.ItemByKey(10)
속성 읽기	… = Obj.AttValue('…')
속성 설정	Obj.SetAttValue('…', ...)
COM 서버 닫기	비심 = 없음

7.3 씨++

행동	C++ 명령어
COM 초기화	#import "C:\\Program Files\\PTV Vision\\PTV Vissim 2023\\Exe\\Vissim230.exe" rename_namespace ("VISSIMLIB") CoInitialize(NULL);
COM 서버 생성	VISSIMLIB::IVissimPtr Vissim; Vissim.CreateInstance("Vissim.Vissim");
특정 Vissim 버전으로 COM 서버 생성	Vissim.CreateInstance("Vissim.Vissim.23");
부하 네트워크	Vissim->LoadNet(파일명, false);
네트워크 저장	Vissim->SaveNetAs(파일 이름);
단일 단계로 시뮬레이션을 실행합니다	Vissim->GetSimulation()->RunSingleStep();
시뮬레이션을 지속적으로 실행합니다	Vissim->GetSimulation()->RunContinuous();
특정 객체에 접근	Vissim->GetNet()->링크->GetItemByKey(10);
속성 읽기	Obj->GetAttValue("...")
속성 설정	Obj->PutAttValue("...", "...");
COM 서버 닫기	Vissim.Detach()->Release();
COM 초기화 해제	CoUninitialize();

7.4 C#

사용된 개발 환경에서 Vissim COM 객체를 사용하려면 설치된 Vissim 실행 파일(현재는 VISSIM230.EXE)에 대한 참조를 설정해야 합니다. 그러면 Vissim COM 객체 모델의 객체와 메서드를 C# 프로젝트에서 사용할 수 있습니다.

Microsoft Visual C# .NET 개발 프레임워크를 사용하는 경우 VBA의 개체 카탈로그와 유사한 "개체 브라우저"를 사용하여 개체 모델을 탐색할 수 있습니다.

내장된 Interop 유형은 지원되지 않습니다. VISSIMLIB의 "Embed Interop Types" 속성을 false로 설정해야 합니다. 그렇지 않으면 코드를 컴파일할 수 없습니다.

행동	C# 명령
COM 초기화	시스템 사용; VISSIMLIB 사용;
COM 서버 생성	VissimClass Vissim = new VissimClass();
부하 네트워크	Vissim.LoadNet(파일명, false);
네트워크 저장	Vissim.SaveNetAs(파일 이름);
단일 단계 시뮬레이션을 실행합니다	Vissim.Simulation.RunSingleStep();
시뮬레이션을 지속적으로 실행합니다	Vissim.Simulation.RunContinuous();
특정 객체에 접근	ILink ObjLink10 = (ILink)Vissim.Net.Links.ItemByKey(10);
속성 읽기	... = (var)Obj.AttValue["..."];
속성 설정	Obj.set_AttValue("...", ...);
COM 서버 닫기	Vissim.Exit();

7.5 매트랩

기본적으로 MATLAB는 2차원 배열을 COM에 전달합니다. PTV Vissim은 항상 1차원 배열을 예상합니다. (SafeArray[0,0], SafeArray[0,1], … 대신 SafeArray[0], SafeArray[1], …). MATLAB에서 다음 명령을 사용하여 동작을 변경할 수 있습니다: feature('COM_SafeArraySingleDim', 1)

행동	MATLAB 명령어
COM 서버 생성	Vissim = actxserver('Vissim.Vissim');
특정 Vissim 버전으로 COM 서버 생성	Vissim = actxserver('Vissim.Vissim.23');
부하 네트워크	Vissim.LoadNet(파일 이름)
네트워크 저장	Vissim.SaveNetAs(파일 이름)
Vissim.Simulation.RunSingleStep으로 단일 단계 시뮬레이션을 실행합니다.	
시뮬레이션을 지속적으로 실행	Vissim.Simulation.RunContinuous
특정 객체에 접근	ObjLink10 = Vissim.Net.Links.ItemByKey(10)
속성 읽기	Obj.AttValue('…')
속성 설정	set(Obj, 'AttValue', '...', ...);
COM 서버 닫기	Vissim.Exit; Vissim.release

7.6 자바

이 명령은 Jacob (링크)을 사용합니다. COM 인터페이스에 접근하려면 Jacob을 다운로드한 후 jacob-*.dll 파일을 c:\windows\system32에 복사하고 jacob.jar 파일을 Java 프로젝트의 라이브러리에 추가하세요.

행동	JAVA 명령어
COM 초기화	com.jacob.com.*을 가져옵니다. com.jacob.activeX.*을 가져옵니다.
COM 서버 생성	ActiveXComponent Vissim = new ActiveXComponent("VISSIM.Vissim");
특정 Vissim 버전으로 COM 서버 생성	ActiveXComponent Vissim = new ActiveXComponent("VISSIM.Vissim.23");
부하 네트워크	Vissim.invoke("LoadNet", 파일 이름);
네트워크 저장	Vissim.invoke("SaveNetAs", 파일 이름);
시뮬레이션을 단일 단계로 실행합니다	Vissim.invokeGetComponent("시뮬레이션").invoke("RunSingleStep");
지속적으로 시뮬레이션 실행	Vissim.invokeGetComponent("Simulation").invoke("RunContinuous");
특정 객체에 접근	ActiveXComponent ObjLink10 = Vissim.invokeGetComponent ("Net").invokeGetComponent("링크").invokeGetComponent("ItemByKey", 새로운 Variant(linkNumber));
속성 읽기	Obj.invoke("AttValue", "..."); // Obj는 ActiveXComponent 클래스입니다.
속성 설정	Dispatch.invoke(Obj, "AttValue", Dispatch.Put, new Object[]{"...", "..."}, new int[1]);
COM 서버 닫기	Vissim.safeRelease();