Using Scripting API

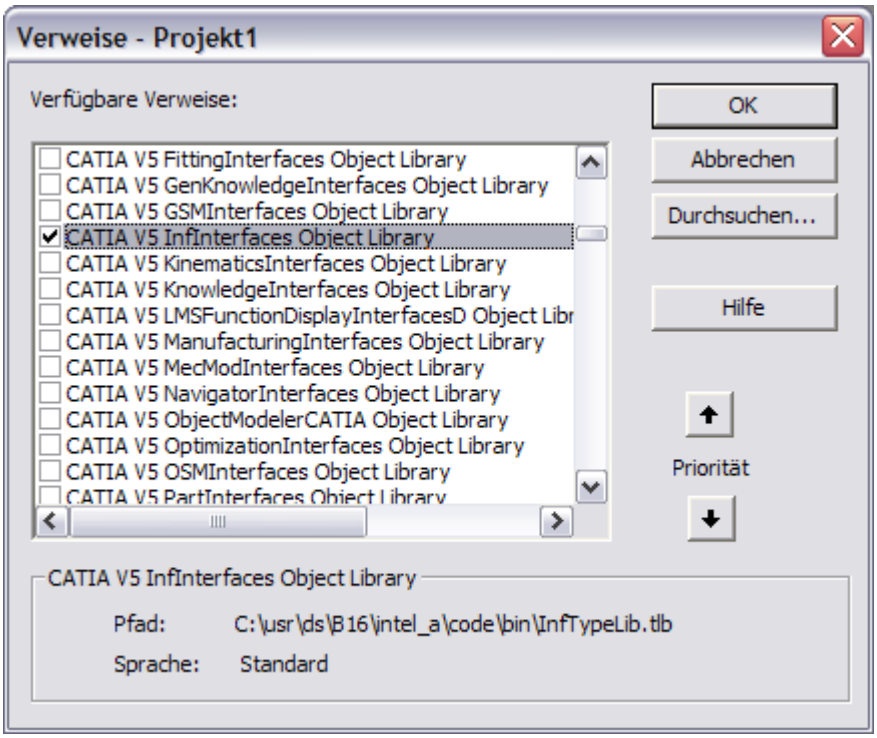
When we talk about scripting on Windows we talk in fact about the COM-enabled DLLs accessed through TypeLibraries CATIA provides. This is a short introduction how to use this Scripting API in CATIA-VBA, VB6 and VB.Net

Using the Scripting API with CATIA-VBA

You need to do nothing as all references are automatically added.
You access the CATIA application object directly with the predefined CATIA object.

Using the Scripting API with VB6

You need to add the required CATIA references to your project through the Add references panel



To get the CATIA application object you need to use the GetObject or CreateObject method, e.g. when CATIA is already running

```
Dim CATIA As INFITF.Application

Set CATIA = GetObject(, "CATIA.Application")
```

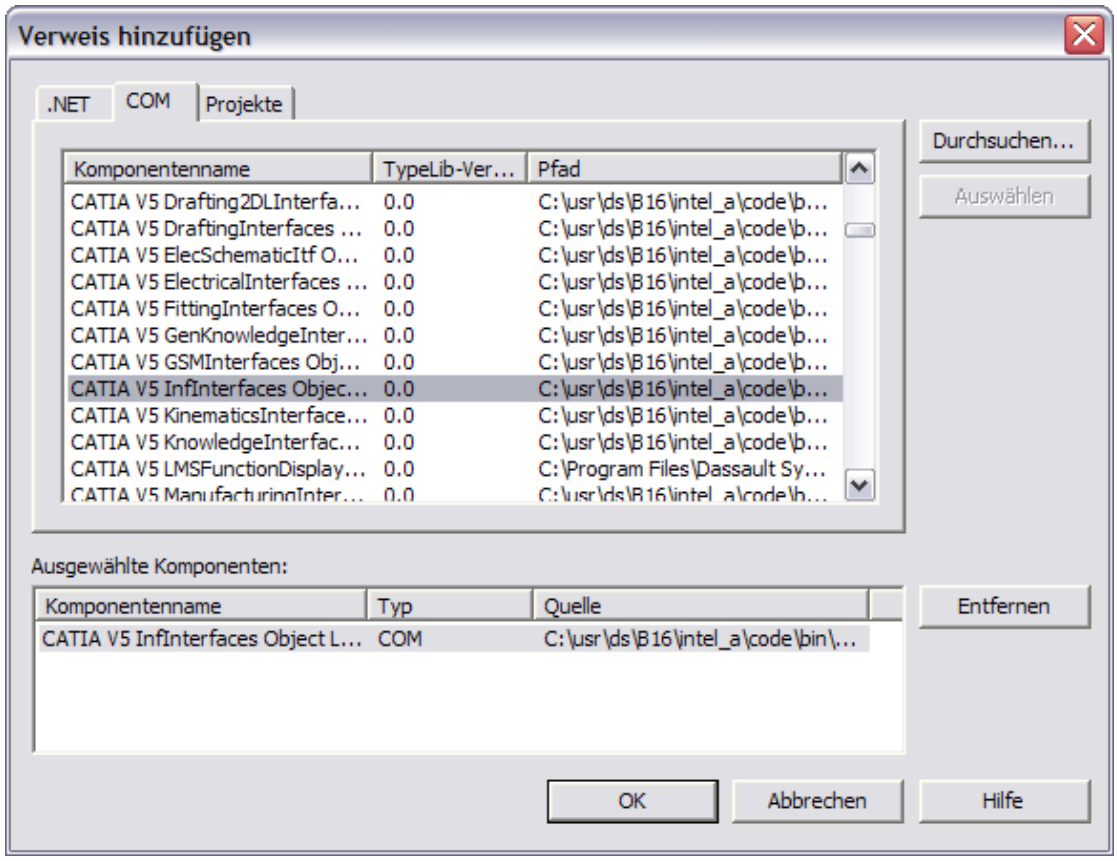
Using the Scripting API with the .Net Languages

Creating the InterOp libraries

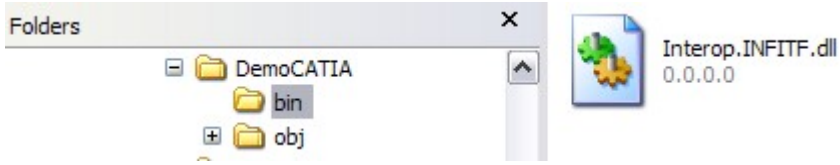
In a .Net environment you need to use so-called InterOp libraries to communicate with CATIAs COM objects. How to create these is explained in the VB.Net section.

Using the Scripting API with VB.Net

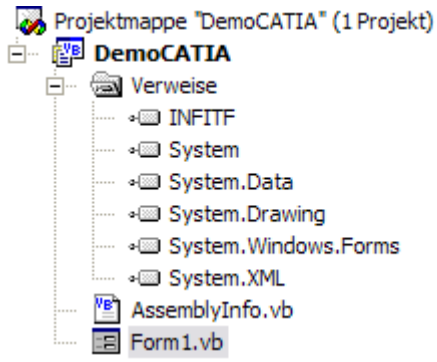
There are 2 methods to do it:
The simple method when using Visual Studio .Net
You need to add the required CATIA references to your project.
Therefor you need to select it under the COM tab in the Add references panel



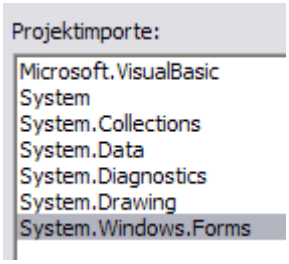
This will create an Interop DLL in the bin folder of your VB.Net project



and add this reference to your project



Let's see how this works in a standard VB.Net Windows application. Such an application automatically imports the following libraries



The following would be possible

```
Dim CATIA As INFITF.Application

CATIA = Microsoft.VisualBasic.GetObject(, "CATIA.Application")
```

However, good practice is to have the Strict option set to on which would lead to a compiler error with the above code as the GetObject method returns a generic object not an Application object.
So it needs to be changed to the following

```
Dim CATIA As INFITF.Application

CATIA = CType(Microsoft.VisualBasic.GetObject(, "CATIA.Application"), _
              INFITF.Application)
```

The second method is to generate the interop libraries manually using the tlbimp utility of the .Net framework. This can be found in the installation path of the framework, e.g.

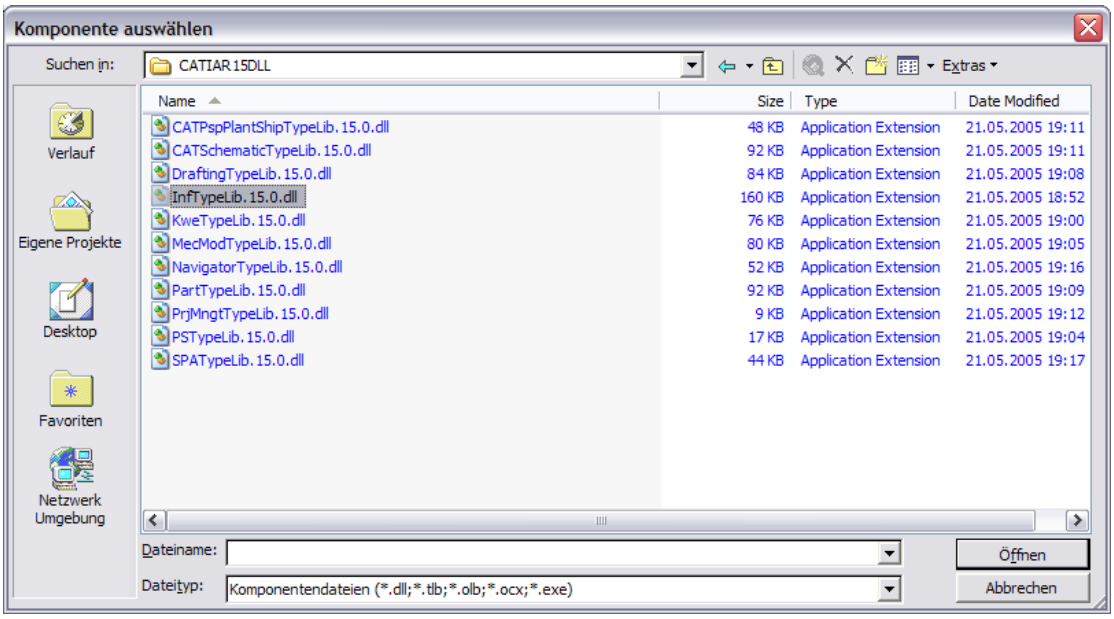
C:\Program Files\Microsoft.NET\SDK\v1.1\Bin\tlbimp.exe

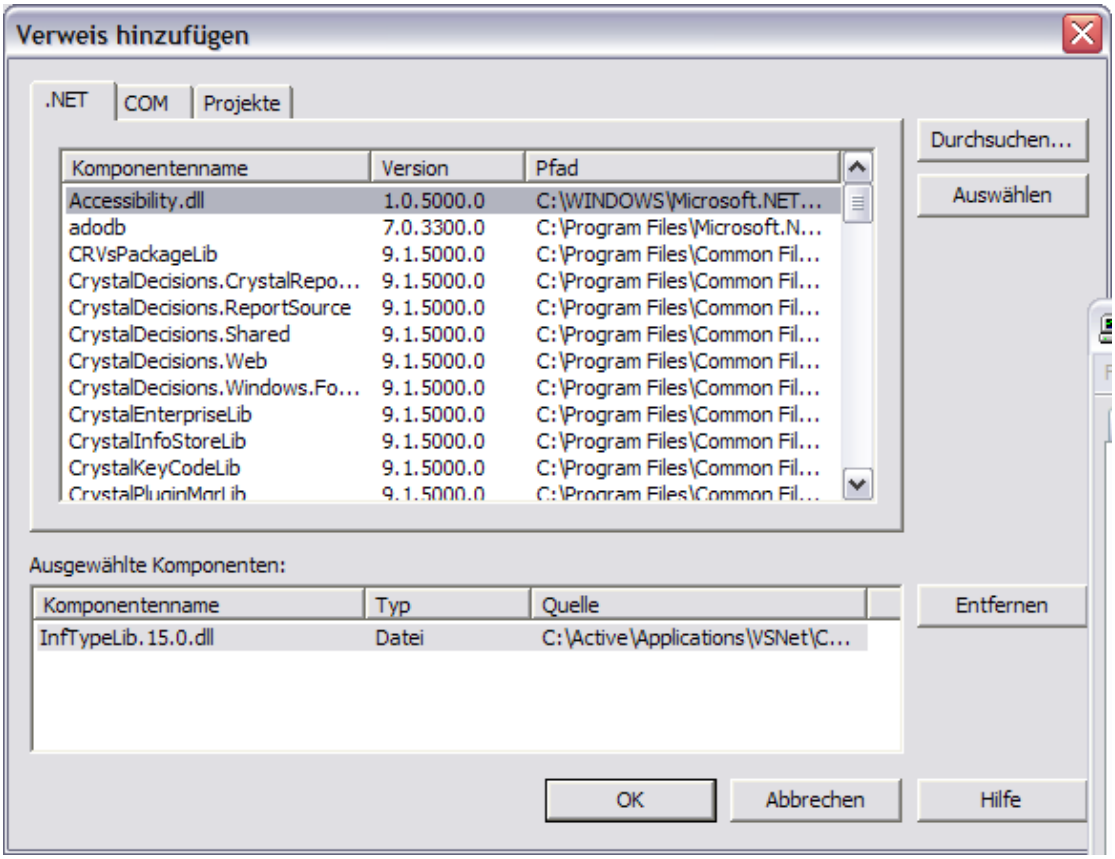
To create a interop library for the basic API (the environment variables are used to make the command line more simple (which is represented as 2 lines in the following)

```
set TLBIMP="C:\Program Files\Microsoft.NET\SDK\v1.1\Bin\tlbimp.exe"
set CATIA=C:\Program Files\Dassault Systemes\B15\intel_a\code\bin
set TYPELIB=InfTypeLib
set OUT=C:\Active\Applications\VSNet\CATIAR15DLL
```

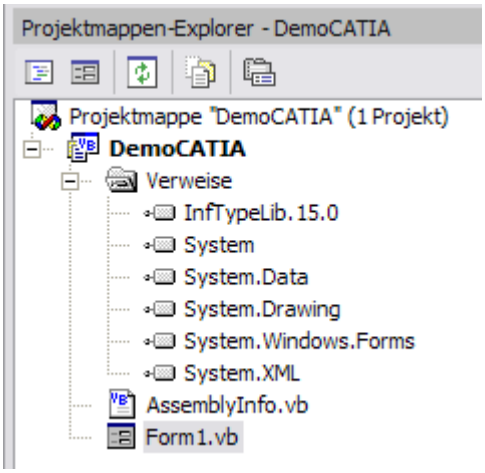
```
%TLBIMP% "%CATIA%\%TYPELIB%.tlb" /out:%OUT%\%TYPELIB%.
15.0.dll /namespace:CATIA.%TYPELIB% /asmversion:5.15.0.0
```

This creates the interop library InfTypeLib.15.0.dll in the directory %OUT% In addition a namespace is defined for this library as well as a version.
You have to add this as a reference to your project using the Add references panels via Browse and select the newly created dll





which adds this library to your project



which then can be used with the following code (you have to use the Namespace you have defined with tlbimp utility).

```
Dim CATIA As CATIA.InfTypeLib.Application

CATIA = CType(Microsoft.VisualBasic.GetObject(, "CATIA.Application"), _
                    CATIA.InfTypeLib.Application)
```

Optionally without using the Microsoft.VisualBasic compatibility library it is possible to use the InterOpServices

```
CATIA = CType(System.Runtime.InteropServices.Marshal.GetActiveObject( _
                "CATIA.Application"), _
                CATIA.InfTypeLib.Application)
```

Using tlbimp you have to take care that some typelibs are dependant on others, e.g. KweTypeLib depends on InfTypeLib.
In this case tlbimp would create an interop library automatically (like in the Visual Studio environment). To avoid this you first have to create the base interops and the reference them in the tlbimp, e.g. for KweTypeLib

```
set TLBIMP="C:\Program Files\Microsoft.NET\SDK\v1.1\Bin\tlbimp.exe"
set CATIA=C:\Program Files\Dassault Systemes\B15\intel_a\code\bin
set TYPELIB=InfTypeLib
set OUT=C:\Active\Applications\VSNet\CATIAR15DLL

%TLBIMP% "%CATIA%\%TYPELIB%.tlb" /out:%OUT%\%TYPELIB%.
15.0.dll /namespace:CATIA.%TYPELIB% /asmversion:5.15.0.0

set TYPELIB=KweTypeLib

%TLBIMP% "%CATIA%\%TYPELIB%.tlb" /out:%OUT%\%TYPELIB%.
15.0.dll /namespace:CATIA.%TYPELIB% /asmversion:5.15.0.0 /reference:
%OUT%\InfTypelib.15.0.dll
```

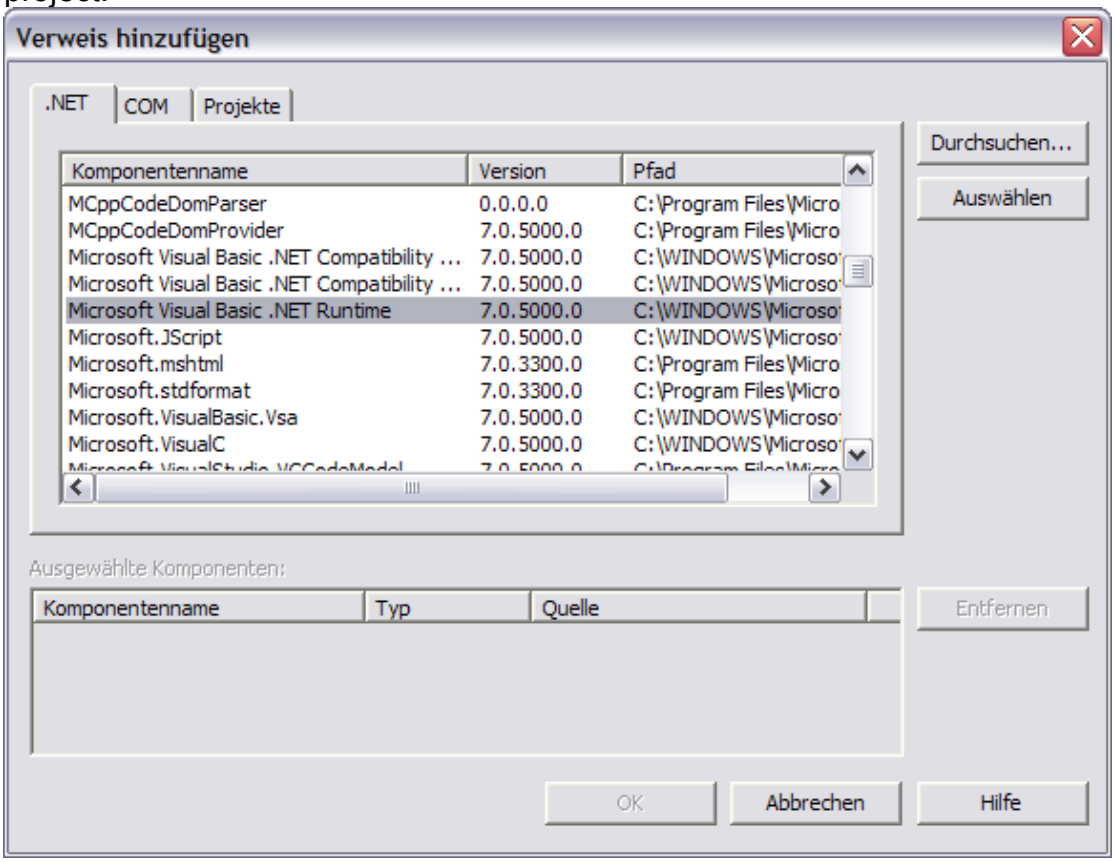
creating these dlls


InfTypeLib.15.0.dll
5.15.0.0


KweTypeLib.15.0.dll
5.15.0.0

Using the Scripting API with C#

Of course the above said is valid for C# as well.
However it is necessary if using the GetObject method from the VisualBasic library to load the Microsoft VisualBasic .Net Runtime as a reference to your project.



Then you can get CATIAs Application object as follows

```
CATIA.InfTypeLib.Application CATIA;
CATIA=(CATIA.InfTypeLib.Application)
Microsoft.VisualBasic.Interaction.GetObject(null,"CATIA.Application");
```

Or using the InterOpServices

```
CATIA=(CATIA.InfTypeLib.Application)
System.Runtime.InteropServices.Marshal.GetActiveObject("CATIA.Application");
```

Using the Scripting API with C++ in .Net

Of course the above said is valid for C++ in a .Net application as well.
So you have to include your reference for the interop CATIA dll in your C++ project.

To get the CATIA Application object this code can be used
To make the code shorter

```
using namespace System::Runtime::InteropServices;
```

Then in your code (just one checking at the end)

```
System::Object* CATIAObj;
CATIAObj = Marshal::GetActiveObject("CATIA.Application");
CATIA::InfTypeLib::Application* CATIA;
CATIA = dynamic_cast<CATIA::InfTypeLib::Application*>(CATIAObj);
if (CATIA == 0)
       System::Diagnostics::Debug::WriteLine("CATIA not retrieved");
```