Abstract
This document describes a CSharp interface to SWI-Prolog. The
described interface provides a layer around the C-interface for natural
programming from C#. The interface deals with automatic type-
conversion to and from Prolog, mapping of exceptions and making
queries to Prolog in an easy way. There is a call-back from Prolog to C#

Introduction
The first version of this Interface was more or less a port of the C++
interface. Now the naming is more '.Net' like and the interface provides
a number of features that make queries to SWI-Prolog very easy and
powerful. Using programmable type-conversion (casting), native data-
types can be translated automatically into appropriate Prolog types.
Automatic destruction deals with most of the cleanup required.

Acknowledgements
I would like to thank Jan Wielemaker for answering many questions and
for his comments.
Also to Arne Skjærholt for the 64-Bit version (SwiPICs64.dll) and Batu
Akan for the Mono code. Foutelet Joel provide the F# sample.

Download binaries
Here is the link to download the latest binaries or older versions.
Download page
At present I only publish the binaries including the documentation on this
site.
The sources, which are under LGPL 2, are on GitHub/SWI-
Prolog/contrib-swiplcs.

Versions
The latest version work with SWI-Prolog 6.3.1 and higher.
The AssemblyVersion number, e.g. 1.1.60601.0, can be interpreted as follows:

- 1 - major version
- 1 - minor version
- 60601 - SWI-Prolog version 6.6.1 (test cases run against this prolog version)
- 0 - patch level version

**Getting started**

Copy SwiPICs.dll or SwiPICs64.dll and SwiPICs.XML where ever you want and add a project reference to SwiPICs.dll.

After that IntelliSense and tool tips should be available.

Make sure that libswipl.dll and its dependencies could be found by the system. For the sample below it is: libswipl.dll, pthreadGC2.dll, libgmp-10.dll, files.dll For a big application it could be a lot more.

TIP: For development add the SWI-prolog bin directory to the PATH environment variable.
NOTE: Don't forget to restart Visual Studio after that. VS must recognize the new environment for debugging.

Basically windows search first in the folder where the executable resist than in the windows system directory and at least in the directories that
are listed in the PATH environment variable. For details see "Dynamic-Link Library Search Order"

If libswipl.dll or one of its dependencies could not found you will receive an error like
System.IO.FileNotFoundException: Das angegebene Modul wurde nicht gefunden. (Ausnahme von HRESULT: 0x8007007E)

An other common error is:
SWI-Prolog: [FATAL ERROR:
   Could not find system resources]
Failed to release stacks

To fix this add the SWI_HOME_DIR environment variable as described
SWI-Prolog FAQ FindResources with a statment like this before calling
PlEngine.Initialize.

Environment.SetEnvironmentVariable("SWI_HOME_DIR",
"@"the_PATH_to_boot32.prc");

**First program**

A sample says more then I want to write here.

C#```
using System;
using SbsSW.SwiPlCs;

namespace HelloWorldDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //Environment.SetEnvironmentVariable("SWI_HOME_DIR",
            if (!PlEngine.IsInitialized)
            {
                String[] param = { "-q" };  // suppr
PlEngine.Initialize(param);
PlQuery.PlCall("assert(father(martin,
PlQuery.PlCall("assert(father(uwe, g
PlQuery.PlCall("assert(father(uwe, m
PlQuery.PlCall("assert(father(uwe, a
using (var q = new PlQuery("father(P
{
    foreach (PlQueryVariables v in q
        Console.WriteLine(v["L"]).ToS

    Console.WriteLine("all children
    q.Variables["P"].Unify("uwe");
    foreach (PlQueryVariables v in q
        Console.WriteLine(v["C"]).ToS

    PlEngine.PlCleanup();
    Console.WriteLine("finshed!");
}

Here is how to use the library in F#. Manny thanks to Foutelet Joel for this sample.

F#

// Learn more about F# at http://fsharp.net
open System
open SbsSW.SwiPlCs;

let ple = PlEngine.IsInitialized in
if ple then printfn "Echec initialisation" else
begin
    PlEngine.Initialize(["-q"]
    PlQuery.PlCall("assert(father(martin, i
    PlQuery.PlCall("assert(father(uwe, glor
    PlQuery.PlCall("assert(father(uwe, mela
    PlQuery.PlCall("assert(father(uwe, ayal

Copied
let q = new PlQuery "father(P, C), atomic_list_concat([P,'is_father_of ',C], L)"
begin
  Seq.iter (fun (x : PlQueryVariables) -> printfn "all children from uwe:
  let r : PlQueryVariables = q.Variables
  Seq.iter (fun (x : PlQueryVariables) -> printfn end
  PlEngine.PlCleanup()
  printfn "%A" "finished!"
end

For further samples see the examples in SbsSW.SwiPICs and SbsSW.SwiPICs.PlEngine.

The class SbsSW.SwiPICs.PlQuery is the key to ask SWI-Prolog for proofs or solutions.

The SbsSW.SwiPICs.PlTerm plays a central role in conversion and operating on Prolog data.

**Programming tips**

I strongly recomanate to use SbsSW.SwiPICs.PlQuery in a using statment using statment like in the sample program above. An alternative is to call Dispose explicitly like in the sample below.

C#

```c#
const string strRef = "a;e;";
PlQuery.PlCall("assert(n('" + strRef + "'))");
var q = new PlQuery("n(X)");
Assert.IsTrue(q.NextSolution());
Assert.AreEqual(strRef, q.Variables["X"].ToString());
var q2 = new PlQuery("n('" + strRef + "')");
Assert.IsTrue(q2.NextSolution());
Assert.AreEqual(strRef, q.Variables["X"].ToString());
```
q2.Dispose();
q.Dispose();

Note that access to the query Variables is impossible after Dispose().

**Known Bugs**

*SbsSW.SwiPICs.PIEngine.Initialize* work *not* as expected if there are e.g. German umlauts in the parameters e.g. in the path or filename for *qlf* file (switch `-x`)

See marshalling in the source `NativeMethods.cs`

by Uwe Lesta, SBS-Softwaresysteme GmbH
The online documentation home is [here](#).

This namespace SbsSW.SwiPICs provides an .NET interface to [SWI-Prolog](#).

### Overview

Prolog variables are dynamically typed and all information is passed around using the C-interface type term_t which is an int. In C#, term_t is embedded in the lightweight struct `PlTerm`. Constructors and operator definitions provide flexible operations and integration with important C#-types (`string`, `int` and `double`).

The list below summarises the important classes / struct defined in the C# interface.

<table>
<thead>
<tr>
<th>class / struct</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PlEngine</code></td>
<td>A static class represents the prolog engine.</td>
</tr>
<tr>
<td><code>PlTerm</code></td>
<td>A struct representing prolog data.</td>
</tr>
<tr>
<td><code>PlTermV</code></td>
<td>A vector of <code>PlTerm</code>.</td>
</tr>
<tr>
<td><code>PlQuery</code></td>
<td>A class to query Prolog.</td>
</tr>
</tbody>
</table>

#### Types

- **All Types**
  - `PlEngine`

  This static class represents the prolog engine.
The class PlFrame provides an interface to discard unused term-references as well as rewinding unifications (data-backtracking). Reclaiming unused term-references is automatically performed after a call to a C#-defined predicate has finished and returns control to Prolog. In this scenario PlFrame is rarely of any use.

This class comes into play if the top level program is defined in C# and calls Prolog multiple times. Setting up arguments to a query requires term-references and using PlFrame is the only way to reclaim them.

This class allows queries to prolog.

A query can be created by a string or by constructing compound terms see Constructors for details.

All resources an terms created by a query are reclaimed by Dispose(). It is recommended to build a query in a using scope.

There are four possible opportunities to query Prolog:

<table>
<thead>
<tr>
<th>Query type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A static call</td>
<td>To ask prolog for</td>
</tr>
</tbody>
</table>
A proof. Return only true or false.

<table>
<thead>
<tr>
<th><strong>A</strong></th>
<th><strong>Return</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PICallQuery(String)</strong></td>
<td>To get the first result of a goal</td>
</tr>
<tr>
<td><strong>Construct a PlQuery object by a string.</strong></td>
<td>The most convenient way.</td>
</tr>
<tr>
<td><strong>Construct a PlQuery object by compound terms.</strong></td>
<td>The most flexible and fast (runtime) way.</td>
</tr>
</tbody>
</table>

For examples see **PlQuery(String)** and **PlQuery(String, P1TermV)**

<table>
<thead>
<tr>
<th><strong>PlQuerySwitch</strong></th>
<th>Flags that control for the foreign predicate parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Swi-Prolog Manual - 9.6.16 Querying Prolog.</strong></td>
<td></td>
</tr>
</tbody>
</table>

| **PlQueryVar** | Represents one variable of a Query result. |

<table>
<thead>
<tr>
<th><strong>PlQueryVariables</strong></th>
<th>Represents the set variables of a Query if it was created from a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This class is also used to represent the results of a PlQuery after <strong>ToList()</strong> or <strong>SolutionVariables</strong> was called.</td>
<td></td>
</tr>
</tbody>
</table>
The `PlTerm` struct plays a central role in conversion and operating on Prolog data.

`PlTerm` implements `IComparable` to support ordering in `System.Linq` queries if `PlTerm` is a List.

Creating a `PlTerm` can be done by the `Constructors` or by the following static methods:

`PlVar()`, `PlTail()`, `PlCompound`, `PlString()`, `PlCodeList()`, `PlCharList()` (see remarks)

The struct `PlTermV` represents an array of term-references.

This type is used to pass the argument to a foreign defined predicate (see `DelegateParameterVarArgs`), construct compound terms (see `PlCompound(String, PlTermV)`) and to create queries (see `PlQuery`).

The only useful member function is the overloading of [], providing (0-based) access to the elements. `Item[Int32]` Range checking is performed and raises a `ArgumentOutOfRangeException` exception.
Obtain the type of a term, which should be a term returned by one of the other interface predicates or passed as an argument. The function returns the type of the Prolog term. The type identifiers are listed below.

### Examples

Before going into a detailed description of the CSharp classes let me present a few examples illustrating the `feel` of the interface. The Assert class in the sample is from the test framework and has nothing to do with the interface. It shows only which return values are expected.

**Creating terms**

This very simple example shows the basic creation of a Prolog term and how a Prolog term is converted to C#-data:

```csharp
PlTerm t1 = new PlTerm("x(A)");
PlTerm t2 = new PlTerm("x(1)");
Assert.IsTrue(t1.Unify(t2));
Assert.AreEqual("x(1)", t1.ToString());
```

**Calling Prolog**

This example shows how to make a simple call to prolog.

```csharp
PlTerm l1 = new PlTerm("[a,b,c,d]");
Assert.IsTrue(PlQuery.P1Call("is_list", l1));
```
Getting the solutions of a query

This example shows how to obtain all solutions of a prolog query.

**PlQuery** takes the name of a predicate and the goal-argument vector as arguments. From this information it deduces the arity and locates the predicate. The member-function NextSolution() yields true if there was a solution and false otherwise. If the goal yielded a Prolog exception it is mapped into a C# exception.

```csharp
PlQuery q = new PlQuery("member", new PlTermV
while (q.NextSolution())
    Console.WriteLine(s[0].ToString());
```

There is an other constructor of **PlQuery** which simplify the sample above.

```csharp
PlQuery q = new PlQuery("member(A, [a,b,c])")
foreach (PlTermV s in q.Solutions)
    Console.WriteLine(s[0].ToString());
```

An other way to get the results is to use **SolutionVariables** to iterate over **PlQueryVariables**.

```csharp
PlQuery q = new PlQuery("member(A, [a,b,c])")
foreach (PlQueryVariables vars in q.SolutionVariables)
    Console.WriteLine(vars["A"].ToString());
```

It is also possible to get all solutions in a list by **ToList()**. This could be used to work with LinQ to objects which is really nice. **PlQuery** and
ToList for further samples.

```csharp
var results = from n in new PlQuery("member(A
foreach (var s in results)
    Console.WriteLine(s.A);
```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de
This static class represents the prolog engine.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static class PlEngine</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Public NotInheritable Class Class PlEngine</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public ref class PlEngine abstract sealed</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>[&lt;AbstractClassAttribute&gt;]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>[&lt;SealedAttribute&gt;]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>type PlEngine = class end</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Members

- **Initialize(String[])**

  Initialise SWI-Prolog

  The write method of the output stream is redirected to `SbsSW.SwiPlCs.Streams` before Initialize. The read method of the input stream just after Initialize.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsInitialized</code></td>
<td>To test if the prolog engine is initialized.</td>
</tr>
<tr>
<td><code>PlCleanup()</code></td>
<td>Try a clean up but it is buggy. Try the web for &quot;possible regression from pl-5.4.7 to pl-5.6.27&quot; to see reasons.</td>
</tr>
<tr>
<td><code>PlHalt()</code></td>
<td>Stops the PlEngine and returns the reference count of the engine.</td>
</tr>
<tr>
<td><code>PlThreadAttachEngine()</code></td>
<td>Return the reference count of the engine. If an error occurs, -1 is returned. If this Prolog is not compiled for multithreading, -2 is returned.</td>
</tr>
<tr>
<td><code>PlThreadDestroyEngine()</code></td>
<td>Destroy the Prolog engine in the calling thread. Only takes effect if PL_thread_destroy_engine() is called as many times as PL_thread_attach_engine() in this thread. Please note that construction and destruction of engines are relatively expensive operations. Only destroy an engine if performance is not critical and memory is a critical resource.</td>
</tr>
<tr>
<td><code>PlThreadSelf()</code></td>
<td>This method is also provided in the single-threaded version of SWI-Prolog.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>RegisterForeign(Delegate)</code></td>
<td>Register a C# callback</td>
</tr>
<tr>
<td><code>RegisterForeign(String, Delegate)</code></td>
<td>Register a C# callback</td>
</tr>
<tr>
<td><code>RegisterForeign(String, Int32, Delegate)</code></td>
<td>Register a C# callback</td>
</tr>
<tr>
<td><code>RegisterForeign(String, String, Int32, Delegate)</code></td>
<td>Register a C# callback</td>
</tr>
<tr>
<td><code>SetStreamFunctionRead(PlStreamType, DelegateStreamReadFunction)</code></td>
<td>TODO</td>
</tr>
<tr>
<td><code>SetStreamFunctionWrite(PlStreamType, DelegateStreamWriteFunction)</code></td>
<td>This is a primitive approach to output from a stream.</td>
</tr>
</tbody>
</table>

**Examples**

A sample

```csharp
if (!PlEngine.IsInitialized)
{
    String[] empty_param = { "" };  
    PlEngine.Initialize(empty_param);
```
The following sample show how a file is consult via command-line options.

```csharp
public void Demo_consult_pl_file_by_param()
{
    string[] ref_values = { "gloria", "melanie" };
    Console.WriteLine("Demo_consult_pl_file_by_param");

    // Build a prolog source file (skip this step if you already have one :-)
    string filename = Path.GetTempFileName();
    StreamWriter sw = File.CreateText(filename);
    sw.WriteLine("father(martin, inka)." );
    sw.WriteLine("father(uwe, gloria)." );
    sw.WriteLine("father(uwe, melanie)." );
    sw.WriteLine("father(uwe, ayala)." );
    sw.Close();

    // build the parameterstring to Initialize PlEngine with the generated file
    String[] param = { "-q", "-f", filename };
    try
    {
        PlEngine.Initialize(param);
        Console.WriteLine("all child's from uwe:");
        using (PlQuery q = new PlQuery("father(uwe, Child)" ))
        {
            int idx = 0;
            foreach (PlQueryVariables v in q."
```
Console.WriteLine(v["Child"].ToString());
Assert.AreEqual(ref_values[idx++],
}
}
catch (PlException e)
{
    Console.WriteLine(e.MessagePl);
    Console.WriteLine(e.Message);
}
finally
{
    PlEngine.PlCleanup();
}
} // Demo_consult_pl_file_by_param

- Inheritance Hierarchy
  - Object
    - PlEngine

- See Also
  - SbsSW.SwiPICs.Callback

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Initialise SWI-Prolog

The write method of the output stream is redirected to \texttt{SbsSW.SwiPlCsStreams} before \texttt{Initialize}. The read method of the input stream just after \texttt{Initialize}.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|}
\hline
\textbf{C\#} & \textbf{Visual Basic} & \textbf{Visual C++} & \textbf{F\#} \\
\hline
\texttt{public static void Initialize(}
\begin{align*}
\text{string[]} & \quad \text{argv} \\
\end{align*}
\texttt{)} \\
\texttt{Public Shared Sub Initialize (}
\begin{align*}
\text{argv As String()} \\
\end{align*}
\texttt{)} \\
\texttt{public:}
\begin{align*}
\text{static void Initialize(}
\text{array<string[^]>[^} argv} \\
\end{align*}
\texttt{)} \\
\texttt{static member Initialize :}
\begin{align*}
\text{argv : string[]} -> \text{unit} \\
\end{align*}
\end{tabular}
\end{table}

\textbf{Parameters}

\texttt{argv (String[])}

For a complete parameter description see the \texttt{SWI-Prolog reference}.
manual section 2.4 Command-line options.

sample parameter:

```c
String[] param = { "-q", "-f", @"some\file"};
```

At the first position a parameter """" is added in this method.

PL_initialise

Remarks

A known bug: Initialize work *not* as expected if there are e.g. German umlauts in the parameters See marshalling in the source
NativeMethods.cs

Examples

For an example see PLEngine

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**IsInitialized Property**

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PIEngine ➤ IsInitialized

To test if the prolog engine is up.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| ```
public static bool IsInitialized { get; }
``` | ```
Public Shared Readonly Property IsInitialized
Get
``` | ```
public:
static property bool IsInitialized { 
bool get ();
}
``` | ```
static member IsInitialized : bool with get
``` |

- **Property Value**

**Boolean**

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Prolog**

**PlCleanup Method**

SwiPICs interface ▶ SbsSW.SwiPICs ▶ PLEngine ▶ PlCleanup()

Try a clean up but it is buggy search the web for "possible regression from pl-5.4.7 to pl-5.6.27" to see reasons

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static void PlCleanup()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Sub PlCleanup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: static void PlCleanup()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static member PlCleanup : unit -&gt; unit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Remarks**

  Use this method only at the last call before run program ends

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Method**

SwiPlCs interface ► SbsSW.SwiPlCs ► PlEngine ► PlHalt()

Stops the PlEngine and **the program**

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static void PlHalt()</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Public Shared Sub PlHalt</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public:</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>static void PlHalt()</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>static member PlHalt : unit -&gt; unit</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Remarks**

SWI-Prolog calls internally `pl_cleanup` and then `exit(0)`

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Prolog**

**PIThreadAttachEngine Method**

SwiPICs interface ► SbsSW.SwiPICs ► PlEngine ► PlThreadAttachEngine()  

```
return : reference count of the engine
```

If an error occurs, -1 is returned.

If this Prolog is not compiled for multi-threading, -2 is returned.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static int PlThreadAttachEngine()</code></td>
<td><code>Public Shared Function PlThreadAttachEngine</code></td>
<td><code>public int PlThreadAttachEngine()</code></td>
<td><code>static member PlThreadAttachEngine : unit -&gt;</code></td>
</tr>
</tbody>
</table>

**Return Value**

**Int32**

A reference count of the engine

---

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**PIThreadDestroyEngine Method**

**SwiPlCs interface** ► **SbsSW.SwiPlCs** ► **PlEngine** ► **PIThreadDestroyEngine()**

Destroy the Prolog engine in the calling thread. Only takes effect if `PL_thread_destroy_engine()` is called as many times as `PL_thread_attach_engine()` in this thread.

Please note that construction and destruction of engines are relatively expensive operations. Only destroy an engine if performance is not critical and memory is a critical resource.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static bool PlThreadDestroyEngine()</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Public Shared Function PlThreadDestroyEngine</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public static bool PlThreadDestroyEngine()</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>static member PlThreadDestroyEngine : unit -&gt;</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Return Value**

**Boolean**

Returns `true` on success and `false` if the calling thread has no engine or this Prolog does not support threads.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
This method is also provided in the single-threaded version of SWI-Prolog, where it returns -2.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public static int PlThreadSelf()` | `Public Shared Function PlThreadSelf As Integer` | `public:
static int PlThreadSelf()` | `static member PlThreadSelf : unit -> int` |

### Return Value

**Int32**

Returns the integer Prolog identifier of the engine or -1 if the calling thread has no Prolog engine.

---

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
RegisterForeign Method

Register a C#-function to implement a Prolog predicate.

After this call returns successfully, a predicate with name (a string) and arity (a C# int) is created in module.

If module is NULL, the predicate is created in the module of the calling context or if no context is present in the module user.

Remarks

Add a additional namespace by:

using SbsSW.SwiPICs.Callback;

Examples

For an example see DelegateParameter2 and DelegateParameter1.

Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>RegisterForeign(Delegate)</td>
<td>Register a C# callback method</td>
</tr>
<tr>
<td>🔄</td>
<td>RegisterForeign(String, Delegate)</td>
<td>Register a C# callback method</td>
</tr>
<tr>
<td>🔄</td>
<td>RegisterForeign(String, Int32, Delegate)</td>
<td>Register a C# callback method</td>
</tr>
</tbody>
</table>
RegisterForeign(String, String, Int32, Delegate)

Register a C# callback method

See Also

SbsSW.SwiPICs.Callback

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Register a C# callback method

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>`public static bool RegisterForeign(</td>
<td>`Public Shared Function RegisterForeign (</td>
<td>`public static bool RegisterForeign(</td>
<td>`static</td>
</tr>
<tr>
<td>Delegate <code>method</code> )`</td>
<td><code>method As Delegate</code></td>
<td><code>Delegate</code> <code>method</code></td>
<td><code>member</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>)</code></td>
<td><code>)</code></td>
<td><code>)</code></td>
<td></td>
</tr>
<tr>
<td>`Public Shared Function RegisterForeign (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>method As Delegate</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>) As Boolean</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>`public: static bool RegisterForeign(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delegate^ <code>method</code> )`</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>`static member RegisterForeign :</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>method</code> : Delegate -&gt; bool`</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Parameters**

  - `method` (**Delegate**)  
    a delegate to a c# method `SbsSW.SwiPICs.Callback`

- **Return Value**

  - `Boolean`
  
    true if registration succeed otherwise false
Examples

For an example see DelegateParameter2 and DelegateParameter1.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Register a C# callback method

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| ```
public static bool RegisterForeign(
    string module,
    Delegate method
)
``` |
| ```
Public Shared Function RegisterForeign ( 
    module As String,
    method As Delegate
) As Boolean
``` |
| ```
public:
static bool RegisterForeign(
    String^ module,
    Delegate^ method
)
``` |
| ```
static member RegisterForeign : 
    module : string *
    method : Delegate -> bool
``` |

- Parameters

  **module (String)**
  - the name of a prolog module [Using Modules](#)
**method (Delegate)**

A delegate to a C# method `SbsSW.SwiPICs.Callback`

- **Return Value**
  - `Boolean`
  - True if registration succeed otherwise false

- **Examples**
  - For an example see `DelegateParameter2` and `DelegateParameter1`.

---

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

RegisterForeign Method (name, arity, method)

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PlEngine ➤

RegisterForeign(String, Int32, Delegate)

Register a C# callback method

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegisterForeign(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>string name,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int arity,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delegate method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Public Shared Function RegisterForeign ( name As String,
                                         arity As Integer,
                                         method As Delegate
) As Boolean

public:
static bool RegisterForeign(
    String^ name,
    int arity,
    Delegate^ method
)

static member RegisterForeign : 
    name : string *
    arity : int *
**method**: Delegate -> bool

- **Parameters**

  - **name** *(String)*
    The name of a static C# method

  - **arity** *(Int32)*
    The amount of parameters

  - **method** *(Delegate)*
    a delegate to a c# method **SbsSW.SwiPICs.Callback**

- **Return Value**

  - **Boolean**
    true if registration succeed otherwise false

- **Examples**

  For an example see **DelegateParameterVarArgs**

  by Uwe Lesta, SBS-Softwaresysteme GmbH
  Send comments on this topic to **Lesta at SBS-Softwaresysteme.de**

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
RegisterForeign Method (module, name, arity, method)

RegisterForeign(String, String, Int32, Delegate)

Register a C# callback method

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public static bool RegisterForeign(  
  string module,  
  string name,  
  int arity,  
  Delegate method  
) | Public Shared Function RegisterForeign (  
  module As String,  
  name As String,  
  arity As Integer,  
  method As Delegate  
) As Boolean | public:  
static bool RegisterForeign(  
  String^ module,  
  String^ name,  
  int arity,  
  Delegate^ method  
) |
static member RegisterForeign :
  module : string *
  name : string *
  arity : int *
  method : Delegate -> bool

- **Parameters**

  *module* (**String**)
  The name of the module (Prolog module system)

  *name* (**String**)
  The name of a static C# method

  *arity* (**Int32**)
  The amount of parameters

  *method* (**Delegate**)
  a delegate to a c# method **SbsSW.SwiPlCs.Callback**

- **Return Value**

  **Boolean**
  true if registration succeed otherwise false

- **Examples**

  For an example see **DelegateParameterVarArgs**

  by Uwe Lesta, SBS-Softwaresysteme GmbH
  Send comments on this topic to **Lesta at SBS-Softwaresysteme.de**

  Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
  (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

SetStreamFunctionRead Method (streamType, function)

SwiPlCs interface ► SbsSW.SwiPlCs ► PIEngine ►
SetStreamFunctionRead(PlStreamType, DelegateStreamReadStreamFunction)

TODO

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static void SetStreamFunctionRead(PlStreamType streamType, DelegateStreamReadStreamFunction function)</td>
<td>Public Shared Sub SetStreamFunctionRead (streamType As PlStreamType, function As DelegateStreamReadStreamFunction)</td>
<td>public: static void SetStreamFunctionRead(PlStreamType streamType, DelegateStreamReadStreamFunction^ function)</td>
<td>static member SetStreamFunctionRead : streamType : PlStreamType * function : DelegateStreamReadStreamFunction</td>
</tr>
</tbody>
</table>

- Parameters

*streamType* (PlStreamType)
Determine which stream to use `PlStreamType` function

A `DelegateStreamReadFunction` example:

```csharp
private const string ValidationStringRead = "hello_dotnet_world_

static internal long Sread(IntPtr handle, IntPtr)
{
    const string s = ValidationStringRead + "\n";
    byte[] array = System.Text.Encoding.Unicode.GetBytes(s);
    System.Runtime.InteropServices.Marshal.Copy(array,
    return array.Length;
}

[TestMethod]
public void StreamRead()
{
    var rf = new DelegateStreamReadFunction(S
    PlEngine.SetStreamFunctionRead(PlStreamTy/// NOTE: read/1 needs a dot (".") at the
    PlQuery.PlCall("assert( (test_read(A) :-
    PlTerm t = PlQuery.PlCallQuery("test_read
    Assert.AreEqual(ValidationStringRead, t.T
```
(1.1.60605.0)
Swi-plCs is a C# class library to connect .NET languages with SWI-Prolog.

**SetStreamFunctionWrite Method**

(setStreamType, function)

This is a primitive approach to enter the output from a stream.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static void SetStreamFunctionWrite(PlStreamType streamType, DelegateStreamWriteFunction function)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Public Shared Sub SetStreamFunctionWrite (streamType As PlStreamType, function As DelegateStreamWriteFunction)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public: static void SetStreamFunctionWrite(PlStreamType streamType, DelegateStreamWriteFunction function)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>static member SetStreamFunctionWrite : streamType : PlStreamType * function : DelegateStreamWriteFunction</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

`streamType` ([PlStreamType](#)())
Determine which stream to use \texttt{PlStreamType} \texttt{function (DelegateStreamWriteFunction)}

\texttt{A DelegateStreamWriteFunction}

\section*{Examples}

\begin{verbatim}
static string _testString;

static long Swrite(IntPtr handle, string buff)
{
    string s = buffer.Substring(0, (int)buffersize);
    _testString = s;
    return buffersize;
}

[TestMethod]
public void StreamWrite()
{
    // NOTE: the Swrite function is only call
    const string validationString = "Hello .n"
    PlQuery.PlCall("assert( (test_write :- wr
    var wf = new DelegateStreamWriteFunction(PlEngine.SetStreamFunctionWrite(PlStreamT
    PlQuery.PlCall("test_write");
    Assert.AreEqual(validationString+"\r\n",
}
\end{verbatim}

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
The class PIFrame provides an interface to discard unused term-references as well as rewinding unifications (data-backtracking). Reclaiming unused term-references is automatically performed after a call to a C#-defined predicate has finished and returns control to Prolog. In this scenario PIFrame is rarely of any use.

This class comes into play if the top level program is defined in C# and calls Prolog multiple times. Setting up arguments to a query requires term-references and using PIFrame is the only way to reclaim them.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class PIFrame : IDisposable</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Public Class PIFrame
    Implements IDisposable

public ref class PIFrame : IDisposable

type PIFrame =
    class
        interface IDisposable
    end

### Members

<table>
<thead>
<tr>
<th>All Members</th>
<th>Constructors</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑ Public</td>
<td></td>
<td>☑ Instance</td>
</tr>
<tr>
<td>☑ Protected</td>
<td></td>
<td>☑ Static</td>
</tr>
<tr>
<td>Icon</td>
<td>Member</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>![Icon]</td>
<td>PlFrame()</td>
<td>Creating an instance of this class marks all term-references created afterwards to be valid only in the scope of this instance.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Dispose()</td>
<td>Implement IDisposable.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Finalize()</td>
<td>Reclaims all term-references created after constructing the instance. (Overrides Object.Finalize().)</td>
</tr>
<tr>
<td>![Icon]</td>
<td>GetHashCode()</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>![Icon]</td>
<td>GetType()</td>
<td>Gets the type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>![Icon]</td>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Rewind()</td>
<td>Discards all term-references and global-stack data created as well as undoing all unifications after the instance was created.</td>
</tr>
</tbody>
</table>
**ToString()**

Returns a string that represents the current object.

(Inherited from **Object**.)

**Remarks**


**Examples**

A typical use for PlFrame is the definition of C# methods that call Prolog and may be called repeatedly from C#. Consider the definition of assertWord(), adding a fact to word/1:

```csharp
void AssertWord2(string word)
{
    PlFrame fr = new PlFrame();
    PlTermV av = new PlTermV(1);
    av[0] = PlTerm.PlCompound("word", new PlT
    PlQuery q = new PlQuery("assert", av);
    q.NextSolution();
    q.Dispose();       // IMPORTANT ! never forge
    fr.Dispose();
}
```

Alternatively you can use

```csharp
void AssertWord(string word)
{
    using (PlFrame fr = new PlFrame())
    {
        PlTermV av = new PlTermV(1);
```
av[0] = PlTerm.PlCompound("word", new
using (PlQuery q = new PlQuery("asser
{
    q.NextSolution();
}

Caution: NOTE: in any case you have to destroy any query object used inside a PlFrame

Inheritance Hierarchy

Object
  ▼ PlFrame

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
## Constructor

SwiPlCs interface ➤ SbsSW.SwiPlCs ➤ PlFrame ➤ PlFrame()

Creating an instance of this class marks all term-references created afterwards to be valid only in the scope of this instance.

### Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public PlFrame()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Public Sub New</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td>PlFrame()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new:</td>
<td>unit -&gt; PlFrame</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Implement IDisposable.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>Language</th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>void Dispose()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public</td>
<td>Sub Dispose</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td>virtual void Dispose() sealed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstract</td>
<td>Dispose : unit -&gt; unit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>override</td>
<td>Dispose : unit -&gt; unit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Remarks**

Do not make this method virtual.

A derived class should not be able to override this method.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Finalize Method

SwiPlCs interface ► SbsSW.SwiPlCs ► PlFrame ► Finalize()

Reclaims all term-references created after constructing the instance.

■ Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected override void</td>
<td>Finalize()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protected Overrides</td>
<td>Sub Finalize</td>
<td></td>
<td></td>
</tr>
<tr>
<td>protected:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>virtual void</td>
<td>Finalize() override</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstract Finalize : unit</td>
<td>-&gt; unit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>override Finalize : unit</td>
<td>-&gt; unit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Discards all term-references and global-stack data created as well as undoing all unifications after the instance was created.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void Rewind()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub Rewind</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void Rewind()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member Rewind : unit -&gt; unit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
This class allows queries to prolog.

A query can be created by a string or by constructing compound terms see [Constructors](#) for details.

All resources and terms created by a query are reclaimed by [Dispose()](#). It is recommended to build a query in a using scope.

There are four possible opportunities to query Prolog:

<table>
<thead>
<tr>
<th>Query type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A <strong>static call</strong></td>
<td>To ask prolog for a proof. Return only true or false.</td>
</tr>
<tr>
<td>A <strong>PlCallQuery(String)</strong></td>
<td>To get the first result of a goal</td>
</tr>
<tr>
<td><strong>Construct</strong> a <strong>PlQuery</strong> object by a string.</td>
<td>The most convenient way.</td>
</tr>
<tr>
<td><strong>Construct</strong> a <strong>PlQuery</strong> object by compound terms.</td>
<td>The most flexible and fast (runtime) way.</td>
</tr>
</tbody>
</table>

For examples see [PlQuery(String)](#) and [PlQuery(String, PlTermV)](#)

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class PlQuery : IDisposable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Class PlQuery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implements IDisposable</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```csharp
public ref class PlQuery : IDisposable

type PlQuery =
    class
        interface IDisposable
    end
end
```

## Members

<table>
<thead>
<tr>
<th>All Members</th>
<th>Constructors</th>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td></td>
<td>Instance</td>
<td>Static</td>
</tr>
<tr>
<td>Protected</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Icon | Member | Description
---|--------|--------------------------------------------------
<p>| ![Icon] | <strong>PlQuery(String)</strong> | With this constructor a query is created from a string. Uppercase parameters are interpreted as variables but can't be nested in subterms. If you need a variable in a nested term use <strong>PlQuery(String, PlTermV)</strong>. See the examples for details. |
| ![Icon] | <strong>PlQuery(String, String)</strong> | locating the predicate in the named module. |
| ![Icon] | <strong>PlQuery(String, PlTermV)</strong> | Create a query where name defines the name of the predicate and av the argument vector. The arity is deduced from av. The predicate is located in... |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>the Prolog module user.</td>
<td></td>
</tr>
<tr>
<td>PLQuery(String, String, PITermV)</td>
<td>locating the predicate in the named module.</td>
</tr>
<tr>
<td>Args</td>
<td>Provide access to the Argument vector for the query</td>
</tr>
<tr>
<td>Dispose()</td>
<td>Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.</td>
</tr>
<tr>
<td>Dispose(Boolean)</td>
<td>Release all resources from the query</td>
</tr>
</tbody>
</table>
| Equals(Object) | Determines whether the specified Object is equal to the current Object.  
(Inherited from Object.) |
| Finalize() | Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.  
(Overrides Object.Finalize().) |
| GetHashCode() | Serves as a hash function for a particular type.  
(Inherited from Object.) |
<p>| GetType() | Gets the type of the current instance. |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>NextSolution()</td>
<td>Provide the next solution to the query. Prolog exceptions are mapped to C# exceptions.</td>
</tr>
<tr>
<td>PlCall(String, PlTermV)</td>
<td>Create a query where name defines the name of the predicate and av the argument vector. The arity is deduced from av. The predicate is located in the Prolog module user.</td>
</tr>
<tr>
<td>PlCall(String, String, PlTermV)</td>
<td>As PlCall(String, PlTermV) but locating the predicate in the named module.</td>
</tr>
<tr>
<td>PlCall(String)</td>
<td>Call a goal once.</td>
</tr>
<tr>
<td>PlCallQuery(String)</td>
<td>NOTE: will be changed in the near future. Return the solution of a query which is called once by call. Throw an ArgumentException if there is no or more than one variable in the goal.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>PlCallQuery(String, String)</code></td>
<td>As <code>PlCallQuery(String)</code> but executed in the named module.</td>
</tr>
<tr>
<td><code>Query(PIQuerySwitch)</code></td>
<td>Obtain status information on the Prolog system. The actual argument type depends on the information required. The parameter <code>queryType</code> describes what information is wanted. Returning pointers and integers as a <code>long</code> is bad style. The signature of this function should be changed.</td>
</tr>
<tr>
<td><strong>Solutions</strong></td>
<td>Enumerate the solutions. For examples see <code>PlQuery(String)</code></td>
</tr>
<tr>
<td><strong>SolutionVariables</strong></td>
<td>Enumerate the <code>PIQueryVariables</code> of one solution.</td>
</tr>
<tr>
<td><strong>ToList()</strong></td>
<td>Create a <code>ReadOnlyCollection&lt;T&gt;</code> of <code>PIQueryVariables</code>. If calling <code>ToList()</code> all solutions of the query are generated and stored in the Collection.</td>
</tr>
</tbody>
</table>
### Remarks

The query will be opened by `NextSolution()` and will be closed if `NextSolution()` return false.

### Inheritance Hierarchy

- **Object**
  - **PlQuery**

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
With these constructors a Prolog query can be created but not opened. To get the results see `NextSolution()`.

A Query can be created from a string or by a name and `PlTermV`. The later is a native way and available for compatibility.

If a Query is created from a string representing arbitrary prolog text the helper classes `PlQueryVar` and `PlQueryVariables` comes into the game. In this case the most convenient way to get the results is to use `SolutionVariables` or `ToList()`.

For examples see `PlQuery(String)`.

### Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📊</td>
<td><code>PlQuery(String)</code></td>
<td>With this constructor a query is created from a string. Uppercase parameters are interpreted as variables but can't be nested in subterms. If you need a variable in a nested term use <code>PlQuery(String, PlTermV)</code>. See the examples for details.</td>
</tr>
<tr>
<td>📊</td>
<td><code>PlQuery(String, String)</code></td>
<td>locating the predicate in the named module.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><code>PlQuery(String, PlTermV)</code></td>
<td>Create a query where name defines the name of the predicate and av the argument vector. The arity is deduced from av. The predicate is located in the Prolog module user.</td>
<td></td>
</tr>
<tr>
<td><code>PlQuery(String, String, PlTermV)</code></td>
<td>locating the predicate in the named module.</td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
With this constructor a query is created from a string.

Uppercase parameters are interpreted as variables but can't be nested in subterms. If you need a variable in a nested term use `PlQuery(String, PlTermV)`. See the examples for details.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| `public` PlQuery(  
|       |    string `goal`  |
| )  |              |            |    |
| `Public Sub New (  
|       | `goal` As String  |
| )  |              |            |    |
| `public:`  
| `PlQuery(  
|       | `goal` As String  |
| )  |              |            |    |
| `new` :  
| `goal` : string -> PlQuery  |
Examples

```csharp
public void QueryStringForeach()
{
    string[] mm = { "aa", "bb", "cc" };
    var q = new PlQuery("member(A, [aa, bb, cc])");
    int i = 0;
    foreach (PlTermV s in q.Solutions)
    {
        Assert.AreEqual(mm[i++], s[0].ToString());
    }
    // or with named variables
    i = 0;
    foreach (PlQueryVariables s in q.SolutionVariables)
    {
        Assert.AreEqual(mm[i++], s["A"].ToString());
    }
}
```

This sample shows a query with two variables.

```csharp
public void QueryString2()
{
    var q = new PlQuery("append(A, B, [a,b,c])");
    Assert.IsTrue(q.NextSolution());
    Assert.AreEqual(["\[", q.Args[0].ToString());
    Assert.AreEqual(["[a,b,c]\], q.Args[1].ToString());
    Assert.IsTrue(q.NextSolution());
    Assert.AreEqual(["[a]\], q.Args[0].ToString());
```
And the same with named variables.

```csharp
public void QueryStringNamed()
{
    var q = new PlQuery("append(A, B, [a,b,c])
    Assert.IsTrue(q.NextSolution());
    Assert.AreEqual("["]	, q.Variables["A"].ToString());
    Assert.AreEqual("[a,b,c]", q.Variables["B"].ToString());
    Assert.IsTrue(q.NextSolution());
    Assert.AreEqual("[a]", q.Variables["A"].ToString());
    Assert.AreEqual("[b,c]", q.Variables["B"].ToString());
}
```

This sample shows what happens if the argument vector is used with compound terms.

```csharp
public void PlCallQueryCompound_string()
{
    string[] mm = { "comp(aa,aa1)", "comp(aa,
build_pred(); // create: test(comp(X,Y))
    var q = new PlQuery("test(comp(aa,X))")
    int i = 0;
    foreach (PlTermV s in q.Solutions)
    {
        Assert.AreEqual(mm[i++], s[0].ToString());
    }
```
And here how to get the results with named variables with compound terms.

```csharp
public void PlCallQueryCompoundNamed_string()
{
    string[] mm = { "aa1", "aa2", "aa3" };
    build_pred();  // create: test(comp(X,Y))
    var q = new PlQuery("test(comp(aa,X))");
    int i = 0;
    foreach (PlQueryVariables v in q.Solution)
    {
        Assert.AreEqual(mm[i++], v["X"].ToString());
    }
}
```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Constructor (name, termV)

Create a query where name defines the name of the predicate and av the argument vector. The arity is deduced from av. The predicate is located in the Prolog module user.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlQuery(</td>
<td>Public Sub New (</td>
<td>public: PlQuery(</td>
<td>new :</td>
</tr>
<tr>
<td>string name,</td>
<td>name As String,</td>
<td>String^ name,</td>
<td>name : string *</td>
</tr>
<tr>
<td>PlTermV termV</td>
<td>termV As PlTermV</td>
<td>PlTermV termV</td>
<td>termV : PlTermV -&gt; PlQuery</td>
</tr>
</tbody>
</table>

**Parameters**

*name* (*String*)
the name of the predicate

\( termV (PlTermV) \)

the argument vector containing the parameters

## Examples

This sample shows a query with a compound term as an argument.

```csharp
public void PlCallQueryCompound_termv()
{
    string[] mm = { "aa1", "aa2", "aa3" };
    build_pred();  // create: test(comp(X,Y)
    PlTerm var1 = PlTerm.PlVar();
    PlTerm comp = PlTerm.PlCompound("comp", new using (var q = new PlQuery("test", new Pl{

        int i = 0;
        foreach (PlTermV s in q.Solutions)
        {
            Assert.AreEqual(mm[i++], var1.ToS
            Assert.AreEqual(comp.ToString(),
        }
    }
}
```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog.

**Constructor (module, goal)**

SwiPICs interface ► SbsSW.SwiPICs ► PlQuery ► PlQuery(String, String)

locating the predicate in the named module.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public PlQuery(  
  string module,  
  string goal  
)  

Public Sub New (  
  module As String,  
  goal As String  
)  

public:  
PlQuery(  
  String^ module,  
  String^ goal  
)  

new :  
  module : string *  
  goal : string -> PlQuery

**Parameters**

*module* (String)

locating the predicate in the named module.
goal (String)
A string for a prolog query

Remarks
Muddy Waters sang: "I'am build for comfort, I ain't build for speed"

Examples

```csharp
public void QueryStringForeach()
{
    string[] mm = { "aa", "bb", "cc" };
    var q = new PlQuery("member(A, [aa, bb, c"
    int i = 0;
    foreach (PlTermV s in q.Solutions)
    {
        Assert.AreEqual(mm[i++], s[0].ToString);
    }
    // or with named variables
    i = 0;
    foreach (PlQueryVariables s in q.SolutionVariables)
    {
        Assert.AreEqual(mm[i++], s["A"].ToString);
    }
}
```

This sample shows a query with two variables.

```csharp
public void QueryString2()
{
    var q = new PlQuery("append(A, B, [a,b,c]
    Assert.IsTrue(q.NextSolution());
    Assert.AreEqual(""][", q.Args[0].ToString(0))
```
And the same with named variables.

```csharp
public void QueryStringNamed()
{
    var q = new PlQuery("append(A, B, [a,b,c])
    Assert.IsTrue(q.NextSolution());
    Assert.AreEqual([], q.Variables["A"].ToString());
    Assert.AreEqual(["a,b,c"], q.Variables["B"].ToString());
    Assert.IsTrue(q.NextSolution());
    Assert.AreEqual(["a"], q.Variables["A"].ToString());
    Assert.AreEqual(["b,c"], q.Variables["B"].ToString());
}
```

This sample shows what happens if the argument vector is used with compound terms.

```csharp
public void PlCallQueryCompound_string()
{
    string[] mm = { "comp(aa,aa1)", "comp(aa,aa2)" };
    Assert.IsTrue(build_pred()); // create: test(comp(X,Y)
    var q = new PlQuery("test(comp(aa,X))");
    int i = 0;
    foreach (PlTermV s in q.Solutions)
    {
    }
```
And here how to get the results with named variables with compound terms.

```csharp
public void PlCallQueryCompoundNamed_string()
{
    string[] mm = { "aa1", "aa2", "aa3" };
    build_pred();  // create: test(comp(X,Y))
    var q = new PlQuery("test(comp(aa,X))");
    int i = 0;
    foreach (PlQueryVariables v in q.SolutionVariables)
    {
        Assert.AreEqual(mm[i++], v["X"].ToString());
    }
}
```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Constructor (module, name, termV)**

SwiPLCs interface ► SbsSW.SwiPLCs ► PlQuery ► PlQuery(String, String, PlTermV)

locating the predicate in the named module.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlQuery(</td>
<td>Public Sub New (</td>
<td>public: PlQuery(</td>
<td>new:</td>
</tr>
<tr>
<td>string module,</td>
<td>module As String,</td>
<td>String^ module,</td>
<td>module : string *</td>
</tr>
<tr>
<td>string name,</td>
<td>name As String,</td>
<td>String^ name,</td>
<td>name : string *</td>
</tr>
<tr>
<td>PlTermV termV</td>
<td>termV As PlTermV</td>
<td>PlTermV termV</td>
<td>termV : PlTermV -&gt; PlQuery</td>
</tr>
<tr>
<td>)</td>
<td>)</td>
<td>)</td>
<td></td>
</tr>
</tbody>
</table>
Parameters

module (String)
locating the predicate in the named module.

name (String)
the name of the predicate

termV (PlTermV)
the argument vector containing the parameters

Examples

This sample shows a query with a compound term as an argument.

```java
public void PlCallQueryCompound_termv()
{
    string[] mm = { "aa1", "aa2", "aa3" };
    build_pred();  // create: test(comp(X,Y)
    PlTerm var1 = PlTerm.PlVar();
    PlTerm comp = PlTerm.PlCompound("comp", new
    using (var q = new PlQuery("test", new Pl
    {
        int i = 0;
        foreach (PlTermV s in q.Solutions) {
            Assert.AreEqual(mm[i++], var1.ToS
            Assert.AreEqual(comp.ToString(),
        }
    }
}
```
Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Property**

SwiPICs interface ► SbsSW.SwiPICs ► PlQuery ► Args

Provide access to the Argument vector for the query

- **Declaration Syntax**

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public PlTermV Args { get; }</td>
<td>Public ReadOnly Property Args As PlTermV Get</td>
<td>public: property PlTermV Args { PlTermV get (); }</td>
<td>member Args : PlTermV with get</td>
</tr>
</tbody>
</table>

- **Property Value**

`PlTermV`

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
## Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📦</td>
<td>Dispose()</td>
<td>Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.</td>
</tr>
<tr>
<td>🎨</td>
<td>Dispose(Boolean)</td>
<td>Release all resources from the query</td>
</tr>
</tbody>
</table>

---

by Uwe Lesta, SBS-Softwaresysteme GmbH  
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Dispose Method**

SwiPlCs interface ► SbsSW.SwiPlCs ► PLQuery ► Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void Dispose()</td>
<td>Public Sub Dispose</td>
<td>public: virtual void Dispose() sealed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>abstract Dispose : unit -&gt; unit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>override Dispose : unit -&gt; unit</td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Dispose Method (disposing)**

SwiPlCs interface ➤ SbsSW.SwiPlCs ➤ PlQuery ➤
Dispose(Boolean)

Release all resources from the query

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| protected virtual void Dispose(
  bool disposing
) |                     |            |             |
| Protected Overridable Sub Dispose (  
  disposing As Boolean  
) |                     |            |             |
| protected: virtual void Dispose(
  bool disposing
) |                     |            |             |
| abstract Dispose :  
  disposing : bool -> unit |                     |            |             |
| override Dispose :  
  disposing : bool -> unit |                     |            |             |

**Parameters**

*disposing (Boolean)*

if true all is deleted

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Method**

SwiPlCs interface ► SbsSW.SwiPlCs ► PlQuery ► **Finalize()**

Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected override void Finalize()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protected Overrides Sub Finalize</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>protected: virtual void Finalize() override</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstract Finalize : unit -&gt; unit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>override Finalize : unit -&gt; unit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
**Swi-cs-pl** - A CSharp class library to connect .NET languages with SWI-Prolog

**NextSolution Method**

SwiPICs interface ► SbsSW.SwiPICs ► PIQuery ► NextSolution()

Provide the next solution to the query. Prolog exceptions are mapped to C# exceptions.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool NextSolution()</td>
<td>Public Function</td>
<td>public:</td>
<td>member NextSolution:</td>
</tr>
<tr>
<td></td>
<td>NextSolution As</td>
<td>bool</td>
<td>unit -&gt; bool</td>
</tr>
<tr>
<td></td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Return Value**

  Boolean

  return true if successful and false if there are no (more) solutions.

- **Remarks**

  If the query is closed it will be opened. If the last solution was generated the query will be closed.

  If an exception is thrown while parsing (open) the query the _qid is set to zero.

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIException</td>
<td>Is thrown if SWI-Prolog Manual PL_next_solution() returns false</td>
</tr>
</tbody>
</table>
The main purpose of the static PlCall methods is to call a prolog prove or to do some site effects.

**Examples**

```csharp
Assert.IsTrue(PlQuery.PlCall("is_list", new PlTerm(...)))
```

```csharp
Assert.IsTrue(PlQuery.PlCall("consult", new PlTerm(...)))
```

// or

```csharp
Assert.IsTrue(PlQuery.PlCall("consult('some_file_name')", new PlTerm(...)))
```

---

**Members**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="SwiPlCs interface" /></td>
<td>PlCall(String, PlTermV)</td>
<td>Create a query where name defines the name of the predicate and av the argument vector. The arity is deduced from av. The predicate is located in the Prolog module user.</td>
</tr>
<tr>
<td><img src="image" alt="SwiPlCs interface" /></td>
<td>PlCall(String, String, PlTermV)</td>
<td>As PlCall(String, PlTermV) but locating the predicate in the named module.</td>
</tr>
<tr>
<td><img src="image" alt="SwiPlCs interface" /></td>
<td>PlCall(String)</td>
<td>Call a goal once.</td>
</tr>
</tbody>
</table>
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlCall**

Method (goal)

SwiPlCs interface ➤ SbsSW.SwiPlCs ➤ PlQuery ➤ PlCall(String)

Call a goal once.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static bool PlCall(</code></td>
<td><code>Public Shared Function PlCall (</code></td>
<td><code>public:</code></td>
<td><code>public: static bool PlCall(</code></td>
</tr>
<tr>
<td><code>    string goal</code></td>
<td><code>    goal As String</code></td>
<td><code>    String</code>^ <code>goal</code> ^</td>
<td><code>    static member PlCall :</code></td>
</tr>
<tr>
<td>`)</td>
<td><code>) As Boolean</code></td>
<td></td>
<td><code>goal : string -&gt; bool</code></td>
</tr>
</tbody>
</table>

- **Parameters**

  `goal (String)`  
  The complete goal as a string

- **Return Value**

  `Boolean`  
  Return true or false as the result of NextSolution() or throw an exception

- **Remarks**
Create a PlQuery from the arguments, generates the first solution by NextSolution() and destroys the query.

- **Examples**

  ```csharp
  Assert.IsTrue(PlQuery.PlCall("is_list([a,b,c,d])")
  
  Assert.IsTrue(PlQuery.PlCall("consult('some_file_name')")
  ```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Create a query where name defines the name of the predicate and av the argument vector. The arity is deduced from av. The predicate is located in the Prolog module user.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public static bool PlCall(  
  string predicate,
  PlTermV args
)  |
| Public Shared Function PlCall (  
  predicate As String,
  args As PlTermV
) As Boolean |
| public:
  static bool PlCall(  
    String^ predicate,
    PlTermV args
  ) |
| static member PlCall :  
  predicate : string *
  args : PlTermV -> bool |

### Parameters

*predicate* (String)
defines the name of the predicate

**args** ([PlTermV](#))
Is a [PlTermV](#) of arguments for the predicate

- **Return Value**
  
  **Boolean**
  Return true or false as the result of NextSolution() or throw an exception.

- **Remarks**
  
  Create a PlQuery from the arguments, generates the first solution by NextSolution() and destroys the query.

- **Examples**
  
  This sample shows a query with a compound term as an argument.

```csharp
public void PlCallQueryCompound_termv()
{
    string[] mm = { "aa1", "aa2", "aa3" };
    build_pred();  // create: test(comp(X,Y))
    PlTerm var1 = PlTerm.PlVar();
    PlTerm comp = PlTerm.PlCompound("comp", n
    using (var q = new PlQuery("test", new Pl
    {
        int i = 0;
        foreach (PlTermV s in q.Solutions)
        {
            Assert.AreEqual(mm[i++], var1.ToString());
            Assert.AreEqual(comp.ToString(),
        }
    })
}
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Method (module, predicate, args)

As `PlCall(String, PlTermV)` but locating the predicate in the named module.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public static bool PlCall(  
  string module,  
  string predicate,  
  PlTermV args  
) | Public Shared Function PlCall (  
  module As String,  
  predicate As String,  
  args As PlTermV  
) As Boolean | public:  
static bool PlCall(  
  String^ module,  
  String^ predicate,  
  PlTermV args  
) | static member PlCall :  
  module : string *  
  predicate : string *  
  args : PlTermV -> bool |
- **Parameters**

  - **module (String)**
    locating the predicate in the named module.
  - **predicate (String)**
    defines the name of the predicate
  - **args (PlTermV)**
    is a *PlTermV* of arguments for the predicate

- **Return Value**

  - **Boolean**
  Return true or false as the result of NextSolution() or throw an exception.

- **Remarks**

  Create a PIQuery from the arguments, generates the first solution by NextSolution() and destroys the query.

- **Examples**

  This sample shows a query with a compound term as an argument.

```java
public void PlCallQueryCompound_termv() {
    string[] mm = { "aa1", "aa2", "aa3" };
    build_pred(); // create: test(comp(X,Y)
    PlTerm var1 = PlTerm.PlVar();
    PlTerm comp = PlTerm.PlCompound("comp", n
    using (var q = new PlQuery("test", new Pl
    {
        int i = 0;
        foreach (PlTermV s in q.Solutions)
        {
```
Assert.AreEqual(mm[i++], var1.ToString());

```csharp
Assert.AreEqual(comp.ToString(),
```
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlCallQuery Method**

**SwiPICs interface** ► **SbsSW.SwiPICs** ► **PlQuery** ► **PlCallQuery**

### Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S]</td>
<td><strong>PlCallQuery(String)</strong></td>
<td>NOTE: will be changed in the near future. Return the solution of a query which is called once by call. Throw an ArgumentException if there is no or more than one variable in the goal.</td>
</tr>
<tr>
<td>![S]</td>
<td><strong>PlCallQuery(String, String)</strong></td>
<td>As <strong>PlCallQuery(String)</strong> but executed in the named module.</td>
</tr>
</tbody>
</table>

---

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
SwiPlCs interface ► SbsSW.SwiPlCs ► PlQuery ►

PlCallQuery(String)

NOTE: will be changed in the near future.

return the solution of a query which is called once by call. Throw an ArgumentException if there is no or more than one variable in the goal.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>Language</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>C#</td>
<td>public static PlTerm PlCallQuery( string goal )</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>Public Shared Function PlCallQuery ( goal As String ) As PlTerm</td>
</tr>
<tr>
<td>Visual C++</td>
<td>public: static PlTerm PlCallQuery( String^ goal )</td>
</tr>
<tr>
<td>F#</td>
<td>static member PlCallQuery : goal : string -&gt; PlTerm</td>
</tr>
</tbody>
</table>

- **Parameters**

  goal (String)
  a goal with *one* variable

- **Return Value**
**pTerm**
the bound variable of the first solution

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Prolog**

**PlCallQuery Method (module, goal)**

SwiPlCs interface ▶ SbsSW.SwiPlCs ▶ PlQuery ▶

PlCallQuery(String, String)

As PlCallQuery(String) but executed in the named module.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static PlTerm PlCallQuery(</code></td>
<td><code>Public Shared Function PlCallQuery (</code></td>
<td><code>public:</code></td>
<td><code>static</code></td>
</tr>
<tr>
<td><code>     string module,</code></td>
<td><code>     module As String,</code></td>
<td><code>static</code></td>
<td><code>static member</code></td>
</tr>
<tr>
<td><code>     string goal</code>)`</td>
<td><code>     goal As String</code></td>
<td><code>)</code></td>
<td><code>:</code></td>
</tr>
</tbody>
</table>

**Parameters**

- **module (String)**
  - The modulename in which the query is executed

- **goal (String)**
a goal with *one* variable

- **Return Value**
  
  `PlTerm`
  
  the bound variable of the first solution

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Method (queryType)**

SwiPLCs interface ➤ SbsSW.SwiPLCs ➤ PIQuery ➤

Query(PIQuerySwitch)

Obtain status information on the Prolog system. The actual argument type depends on the information required. The parameter `queryType` describes what information is wanted.

Returning pointers and integers as a long is bad style. The signature of this function should be changed.

**PIQuerySwitch**

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public static long Query(
            PIQuerySwitch queryType
        )` | `Public Shared Function Query (  
            queryType As PIQuerySwitch
        ) As Long` | `public:
        static long long Query(
            PIQuerySwitch queryType
        )` | `static member Query :  
            queryType : PIQuerySwitch -> int64` |

- **Parameters**
queryType (PlQuerySwitch)
A PlQuerySwitch.

Return Value
Int64
A int depending on the given queryType

Examples
This sample shows how to get SWI-Prologs version number

```csharp
public void Pl_query_version()
{
    long v = PlQuery.Query(PlQuerySwitch.Version);
    Assert.AreEqual(60605, v, "SWI-Prolog version number by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de
Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)```
Property
SwiPlCs interface ► SbsSW.SwiPlCs ► PIQuery ► Solutions

Enumerate the solutions.

For examples see PIQuery(String)

_declarationSyntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public IEnumerable&lt;PlTermV&gt; Solutions { get; }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public ReadOnly Property Solutions As IEnumerable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: property IEnumerable&lt;PlTermV&gt;^ Solutions { IEnumerable&lt;PlTermV&gt;^ get (); }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member Solutions : IEnumerable&lt;PlTermV&gt; with</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

_propertyValue
IEnumerable<PlTermV>

_seeAlso
NextSolution() Constructor

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
Enumerate the `PlQueryVariables` of one solution.

## Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public <code>IEnumerable&lt;PlQueryVariables&gt;</code> SolutionVariables</td>
<td>Public ReadOnly Property <code>SolutionVariables</code> As Get</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: property <code>IEnumerable&lt;PlQueryVariables&gt;^</code>^ <code>SolutionVariables</code> As Get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>member SolutionVariables : IEnumerable&lt;PlQueryVariables&gt;</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Property Value

`IEnumerable<PlQueryVariables>`

## Examples

```csharp
public void TestCompoundQuery()
{
    string[] refValues = { "gloria", "melanie" }
    using (new PlFrame())
    {
        PlQuery.PlCall("assert(father(uwe, gl")
    }
}
PlQuery.PlCall("assert(father(uwe, me)
PlQuery.PlCall("assert(father(uwe, ay
var plq = new PlQuery("father(P,C), a
int i = 0;
foreach (PlQueryVariables vars in plq
{
    Assert.AreEqual("uwe", (string)vars
    Assert.AreEqual(refValues[i++], (s
})

See Also
NextSolution()

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**ToList**

Create a `ReadOnlyCollection<>` of `PlQueryVariables`.

If calling `ToList()` all solutions of the query are generated and stored in the collection.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public</code> <code>ReadOnlyCollection&lt;PlQueryVariables&gt; ToList();</code></td>
<td><code>Public Function ToList As ReadOnlyCollection(PlQueryVariables) ;</code></td>
<td><code>public:</code> <code>ReadOnlyCollection&lt;PlQueryVariables&gt;^&gt;^ ToList;</code></td>
<td><code>member ToList : unit -&gt; ReadOnlyCollection&lt;PlQueryVariables&gt;;</code></td>
</tr>
</tbody>
</table>

- **Return Value**

  `ReadOnlyCollection<PlQueryVariables>`

  A `ReadOnlyCollection` of `PlQueryVariables` containing all solutions of the query.

- **Examples**

  ```csharp
  public void Test_multi_goal_ToList()
  {
    var results = from n in new PlQuery("L=[a,b,c], member(A,L)"
    select new { A = (string)n[0] );
  ```
int i = 0;
foreach (var t in results)
    Assert.AreEqual(_abc[i++], t.A);

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

VariableNames Property

SwiPICs interface ► SbsSW.SwiPICs ► PIQuery ► VariableNames

C#

Gets a Collection<T> of the variable names if the query was built by a string.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>

- Property Value

Collection<String>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Property

SwiPIcs interface ► SbsSW.SwiPIcs ► PIQuery ► Variables

The List of PIQueryVariables of this PIQuery.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>PlQueryVariables</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variables { get; }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public</td>
<td>ReadOnly Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variables As PIQuery</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Get</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td>property</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlQueryVariables^</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variables {</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlQueryVariables^</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get ()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>member</td>
<td>Variables : PIQueryVariables with get</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Property Value

PIQueryVariables

- Examples

In the following example you see how the query Variables can be used to set a variable.

```csharp
public void TestCompoundQueryWithVariables()
{
    string[] refValues = { "x_a", "x_b", "x_c"
    using (new PlFrame())
```
Here is a more complex sample where the variables of two queries are connected.

```csharp
public void TestCompoundNestedQueryWithVariables()
{
    string[] refValues = { "x_a", "x_b", "x_c"
    string[] refValuesInner = { "a1", "a2", "b1"
    int innerIdx = 0;
    using (new PlFrame())
    {
        var plq = new PlQuery("member(X, [a,b,c]), atomic_list_concat([P,X], L)"
            .Variables["P"].Unify("x_");
        int i = 0;
        foreach (PlQueryVariables vars in plq)
        {
            Assert.AreEqual("x_", (string)vars["P");
            Assert.AreEqual(refValues[i++], (string)vars["P");
        }
    }
}
```
var results = from n in q.ToList()
foreach (var v in results)
{
    Assert.AreEqual(refValuesInneree[innerIdx++], v.L);
}

See Also
- PlQueryVariables

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlQuerySwitch Enumeration**

Flags that control for the foreign predicate parameters


### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public enum PlQuerySwitch</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Enumeration PlQuerySwitch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public enum class PlQuerySwitch</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>type</strong> PlQuerySwitch</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Members

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>The default value.</td>
</tr>
<tr>
<td>Argc</td>
<td>Return an integer holding the number of arguments given to Prolog from Unix.</td>
</tr>
<tr>
<td>Argv</td>
<td>Return a char ** holding the argument vector given to Prolog from Unix.</td>
</tr>
<tr>
<td>GetChar</td>
<td>Read character from terminal.</td>
</tr>
<tr>
<td>MaxInteger</td>
<td>Return a long, representing the maximal integer value represented by a Prolog integer.</td>
</tr>
<tr>
<td>MinInteger</td>
<td>Return a long, representing the minimal integer value.</td>
</tr>
<tr>
<td>Version</td>
<td>Return a long, representing the version as a number.</td>
</tr>
</tbody>
</table>
10,000 \times M + 100 \times m + p, \text{ where } M \text{ is the major, } m \text{ the minor version number and } p \text{ the patch-level. For example, 20717 means 2.7.17}

| **MaxThreads** | Return the maximum number of threads that can be created in this version. Return values of PL_thread_self() are between 0 and this number. |
| **Encoding** | Return the default stream encoding of Prolog (of type IOENC). |
| **UserCpu** | Get amount of user CPU time of the process in milliseconds. |

- **See Also**
  - Query(PlQuerySwitch)

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

### PiQueryVar Class

**SwiPICS interface ➤ SbsSW.SwiPICS ➤ PiQueryVar**

Represents one variable of a Query result.

#### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public class PiQueryVar</code></td>
<td><code>Public Class PiQueryVar</code></td>
<td><code>public ref class PiQueryVar</code></td>
<td><code>type PiQueryVar = class end</code></td>
</tr>
</tbody>
</table>

#### Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="Public" /> <img src="icon" alt="Protected" /></td>
<td><code>Equals(Object)</code></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><img src="icon" alt="Public" /> <img src="icon" alt="Protected" /></td>
<td><code>Finalize()</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><img src="icon" alt="Public" /> <img src="icon" alt="Protected" /></td>
<td><code>GetHashCode()</code></td>
<td>Serves as a hash function for a</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>GetType()</code></td>
<td>Gets the type of the current instance. (Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>MemberwiseClone()</code></td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
<td></td>
</tr>
<tr>
<td><code>Name</code></td>
<td>The name of a variable in a Query</td>
<td></td>
</tr>
<tr>
<td><code>ToString()</code></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>Value</code></td>
<td>The Value (PlTerm) of a variable in a Query</td>
<td></td>
</tr>
</tbody>
</table>

### Inheritance Hierarchy

- **Object**
  - **PlQueryVar**

by Uwe Lesta, SBS-Softwaresysteme GmbH  
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
The name of a variable in a Query

## Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public string Name { get; internal set; }</code></td>
<td><code>Public Property Name As String Get Friend Set</code></td>
<td><code>public: property String^ Name { String^ get (); internal: void set (String^ value); }</code></td>
<td><code>member Name : string with get, internal set</code></td>
</tr>
</tbody>
</table>

## Property Value

String

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
The Value (PlTerm) of a variable in a Query

### Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C#</strong></td>
<td><code>public PlTerm Value { get; internal set; }</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td><code>Public Property Value As PlTerm</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td><code>Get</code></td>
<td><code>Friend Set</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td><code>public:</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td>`property PlTerm Value {</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td><code>    PlTerm get ();</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td><code>   internal: void set (PlTerm</code>value<code>);</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td><code>}</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C#</strong></td>
<td><code>member Value : PlTerm with get, internal set</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Property Value

**PlTerm**

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
PIQueryVariables Class

Represents the set variables of a Query if it was created from a string.

This class is also used to represent the results of a PLQuery after `ToList` or `SolutionVariables` was called.

### Declaration Syntax

<table>
<thead>
<tr>
<th>Language</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>C#</td>
<td><code>public class PlQueryVariables</code></td>
</tr>
<tr>
<td>Visual Basic</td>
<td><code>Public Class PlQueryVariables</code></td>
</tr>
<tr>
<td>Visual C++</td>
<td><code>public ref class PlQueryVariables</code></td>
</tr>
<tr>
<td>F#</td>
<td><code>type PlQueryVariables = class end</code></td>
</tr>
</tbody>
</table>

### Members

<table>
<thead>
<tr>
<th>All Members</th>
<th>Constructors</th>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td>Instance</td>
<td>Static</td>
<td></td>
</tr>
<tr>
<td>Protected</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Icon | Member | Description
---|--------|--------------------------------------------------
[ ] | `PlQueryVariables()` | Initializes a new instance of the `PlQueryVariables` class |
[ ] | `Count` | Returns the number of elements in the sequence. (Defined by `Enumerable.ToList<PlQueryVar>`) |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals(Object)</code></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Finalize()</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode()</code></td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetType()</code></td>
<td>Gets the type of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Item[String]</code></td>
<td>Gets the <code>PITerm</code> of the given variable name or throw an <code>ArgumentException</code>.</td>
</tr>
<tr>
<td><code>MemberwiseClone()</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>ToString()</code></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

**Examples**

This sample shows both `Variables` is used to unify the variables of two nested queries and the result
public void TestCompoundNestedQueryWithVariables3()
{
    string[] refValuesInner = { "_gap_abc1" },
    int innerIdx = 0;
    using (new PlFrame())
    {
        try
        {
            var outerQuery = new PlQuery("append(A,B,C),
                                             atomic_list_concat(A,
                                             A1),
                                             atomic_list_concat(B,
                                             B1)"
                                             .Unify(outerQuery.Variables["C"].Unify(n
                                             var innerQuery = new PlQuery("member(Count, [1,2]),
                                             atomic_list_concat([X,Y,Z,Count], ATOM)"
                                             .Unify(o
                                             innerQuery.Variables["X"].Unify(y
                                             innerQuery.Variables["Y"].Unify(z
                                             innerQuery.Variables["Z"].Unify(o
                                             foreach (PlQueryVariables vars in
                                             {
                     var results = from n in inner
                                      select new { L = n["ATOM"
                                      foreach (var v in results)
            {
                Assert.AreEqual(refValuesInner[innerIdx++], v.L);
            }
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Trace.WriteLine(ex.Message);
        Assert.IsFalse(true);
    }
}
Inheritance Hierarchy

Object
  ↓ PiQueryVariables

See Also

Variables

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

PlQueryVariables Constructor

SwiPlCs interface ► SbsSW.SwiPlCs ► PlQueryVariables ►
PlQueryVariables()

Initializes a new instance of the PlQueryVariables class

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlQueryVariables()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlQueryVariables()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new : unit -&gt; PlQueryVariables</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Count**

### Property

SwiPICs interface ▶ SbsSW.SwiPICs ▶ PIQueryVariables ▶ Count

Returns the number of elements in the sequence. (Defined by Enumerable of List<PIQueryVar>.)

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public int Count { get; }` | `Public Readonly Property Count As Integer Get` | `public:
property int Count {
    int get ();
}

`member Count : int with get` |

### Property Value

Int32

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Leste@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

Item

Property (name)
SwiPlCs interface ► SbsSW.SwiPlCs ► PIQueryVariables ► Item[String]

Gets the PI Term of the given variable name or throw an ArgumentException.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public PI Term this[string name] { get; } | Public ReadOnly Default Property Item (name As String) As PI Term | public:
property PI Term default[String^ name] { PI Term get (String^ name); } | member Item : PI Term with get |

- Parameters

  name (String)
  The name of the variable

- Return Value

  PI Term
  The PI Term (value) of the variable
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Is thrown if the name is not the name of a variable.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlTerm**

**Structure**

SwiPlCs interface ► SbsSW.SwiPlCs ► PlTerm

struct plays a central role in conversion and operating on elements **IComparable** to support ordering in [!:System.Linq] Term is a List.

PlTerm can be done by the **Constructors** or by the following:

- PlVar(), PlCompound, PlString(), PlCodeList(), PlCharList() (see

### Declaration Syntax

<table>
<thead>
<tr>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>struct</em> PlTerm : IComparable, IEnumerable&lt;PlTerm&gt;, IEnumerable</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Structure</em> PlTerm</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Implements</em> IComparable, IEnumerable(Of PlTerm), IEnumerable</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>value class</em> PlTerm : IComparable,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEnumerable&lt;PlTerm&gt;, IEnumerable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| dAttribute>]
<p>| Term = |
| <em>uct</em> |
|  <em>interface</em> IComparable |
|  <em>interface</em> IEnumerable&lt;PlTerm&gt; |
|  <em>interface</em> IEnumerable |</p>
<table>
<thead>
<tr>
<th>Members</th>
<th>Constructors</th>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>✓ Instance ✓ Static</td>
<td>✓ Declared ✓ Inherited</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlTerm(String)</td>
<td>Creates a term-references holding a Prolog term representing text.</td>
</tr>
<tr>
<td>PlTerm(Int32)</td>
<td>Creates a term-references holding a Prolog integer representing value.</td>
</tr>
<tr>
<td>PlTerm(Double)</td>
<td>Creates a term-references holding a Prolog float representing value.</td>
</tr>
</tbody>
</table>
| Add(PlTerm) | Appends an element to a list by creating a new one and copy all elements  
Note: This is a slow version see my mail from Jan from 2007.11.06 14:44 |
<p>| AddList(PlTerm) | Appends a list ( PlTail ) to a list by creating a new one and copy all elements |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Append(PlTerm)</code></td>
<td>Appends element to the list and make the PlTail reference point to the new variable tail. If A is a variable, and this method is called on it using the argument &quot;gnat&quot;, a list of the form [gnat</td>
</tr>
<tr>
<td><code>arity()</code></td>
<td>Get the arity of the functor if <code>PlTerm</code> is a compound term.</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>Unifies the term with [] and returns the result of the unification.</td>
</tr>
<tr>
<td><code>CompareTo(Object)</code></td>
<td>Compares the current instance with another object of the same type and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other object.</td>
</tr>
<tr>
<td><code>Equality(PlTerm, PlTerm)</code></td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an</td>
</tr>
<tr>
<td>Equality(PlTerm, Int32)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same integer as the argument, FALSE if the conversion was not successful. conversion of the term is done by PL_get_long</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Equality(Int32, PlTerm)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>Equality(PlTerm, String)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>Equality(String, PlTerm)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>type_error</td>
<td>exception if the conversion failed.</td>
</tr>
<tr>
<td><code>Equals(Object)</code></td>
<td>Indicates whether this instance and a specified object are equal. (Overrides <code>ValueType.Equals(Object)</code>.)</td>
</tr>
<tr>
<td><code>Explicit (PlTerm to String)</code></td>
<td>Converts the Prolog argument into a string which implies Prolog atoms and strings are converted to the represented text or throw a <code>PlTypeException</code>.</td>
</tr>
<tr>
<td><code>Explicit (PlTerm to Int32)</code></td>
<td>Yields a int if the PlTerm is a Prolog integer or float that can be converted without loss to a int. Throws a <code>PlTypeException</code> exception otherwise.</td>
</tr>
<tr>
<td><code>Explicit (PlTerm to Double)</code></td>
<td>Yields the value as a C# double if PlTerm represents a Prolog integer or float. Throws a <code>PlTypeException</code> exception otherwise.</td>
</tr>
<tr>
<td><code>Finalize()</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetEnumerator()</td>
<td>Returns an enumerator that iterates through the collection.</td>
</tr>
<tr>
<td>GetHashCode()</td>
<td>Returns the hash code for this instance.</td>
</tr>
<tr>
<td></td>
<td>(Overrides ValueType.GetHashCode().)</td>
</tr>
<tr>
<td>GetType()</td>
<td>Gets the type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GreaterThan(PlTerm, m)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>GreaterThanOrEqual(PlTerm, m)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>Inequality(PlTerm, PlTerm)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>type_error</code></td>
<td>Exception if the conversion failed.</td>
</tr>
<tr>
<td><code>summary(PlTerm, Int32)</code></td>
<td>Return true if <code>PlTerm</code> is an atom.</td>
</tr>
<tr>
<td><code>summary(Int32, PlTerm)</code></td>
<td>Return true if <code>PlTerm</code> is atomic (not variable or compound).</td>
</tr>
<tr>
<td><code>summary(PlTerm, String)</code></td>
<td>Return true if <code>PlTerm</code> is a compound term. Note that a list is a compound term.</td>
</tr>
<tr>
<td><code>summary(String, PlTerm)</code></td>
<td>Return true if <code>PlTerm</code> is a float.</td>
</tr>
<tr>
<td><code>summary(PlTerm)</code></td>
<td>Return true if <code>PlTerm</code> is a ground term. See also <code>ground/1</code>. This function is cycle-safe.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>IsInitialized</td>
<td>return false for a PLTerm variable which is only declared and true if it is also Initialized</td>
</tr>
<tr>
<td>IsInteger</td>
<td>Return true if PLTerm is an integer.</td>
</tr>
<tr>
<td>IsCompound</td>
<td>Return true if PLTerm is a compound term with functor  */2 or the atom [].</td>
</tr>
<tr>
<td>IsNumber</td>
<td>Return true if PLTerm is an integer or float.</td>
</tr>
<tr>
<td>IsString</td>
<td>Return true if PLTerm is a string.</td>
</tr>
<tr>
<td>IsVariable</td>
<td>Return true if PLTerm is a variable</td>
</tr>
<tr>
<td>Int32</td>
<td>If PLTerm is compound and index is between 0 and Arity (including), the nth PLTerm is returned. If pos is 0 the functor of the term is returned (For a list '.').</td>
</tr>
</tbody>
</table>

See: PL_get_arg/3
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LessThan(PlTerm, PlTerm)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>LessThanOrEqual(PlTerm, PlTerm)</td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Name()</td>
<td>Get a holding the name of the functor if PlTerm is a compound term.</td>
</tr>
<tr>
<td>NextValue()</td>
<td>return a PlTerm bound to the next element of the list PlTail and advance PlTail. Returns the element on success or a free PlTerm (Variable) if PlTail represents the empty list. If PlTail is neither a list nor the empty list, a</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PlCharList(String)</td>
<td>Create a Prolog list of one-character atoms from a C#-string.</td>
</tr>
<tr>
<td>PlCodeList(String)</td>
<td>Create a Prolog list of ASCII codes from a 0-terminated C-string.</td>
</tr>
<tr>
<td>PlCompound(String, PlTermV)</td>
<td>Create a compound term with the given name from the given vector of arguments. See PlTermV for details.</td>
</tr>
<tr>
<td>PlCompound(String, PlTerm)</td>
<td>Create a compound term with the given name and the arguments</td>
</tr>
<tr>
<td>PlCompound(String, PlTerm, PlTerm)</td>
<td>Create a compound term with the given name and the arguments</td>
</tr>
<tr>
<td>PlCompound(String, PlTerm, PlTerm, PlTerm)</td>
<td>Create a compound term with the given name and the arguments</td>
</tr>
<tr>
<td>PlString(String)</td>
<td>A SWI-Prolog string represents a byte-string on the global stack. It's</td>
</tr>
</tbody>
</table>
Lifetime is the same as for compound terms and other data living on the global stack. Strings are not only a compound representation of text that is garbage-collected, but as they can contain 0-bytes, they can be used to contain arbitrary C-data structures.

A SWI-Prolog string represents a byte-string on the global stack. It's lifetime is the same as for compound terms and other data living on the global stack. Strings are not only a compound representation of text that is garbage-collected, but as they can contain 0-bytes, they can be used to contain arbitrary C-data structures.

PITail is for analysing and constructing lists. It is called PITail as enumeration-steps make the term-reference follow the `tail' of the list.

A PITail is created by making a new term-reference pointing to the same object. As PITail is used to enumerate or build a Prolog list, the initial list term-reference keeps pointing to the head of the list.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PlType</code></td>
<td>Get the <strong>PlType</strong> of a <strong>PlTerm</strong>.</td>
</tr>
<tr>
<td><code>()</code></td>
<td>Creates a new initialised term (holding a Prolog variable).</td>
</tr>
<tr>
<td><code>t()</code></td>
<td>Converts to a strongly typed <strong>ReadOnlyCollection</strong> of <strong>PlTerm</strong> objects that can be accessed by index</td>
</tr>
<tr>
<td><code>toString()</code></td>
<td>Converts to a strongly typed <strong>Collection</strong> of strings that can be accessed by index</td>
</tr>
<tr>
<td><code>ToString()</code></td>
<td>If <strong>PlTerm</strong> is a list the string is build by calling <code>ToString()</code> for each element in the list separated by<code>,</code> and put the brackets around <code>[' ']'.</code> (Overrrides **Value<code>Type.ToString()</code>.)</td>
</tr>
<tr>
<td><code>ToStringCanonical()</code></td>
<td>Convert a <strong>PlTerm</strong> to a string by <strong>PL_get_chars/1</strong> with the CVT_WRITE_CANONICAL flag. If it fails <strong>PL_get_chars/3</strong> is called again with REP_MB flag.</td>
</tr>
<tr>
<td>(PITerm)</td>
<td>Unify a PITerm with a PITerm</td>
</tr>
<tr>
<td>(String)</td>
<td>Unify a PITerm with a PITerm</td>
</tr>
</tbody>
</table>

**static method**

**Description**

Creates a new initialised term (holding a Prolog variable).

- **PlTail(PlTerm)**
  - PITail is for analysing and constructing lists.

- **PlCompound(string)**
  - Create compound terms. E.g. by parsing (as read/1) the given text.

- **PlString(String)**
  - Create a SWI-Prolog string.

- **PlCodeList(String)**
  - Create a Prolog list of ASCII codes from a 0-terminated C-string.

- **PlCharList(String)**
  - Create a Prolog list of one-character atoms from a 0-terminated C-string.

---

Copyright by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
A new PITerm can be also created by the static methods:

<table>
<thead>
<tr>
<th>static method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PITVar()</td>
<td>Creates a new initialised term (holding a Prolog variable).</td>
</tr>
<tr>
<td>PITail(PITerm)</td>
<td>PITail is for analysing and constructing lists.</td>
</tr>
<tr>
<td>PICompound(string)</td>
<td>Create compound terms. E.g. by parsing (as read/1) the given text.</td>
</tr>
<tr>
<td>PISTring(String)</td>
<td>Create a SWI-Prolog string.</td>
</tr>
<tr>
<td>PICODEList(String)</td>
<td>Create a Prolog list of ASCII codes from a 0-terminated C-string.</td>
</tr>
<tr>
<td>PICCharList(String)</td>
<td>Create a Prolog list of one-character atoms from a 0-terminated C-string.</td>
</tr>
</tbody>
</table>

- **Members**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🧮</td>
<td>PITerm(String)</td>
<td>Creates a term-references holding a Prolog term representing text.</td>
</tr>
<tr>
<td>🧮</td>
<td>PITerm(Int32)</td>
<td>Creates a term-references holding a Prolog integer representing value.</td>
</tr>
<tr>
<td>🧮</td>
<td>PITerm(Double)</td>
<td>Creates a term-references holding a Prolog float representing value.</td>
</tr>
</tbody>
</table>
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Constructor (value)**

SwiPlCs interface ➤ SbsSW.SwiPlCs ➤ PlTerm ➤ PlTerm(Double)

Creates a term-references holding a Prolog float representing value.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public PlTerm(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>double value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value As Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>double value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value : float -&gt; PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Parameters**

  `value` (**Double**)

  a double value

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Constructor (value)

Creates a term-references holding a Prolog integer representing value.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlTerm(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value As Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value : int -&gt; PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Parameters**

  - **value** (**Int32**)  
    a integer value

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlTerm Constructor (text)**

SwiPICs interface ★ SbsSW.SwiPICs ★ PlTerm ★ PlTerm(String)

Creates a term-references holding a Prolog term representing text.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlTerm(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>string text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text As String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>String^ text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text : string -&gt; PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Parameters**

  **text (String)**
  
  the text

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Add Method (term)

SwiPlCs interface ► SbsSW.SwiPlCs ► PlTerm ► Add(PlTerm)

Apssends an element to a list by creating a new one and copy all elements Note This is a slow version see my mail from Jan from 2007.11.06 14:44

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm</td>
<td>term</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public</td>
<td>Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>term</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>term</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>As</td>
<td>PlTerm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>PlTerm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member</td>
<td>Add</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>PlTerm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-&gt;</td>
<td>bool</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Parameters

  term (PlTerm)
a closed list

- Return Value

  Boolean
True if Succeed
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**AddList**

**Method (listToAppend)**

SwiPIcs interface ➤ SbsSW.SwiPIcs ➤ PlTerm ➤ AddList(PlTerm)

Appends a list (PlTail) to a list by creating a new one and copy all elements

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public bool AddList(
    PlTerm *listToAppend*
) |
| Public Function AddList (
    *listToAppend* As PlTerm |
) As Boolean |
| public:
    bool AddList(
        PlTerm *listToAppend*
    ) |
| member AddList :
    *listToAppend* : PlTerm -> bool |

- **Parameters**

  *listToAppend* *(PlTerm)*
  a closed list

- **Return Value**

  Boolean
  True if Succeed
Examples

```csharp
public void List_Add_list_doc()
{
    PlTerm t = new PlTerm("[x,y]");
    PlTerm l = new PlTerm("[a,b]");

    Assert.IsTrue(l.IsList);
    Assert.IsTrue(l.AddList(t));
    Assert.IsTrue(l.IsList);
    Assert.AreEqual("[a,b,x,y]", l.ToString());
    Assert.AreEqual("a", l.NextValue().ToString());
    Assert.AreEqual("b", l.NextValue().ToString());
    Assert.AreEqual("[x,y]", l.ToString());
}
```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Append

Method (term)

SwiPlCs interface ► SbsSW.SwiPlCs ► PlTerm ► Append(PlTerm)

Applies to the list and makes the PITail reference point to the new variable tail. If A is a variable, and this method is called on it using the argument "gnat", a list of the form [gnat|B] is created and the PITail object now points to the new variable B. This method returns TRUE if the unification succeeded and FALSE otherwise. No exceptions are generated.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| ```
public bool Append(
    PlTerm term
)
``` |
| ```
Public Function Append (  
    term As PlTerm  
) As Boolean  
``` |
| ```
public:
bool Append(  
    PlTerm term
)
``` |
| ```
member Append :  
    term : PlTerm -> bool  
``` |

- **Parameters**

`term (PlTerm)`

The PlTerm to append on the list.
- **Return Value**
  
  **Boolean**
  
  true if successful otherwise false

- **Examples**

  ```csharp
  public void List_Append()
  {
      PlTerm t = PlTerm.PlVar();
      PlTerm l = PlTerm.PlTail(t);

      Assert.IsTrue(l.Append(new PlTerm("one")))
      Assert.IsTrue(l.Append(new PlTerm("two")))
      Assert.IsTrue(l.Append(new PlTerm("three"))

      Assert.AreEqual(1, l.Close());
      Assert.AreEqual(3, list_length(t));
      Assert.AreEqual("[one,two,three]", t.ToSt
  }
  ```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Property**

SwiPlCs interface ► SbsSW.SwiPlCs ► PlTerm ► Arity

Get the arity of the functor if PlTerm is a compound term.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>```c#</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public int Arity { get; }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public ReadOnly Property Arity As Integer Get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>property int Arity {</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int get ();</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member Arity : int with get</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Property Value**

Int32

- **Remarks**

Arity and Name are for compound terms only

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Is thrown if the term isn't compound</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Method**

SwiPlCs interface ➤ SbsSW.SwiPlCs ➤ PITerm ➤ Close()

Unifies the term with [] and returns the result of the unification.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public int Close()</td>
<td>Public Function Close As Integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: int Close()</td>
<td></td>
<td></td>
<td>member Close : unit -&gt; int</td>
</tr>
</tbody>
</table>

**Return Value**

Int32

The int value of PL_unify_nil(TermRef)

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**CompareTo Method (obj)**

**SwiPLCs interface** ▶ **SbsSW.SwiPLCs** ▶ **PITerm** ▶ **compareTo(Object)**

Compares the current instance with another object of the same type and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other object.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public int compareTo(
    Object *obj*
) |
| Public Function CompareTo ( |
    *obj* As Object
) As Integer |
| public:
  virtual int compareTo(
    Object^ *obj*
) sealed |
| abstract compareTo :
  *obj* : Object -> int |
| override compareTo :
  *obj* : Object -> int |

### Parameters

**obj (Object)**

An object to compare with this instance.
- **Return Value**

**Int32**
A 32-bit signed integer that indicates the relative order of the objects being compared. The return value has these meanings:
- Less than zero: This instance precedes `obj` in the sort order.
- Zero: This instance occurs in the same position in the sort order as `obj`.
- Greater than zero: This instance follows `obj` in the sort order.

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td><code>obj</code> is not the same type as this instance.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
## Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="SwiPICs interface" /></td>
<td><strong>Equality(PlTerm, PlTerm)</strong></td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td><img src="image" alt="SbsSW.SwiPICs" /></td>
<td><strong>Equality(PlTerm, Int32)</strong></td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same integer as the argument, FALSE if the conversion was not successful. Conversion of the term is done by PL_get_long.</td>
</tr>
<tr>
<td><img src="image" alt="PTerm" /></td>
<td><strong>Equality(Int32, PlTerm)</strong></td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td><img src="image" alt="Equality" /></td>
<td><strong>Equality(PlTerm, String)</strong></td>
<td>Yields TRUE if the PlTerm is an atom or string representing the same text as</td>
</tr>
</tbody>
</table>
the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

| ![Equality](String, PlTerm) | Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed. |

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Equality Operator**

**SwiPICs interface** ➔ **SbsSW.SwiPICs** ➔ **PlTerm** ➔ **Equality(PlTerm, PlTerm)**

Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator ==(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Public Shared Operator = (**

```vbnet
term1 As PlTerm,
term2 As PlTerm
```

) As Boolean

```csharp
due to
```
```csharp
public:
static bool operator ==(                        |
PlTerm term1,    |
PlTerm term2)    |
)          |
```

```fsharp
due to
```
```fsharp
static let inline (=
```
```fsharp
term1 : PlTerm *
term2 : PlTerm : bool
```

### Parameters

`term1` (**PlTerm**)
a PITerm

\textit{term2 (PITerm)}
a PITerm

- \textbf{Return Value}
  
  \texttt{Boolean}
  
  true or false

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to \texttt{Lesta at SBS-Softwaresysteme.de}

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Operator**

SwiPLCs interface ➤ SbsSW.SwiPLCs ➤ PITerm ➤ Equality(PITerm, Int32)

Yields TRUE if the PITerm is an atom or string representing the same integer as the argument, FALSE if the conversion was not successful. Conversion of the term is done by PL_get_long

### Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator ==</td>
<td>(PITerm term, int value)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Operator =</td>
<td>(term As PITerm, value As Integer) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: static bool operator ==</td>
<td>(PITerm term, int value)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static let inline (=)</td>
<td>term : PITerm * value : int : bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

*term (PITerm)*
a PLTerm

value (Int32)
a int

Return Value

Boolean
A bool

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an `type_error` exception if the conversion failed.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static bool operator ==(PlTerm term, string value)</code></td>
<td><code>Public Shared Operator = (term As PlTerm, value As String) As Boolean</code></td>
<td><code>public: static bool operator ==(PlTerm term, String^ value)</code></td>
<td><code>static let inline (term : PlTerm * value : string : bool</code></td>
</tr>
</tbody>
</table>

### Parameters

- `term` (**PlTerm**)

---

**Swi-cs-pl** - A CSharp class library to connect .NET languages with SWI-Prolog.

**Equality**

**Opertaor**

**SwiPIcs interface** ➤ **SbsSW.SwiPIcs** ➤ **PlTerm** ➤ **Equality(PlTerm, String)**

C#
value (String)

Return Value

Boolean
ture or false

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Equality**

**Operator**

| SwiPICs interface | SbsSW.SwiPICs | PITerm | Equality(Int32, PITerm) |

Yields TRUE if the PITerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public static bool operator ==(
  int value,
  PITerm term
)
| Public Shared Operator = (
  value As Integer,
  term As PITerm
) As Boolean |
| public:
  static bool operator ==(
  int value,
  PITerm term
)
| static let inline (=)
  value : int *
  term : PITerm : bool |

**Parameters**

*value* (*Int32*)
documentation for "M:SbsSW.SwiPICs.PlTerm.op_Equality(System.Int32,SbsSW."

**term** *(PlTerm)*

documentation for "M:SbsSW.SwiPICs.PlTerm.op_Equality(System.Int32,SbsSW."

**Return Value**

**Boolean**

true or false

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-plCs - A CSharp class library to connect .NET languages with SWI-Prolog

**Equality Operator**

SwiPlCs interface ► SbsSW.SwiPlCs ► PIPterm ► Equality(String, PIPterm)

Yields TRUE if the PIPterm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator ==(string value, PIPterm term)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Operator = (value As String, term As PIPterm) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| public:
  static bool operator ==(String^ value, PIPterm term) |
| static let inline (=) value : string *
  term : PIPterm : bool |

- **Parameters**
  
  *value* (String)
term (PlTerm)

Return Value

Boolean
ture or false

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Equals**

**Method (obj)**

SwiPIcs interface ► SbsSW.SwiPICs ► PITerm ► Equals(Object)

Indicates whether this instance and a specified object are equal.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public override bool Equals(  
  Object \textit{obj}  
)  |
| Public Overrides Function Equals (  
  \textit{obj} As Object  
) As Boolean  |
| public:  
virtual bool Equals(  
  Object^ \textit{obj}  
) override  |
| abstract Equals :  
  \textit{obj} : Object -> bool  
override Equals :  
  \textit{obj} : Object -> bool |

- **Parameters**

\textit{obj} (Object)

Another object to compare to.

- **Return Value**

\textit{Boolean}

true if \textit{obj} and this instance are the same type and represent the same
value; otherwise, false.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
## Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌀</td>
<td>Explicit(PlTerm to String)</td>
<td>Converts the Prolog argument into a string which implies Prolog atoms and strings are converted to the represented text or throw a PlTypeException.</td>
</tr>
<tr>
<td>🌀</td>
<td>Explicit(PlTerm to Int32)</td>
<td>Yields a int if the PlTerm is a Prolog integer or float that can be converted without loss to a int. Throws a PlTypeException exception otherwise</td>
</tr>
<tr>
<td>🌀</td>
<td>Explicit(PlTerm to Double)</td>
<td>Yields the value as a C# double if PlTerm represents a Prolog integer or float. Throws a PlTypeException exception otherwise.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Explicit Operator

SwiPlCs interface ▶ SbsSW.SwiPlCs ▶ PlTerm ▶ Explicit(PlTerm to Double)

Yields the value as a C# double if PlTerm represents a Prolog integer or float. Throws a PITypeException exception otherwise.

 Scalars Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static explicit operator double ( PlTerm term )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Narrowing Operator CType ( term As PlTerm ) As Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static explicit operator double ( PlTerm term )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F# does not support the declaration of new casting operators.

Parameters

term (PlTerm)

A PlTerm represents a Prolog integer or float

Return Value

Double

A C# double

Exceptions
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PITypeException</td>
<td>Throws a PITypeException exception if PIType is not a PIType.PlInteger or a PIType.PlFloat.</td>
</tr>
<tr>
<td>PreconditionException</td>
<td>Is thrown if the operator is used on an uninitialized PITerm.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
 Explicit Operator

SwiPlCs interface ► SbsSW.SwiPlCs ► PITerm ► Explicit(PITerm to String)

Converts the Prolog argument into a string which implies Prolog atoms and strings are converted to the represented text or throw a PlTypeException.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static explicit operator string (PlTerm term)</td>
<td>Public Shared Narrowing Operator CType (term As PlTerm) As String</td>
<td>static explicit operator String^ (PlTerm term)</td>
<td>F# does not support the declaration of new casting operators.</td>
</tr>
</tbody>
</table>

- **Parameters**

  term (PITerm)
  
  A PITerm that can be converted to a string

- **Return Value**

  String
  
  A C# string
Remarks

Converts the Prolog argument using PL_get_chars() using the flags CVT_ALL|CVT_WRITE|BUF_RING, which implies Prolog atoms and strings are converted to the represented text or throw a PITypeException.

If the above call return 0 PL_get_chars is called a second time with the flags CVT_ALL|CVT_WRITE|BUF_RING|REP_UTF8.

All other data is handed to write/1.

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PITypeException</td>
<td>Throws a PITypeException exception</td>
</tr>
<tr>
<td>PreconditionException</td>
<td>Is thrown if the operator is used on an uninitialized PITerm</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Explicit Operator**

*SwiPLCs interface ➤ SbsSW.SwiPLCs ➤ PlTerm ➤ Explicit(PlTerm to Int32)*

Yields a `int` if the `PlTerm` is a Prolog integer or float that can be converted without loss to a `int`. Throws a `PITypeException` exception otherwise.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public static explicit operator int (  
  PlTerm term)

Public Shared Narrowing Operator CType ( 
  term As PlTerm
)

static explicit operator int ( 
  PlTerm term
)

F# does not support the declaration of new casting operators.

### Parameters

*term (PlTerm)*

A `PlTerm` is a Prolog integer or float that can be converted without loss to a `int`.

### Return Value

*Int32*

A C# `int`
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlTypeException</td>
<td>Throws a PlTypeException exception if PlType is not a PlType.PlInteger or a PlType.PlFloat.</td>
</tr>
<tr>
<td>PreconditionException</td>
<td>Is thrown if the operator is used on an uninitialized PlTerm</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog**

**GetEnumerable Method**

SwiPICs interface ► SbsSW.SwiPICs ► PTerm ► GetEnumerable()

Returns an enumerator that iterates through the collection.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public IEnumerator&lt;PlTerm&gt; GetEnumerable()</td>
<td>public Function GetEnumerable As IEnum&lt;PlTerm&gt;</td>
<td>public: virtual IEnum&lt;PlTerm&gt; ^ GetEnumerable()</td>
<td>abstract GetEnumerable : unit -&gt; IEnum&lt;PlTerm&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>override GetEnumerable : unit -&gt; IEnum&lt;PlTerm&gt;</td>
</tr>
</tbody>
</table>

- **Return Value**

IEnumerable<PlTerm>

A System.Collections.Generic.IEnumerable<T> that can be used to iterate through the collection.

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

GetHashCode Method

SwiPICs interface ► SbsSW.SwiPICs ► PlTerm ► GetHashCode

Returns the hash code for this instance.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>

- **Return Value**

**Int32**
A 32-bit signed integer that is the hash code for this instance.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
### GreaterThan Operator

Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

#### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static bool operator &gt;(</code></td>
<td><code>Public Shared Operator &gt; (</code></td>
<td><code>public:</code></td>
<td><code>static let inline (&gt;</code></td>
</tr>
<tr>
<td><code>PlTerm term1,</code></td>
<td><code>term1 As PlTerm,</code></td>
<td><code>static bool operator &gt;(</code></td>
<td><code>term1 : PlTerm</code></td>
</tr>
<tr>
<td><code>PlTerm term2</code></td>
<td><code>term2 As PlTerm</code></td>
<td><code>:</code></td>
<td><code>*</code></td>
</tr>
<tr>
<td><code>)</code></td>
<td><code>)</code></td>
<td><code>)</code></td>
<td><code>)</code></td>
</tr>
<tr>
<td><code>As Boolean</code></td>
<td><code>As Boolean</code></td>
<td><code>As bool</code></td>
<td><code>term2 : PlTerm</code></td>
</tr>
<tr>
<td><code>)</code></td>
<td><code>)</code></td>
<td><code>)</code></td>
<td><code>bool</code></td>
</tr>
</tbody>
</table>

#### Parameters

- `term1 (PlTerm)`
a PlTerm

\textbf{term2 (PlTerm)}
a PlTerm

\begin{itemize}
\item Return Value
\end{itemize}

\textbf{Boolean}

true or false

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to \texttt{Lesta at SBS-Softwaresysteme.de}

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
GreaterThanOrEqual Operator

Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator &gt;= (PlTerm term1, PlTerm term2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Operator &gt;= (term1 As PlTerm, term2 As PlTerm) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public static bool operator &gt;= (PlTerm term1, PlTerm term2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static let inline (&gt;=) (term1 : PlTerm * term2 : PlTerm : bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters

term1 (PlTerm)
a PiTerm

term2 (PiTerm)
a PiTerm

Return Value

Boolean
true or false

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Inequality Method overload

Equality(PLTerm, PLTerm)  a  Equality(PLTerm, Int32)

Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Symbol]</td>
<td>Inequality(PLTerm, PLTerm)</td>
<td>Yields TRUE if the PLTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Inequality(PLTerm, Int32)</td>
<td>summary</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Inequality(Int32, PLTerm)</td>
<td>summary</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Inequality(PLTerm, String)</td>
<td>test</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Inequality(String, PLTerm)</td>
<td>test</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Inequality Operator**

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PlTerm ➤ Inequality(PlTerm, PlTerm)

Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an `type_error` exception if the conversion failed.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator !=(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Operator &lt;&gt; (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term1 As PlTerm,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term2 As PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static bool operator !=(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static let inline (&lt;&gt;)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term1 : PlTerm *</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term2 : PlTerm : bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

*term1 (PlTerm)*
a PlTerm

\textit{term2 (PlTerm)}

a PlTerm

**Return Value**

\textbf{Boolean}

ture or false

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to \texttt{Lesta at SBS-Softwaresysteme.de}

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0

(1.1.60605.0)
### Summary

**Operator**

**SwiPLCs interface** ▶ **SbsSW.SwiPLCs** ▶ **PlTerm** ▶ **Inequality**

**Inequality(PlTerm, Int32)**

C# version:

```csharp
public static bool operator !=(PlTerm term, int value)
```

Public Shared Operator `<>` (term As PlTerm, value As Integer) As Boolean

F# version:

```fsharp
static let inline (<>)
    term : PlTerm *
    value : int : bool
```

### Parameters

- **term** *(PlTerm)*

[Missing <param name="term"/> documentation for
"M:SbsSW.SwiPICs.PlTerm.op_Inequality(SbsSW.SwiPICs.PlTerm,System.Int32)"

value (Int32)

[Missing <param name="value"/> documentation for "M:SbsSW.SwiPICs.PlTerm.op_Inequality(SbsSW.SwiPICs.PlTerm,System.Int32)"

Return Value

Boolean

[Missing <returns> documentation for "M:SbsSW.SwiPICs.PlTerm.op_Inequality(SbsSW.SwiPICs.PlTerm,System.Int32)"

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Operator

SwiPLCs interface ▶ SbsSW.SwiPLCs ▶ PlTerm ▶ Inequality(PlTerm, String)

Test

» Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator !=(PlTerm term, string value)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Operator &lt;&gt; (term As PlTerm, value As String) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| public:
static bool operator !=(PlTerm term,
    String^ value)
| |
| static let inline (<>)
    term : PlTerm *
    value : string : bool |

» Parameters

term (PlTerm)

[Missing <param name="term"/> documentation for
value (String)

[Missing <param name="value"/> documentation for "M:SbsSW.SwiPICs.PlTerm.op_Inequality(SbsSW.SwiPICs.PlTerm,System.String)"

Return Value

Boolean

[Missing <returns> documentation for "M:SbsSW.SwiPICs.PlTerm.op_Inequality(SbsSW.SwiPICs.PlTerm,System.String)"

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

Inequality

**Operator**
SwiPIcs interface ► SbsSW.SwiPIcs ► PlTerm ► Inequality(Int32, PlTerm)

summary

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator != (int value, PlTerm term)</td>
<td>Public Shared Operator &lt;&gt; (value As Integer, term As PlTerm) As Boolean</td>
<td>public: static bool operator != (int value, PlTerm term)</td>
<td>static let inline (&lt;&gt;) value : int * term : PlTerm : bool</td>
</tr>
</tbody>
</table>

- **Parameters**

  value ([Int32])

  [Missing <param name="value"/> documentation for]
"M:SbsSW.SwiPICs.PlTerm.op_Inequality(System.Int32,SbsSW.SwiPICs.PlTerm"

term (PlTerm)

[Missing <param name="term"/> documentation for "M:SbsSW.SwiPICs.PlTerm.op_Inequality(System.Int32,SbsSW.SwiPICs.PlTerm"

Return Value

Boolean

[Missing <returns> documentation for "M:SbsSW.SwiPICs.PlTerm.op_Inequality(System.Int32,SbsSW.SwiPICs.PlTerm"

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Operator**

`SwiPLCs interface ▶ SbsSW.SwiPLCs ▶ PlTerm ▶ Inequality(String, PlTerm)`

```
public static bool operator !=(  
    string value,  
    PlTerm term  
)  

Public Shared Operator <> (  
    value As String,  
    term As PlTerm  
) As Boolean  

public:  
static bool operator !=(  
    String^ value,  
    PlTerm term  
)  

static let inline (<>)  
    value : string *  
    term : PlTerm : bool
```

**Parameters**

`value (String)`

[Missing <param name="value"/> documentation for
"M:SbsSW.SwiPICs.PIterm.op_Inequality(System.String,SbsSW.SwiPICs.PIterm)"

**term** *(PIterm)*

[Missing <param name="term"/>

documentation for
"M:SbsSW.SwiPICs.PIterm.op_Inequality(System.String,SbsSW.SwiPICs.PIterm)"

- **Return Value**
  
  *Boolean*

[Missing <returns> documentation for
"M:SbsSW.SwiPICs.PIterm.op_Inequality(System.String,SbsSW.SwiPICs.PIterm)"

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Property

SwiPICs interface ► SbsSW.SwiPICs ► PlTerm ► IsAtom

Return true if PlTerm is an atom.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public bool IsAtom { get; }</td>
<td>Public ReadOnly Property IsAtom As Boolean</td>
<td>public: property bool IsAtom {</td>
<td>member IsAtom : bool with get</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Get</td>
<td>bool get ();</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Property Value**

Boolean

- **See Also**

PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Property

SwiPICs interface ► SbsSW.SwiPICs ► PTerm ► IsAtomic

Return true if PTerm is atomic (not variable or compound).

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public bool IsAtomic { get; } | Public ReadOnly Property IsAtomic As Boolean Get | public: property bool IsAtomic {
|             |              |            | bool get (); } |
|             |              |            | member IsAtomic : bool with get |

- Property Value

Boolean

- See Also

PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**IsCompound Property**

SwiPICs interface ► SbsSW.SwiPICs ► PlTerm ► IsCompound

Return true if PlTerm is a compound term. Note that a list is a compound term \( /2 \)

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool IsCompound { get; }</td>
<td>Public ReadOnly Property IsCompound As Boolean Get</td>
<td>public: property bool IsCompound { bool get (); }</td>
<td>member IsCompound : bool with get</td>
</tr>
</tbody>
</table>

### Property Value

Boolean

### See Also

PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
IsFloat

Property

SwiPICs interface ► SbsSW.SwiPICs ► PITerm ► IsFloat

Return true if PITerm is a float.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool IsFloat { get; }</td>
<td>Public ReadOnly Property IsFloat As Boolean Get</td>
<td>public: property bool IsFloat { bool get (); }</td>
<td>member IsFloat : bool with get</td>
</tr>
</tbody>
</table>

- Property Value

Boolean

- See Also

PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Property**

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PITerm ➤ IsGround

Return true if PI Term is a ground term. See also ground/1. This function is cycle-safe.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
  | public bool IsGround { get; } | Public ReadOnly Property IsGround As Boolean Get | public: property bool IsGround {
  |                           |                        | boolean get ();           |                           |
  |                           |                        | }                          |                           |
  |                           |                        | member IsGround : bool with get |                           |

- **Property Value**

  Boolean

- **See Also**

  PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**IsInitialized Property**

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PITerm ➤ IsInitialized

return false for a PITerm variable which is only declared and true if it is also initialized

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool IsInitialized { get; }</td>
<td>Public ReadOnly Property IsInitialized As Boolean Get</td>
<td>public: property bool IsInitialized { bool get (); }</td>
<td>member IsInitialized : bool with get</td>
</tr>
</tbody>
</table>

### Property Value

Boolean

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Property**

**SwiPICs interface** ► **SbsSW.SwiPICs** ► **PlTerm** ► **IsInteger**

Return true if **PlTerm** is an integer.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public bool IsInteger { get; }</code></td>
<td><code>Public ReadOnly Property IsInteger As Boolean Get</code></td>
<td></td>
<td><code>member IsInteger : bool with get</code></td>
</tr>
</tbody>
</table>

- **Property Value**

  **Boolean**

- **See Also**

  **PlType**

---

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**IsList Property**

SwiPicS interface ★ SbsSW.SwiPICs ★ PTerm ★ IsList

Return true if PTerm is a compound term with functor ./2 or the atom [].

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool IsList { get; }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public ReadOnly Property IsList As Boolean Get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: property bool IsList {</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chunk get ();</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member IsList : bool with get</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Property Value**
  - Boolean

- **See Also**
  - PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Property**

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PlTerm ➤ IsNumber

Return true if `PlTerm` is an integer or float.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public bool IsNumber { get; }` | `Public ReadOnly Property IsNumber As Boolean Get` | `public: property bool IsNumber {
    bool get ();
} member IsNumber : bool with get` | |

**Property Value**

Boolean

**See Also**

PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Property**

SwiPICs interface ► SbsSW.SwiPICs ► PTerm ► IsString

Return true if PTerm is a string.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool IsString { get; }</td>
<td>Public ReadOnly Property IsString As Boolean Get</td>
<td>public: property bool IsString { bool get (); }</td>
<td>member IsString : bool with get</td>
</tr>
</tbody>
</table>

- **Property Value**

Boolean

- **See Also**

PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Property**

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PTerm ➤ IsVar

Return true if **PTerm** is a variable

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool IsVar { get; }</td>
<td>Public ReadOnly Property IsVar As Boolean Get</td>
<td>public: property bool IsVar { bool get (); }</td>
<td>member IsVar : bool with get</td>
</tr>
</tbody>
</table>

- **Property Value**

Boolean

- **See Also**

 PlType

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Property (position)

If PlTerm is compound and index is between 0 and Arity (including), the nth PlTerm is returned.

If pos is 0 the functor of the term is returned (For a list '.').

See: PL_get_arg/3

Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlTerm this[ int position ] { get; }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public ReadOnly Default Property Item ( position As Integer ) As PlTerm Get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: property PlTerm default[int position] { PlTerm get (int position); }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member Item : PlTerm with get</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters

position (Int32)

To Get the nth PlTerm
Return Value

PlTerm

a PlTerm

Examples

[TestMethod]
public void PlTermIndexer2()
{
    var t = new PlTerm("foo(bar, x(y))");
    Assert.AreEqual("foo", t[0].ToString());
    Assert.AreEqual("bar", t[1].ToString());
    Assert.AreEqual("x(y)", t[2].ToString());
}

[TestMethod]
public void PlTermIndexer_list1()
{
    var t = new PlTerm("[a,b,c]"");
    Assert.AreEqual(".", t[0].ToString());
    Assert.AreEqual("a", t[1].ToString());
    Assert.AreEqual("[b,c]", t[2].ToString());
}

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Is thrown if PI Term is not of Type PI Compound see IsCompound</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Thrown if (pos &lt; 0</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>Is thrown if PL_get_arg returns 0.</td>
</tr>
</tbody>
</table>
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Yields TRUE if the PlTerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator &lt;(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Operator &lt; (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term1 As PlTerm,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term2 As PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static bool operator &lt;(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static let inline (&lt;)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term1 : PlTerm *</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term2 : PlTerm : bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

term1 (PlTerm)
a PlTerm

\( \text{term2 (PlTerm)} \)
a PlTerm

- **Return Value**
  - **Boolean**
  - true or false

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

### LessThanOrEqual Operator

#### SwiPICs interface ► SbsSW.SwiPICs ► PITerm ► LessThanOrEqual(PITerm, PITerm)

Yields TRUE if the PITerm is an atom or string representing the same text as the argument, FALSE if the conversion was successful, but the strings are not equal and an type_error exception if the conversion failed.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator &lt;=(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITerm term1,</td>
<td>Public Shared Operator &lt;= (</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITerm term2</td>
<td>term1 As PITerm,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>term2 As PITerm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td>) As Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td>public:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>static bool operator &lt;=(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITerm term1,</td>
<td>static let inline (&lt;=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PITerm term2</td>
<td>term1 : PITerm *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td>term2 : PITerm : bool</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Parameters**

  - *term1 (PITerm)*
a PIterm

\texttt{term2 (PIterm)}
a PIterm

- **Return Value**
  
  \texttt{Boolean}
  
  true or false

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to \texttt{Lesta at SBS-Softwaresysteme.de}

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Property

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PlTerm ➤ Name

Get a holding the name of the functor if PlTerm is a compound term.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public string Name { get; }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public ReadOnly Property Name As String Get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: property String^ Name {</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>String^ get ();</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member Name : string with get</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Property Value**

  String

- **Remarks**

  Arity and Name are for compound terms only

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Is thrown if the term isn't compound</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

NextValue Method

SwiPIcs interface ► SbsSW.SwiPIcs ► PlTerm ► NextValue()

C# return a PlTerm bound to the next element of the list PITail and advance PITail. Returns the element on success or a free PlTerm (Variable) if PITail represents the empty list. If PITail is neither a list nor the empty list a PlTypeException (type_error) is thrown.

Decleration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlTerm NextValue()</td>
<td>Public Function NextValue As PlTerm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: PlTerm NextValue()</td>
<td>member NextValue : unit -&gt; PlTerm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return Value

PlTerm

The Next element in the list as a PlTerm which is a variable for the last element or an empty list

Examples

public void List_Add_list_doc()
{
    PlTerm t = new PlTerm("[x,y]"),
    PlTerm l = new PlTerm("[a,b]"),

    Assert.IsTrue(l.IsList);
Assert.IsTrue(l.AddList(t));
Assert.IsTrue(l.IsList);
Assert.AreEqual("[a,b,x,y]", l.ToString());
Assert.AreEqual("a", l.NextValue().ToString());
Assert.AreEqual("b", l.NextValue().ToString());
Assert.AreEqual("[x,y]", l.ToString());
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlCharList Method (text)**

Create a Prolog list of one-character atoms from a C#-string.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static PlTerm PlCharList( string text )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Function PlCharList ( text As String ) As PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static PlTerm PlCharList( String^ text )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static member PlCharList :</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text : string -&gt; PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

- **text (String)**
  a string

### Return Value

- **PlTerm**
  A new PlTerm containing a prolog list of character

### Remarks

Character lists are compliant to Prolog's `atom_chars/2` predicate.
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlCodeList Method (text)**

Create a Prolog list of ASCII codes from a 0-terminated C-string.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public static PlTerm PlCodeList(
    string text
) ` | | | |
| `Public Shared Function PlCodeList ( 
    text As String
) As PlTerm` | | | |
| `public:
static PlTerm PlCodeList( 
    String^ text
) ` | | | |
| `static member PlCodeList : 
    text : string -> PlTerm` | | | |

### Parameters

- **text (String)**
  The text

### Return Value

- **PlTerm**
  a new PlTerm

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
### Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="swi.png" alt="SwiPic" /></td>
<td>PlCompound(String, PlTermV)</td>
<td>Create a compound term with the given name from the given vector of arguments. See <code>PlTermV</code> for details.</td>
</tr>
<tr>
<td><img src="swi.png" alt="SwiPic" /></td>
<td>PlCompound(String, PlTerm)</td>
<td>Create a compound term with the given name and the arguments</td>
</tr>
<tr>
<td><img src="swi.png" alt="SwiPic" /></td>
<td>PlCompound(String, PlTerm, PlTerm)</td>
<td>Create a compound term with the given name and the arguments</td>
</tr>
<tr>
<td><img src="swi.png" alt="SwiPic" /></td>
<td>PlCompound(String, PlTerm, PlTerm, PlTerm)</td>
<td>Create a compound term with the given name and the arguments</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlCompound Method (functor, arg1)**

**SwiPICs interface ► SbsSW.SwiPICs ► PlTerm ► PlCompound(String, PlTerm)**

Create a compound term with the given name and the arguments

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public static PlTerm PlCompound(  
  string functor,  
  PlTerm arg1  
)  
| Public Shared Function PlCompound (  
  functor As String,  
  arg1 As PlTerm  
) As PlTerm  
| public:  
static PlTerm PlCompound(  
  String^ functor,  
  PlTerm arg1  
)  
| static member PlCompound :  
functor : string ^  
arg1 : PlTerm -> PlTerm |

### Parameters

**functor (String)**  
The functor (name) of the compound term
arg1 (**PlTerm**)  
The first Argument as a **PlTerm**

- **Return Value**  
  **PlTerm**  
a new **PlTerm**

by Uwe Lesta, SBS-Softwaresysteeme GmbH  
Send comments on this topic to Lesta at SBS-Softwaresysteeme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0  
(1.1.60605.0)
Create a compound term with the given name and the arguments

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public static PlTerm PlCompound(
    string functor,
    PlTerm arg1,
    PlTerm arg2
)` |                       |                        |                         |
| `Public Shared Function PlCompound (``
    functor As String,
    arg1 As PlTerm,
    arg2 As PlTerm``) As PlTerm` |                       |                        |                         |
| `public:
    static PlTerm PlCompound(
        String^ functor,
        PlTerm arg1,
        PlTerm arg2``)
    )` |                       |                        |                         |
| `static member PlCompound :``
    functor : string *
    arg1 : PlTerm *
    arg2 : PlTerm -> PlTerm` |                       |                        |                         |
- **Parameters**

  **functor (String)**
  The functor (name) of the compound term

  **arg1 (PITerm)**
  The first Argument as a PITerm

  **arg2 (PITerm)**
  The second Argument as a PITerm

- **Return Value**

  **PITerm**
  a new PITerm

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PlCompound Method (functor, arg1, arg2, arg3)**

SwiPICs interface ► SbsSW.SwiPICs ► PlTerm ►
PlCompound(String, PlTerm, PlTerm, PlTerm)

Create a compound term with the given name and the arguments

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>**public static PlTerm PlCompound(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>string functor,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm arg1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm arg2,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm arg3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Public Shared Function PlCompound (**

```csharp
functor As String,
arg1 As PlTerm,
arg2 As PlTerm,
arg3 As PlTerm
) As PlTerm
```

**public static PlTerm PlCompound(**

```csharp
String^ functor,
PlTerm arg1,
PlTerm arg2,
PlTerm arg3
)```

static member PlCompound :
    functor : string *
    arg1 : PlTerm *
    arg2 : PlTerm *
    arg3 : PlTerm -> PlTerm

- **Parameters**
  
  *functor (String)*
  The functor (name) of the compound term

  *arg1 (PlTerm)*
  The first Argument as a PlTerm

  *arg2 (PlTerm)*
  The second Argument as a PlTerm

  *arg3 (PlTerm)*
  The third Argument as a PlTerm

- **Return Value**
  PlTerm
  a new PlTerm

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

Prolog **PICompound Method (functor, args)**

SwiPlCs interface ► SbsSW.SwiPlCs ► PTerm ►
PICompound(String, PTermV)

Create a compound term with the given name from the given vector of arguments. See **PTermV** for details.

### Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public static</strong> PlTerm PlCompound(</td>
<td>string <strong>functor</strong>, PlTermV <strong>args</strong> )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Public Shared Function</strong> PlCompound (</td>
<td><strong>functor As</strong> String, <strong>args As</strong> PTermV )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>As</strong> PTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>public:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>static</strong> PlTerm PlCompound(</td>
<td>String^ <strong>functor</strong>, PlTermV <strong>args</strong> )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>static member</strong> PlCompound :</td>
<td><strong>functor</strong> : string *</td>
<td><strong>args</strong> : PTermV -&gt; PTerm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

- **functor (String)**
The functor (name) of the compound term

\textit{args (PlTermV)}
the arguments as a \textit{PlTermV}

- Return Value
\textit{PlTerm}
a new \textit{PlTerm}

- Examples

The example below creates the Prolog term hello(world).

\begin{verbatim}
PlTerm t = PlTerm.PlCompound("hello", new PlTermV
\end{verbatim}

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
## Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon]</td>
<td><code>PlString(String)</code></td>
<td>A SWI-Prolog string represents a byte string on the global stack. It’s lifetime is the same as for compound terms and other data living on the global stack. Strings are not only a compound representation of text that is garbage-collected, but as they can contain 0-bytes, they can be used to contain arbitrary C-data structures.</td>
</tr>
<tr>
<td>![Icon]</td>
<td><code>PlString(String, Int32)</code></td>
<td>A SWI-Prolog string represents a byte string on the global stack. It’s lifetime is the same as for compound terms and other data living on the global stack. Strings are not only a compound representation of text that is garbage-collected, but as they can contain 0-bytes, they can be used to contain arbitrary C-data structures.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
A SWI-Prolog string represents a byte-string on the global stack. Its lifetime is the same as for compound terms and other data living on the global stack. Strings are not only a compound representation of text that is garbage-collected, but as they can contain 0-bytes, they can be used to contain arbitrary C-data structures.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static PlTerm PlString(</td>
<td>Public Shared Function PlString (</td>
<td>public: static PlTerm PlString(</td>
<td>static member PlString :</td>
</tr>
<tr>
<td>string text</td>
<td>text As String</td>
<td>String^ text</td>
<td>text : string -&gt; PlTerm</td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

- **text (String)**
  
  the string

### Return Value
PITerm
a new PITerm

 Remarks
NOTE: this Method do *not* work with unicode characters. Consider to use new PITerm(test) instead.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
A SWI-Prolog string represents a byte-string on the global stack. It's lifetime is the same as for compound terms and other data living on the global stack. Strings are not only a compound representation of text that is garbage-collected, but as they can contain 0-bytes, they can be used to contain arbitrary C-data structures.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static PlTerm PlString(</code></td>
<td><code>Public Shared Function PlString (</code></td>
<td><code>public: static PlTerm PlString(</code></td>
<td><code>static member PlString : </code></td>
</tr>
<tr>
<td><code>string text,</code></td>
<td><code>text As String,</code></td>
<td><code>String^ text,</code></td>
<td><code>text : string *</code></td>
</tr>
<tr>
<td><code>int len</code></td>
<td><code>len As Integer</code></td>
<td><code>int len</code></td>
<td><code>len : int -&gt; PlTerm</code></td>
</tr>
</tbody>
</table>
- **Parameters**

  *text* (*String*)
  the string

  *len* (*Int32*)
  the length of the string

- **Return Value**

  *PlTerm*
  a new PlTerm

- **Remarks**

  NOTE: this Method do *not* work with unicode characters. Consider to use new PlTerm(test) instead.

  by Uwe Lesta, SBS-Softwaresysteme GmbH
  Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

  Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
  (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Method (list)**

SwiPICs interface ► SbsSW.SwiPICs ► PlTerm ► PlTail(PlTerm)

PlTail is for analysing and constructing lists. It is called PlTail as enumeration-steps make the term-reference follow the `tail' of the list.

A PlTail is created by making a new term-reference pointing to the same object. As PlTail is used to enumerate or build a Prolog list, the initial list term-reference keeps pointing to the head of the list.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static PlTerm PlTail(</code></td>
<td><code>Public Shared Function PlTail (</code></td>
<td><code>public:</code></td>
<td><code>static member PlTail :</code></td>
</tr>
<tr>
<td><code>PlTerm </code>list`</td>
<td><code>list As PlTerm</code></td>
<td><code>static PlTerm PlTail(</code></td>
<td><code>list</code> : PlTerm -&gt; PlTerm</td>
</tr>
<tr>
<td><code>)</code></td>
<td><code>)</code></td>
<td><code>)</code></td>
<td></td>
</tr>
</tbody>
</table>

- **Parameters**

`list (PlTerm)`

The initial PlTerm
Return Value

**PlTerm**
A PlTerm for which is_list/1 succeed.

Examples

```csharp
public void List_Append()
{
    PlTerm t = PlTerm.PlVar();
    PlTerm l = PlTerm.PlTail(t);

    Assert.IsTrue(l.Append(new PlTerm("one"))
    Assert.IsTrue(l.Append(new PlTerm("two"))
    Assert.IsTrue(l.Append(new PlTerm("three")

    Assert.AreEqual(1, l.Close());
    Assert.AreEqual(3, list_length(t));
    Assert.AreEqual("[one,two,three]", t.ToSt
```

See Also

Append(PlTerm)
Add(PlTerm)
AddList(PlTerm)
Close()
NextValue()

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Get the **PlType** of a **PlTerm**.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public PlType PlType { get; } | Public ReadOnly Property PlType As PlType Get | public: property PlType PlType {  
PlType get ();  
} | member PlType : PlType with get |

### Property Value

**PlType**

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Method**

SwiPICs interface ► SbsSW.SwiPICs ► PlTerm ► PlVar()

Creates a new initialised term (holding a Prolog variable).

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static PlTerm PlVar()</td>
<td>Public Shared Function PlVar As PlTerm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
public static PlTerm PlVar()

Public Shared Function PlVar As PlTerm
```

```
public:
static PlTerm PlVar()

static member PlVar : unit -> PlTerm
```

- **Return Value**

<table>
<thead>
<tr>
<th>PlTerm</th>
</tr>
</thead>
</table>

a PlTerm

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
To List

Method

SwiPICs interface ► SbsSW.SwiPICs ► PIFerm ► ToList()

Converts to a strongly typed ReadOnlyCollection of PIFerm objects that can be accessed by index

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ReadOnlyCollection&lt;PITerm&gt; ToList()</td>
<td>Public Function ToList As ReadOnlyCollection&lt;PITerm&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>member ToList : unit -&gt; ReadOnlyCollection&lt;PITerm&gt;^ ToList()</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Return Value**

**ReadOnlyCollection<PITerm>**

A strongly typed ReadOnlyCollection of PIFerm objects

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**ToListString Method**

SwiPICs interface ▶ SbsSW.SwiPICs ▶ PITerm ▶ **ToListString()**

Converts to a strongly typed Collection of strings that can be accessed by index

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public Collection&lt;string&gt; ToListString()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Function ToListString As Collection(Of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collection&lt;String^^ ToListString()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member ToListString : unit -&gt; Collection&lt;string&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Return Value**

**Collection<String>**

A strongly typed string Collection

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**ToString**

**SwiPlCs interface** ➤ **SbsSW.SwiPlCs** ➤ **PITerm** ➤ **ToString()**

If PITerm is a list the string is build by calling ToString() for each element in the list separated by ',' and put the brackets around '[' ']'.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public override string ToString()</td>
<td>Public Overrides Function ToString As String</td>
<td>public: virtual String^ ToString() override</td>
<td>abstract ToString : unit -&gt; string override ToString : unit -&gt; string</td>
</tr>
</tbody>
</table>

- **Return Value**

  **String**

  A string representing the PITerm.

- **See Also**

  [Overload:string]

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Prolog **ToStringCanonical Method**

SwiPlCs interface ► SbsSW.SwiPlCs ► PITerm ► ToStringCanonical()

Convert a PITerm to a string by `PL_get_chars/1` with the CVT_WRITE_CANONICAL flag. If it fails `PL_get_chars/3` is called again with REP_MB flag.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public string ToStringCanonical()</code></td>
<td><code>Public Function ToStringCanonical As String</code></td>
<td><code>public: String^ ToStringCanonical()</code></td>
<td><code>member ToStringCanonical : unit -&gt; string</code></td>
</tr>
</tbody>
</table>

- **Return Value**

  **String**

  return the string of a PITerm

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PITypeException</td>
<td>Throws a PITypeException if <code>PL_get_chars/3</code> didn't succeeds.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Unify**

This method performs Prolog unification and returns true if successful and false otherwise. It is equal to the prolog =/2 operator.

See **Unify(PlTerm)** for an example.

- **Remarks**
  This methods are introduced for clear separation between the destructive assignment in C# using = and prolog unification.

- **Members**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon.png" alt="Icon" /></td>
<td><strong>Unify(PlTerm)</strong></td>
<td>Unify a PlTerm with a PlTerm</td>
</tr>
<tr>
<td><img src="icon.png" alt="Icon" /></td>
<td><strong>Unify(String)</strong></td>
<td>Unify a PlTerm with a PlTerm</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Unify a PlTerm with a PlTerm

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public bool Unify(</code></td>
<td><code>Public Function Unify (</code></td>
<td><code>public:</code></td>
<td><code>member Unify :</code></td>
</tr>
<tr>
<td><code>    PlTerm term</code>)`</td>
<td><code>    term As PlTerm)</code></td>
<td><code>    </code></td>
<td><code>    term : PlTerm -&gt; bool</code></td>
</tr>
</tbody>
</table>

### Parameters

- **term (PlTerm)**
  - the second term for unification

### Return Value

- **Boolean**
  - true or false

### Examples
public void UnifyTermVar_doc()
{
    var t1 = new PlTerm("x(A, 2)" starring: 1 Agrolab GmbH);
    var t2 = new PlTerm("x(1, B)"");
    Assert.IsTrue(t1.Unify(t2));
    Assert.AreEqual("x(1,2)", t1.ToString());
    Assert.AreEqual("x(1,2)", t2.ToString());
}
**Unify Method (atom)**

SwiPlCs interface ► SbsSW.SwiPlCs ► PITerm ► Unify(String)

Unify a PITerm with a PITerm

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public bool Unify(  
  string atom  
)` | | | |
| `Public Function Unify (  
  atom As String  
)` As Boolean | | | |
| `public:  
bool Unify(  
  String^ atom  
)` | | | |
| `member Unify :  
atom : string -> bool` | | | |

- **Parameters**

  *atom (String)*

  A string to unify with

- **Return Value**

  *Boolean*

  true or false

- **Examples**
public void UnifyTermVar_doc()
{
    var t1 = new PlTerm("x(A, 2)" );
    var t2 = new PlTerm("x(1, B)" );
    Assert.IsTrue(t1.Unify(t2));
    Assert.AreEqual("x(1,2)", t1.ToString());
    Assert.AreEqual("x(1,2)", t2.ToString());
}
The struct PlTermV represents an array of term-references.

This type is used to pass the arguments to a foreign defined predicate (see DelegateParameterVarArgs), construct compound terms (see PlCompound(String, PlTermV) and to create queries (see PlQuery).

The only useful member function is the overloading of [], providing (0-based) access to the elements. Item[Int32] Range checking is performed and raises a ArgumentOutOfRangeException exception.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public struct PlTermV : IEquatable&lt;PlTermV&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Structure PlTermV</td>
<td></td>
<td></td>
<td>Implement IEquatable(Of PlTermV)</td>
</tr>
<tr>
<td>public value class PlTermV : IEquatable&lt;PlTermV&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SealedAttribute]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type PlTermV =</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>struct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>interface IEquatable&lt;PlTermV&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Members**

<table>
<thead>
<tr>
<th>All Members</th>
<th>Constructors</th>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td></td>
<td>Instance</td>
<td></td>
</tr>
<tr>
<td>Icon</td>
<td>Member</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>PlTermV(Int32)</td>
<td>Create a vector of PITerms with size elements</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>PlTermV(PlTerm)</td>
<td>Create a PlTermV from the given PITerms.</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>PlTermV(PlTerm, PlTerm)</td>
<td>Create a PlTermV from the given PITerms.</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>PlTermV(PlTerm, PlTerm, PlTerm)</td>
<td>Create a PlTermV from the given PITerms.</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>PlTermV(PlTerm[])</td>
<td>Create a PlTermV from the given PITerm[] array.</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>Equality(PlTermV, PlTermV)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>Equals(Object)</td>
<td>Indicates whether this instance and a specified object are equal. (Overrides ValueType.Equals(Object).)</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>Equals(PlTermV)</td>
<td>Compare the size and A0 of the PltermV</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>Finalize()</td>
<td>Allows an object to try to free</td>
<td></td>
</tr>
</tbody>
</table>
resources and perform other cleanup operations before it is reclaimed by garbage collection.

(Inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object).)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode()</td>
<td>Returns the hash code for this instance.</td>
</tr>
<tr>
<td>(Overrides ValueType.GetHashCode())</td>
<td></td>
</tr>
<tr>
<td>GetType()</td>
<td>Gets the type of the current instance.</td>
</tr>
<tr>
<td>(Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
<td></td>
</tr>
<tr>
<td>Inequality(PlTermV, PlTermV)</td>
<td>A zero based list</td>
</tr>
<tr>
<td>Item[Int32]</td>
<td></td>
</tr>
<tr>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>(Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>Get the size of a PlTermV</td>
</tr>
<tr>
<td>ToString()</td>
<td>Returns the fully qualified type name of this instance.</td>
</tr>
<tr>
<td>(Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.valueobject">ValueType</a>.)</td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:lesta@sbs-softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

Constructor

Create a PIInterval vector from the given PIInterval parameters

Create a new vector with PIInterval as elements

It can be created with size elements

or

automatically for 1, 2 or 3 PIIntervals

Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PIInterval(Int32)</td>
<td>Create a vector of PIIntervals with size elements</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PIInterval(PIInterval)</td>
<td>Create a PIInterval from the given PIIntervals.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PIInterval(PIInterval, PIInterval)</td>
<td>Create a PIInterval from the given PIIntervals.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PIInterval(PIInterval, PIInterval, PIInterval)</td>
<td>Create a PIInterval from the given PIIntervals.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PIInterval(PIInterval[])</td>
<td>Create a PIInterval from the given PIInterval[] array.</td>
</tr>
</tbody>
</table>
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Constructor (term0)**

Create a PlTermV from the given PlTerms.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlTermV( PlTerm term0 )</td>
<td>Public Sub New ( term0 As PlTerm )</td>
<td>public: PlTermV( PlTerm term0 )</td>
<td>new : term0 : PlTerm -&gt; PlTermV</td>
</tr>
</tbody>
</table>

### Parameters

**term0 (PlTerm)**

The first PlTerm in the vector.

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Constructor (term0, term1)**

Create a `PlTermV` from the given `PlTerm`s.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public PlTermV(</code></td>
<td>`Public Sub New (</td>
<td><code>public:</code></td>
<td><code>new:</code></td>
</tr>
<tr>
<td><code>    PlTerm term0,</code></td>
<td>`    term0 As PlTerm,</td>
<td><code>    term0 : PlTerm *</code></td>
<td>`    term0 : PlTerm *</td>
</tr>
<tr>
<td><code>    PlTerm term1)</code></td>
<td><code>    term1 As PlTerm</code></td>
<td></td>
<td><code>    term1 : PlTerm -&gt; PlTermV</code></td>
</tr>
</tbody>
</table>

### Parameters

**term0 (PlTerm)**

The first `PlTerm` in the vector.
**term1 (PlTerm)**
The second PlTerm in the vector.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PiTermV Constructor (term0, term1, term2)**

SwiPICs interface ➤ SbsSW.SwiPICs ➤ PiTermV ➤ PiTermV(PiTerm, PiTerm, PiTerm)

Create a PiTermV from the given PiTerms.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PiTermV(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PiTerm term0,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PiTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PiTerm term2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Public Sub New (              |              |            |                             |
|     term0 As PiTerm,        |              |            |                             |
|     term1 As PiTerm,        |              |            |                             |
|     term2 As PiTerm         |              |            |                             |
| )                            |              |            |                             |

public:                       |              |            |                             |
| PiTermV(                     |              |            |                             |
|     PiTerm term0,            |              |            |                             |
|     PiTerm term1,            |              |            |                             |
|     PiTerm term2             |              |            |                             |
| )                            |              |            |                             |

new:                         |              |            |                             |
|     term0 : PiTerm *         |              |            |                             |
|     term1 : PiTerm *         |              |            |                             |
|     term2 : PiTerm -> PiTermV |              |            |                             |
- Parameters

**term0 (PITerm)**
The first PITerm in the vector.

**term1 (PITerm)**
The second PITerm in the vector.

**term2 (PITerm)**
The third PITerm in the vector.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog.

**Constructor (terms)**

SwiPICs interface ► SbsSW.SwiPICs ► PlTermV ►

PlTermV(PlTerm[])

Create a PlTermV from the given PlTerm[] array.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public PlTermV(  
PlTerm[] terms  
) | Public Sub New (  
  terms As PlTerm()  
) | public:  
PlTermV(  
  array<PlTerm>^ terms  
) | new:  
  terms : PlTerm[] -&gt; PlTermV |

**Parameters**

*terms (PlTerm[])*

An array of PlTerms to build the vector.

**Examples**

Use of Initializing an Array in CSharp
PlTermV v = new PlTermV(new PlTerm[] {t1, t2, ... })

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-CS-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Constructor (size)**

Create a vector of PlTerms with size elements

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>`public PlTermV(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size As Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: PlTermV(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new: size : int -&gt; PlTermV</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

**size (Int32)**

The amount of PlTerms in the vector

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Operator

SwiPlCs interface ► SbsSW.SwiPlCs ► PITermV ►
Equality(PITermV, PITermV)

[Missing <summary> documentation for
"M:SbsSW.SwiPlCs.PITermV.op_Equality(SbsSW.SwiPlCs.PITermV.

declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static bool operator ==(</td>
<td>PlTermV termVector1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlTermV termVector2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Shared Operator = (</td>
<td>termVector1 As PITermV,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>termVector2 As PITermV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static bool operator ==(</td>
<td>PlTermV termVector1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlTermV termVector2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static let inline (=)</td>
<td>termVector1 : PITermV *,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>termVector2 : PITermV : bool</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters

termVector1 (PITermV)
Return Value

Boolean

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
### Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌻</td>
<td><code>Equals(Object)</code></td>
<td>Indicates whether this instance and a specified object are equal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Overrrides <code>ValueType.Equals(Object)</code>).</td>
</tr>
<tr>
<td>🌻</td>
<td><code>Equals(PlTermV)</code></td>
<td>Compare the size and A0 of the PltermV</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Method (other)**

SwiPIcs interface ➤ SbsSW.SwiPIcs ➤ PlTermV ➤ Equals(PlTermV)

Compare the size and A0 of the PlTermV

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool Equals(</td>
<td>Visual Basic</td>
<td>Visual C++</td>
<td>F#</td>
</tr>
<tr>
<td>PlTermV other</td>
<td>Public Function Equals(</td>
<td>As Boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other As PlTermV</td>
<td>As Boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>public:</td>
<td>Public Function</td>
<td></td>
</tr>
<tr>
<td></td>
<td>virtual bool Equals(</td>
<td>As Boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlTermV other</td>
<td>As Boolean</td>
<td></td>
</tr>
<tr>
<td>secondary sealed</td>
<td>abstract Equals :</td>
<td>override Equals</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other : PlTermV -&gt; bool</td>
<td>Equals :</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>other : PlTermV -&gt; bool</td>
<td></td>
</tr>
</tbody>
</table>

- **Parameters**

  *other* (**PlTermV**)
  
The PlTermV to compare

- **Return Value**

  *Boolean*
  
  Return *false* if size or A0 are not equal otherwise *true*. 
Remarks

// TODO compare each PITerm in PITermV not only the references in A0

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
**Equals**

**Method (obj)**

Indicates whether this instance and a specified object are equal.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public override bool Equals(  
  Object obj  
)  
| Public Overrides Function Equals (  
  obj As Object  
) As Boolean  
| public:  
  virtual bool Equals(  
  Object^ obj  
) override  
| abstract Equals :  
  obj : Object -> bool  
| override Equals :  
  obj : Object -> bool |

**Parameters**

- **obj (Object)**
  - Another object to compare to.

**Return Value**

- **Boolean**
  - true if obj and this instance are the same type and represent the same
value; otherwise, false.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

GetHashCode Method

SwiPICs interface ► SbsSW.SwiPICs ► PITermV ► GetHashCode()

Returns the hash code for this instance.

- Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public override int GetHashCode()</td>
<td>Public Overrides Function GetHashCode As Int</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>public:</td>
<td>public:</td>
<td>public:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>virtual int GetHashCode() override</td>
<td>Function GetHaseCode As Int</td>
<td>abstract GetHashCode : unit -&gt; int</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>override GetHashCode : unit -&gt; int</td>
<td></td>
</tr>
</tbody>
</table>

- Return Value

Int32
A 32-bit signed integer that is the hash code for this instance.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

### Operator

**SwiPICs interface** ► **SbsSW.SwiPICs** ► **PlTermV** ► **Inequality**

Inequality(PlTermV, PlTermV)

[Missing <summary> documentation for "M:SbsSW.SwiPICs.PlTermV.op_Inequality(SbsSW.SwiPICs.PlTerm

#### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>**public static bool operator !=(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTermV termVector1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTermV termVector2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)**</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>**Public Shared Operator &lt;&gt; (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>termVector1 As PlTermV,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>termVector2 As PlTermV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>public:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>**static bool operator !=(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTermV termVector1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTermV termVector2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)**</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>static let inline (&lt;&gt;))</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>termVector1 : PlTermV *</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>termVector2 : PlTermV : bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Parameters

*termVector1 (PlTermV)*
documentation for
"M:SbsSW.SwiPICs.PlTermV.op_Inequality(SbsSW.SwiPICs.PlTermV,SbsSW.SwiPICs.PlTermV)"

termVector2 (PlTermV)

documentation for
"M:SbsSW.SwiPICs.PlTermV.op_Inequality(SbsSW.SwiPICs.PlTermV,SbsSW.SwiPICs.PlTermV)"

Return Value

Boolean

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
A zero based list

## Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlTerm this[ int index ] { get; set; }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Default Property Item ( index As Integer ) As PlTerm</td>
<td>Get</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: property PlTerm default[int index] { PlTerm get (int index); void set (int index, PlTerm value); }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member Item : PlTerm with get, set</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Parameters

index (Int32)

## Return Value

PlTerm
The PlTerm for the given index
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>thrown if (index &lt; 0</td>
</tr>
<tr>
<td>PreconditionException</td>
<td>Is thrown if the operator is used on an uninitialized PIterm</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Get the size of a PlTermV

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public int Size { get; }</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public ReadOnly Property</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size As Integer Get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: property int Size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get ()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member Size : int with get</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Property Value**

Int32

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Obtain the type of a term, which should be a term returned by one of the other interface predicates or passed as an argument. The function returns the type of the Prolog term. The type identifiers are listed below.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public enum PlType</td>
<td>Public Enumeration PlType</td>
<td>public enum class PlType</td>
<td>type PlType</td>
</tr>
</tbody>
</table>

### Members

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLUnknown</td>
<td>0 - PL_UNKNOWN: Undefined</td>
</tr>
<tr>
<td>PIVariable</td>
<td>1 - PL_VARIABLE: An unbound variable. The value of term as such is a unique identifier for the variable.</td>
</tr>
<tr>
<td>PLAtom</td>
<td>2 - PL_ATOM: A Prolog atom.</td>
</tr>
<tr>
<td>PInteger</td>
<td>3 - PL_INTEGER: A Prolog integer.</td>
</tr>
<tr>
<td>PIFloat</td>
<td>4 - PL_FLOAT: A Prolog floating point number.</td>
</tr>
<tr>
<td>PIString</td>
<td>5 - PL_STRING: A Prolog string.</td>
</tr>
<tr>
<td>PITerm</td>
<td>6 - PL_TERM: A compound term. Note that a list is a compound term ./2.</td>
</tr>
</tbody>
</table>
Remarks

see **PL_term_type(term_t)** in the SWI-Prolog Manual.

Examples

In this sample a Prolog variable is created in **PlTerm t** and the **PlType** is checked by his integer representation and his name.

```plaintext
PlTerm t = PlTerm.PlVar();
Assert.AreEqual(1, (int)t.PlType);
Assert.AreEqual(PlType.PlVariable, t.PlType);
```

See Also

**PlType**

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
The namespace SbsSW.SwiPICs.Callback provides the delegates to register .NET methods to be called from SWI-Prolog.

### Types

<table>
<thead>
<tr>
<th>Icon</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Delegate" /></td>
<td>DelegateParameter0</td>
<td>Provide a predefined Delegate to register a C# method to be called from SWI-Prolog</td>
</tr>
<tr>
<td><img src="image" alt="Delegate" /></td>
<td>DelegateParameter1</td>
<td>Provide a predefined Delegate to register a C# method to be called from SWI-Prolog</td>
</tr>
<tr>
<td><img src="image" alt="Delegate" /></td>
<td>DelegateParameter2</td>
<td>Provide a predefined Delegate to register a C# method to be called from SWI-Prolog</td>
</tr>
<tr>
<td><img src="image" alt="Delegate" /></td>
<td>DelegateParameter3</td>
<td>Provide a predefined Delegate to register a C# method to be called from SWI-Prolog</td>
</tr>
<tr>
<td>DelegateParameterBacktrack</td>
<td>NOT IMPLEMENTED YET</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For details to implement see 9.6.17 Registering Foreign Predicates</td>
<td>see also PL_foreign_control</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DelegateParameterVarArgs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With this delegate you can build call-back predicate with a variable amount of parameters.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PLForeignSwitches</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flags that are responsible for the foreign predicate parameters</td>
</tr>
</tbody>
</table>

**Remarks**

**Note:**

It is only possible to call static methods

**See Also**

RegisterForeign(String, Delegate)

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de
Provide a predefined Delegate to register a C# method to be called from SWI-Prolog.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public delegate bool DelegateParameter0()</td>
<td>Public Delegate Function DelegateParameter0()</td>
<td>public delegate bool DelegateParameter0()</td>
<td>type DelegateParameter0 = delegate of unit -&gt; bool</td>
</tr>
</tbody>
</table>

**Return Value**

Boolean

true for succeeding otherwise false for fail

**Examples**

This example is for DelegateParameter2 and shows how to call a C# method with two parameter.

For other samples see the source file CallbackForeignPredicate.cs in the TestSwiPl VS2008 test project.

```csharp
public void t_in_out()
```
Delegate replaceDelegate = new DelegateParameter2(atom_replace);
PlEngine.RegisterForeign(replaceDelegate);

for (int i = 1; i < 10; i++)
{
    var arg = new PlTermV(new PlTerm("test_f"));
    PlQuery.PlCall("atom_replace", arg);
    Assert.AreEqual("test_xx_f", arg[1].ToString());
}

public static bool atom_replace(PlTerm atomIn, PlTerm atomOut)
{
    return atomOut.Unify(atomIn.ToString().Replace...}

See Also

RegisterForeign(Delegate)
PlEngine()

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**DelegateParameter1 Delegate**

Provide a predefined Delegate to register a C# method to be called from SWI-Prolog

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public delegate bool DelegateParameter1( PlTerm term )</td>
<td>Public Delegate Function DelegateParameter1 ( term As PlTerm ) As Boolean</td>
<td>public delegate bool DelegateParameter1( PlTerm term )</td>
<td>type DelegateParameter1 = delegate of term : PlTerm -&gt; bool</td>
</tr>
</tbody>
</table>

### Parameters

*term (PlTerm)*

### Return Value

*Boolean*

true for succeeding otherwise false for fail
Examples

See also the example in DelegateParameter2.

```csharp
[TestMethod]
public void t_creating_a_list()
{
    Delegate d = new DelegateParameter1(create_list);
    PlEngine.RegisterForeign(d);
    for (int i = 1; i < 10; i++)
    {
        PlTerm t = PlQuery.PlCallQuery("create_list(L)"
        Assert.AreEqual("[a,b,c]", t.ToString);
    }
}

class public static bool create_list(PlTerm list)
{
    return list.Unify(new PlTerm("[a,b,c]"));
}
```

See Also

RegisterForeign(Delegate)
PlEngine()

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Provide a predefined Delegate to register a C# method to be called from SWI-Prolog

### Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public delegate bool DelegateParameter2(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlTerm term2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>public delegate Function DelegateParameter2 (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>term1 As PlTerm,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>term2 As PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>public delegate bool DelegateParameter2(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlTerm term1,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PlTerm term2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>type DelegateParameter2 =</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>delegate of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>term1 : PlTerm *</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>term2 : PlTerm -&gt; bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

- **term1 (PlTerm)**
term2 (PlTerm)

- **Return Value**

  **Boolean**
  
  true for succeeding otherwise false for fail

- **Examples**

  This example is for **DelegateParameter2** and shows how to call a C# method with two parameters.

  For other samples see the source file CallbackForeignPredicate.cs in the TestSwiPl VS2008 test project.

```csharp
public void t_in_out()
{
    Delegate replaceDelegate = new DelegateParameter2(atom_replace);
    PlEngine.RegisterForeign(replaceDelegate);
    for (int i = 1; i < 10; i++)
    {
        var arg = new PlTermV(new PlTerm("test_f"));
        PlQuery.PlCall("atom_replace", arg);
        Assert.AreEqual("test_xx_f", arg[1].ToString);
    }
}

public static bool atom_replace(PlTerm atomIn)
{
    return atomOut.Unify(atomIn.ToString().Replace(}
```

- **See Also**

  **RegisterForeign(Delegate)**

  **PlEngine()**
Provide a predefined Delegate to register a C# method to be called from SWI-Prolog

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public delegate bool DelegateParameter3(  
  PlTerm term1,  
  PlTerm term2,  
  PlTerm term3
)  |
| Public Delegate Function DelegateParameter3 (  
  term1 As PlTerm,  
  term2 As PlTerm,  
  term3 As PlTerm
) As Boolean  |
| public delegate bool DelegateParameter3(  
  PlTerm term1,  
  PlTerm term2,  
  PlTerm term3
)  |
| type DelegateParameter3 =  
  delegate of  
  term1 : PlTerm *  
  term2 : PlTerm *  
  term3 : PlTerm -> bool  |
Parameters

term1 (PlTerm)
term2 (PlTerm)
term3 (PlTerm)

Return Value

Boolean
true for succeeding otherwise false for fail

Examples

This example is for DelegateParameter2 and shows how to call a C# method with two parameters.

For other samples see the source file CallbackForeignPredicate.cs in the TestSwiPl VS2008 test project.

```csharp
public void t_in_out()
{
    Delegate replaceDelegate = new DelegateParameter2(atom_replace);
    PlEngine.RegisterForeign(replaceDelegate);
    for (int i = 1; i < 10; i++)
    {
        var arg = new PlTermV(new PlTerm("test_f"));
        PlQuery.PlCall("atom_replace", arg);
        Assert.AreEqual("test_xx_f", arg[1].ToString());
    }
}

public static bool atom_replace(PlTerm atomIn)
{
    return atomOut.Unify(atomIn.ToString().Replace(
```
See Also

RegisterForeign(Delegate)
PlEngine()

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

DelegateParameterBacktrack Delegate

SwiPlCs interface ► SbsSW.SwiPlCs.Callback ► DelegateParameterBacktrack

**NOT IMPLEMENTED YET**

For details to implement see [9.6.17 Registering Foreign Predicates](#)

see also [PL_foreign_control](#)

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public delegate int DelegateParameterBacktrack
            PlTerm term1,
            PlTerm term2,
            IntPtr control
        )` | `Public Delegate Function DelegateParameterBacktrack
            term1 As PlTerm,
            term2 As PlTerm,
            control As IntPtr
        ) As Integer` | `public delegate int DelegateParameterBacktrack
            PlTerm term1,
            PlTerm term2,
            IntPtr control
        )` | `type DelegateParameterBacktrack =
            delegate of` |
**term1** : PlTerm *
**term2** : PlTerm *
**control** : IntPtr -> int

- **Parameters**

  **term1** ([PlTerm])
  TODO

  **term2** ([PlTerm])
  TODO

  **control** ([IntPtr])
  TODO

- **Return Value**

  Int32
  TODO

- **Examples**

  TODO

See "t_backtrack" in TestSwiPl.CallbackForeignPredicate.cs

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
With this delegate you can build a call-back predicate with a variable amount of parameters.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public delegate bool DelegateParameterVarArgs(PlTermV termVector)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Delegate Function DelegateParameterVarArgs(PlTermV termVector As PlTermV) As Boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public delegate bool DelegateParameterVarArgs(PlTermV termVector)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type DelegateParameterVarArgs = delegate of termVector : PlTermV -&gt; bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

**termVector (PlTermV)**

The termVector representing the arguments which can be accessed by the indexer of PlTermV see PlTermV. The amount of parameters is in Size.

### Return Value
Boolean
True for succeeding otherwise false for fail

Remarks

TODO: This do *NOT* work on 64-Bit systems. Hope to Fix this in the future.

It seems to be impossible to marshal two parameter which are bigger than 8 byte into one struct. Perhaps there is a way in CLI :-(

The problem are the parameters of the call back method. These are in

\[ \text{foreign_t (f)(term_t t0, int a, control_t ctx)} \]

(see SWI-cpp.h) If we provide

DelegateParameterVarArgs(PlTerm term, int arity); and do in the callback Method something like

```java
public static bool my_call_back(PlTerm term,
{
    PlTermV args = new PlTermV(term, arity);
}
```

every thing work fine. The drawback is this ugly ctor. It might be better to do

```java
PlTermV args = PlTermV.VarArgs(term, arity);
```

with a strong recommendation to use it *OINLY* in this call back scenario.

Examples

```java
public void t_varargs()
{
    Delegate d = new DelegateParameterVarArgs
    PlEngine.RegisterForeign("my_concat_atom"
```
PlEngine.RegisterForeign("my_concat_atom"
for (int i = 1; i < 10; i++)
{
    PlTerm t = PlQuery.PlCallQuery("my_co
Assert.AreEqual("abc", t.ToString()),
t = PlQuery.PlCallQuery("my_concat_at
Assert.AreEqual("abcdef", t.ToString(}
}
}
public static bool my_concat_atom(PlTermV term
{
    int arity = term1.Size;
    string sRet = ");
    PlTerm termOut = term1[arity -1];

    for (int i = 0; i < arity-1; i++)
    {
        sRet += term1[i].ToString();
    }
    termOut.Unify(sRet);
    return true;
}
**Swi-cs-pl** - A CSharp class library to connect .NET languages with SWI-Prolog

**Prolog**

**PIForeignSwitches Enumeration**

SwiPlCs interface ▶ SbsSW.SwiPlCs.Callback ▶ PIForeignSwitches

Flags that are responsible for the foreign predicate parameters

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[FlagsAttribute]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public enum PIForeignSwitches</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;FlagsAttribute&gt;</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Public Enumeration PIForeignSwitches</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>[FlagsAttribute]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public enum class PIForeignSwitches</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>[&lt;FlagsAttribute&gt;]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>type PIForeignSwitches</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Members

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0 - PL_FA_NOTHING: no flags.</td>
</tr>
<tr>
<td>NoTrace</td>
<td>1 - PL_FA_NOTRACE: Predicate cannot be seen in the tracer.</td>
</tr>
<tr>
<td>Transparent</td>
<td>2 - PL_FA_TRANSPARENT: Predicate is module transparent.</td>
</tr>
<tr>
<td>Nondeterministic</td>
<td>4 - PL_FA_NONDETERMINISTIC: Predicate is non-deterministic. See also PL_retry().</td>
</tr>
<tr>
<td>VarArgs</td>
<td>8 - PL_FA_VARARGS: (Default) Use alternative calling convention.</td>
</tr>
</tbody>
</table>
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
These are the namespace comments for **SbsSW.SwiPICs.Exceptions**.

The namespace SbsSW.SwiPICs.Exceptions provides the Exception classes to catch a prolog exception see [SWI-Prolog Manual - 4.9 ISO compliant Exception handling](https://swi-prolog.github.io/prolog-manual/index.html).

Prolog exceptions are mapped to C# exceptions using the subclass PException of Exception to represent the Prolog exception term.

All type-conversion functions of the interface raise Prolog-compliant exceptions, providing decent error-handling support at no extra work for the programmer.

For some commonly used exceptions, subclasses of PException have been created to exploit both their constructors for easy creation of these exceptions as well as selective trapping in C#.

Currently, these are **PlTypeException** and **PlDomainException**.

### Types

<table>
<thead>
<tr>
<th>Icon</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌏</td>
<td><strong>PlDomainException</strong></td>
<td>A domain exception expresses that a term satisfies the basic Prolog type expected, but is unacceptable to the restricted domain expected by some operation.</td>
</tr>
<tr>
<td>🌍</td>
<td><strong>PlException</strong></td>
<td>This class is the base class to catch</td>
</tr>
</tbody>
</table>
exceptions thrown by prolog in C#.

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PILibException</strong></td>
<td>This exception is thrown if something in the interface went wrong.</td>
</tr>
<tr>
<td><strong>PITypeException</strong></td>
<td>A type error expresses that a term does not satisfy the expected basic Prolog type.</td>
</tr>
</tbody>
</table>

 Remarks

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de
A domain exception expresses that a term satisfies the basic Prolog type expected, but is unacceptable to the restricted domain expected by some operation.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SerializableAttribute] public class PlDomainException : PlException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;SerializableAttribute&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public ref class PlDomainException Inherits PlException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SerializableAttribute]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public ref class PlDomainException : public PlException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[&lt;SerializableAttribute&gt;]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type PlDomainException =</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inherit PlException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Members

<table>
<thead>
<tr>
<th>All Members</th>
<th>Constructors</th>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔ Public</td>
<td>✔ Instance</td>
<td>✔ Declared</td>
<td></td>
</tr>
<tr>
<td>✔ Protected</td>
<td>✔ Static</td>
<td>✔ Inherited</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlDomainException()</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td>PlDomainException(String)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td>PlDomainException(String, Exception)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td>PlDomainException(SerializationInfo, StreamingContext)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td>PlDomainException(PlTerm)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td>Data</td>
<td>Gets a collection of key/value pairs that provides additional user-defined information about the exception.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines whether the specified <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a> is equal to the current <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.</td>
</tr>
<tr>
<td><strong>Finalize()</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>GetBaseException()</strong></td>
<td>When overridden in a derived class, returns the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.exception">Exception</a> that is the root cause of one or more subsequent exceptions.</td>
</tr>
<tr>
<td><strong>GetHashCode()</strong></td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td><strong>GetObjectData(SerializationInfo, StreamingContext)</strong></td>
<td>When overridden in a derived class, sets the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.serialization.serializationinfo">SerializationInfo</a> with information about the exception.</td>
</tr>
<tr>
<td><strong>GetType()</strong></td>
<td>Gets the runtime type of the current instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the Exception instance that caused the current exception.</td>
</tr>
<tr>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception.</td>
</tr>
<tr>
<td>MessagePl</td>
<td>Provide sometimes some additional information about the exceptions reason.</td>
</tr>
<tr>
<td>PlThrow()</td>
<td>Generate an exception (as</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>throw/1()</td>
<td>and return false. (Inherited from PlException.)</td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Gets the method that throws the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>Term</td>
<td>Get the PlTerm of this exception. (Inherited from PlException.)</td>
</tr>
<tr>
<td>Throw()</td>
<td>Throw this PlException. (Inherited from PlException.)</td>
</tr>
<tr>
<td>ToString()</td>
<td>The exception is translated into a message as produced by print_message/2. The character data is stored in</td>
</tr>
</tbody>
</table>
Examples
For example, the standard Prolog open/3 call expect an IO-Mode (read, write, append, ...). If an integer is provided, this is a type error, if an atom other than one of the defined IO-modes is provided it is a domain error.

```java
public void prolog_domain_exception_sample()
{
    try {
        Assert.IsTrue(PlQuery.PlCall("open(temp_kill,
nonsens,
F)"
    } catch (PlDomainException ex)
    {
        Assert.AreEqual("open/3: Domain error: `io_mode'	expected,	found	`nonsens'
    }
}
```

Inheritance Hierarchy
- Object
  - Exception
    - PlException
      - PlDomainException

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PIDomainException()</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td></td>
<td>PIDomainException(String)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td></td>
<td>PIDomainException(String, Exception)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td></td>
<td>PIDomainException(SerializationInfo, StreamingContext)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
<tr>
<td></td>
<td>PIDomainException(PlTerm)</td>
<td>This class is the base class to catch exceptions thrown by prolog in C#.</td>
</tr>
</tbody>
</table>
This class is the base class to catch exceptions thrown by Prolog in C#.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlDomainException()</td>
<td>Public Sub New</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New</td>
<td></td>
<td>new : unit -&gt; PlDomainException</td>
<td></td>
</tr>
</tbody>
</table>

- **Examples**

```csharp
public void prolog_exception_sample()
{
    const string exceptionText = "test_exception"
    Assert.IsTrue(PlQuery.PlCall("assert( (test_throw :- throw(
    try
    {
        Assert.IsTrue(PlQuery.PlCall("test_throw")
    }
    catch (PlException ex)
    {
        Assert.AreEqual(exceptionText, ex.Term.ToString());
    }
```
Assert.AreEqual("Unknown message: "+exceptionText, tx.Message);

See Also

PITypeException
SWI-Prolog Manual - 4.9 ISO compliant Exception handling

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
This class is the base class to catch exceptions thrown by prolog in C#.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public PlDomainException(  
   PlTerm term  
) |
| Public Sub New (  
   term As PlTerm  
) |
| public:  
PlDomainException(  
   PlTerm term  
) |
| new:  
   term : PlTerm -> PlDomainException |

- **Parameters**

  term (**PlTerm**)

  [Missing <param name="term"/> documentation for "M:SbsSW.SwiPICs.Exceptions.PlDomainException.#ctor(SbsSW.SwiPICs.PlTerm)"

- **Examples**
public void prolog_exception_sample()
{
    const string exceptionText = "test_exception"
    Assert.IsTrue(PlQuery.PlCall("assert( (test_throw
    try
    {
        Assert.IsTrue(PlQuery.PlCall("test_throw")
    }
    catch (PlException ex)
    {
        Assert.AreEqual(exceptionText, ex.Term.ToString());
        Assert.AreEqual("Unknown message: " +
    }

--- See Also
- PlTypeException
- SWI-Prolog Manual - 4.9 ISO compliant Exception handling

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
This class is the base class to catch exceptions thrown by prolog in C#.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected PlDomainException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SerializationInfo info,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StreamingContext context</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protected Sub New (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>info As SerializationInfo,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>context As StreamingContext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>protected:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlDomainException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SerializationInfo^ info,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StreamingContext context</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new :</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>info : SerializationInfo *</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>context : StreamingContext -&gt; PlDomain</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
info (SerializationInfo)


corext (StreamingContext)


- Examples

```csharp
public void prolog_exception_sample()
{
    const string exceptionText = "test_exception"
    Assert.IsTrue(PlQuery.PlCall("assert( (try
    {
        Assert.IsTrue(PlQuery.PlCall("test_throw"
    } catch (PlException ex)
    {
        Assert.AreEqual(exceptionText, ex.Term.ToString());
        Assert.AreEqual("Unknown message: "+
    }
}
```

- See Also

PlTypeException
SWI-Prolog Manual - 4.9 ISO compliant Exception handling

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de
Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
This class is the base class to catch exceptions thrown by prolog in C#.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlDomainException(string message)</td>
<td></td>
<td></td>
<td>public Sub New (message As String)</td>
</tr>
<tr>
<td>public: PlDomainException(String^ message)</td>
<td></td>
<td></td>
<td>new : message : string -&gt; PlDomainException</td>
</tr>
</tbody>
</table>

- **Parameters**

  - `message (String)`

    [Missing <param name="message"/> documentation for "M:SbsSW.SwiPICs.Exceptions.PlDomainException.#ctor(Sys..."

- **Examples**
public void prolog_exception_sample()
{
    const string exceptionText = "test_exception"
    Assert.IsTrue(PlQuery.PlCall("assert( (test_"
    try
    {
        Assert.IsTrue(PlQuery.PlCall("test_th"
    }
    catch (PlException ex)
    {
        Assert.AreEqual(exceptionText, ex.Ter
        Assert.AreEqual("Unknown message: " +
    }
}

- See Also
  PlTypeException
  SWI-Prolog Manual - 4.9 ISO compliant Exception handling

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
This class is the base class to catch exceptions thrown by prolog in C#.

### Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public PlDomainException(string message, Exception innerException)</td>
<td>Public Sub New (message As String, innerException As Exception)</td>
<td>public: PlDomainException(String^ message, Exception^ innerException)</td>
<td>new : message : string * innerException : Exception -&gt; PlDomainException</td>
</tr>
</tbody>
</table>

#### Parameters

- **message (String)**
innerException (Exception)

Examples

```csharp
public void prolog_exception_sample()
{
    const string exceptionText = "test_exception"
    Assert.IsTrue(PlQuery.PlCall("assert( (test_throw :-
        try
        {
            Assert.IsTrue(PlQuery.PlCall("test_throw"
        } catch (PlException ex)
        {
            Assert.AreEqual(exceptionText, ex.Term.ToString());
            Assert.AreEqual("Unknown message: " +
```}

See Also

PITypeException

SWI-Prolog Manual - 4.9 ISO compliant Exception handling

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
This class is the base class to catch exceptions thrown by prolog in C#.

**Declaration Syntax**

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[SerializableAttribute]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>public class</strong> PlException : Exception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;SerializableAttribute&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Public Class</strong> PlException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Inherits</strong> Exception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SerializableAttribute]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>public ref class</strong> PlException : public Exception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;SerializableAttribute&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>type</strong> PlException =</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>class</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>inherit</strong> Exception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>end</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Members**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🍃</td>
<td>PlException()</td>
<td>Initializes a new instance of the</td>
</tr>
<tr>
<td>Method/Constructor</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>PlException(String)</code></td>
<td>Initializes a new instance of the class with a specified error message.</td>
<td></td>
</tr>
<tr>
<td><code>PlException(String, Exception)</code></td>
<td>Initializes a new instance of the class with a specified error message and a reference to the inner exception that is the cause of this exception.</td>
<td></td>
</tr>
<tr>
<td><code>PlException(SerializationInfo, StreamingContext)</code></td>
<td>Initializes a new instance of the class with serialized data.</td>
<td></td>
</tr>
<tr>
<td><code>PlException(PlTerm)</code></td>
<td>To catch a exception thrown by Prolog.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For a example see <code>PlException</code>.</td>
<td></td>
</tr>
<tr>
<td><code>Data</code></td>
<td>Gets a collection of key/value pairs that can be used to extend the context of the exception. (Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>Equals(Object)</code></td>
<td>Determines whether the specified object is equal to the current object.     (Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>Finalize()</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
<td></td>
</tr>
<tr>
<td>Method/Property</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>When overridden in a derived class, returns the <code>Exception</code> that is the root cause of one or more subsequent exceptions. (Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetObjectData</strong></td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the exception. (Overrides <code>Exception.GetObjectData(SerializationInfo, StreamingContext)</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the runtime type of the current instance. (Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>HelpLink</strong></td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>HResult</strong></td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>Gets the <code>Exception</code> instance that caused the current exception.</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current Object.</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception.</td>
<td></td>
</tr>
<tr>
<td>MessagePI</td>
<td>Sometimes provide some additional information about the exception.</td>
<td></td>
</tr>
<tr>
<td>PIThrow()</td>
<td>Generate an exception (as throw/1) and return false.</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the application or the object that causes the error.</td>
<td></td>
</tr>
<tr>
<td>StackTrace</td>
<td>Gets a string representation of the frames on the call stack at the time the exception was thrown.</td>
<td></td>
</tr>
<tr>
<td>TargetSite</td>
<td>Gets the method that throws the exception.</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>Get the PIException of this exception.</td>
<td></td>
</tr>
<tr>
<td>Throw()</td>
<td>Throw this PIException.</td>
<td></td>
</tr>
</tbody>
</table>
The exception is translated into a message as produced by print_message/2. The character data is stored in a ring.

(Overrides Exception.ToString())

### Examples

```csharp
public void prolog_exception_sample()
{
    const string exceptionText = "test_exception"
    Assert.IsTrue(PlQuery.PlCall("assert( (test_throw :- throw(
    try
    {
        Assert.IsTrue(PlQuery.PlCall("test_throw"
    }
    catch (PlException ex)
    {
        Assert.AreEqual(exceptionText, ex.Term.ToString());
        Assert.AreEqual("Unknown message: " +
    }
}
```

### Inheritance Hierarchy

- **Object**
  - **Exception**
    - **PlException**
      - **PlDomainException**
      - **PlTypeException**

### See Also
PlTypeException
SWI-Prolog Manual - 4.9 ISO compliant Exception handling

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
### Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Exception class" /></td>
<td><strong>PlException()</strong></td>
<td>Initializes a new instance of the <code>Exception</code> class.</td>
</tr>
<tr>
<td><img src="image" alt="Exception class" /></td>
<td><strong>PlException(String)</strong></td>
<td>Initializes a new instance of the <code>Exception</code> class with a specified error message.</td>
</tr>
<tr>
<td><img src="image" alt="Exception class" /></td>
<td><strong>PlException(String, Exception)</strong></td>
<td>Initializes a new instance of the <code>Exception</code> class with a specified error message and a reference to the inner exception that is the cause of this exception.</td>
</tr>
<tr>
<td><img src="image" alt="Exception class" /></td>
<td><strong>PlException(SerializationInfo, StreamingContext)</strong></td>
<td>Initializes a new instance of the <code>Exception</code> class with serialized data.</td>
</tr>
<tr>
<td><img src="image" alt="Exception class" /></td>
<td><strong>PlException(PlTerm)</strong></td>
<td>To catch an exception thrown by Prolog. For a example see <code>PlException</code>.</td>
</tr>
</tbody>
</table>
by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PIException Constructor**

**SwiPICs interface** ► **SbsSW.SwiPICs.Exceptions** ► **PIException** ► **PIException()**

Initializes a new instance of the **Exception** class.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public PIException()</code></td>
<td><code>Public Sub New</code></td>
<td><code>public:</code></td>
<td><code>new : unit -&gt; PIException</code></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PIException Constructor (term)**

**SwiPICs interface** ★ **SbsSW.SwiPICs.Exceptions** ★ **PIException** ★

PIException(PlTerm)

To catch a exception thrown by prolog

For a example see PIException.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public</code> PIException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>PlTerm</code> <code>term</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Public Sub New</code> (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>term As PlTerm</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public:</code> PLException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>PlTerm</code> <code>term</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>new:</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>term : PlTerm -&gt; PIException</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameters

**term (PlTerm)**

A PlTerm containing the Prolog exception

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de
Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**PIException Constructor (info, context)**

Initializes a new instance of the `Exception` class with serialized data.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| protected PlException(  
  SerializationInfo `info`,  
  StreamingContext `context`  
)  
Protected Sub New (  
  `info As SerializationInfo`,  
  `context As StreamingContext`  
)  
protected:  
PlException(  
  SerializationInfo^ `info`,  
  StreamingContext `context`  
)  
new :  
  `info : SerializationInfo *`  
  `context : StreamingContext`  
  -> PlExcep |

### Parameters

**info (SerializationInfo)**

The `SerializationInfo` that holds the serialized object data about the exception being thrown.
context (StreamingContext)

The StreamingContext that contains contextual information about the source or destination.

 Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>The info parameter is null.</td>
</tr>
<tr>
<td>SerializationException</td>
<td>The class name is null or HRESULT is zero (0).</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
PlException Constructor (message)

PlException(String)

Initializes a new instance of the Exception class with a specified error message.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
  | `public PlException(
  | `string message
  | `)
  | `Public Sub New (
  | `message As String
  | `)
  | `public:
  | `PlException(
  | `String^ message
  | `)
  | `new :
  | `message : string -> PlException

- **Parameters**

  *message (String)*

  The message that describes the error.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
PlException Constructor (message, innerException)

SwiPIcs interface ► SbsSW.SwiPIcs.Exceptions ► PlException ►
PlException(String, Exception)

Initializes a new instance of the Exception class with a specified error message and a reference to the inner exception that is the cause of this exception.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlException(string message, Exception innerException)</td>
<td>Public Sub New ( message As String, innerException As Exception)</td>
<td>public: PlException( String^ message, Exception^ innerException )</td>
<td>new : message : string * innerException : Exception -&gt; PlExcep</td>
</tr>
</tbody>
</table>

- Parameters
message (String)
The error message that explains the reason for the exception.

innerException (Exception)
The exception that is the cause of the current exception, or a null reference (Nothing in Visual Basic) if no inner exception is specified.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**GetObjectData Method (info, context)**

SwiPlCs interface ➤ SbsSW.SwiPlCs.Exceptions ➤ PlException ➤

GetObjectData(SerializationInfo, StreamingContext)

When overridden in a derived class, sets the SerializationInfo with information about the exception.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public override void GetObjectData(</td>
<td>SerializationInfo info, StreamingContext context )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Overrides Sub GetObjectData (</td>
<td>info As SerializationInfo, context As StreamingContext )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: virtual void GetObjectData(</td>
<td>SerializationInfo^ info, StreamingContext context ) override</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstract GetObjectData :</td>
<td>info : SerializationInfo * context : StreamingContext -&gt; unit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>override GetObjectData :</td>
<td>info : SerializationInfo * context : StreamingContext -&gt; unit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- **Parameters**

  *info* (*SerializationInfo*)
  The *SerializationInfo* that holds the serialized object data about the exception being thrown.

  *context* (*StreamingContext*)
  The *StreamingContext* that contains contextual information about the source or destination.

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>The <em>info</em> parameter is a null reference (Nothing in Visual Basic).</td>
</tr>
<tr>
<td>SecurityException</td>
<td>The caller does not have the required permission.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Message

Property

SwiPlCs interface ► SbsSW.SwiPlCs.Exceptions ► PLEException ►

Message

Gets a message that describes the current exception.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public override string Message { get; }</td>
<td>Public Overrides ReadOnly Property Message As Get</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public: virtual property String^ Message { String^ get () override; }</td>
<td></td>
<td>abstract Message : string with get override Message : string with get</td>
<td></td>
</tr>
</tbody>
</table>

- Return Value

String

The error message that explains the reason for the exception, or an empty string("").

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**MessagePl Property**

*SwiPICs interface* ▶ *SbsSW.SwiPICs.Exceptions* ▶ *PIException* ▶

**MessagePl**

provide sometimes some additional information about the exceptions reason.

## Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public string MessagePl { get; }</code></td>
<td><code>Public ReadOnly Property MessagePl As String Get</code></td>
<td><code>public: property String^ MessagePl { String^ get (); }</code></td>
<td><code>member MessagePl : string with get</code></td>
</tr>
</tbody>
</table>

## Property Value

*String*

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Method

SwiPLCs interface ► SbsSW.SwiPLCs.Exceptions ► PIException ►
PIThrow()

Generate an exception (as throw/1) and return false.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public bool PlThrow()</td>
<td>Public Function PlThrow As Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool PlThrow()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member PlThrow : unit -&gt; bool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Return Value

Boolean
Generate an exception (as throw/1) and return false.

- Remarks

Used in the PREDICATE() wrapper to pass the exception to Prolog. See PL_raise_exception().

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Get the **PlTerm** of this exception.

### Declaration Syntax

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public</strong></td>
<td><strong>PlTerm</strong> Term { get; }</td>
<td><strong>Public</strong> ReadOnly <strong>Property</strong> Term As <strong>PlTerm</strong></td>
<td><strong>public</strong>: property <strong>PlTerm</strong> Term { \n</td>
<td><strong>get</strong>; }</td>
</tr>
<tr>
<td></td>
<td><strong>Public</strong> <strong>ReadOnly</strong> <strong>Property</strong> <strong>Term</strong> <strong>As</strong> <strong>PlTerm</strong></td>
<td>\n<strong>Get</strong></td>
<td>\n<strong>get</strong> \n</td>
<td><strong>member</strong> <strong>Term</strong>: <strong>PlTerm</strong> <strong>with</strong> <strong>get</strong></td>
</tr>
</tbody>
</table>

### Property Value

**PlTerm**

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta at SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0

(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**Method**

SwiPlCs interface ➤ SbsSW.SwiPlCs.Exceptions ➤ PlException ➤ Throw()

Throw this PlException.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void Throw()</td>
<td>Public Sub Throw</td>
<td>public: void Throw()</td>
<td>member Throw : unit -&gt; unit</td>
</tr>
</tbody>
</table>

- **Remarks**

  see [http://www.swi-prolog.org/packages/pl2cpp.html#cppThrow()](http://www.swi-prolog.org/packages/pl2cpp.html#cppThrow())

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta at SBS-Softwaresysteme.de)

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
The exception is translated into a message as produced by print_message/2. The character data is stored in a ring.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public override string ToString()</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Public Overrides Function ToString As String</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>public: virtual String^ ToString() override</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>abstract ToString : unit -&gt; string</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>override ToString : unit -&gt; string</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Return Value

**String**
A textual description of the Exception

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@sbs-softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
This exception is thrown if something in the interface went wrong.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[SerializableAttribute]</code></td>
<td><code>public class PlLibException : Exception</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;SerializableAttribute&gt;</code></td>
<td><code>Public Class PlLibException Inherits Exception</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>[SerializableAttribute]</code></td>
<td><code>public ref class PlLibException : public Exception</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;SerializableAttribute&gt;</code></td>
<td><code>type PlLibException = class inherit Exception end</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Members**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td><code>PlLibException()</code></td>
<td>Initializes a new instance of the class.</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>PlLibException(String)</code></td>
<td>Initializes a new instance of the class with a specified error message.</td>
<td></td>
</tr>
<tr>
<td><code>PlLibException(String, Exception)</code></td>
<td>Initializes a new instance of the class with a specified error message and a reference to the inner exception that is the cause of this exception.</td>
<td></td>
</tr>
<tr>
<td><code>PlLibException(SerializationInfo, StreamingContext)</code></td>
<td>Initializes a new instance of the class with serialized data.</td>
<td></td>
</tr>
<tr>
<td><code>Data</code></td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>[Exception](http://docs.microsoft.com/en-us/dotnet/api/system.exception)</code>.</td>
<td></td>
</tr>
<tr>
<td><code>Equals(Object)</code></td>
<td>Determines whether the specified object is equal to the current object.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>[Object](http://docs.microsoft.com/en-us/dotnet/api/system.object).</code></td>
<td></td>
</tr>
<tr>
<td><code>Finalize()</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>[Object](http://docs.microsoft.com/en-us/dotnet/api/system.object).</code></td>
<td></td>
</tr>
<tr>
<td><code>GetBaseException()</code></td>
<td>When overridden in a derived class, returns the <code>Exception</code> that is the root cause of one or more subsequent exceptions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>[Exception](http://docs.microsoft.com/en-us/dotnet/api/system.exception).</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GetHashCode()</td>
<td>Serves as a hash function for a type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td>GetObjectData(SerializationInfo,</td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the</td>
<td></td>
</tr>
<tr>
<td>StreamingContext)</td>
<td>exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Overrides <code>Exception.GetObjectData(SerializationInfo, StreamingContext)</code>.)</td>
<td></td>
</tr>
<tr>
<td>GetType()</td>
<td>Gets the runtime type of the current instance.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the <code>Exception</code> instance that caused the current exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the exception.</td>
<td></td>
</tr>
</tbody>
</table>
| **Source** | Gets or sets the name of the application or the object that causes the error.  
(Inherited from Exception.) |
| **StackTrace** | Gets a string representation of the frames on the call stack at the time the exception was thrown.  
(Inherited from Exception.) |
| **TargetSite** | Gets the method that throws the current exception.  
(Inherited from Exception.) |
| **ToString()** | Creates and returns a string representation of the current exception.  
(Inherited from Exception.) |

### Inheritance Hierarchy

- **Object**
  - **Exception**
    - **PlLibException**

---

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
**Swi-cs-pl** - A CSharp class library to connect .NET languages with SWI-Prolog

**PlLibException Constructor**

**PlLibException()**

**Members**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Icon" /></td>
<td>PlLibException()</td>
<td>Initializes a new instance of the <strong>Exception</strong> class.</td>
</tr>
<tr>
<td><img src="image2" alt="Icon" /></td>
<td>PlLibException(String)</td>
<td>Initializes a new instance of the <strong>Exception</strong> class with a specified error message.</td>
</tr>
<tr>
<td><img src="image3" alt="Icon" /></td>
<td>PlLibException(String, Exception)</td>
<td>Initializes a new instance of the <strong>Exception</strong> class with a specified error message and a reference to the inner exception that is the cause of this exception.</td>
</tr>
<tr>
<td><img src="image4" alt="Icon" /></td>
<td>PlLibException(SerializationInfo, StreamingContext)</td>
<td>Initializes a new instance of the <strong>Exception</strong> class with serialized data.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to **Lesta at SBS-Softwaresysteme.de**

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
PlLibException Constructor

Initializes a new instance of the Exception class.

- **Declaration Syntax**

  C#  

  ```csharp
  public PlLibException()
  ```

  Visual Basic

  ```vbnet
  Public Sub New
  ```

  Visual C++

  ```cpp
  public:
  PlLibException()
  ```

  F#

  ```fsharp
  new : unit -> PlLibException
  ```

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

PlLibException Constructor (info, context)

SwiPlCs interface ► SbsSW.SwiPlCs.Exceptions ► PlLibException ►

PlLibException(SerializationInfo, StreamingContext)

C#

Initializes a new instance of the Exception class with serialized data.

Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected PlLibException( SerializationInfo info, StreamingContext context )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protected Sub New ( info As SerializationInfo, context As StreamingContext )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>protected: PlLibException( SerializationInfo^ info, StreamingContext context )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new : info : SerializationInfo * context : StreamingContext -&gt; PlLibEx</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters

info (SerializationInfo)

The SerializationInfo that holds the serialized object data about the exception being thrown.
*context* (**StreamingContext**)  
The **StreamingContext** that contains contextual information about the source or destination.

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>The <em>info</em> parameter is null.</td>
</tr>
<tr>
<td>SerializationException</td>
<td>The class name is null or <em>HRESULT</em> is zero (0).</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
## PlLibException Constructor (message)

PlLibException(String)

Initializes a new instance of the Exception class with a specified error message.

### Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public PlLibException(</code></td>
<td>`Public Sub New (</td>
<td>`public: PlLibException(</td>
<td>`new :</td>
</tr>
<tr>
<td>`    string message</td>
<td><code>                  </code>message As String)`</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>)</code></td>
<td><code>)</code></td>
<td><code>String^ message)</code></td>
<td><code>message : string -&gt; PlLibException</code></td>
</tr>
</tbody>
</table>

### Parameters

`message (String)`

The message that describes the error.

---

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
PlLibException Constructor (message, innerException)

Initializes a new instance of the Exception class with a specified error message and a reference to the inner exception that is the cause of this exception.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public PlLibException(
    string message,
    Exception innerException
) |
| Public Sub New ( |
    message As String,
    innerException As Exception
) |
| public:
PlLibException(
    String^ message,
    Exception^ innerException
) |
| new :
    message : string *
    innerException : Exception -> PlLibEx |
message (String)
The error message that explains the reason for the exception.

innerException (Exception)
The exception that is the cause of the current exception, or a null reference (Nothing in Visual Basic) if no inner exception is specified.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

GetObjectData Method (info, context)

When overridden in a derived class, sets the SerializationInfo with information about the exception.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public override void GetObjectData(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SerializationInfo info,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StreamingContext context</td>
<td>)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Public Overrides Sub GetObjectData ( |
  info As SerializationInfo, |
  context As StreamingContext |
)

public: |
virtual void GetObjectData( |
  SerializationInfo^ info, |
  StreamingContext context |
) override |

abstract GetObjectData :
  info : SerializationInfo *
  context : StreamingContext -> unit
override GetObjectData :
  info : SerializationInfo *
  context : StreamingContext -> unit
- **Parameters**

  **info (SerializationInfo)**
  The SerializationInfo that holds the serialized object data about the exception being thrown.

  **context (StreamingContext)**
  The StreamingContext that contains contextual information about the source or destination.

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>The info parameter is a null reference (Nothing in Visual Basic).</td>
</tr>
<tr>
<td>SecurityException</td>
<td>The caller does not have the required permission.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PITypeException Class**

SwiPIcs interface ► SbsSW.SwiPIcs.Exceptions ► PITypeException

A type error expresses that a term does not satisfy the expected basic Prolog type.

- **Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[SerializableAttribute]</code> public class PITypeException : PlException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;SerializableAttribute&gt;</code> Public Class PITypeException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Inherits PlException</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>[SerializableAttribute]</code> public ref class PITypeException : public PlException</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>[&lt;SerializableAttribute&gt;] type PITypeException =</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>class</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>inherit PlException</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>end</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Members**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📜</td>
<td>PITypeException()</td>
<td>Initializes a new instance of</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>PlTypeException(String)</code></td>
<td>Initializes a new instance of the <code>Exception</code> class with a specified error message.</td>
<td></td>
</tr>
<tr>
<td><code>PlTypeException(String, Exception)</code></td>
<td>Initializes a new instance of the <code>Exception</code> class with a specified error message and a reference to the inner exception that is the cause of this exception.</td>
<td></td>
</tr>
<tr>
<td><code>PlTypeException(SerializationInfo, StreamingContext)</code></td>
<td>Initializes a new instance of the <code>Exception</code> class with serialized data.</td>
<td></td>
</tr>
<tr>
<td><code>PlTypeException(PlTerm)</code></td>
<td>To catch a exception thrown by prolog</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For a example see <code>PlException</code>.</td>
<td></td>
</tr>
<tr>
<td><code>PlTypeException(String, PlTerm)</code></td>
<td>Creates an ISO standard Prolog error term expressing the expected type and actual term that does not satisfy this type.</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Exception.)</td>
<td></td>
</tr>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>Finalize()</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetBaseException()</strong></td>
<td>When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Exception.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetHashCode()</strong></td>
<td>Serves as a hash function for a particular type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetObjectData(SerializationInfo, StreamingContext)</strong></td>
<td>When overridden in a derived class, sets the SerializationInfo with information about the exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from PException.)</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GetType()</td>
<td>Gets the runtime type of the current instance.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Exception.)</td>
<td></td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Exception.)</td>
<td></td>
</tr>
<tr>
<td>HRESULT</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Exception.)</td>
<td></td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the Exception instance that caused the current exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Exception.)</td>
<td></td>
</tr>
<tr>
<td>MemberwiseClone()</td>
<td>Creates a shallow copy of the current Object.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from PLEException.)</td>
<td></td>
</tr>
<tr>
<td>MessagePI</td>
<td>provide sometimes some additional information about the exceptions reason.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Inherited from PLEException.)</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>PlThrow()</code></td>
<td>Generate an exception (as throw/1) and return <code>false</code>. (Inherited from <code>PlException</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>Source</code></td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>StackTrace</code></td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>TargetSite</code></td>
<td>Gets the method that throws the current exception. (Inherited from <code>Exception</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>Term</code></td>
<td>Get the <code>PlTerm</code> of this exception. (Inherited from <code>PlException</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>Throw()</code></td>
<td>Throw this <code>PlException</code>. (Inherited from <code>PlException</code>.)</td>
<td></td>
</tr>
<tr>
<td><code>ToString()</code></td>
<td>The exception is translated into a message as produced by <code>print_message/2</code>. The character data is stored in a ring. (Inherited from <code>PlException</code>.)</td>
<td></td>
</tr>
</tbody>
</table>

**Examples**
This sample demonstrate how to catch a PlTypeException in C# that is thrown somewhere int the prolog code.

```csharp
public void prolog_type_exception_sample()
{
    try
    {
        Assert.IsTrue(PlQuery.PlCall("sumlist
    }
    catch (PlTypeException ex)
    {
        Assert.AreEqual("is/2: Arithmetic: `e
    }
}
```

- Inheritance Hierarchy

  Object
  ├ Exception
  │  └ PlException
  │     └ PlTypeException

- See Also

  PlTypeException

  SWI-Prolog Manual - 4.9 ISO compliant Exception handling

  by Uwe Lesta, SBS-Softwaresysteme GmbH
  Send comments on this topic to Lesta at SBS-Softwaresysteme.de

  Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
## Members

<table>
<thead>
<tr>
<th>Icon</th>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PITypeException()</td>
<td>Initializes a new instance of the <code>Exception</code> class.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PITypeException(String)</td>
<td>Initializes a new instance of the <code>Exception</code> class with a specified error message.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PITypeException(String, Exception)</td>
<td>Initializes a new instance of the <code>Exception</code> class with a specified error message and a reference to the inner exception that is the cause of this exception.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PITypeException(SerializationInfo, StreamingContext)</td>
<td>Initializes a new instance of the <code>Exception</code> class with serialized data.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>PITypeException(PlTerm)</td>
<td>To catch a exception thrown by prolog For a example see <code>PlException</code>.</td>
</tr>
<tr>
<td>PlTypeException(String, PITerm)</td>
<td>Creates an ISO standard Prolog error term expressing the expected type and actual term that does not satisfy this type.</td>
<td></td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

PlTypeException Constructor

SwiPIcs interface ► SbsSW.SwiPIcs.Exceptions ► PlTypeException
► PlTypeException()

Initializes a new instance of the Exception class.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| `public PlTypeException()` | `Public Sub New` | `public:
PlTypeException()` | `new : unit -> PlTypeException` |

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPIcs (Module: SwiPIcs.dll) Version: 1.1.60605.0 (1.1.60605.0)
To catch a exception thrown by prolog

For a example see PlException.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTypeException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term As PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTypeException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm term</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new :</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term : PlTerm -&gt; PlTypeException</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Parameters

  term (PlTerm)
  A PlTerm containing the Prolog exception

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de
Assembly: SwiPLCs (Module: SwiPLCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

PITypeException Constructor (info, context)

SwiPICs interface ► SbsSW.SwiPICs.Exceptions ► PITypeException ► PITypeException(SerializationInfo, StreamingContext)

Initializes a new instance of the Exception class with serialized data.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| protected PlTypeException(  
  SerializationInfo info,  
  StreamingContext context  
)  |
| Protected Sub New (  
  info As SerializationInfo,  
  context As StreamingContext  
)  |
| protected:  
  PlTypeException(  
    SerializationInfo^ info,  
    StreamingContext context  
)  |
| new :  
  info : SerializationInfo *  
  context : StreamingContext -> PlTypeEx... |

- Parameters

  *info (SerializationInfo)*

  The SerializationInfo that holds the serialized object data about the exception being thrown.
**context** (**StreamingContext**)  
The **StreamingContext** that contains contextual information about the source or destination.

- **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentNullException</code></td>
<td>The <code>info</code> parameter is null.</td>
</tr>
<tr>
<td><code>SerializationException</code></td>
<td>The class name is null or <code>HResult</code> is zero (0).</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
PlTypeException Constructor (message)

Initializes a new instance of the `Exception` class with a specified error message.

**Declaration Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public PlTypeException(String message)</code></td>
<td><code>Public Sub New (message As String)</code></td>
<td><code>public: PlTypeException(String^ message)</code></td>
<td><code>new : message : string -&gt; PlTypeException</code></td>
</tr>
</tbody>
</table>

**Parameters**

`message (String)`

The message that describes the error.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0
SwiPlCs interface ► SbsSW.SwiPlCs.Exceptions ► PlTypeException

PlTypeException constructor (expected, actual)

Creates an ISO standard Prolog error term expressing the expected type and actual term that does not satisfy this type.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PlTypeException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>string expected,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm actual</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Sub New (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expected As String,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>actual As PlTerm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTypeException(</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>String^ expected,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlTerm actual</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new :</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expected : string *</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>actual : PlTerm -&gt; PlTypeException</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Parameters

  expected (String)
The type which was expected

*actual* (**PlTerm**)
The actual term

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPlCs (Module: SwiPlCs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

PlTypeException Constructor (message, innerException)

SwiPIcs interface ► SbsSW.SwiPIcs.Exceptions ► PlTypeException ► PlTypeException(String, Exception)

Initializes a new instance of the Exception class with a specified error message and a reference to the inner exception that is the cause of this exception.

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| ```
public PlTypeException(
    string message,
    Exception innerException
)
``` |
| ```
Public Sub New (
    message As String,
    innerException As Exception
)
``` |
| ```
public:
PlTypeException(
    String^ message,
    Exception^ innerException
)
``` |
| ```
new :
    message : string *
    innerException : Exception -> PlTypeException
``` |

- Parameters
**message** *(String)*

The error message that explains the reason for the exception.

**innerException** *(Exception)*

The exception that is the cause of the current exception, or a null reference (Nothing in Visual Basic) if no inner exception is specified.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
The namespace SbsSW.SwiPICs.Streams provides the delegates to redirect the read and write functions of the SWI-Prolog IO Streams.

When `Initialize(String[])` is called the *Sinput->functions.read is replaced by the .NET method 'Sread_function' and *Sinput->functions.write by 'Swrite_funktion'.

For further examples see the methods `SetStreamFunctionWrite(PlStreamType, DelegateStreamWriteFunction)` and `SetStreamFunctionRead(PlStreamType, DelegateStreamReadFunction)`

### Types

<table>
<thead>
<tr>
<th>Icon</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📨</td>
<td><strong>DelegateStreamReadFunction</strong></td>
<td>See [SetStreamFunctionRead(PlStream, DelegateStreamReadFunction)](SetStreamFunctionRead(PlStream, DelegateStreamReadFunction))</td>
</tr>
<tr>
<td>📨</td>
<td><strong>DelegateStreamWriteFunction</strong></td>
<td>See [SetStreamFunctionWrite(PlStream, DelegateStreamWriteFunction)](SetStreamFunctionWrite(PlStream, DelegateStreamWriteFunction))</td>
</tr>
<tr>
<td>📨</td>
<td><strong>PlStreamType</strong></td>
<td>The standard SWI-Prolog stream input output error</td>
</tr>
</tbody>
</table>

---

Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

SbsSW.SwiPICs.Streams Namespace

SwiPICs interface ► SbsSW.SwiPICsStreams
Remarks

**Note:**
The reason for this is debugging.

Examples

```csharp
static internal long Swrite_function(IntPtr handle, {
    string s = buf.Substring(0, (int)bufsize);
    Console.Write(s);
    System.Diagnostics.Trace.WriteLine(s);
    return bufsize;
}

static internal long Sread_function(IntPtr handle, IntPtr buf, {
    throw new PlLibException("SwiPlCs: Prolog
```

See Also

`SetStreamFunctionRead(PlStreamType, DelegateStreamReadFunction)`

by Uwe Lesta, SBS-Softwaresysteme GmbH

Send comments on this topic to Lesta at SBS-Softwaresysteme.de
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

DelegateStreamReadFunction

See SetStreamFunctionRead(PlStreamType, DelegateStreamReadFunction)

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public delegate long DelegateStreamReadFunction(IntPtr handle, IntPtr buffer, long bufferSize) )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Delegate Function DelegateStreamReadFunction handle As IntPtr, buffer As IntPtr, bufferSize As Long As Long</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public delegate long long DelegateStreamReadFunction(IntPtr handle, IntPtr buffer, long long bufferSize) )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>type DelegateStreamReadFunction = delegate of handle : IntPtr *, buffer : IntPtr *, bufferSize : int64 -&gt; int64</td>
</tr>
</tbody>
</table>
- Parameters

  **handle** ([IntPtr](https://docs.microsoft.com/en-us/dotnet/api/system.IntPtr))
  
  A C stream handle. Simple ignore it.

  **buffer** ([IntPtr](https://docs.microsoft.com/en-us/dotnet/api/system.IntPtr))
  
  A pointer to a string buffer

  **bufferSize** ([Int64](https://docs.microsoft.com/en-us/dotnet/api/system.Int64))
  
  The size of the string buffer

- Return Value

  **Int64**
  
  A Delegate

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
SwiPlCs - A CSharp class library to connect .NET languages with SWI-Prolog

DelegateStreamWriteFunction Delegate
SwiPlCs interface ► SbsSW.SwiPlCs.Streams ► DelegateStreamWriteFunction

See SetStreamFunctionWrite(PlStreamType, DelegateStreamWriteFunction)

- Declaration Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public delegate long DelegateStreamWriteFunction(IntPtr handle, string buffer, long bufferSize) )</td>
<td>Public Delegate Function DelegateStreamWriteFunction handle As IntPtr, buffer As String, bufferSize As Long ) As Long</td>
<td>public delegate long long DelegateStreamWriteFunction IntPtr handle, String^ buffer, long long bufferSize )</td>
<td>type DelegateStreamWriteFunction = delegate of handle : IntPtr *, buffer : string *, bufferSize : int64 -&gt; int64</td>
</tr>
</tbody>
</table>
- **Parameters**

  *handle* ([IntPtr](#))
  A C stream handle. simple ignore it.

  *buffer* ([String](#))
  A pointer to a string buffer

  *bufferSize* ([Int64](#))
  The size of the string buffer

- **Return Value**

  Int64
  A Delegate

---

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta at SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0
(1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

**PIStreamType Enumeration**

[SwiPICs interface] ➤ [SbsSW.SwiPICs.Streams] ➤ PIStreamType

The standard SWI-Prolog streams (input output error)

- **Declaration Syntax**

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>Visual Basic</th>
<th>Visual C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>public enum PIStreamType</td>
<td>Public Enumeration PIStreamType</td>
<td>public enum class PIStreamType</td>
<td>type PIStreamType</td>
</tr>
</tbody>
</table>

- **Members**

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>0 - The standard input stream.</td>
</tr>
<tr>
<td>Output</td>
<td>1 - The standard input stream.</td>
</tr>
<tr>
<td>Error</td>
<td>1 - The standard error stream.</td>
</tr>
</tbody>
</table>

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to [Lesta at SBS-Softwaresysteme.de](mailto:Lesta@SBS-Softwaresysteme.de)

Assembly: SwiPICs (Module: SwiPICs.dll) Version: 1.1.60605.0 (1.1.60605.0)
Swi-cs-pl - A CSharp class library to connect .NET languages with SWI-Prolog

SwiPlCs interface

SbsSW.SwiPlCs

The online documentation home is [here](#).

This namespace SbsSW.SwiPlCs provides an .NET interface to [SWI-Prolog](#).

### Overview

Prolog variables are dynamically typed and all information is passed around using the C-interface type term_t which is an int. In C#, term_t is embedded in the lightweight struct `PlTerm`. Constructors and operator definitions provide flexible operations and integration with important C#-types (`string`, `int` and `double`).

The list below summarises the important classes / struct defined in the C# interface.

<table>
<thead>
<tr>
<th>class / struct</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PlEngine</code></td>
<td>A static class represents the prolog engine.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PlTerm</td>
<td>A struct representing prolog data.</td>
</tr>
<tr>
<td>PlTermV</td>
<td>A vector of PlTerm.</td>
</tr>
<tr>
<td>PIQuery</td>
<td>A class to query Prolog.</td>
</tr>
</tbody>
</table>

**SbsSW.SwiPICs.Callback**

The namespace SbsSW.SwiPICs.Callback provides the delegates to register .NET methods to be called from SWI-Prolog.

**SbsSW.SwiPICs.Exceptions**

These are the namespace comments for SbsSW.SwiPICs.Exceptions.

The namespace SbsSW.SwiPICs.Exceptions provides the Exception classes to catch a prolog exception see SWI-Prolog Manual - 4.9 ISO compliant Exception handling.

Prolog exceptions are mapped to C# exceptions using the subclass PlException of Exception to represent the Prolog exception term.

All type-conversion functions of the interface raise Prolog-compliant exceptions, providing decent error-handling support at no extra work for the programmer.

For some commonly used exceptions, subclasses of PlException have been...
created to exploit both their constructors for easy creation of these exceptions as well as selective trapping in C#.

Currently, these are `PITypeException` and `PIDomainException`.

**SbsSW.SwiPICs.Streams**

The namespace `SbsSW.SwiPICs.Streams` provides the delegates to redirect the read and write functions of the `SWI-Prolog` IO Streams.

When `Initialize(String[])` is called the *Sinput->functions.read is replaced by the .NET method 'Sread_function' and *Sinput->functions.write by 'Swrite_funktion'.

For further examples see the methods `SetStreamFunctionWrite(PIStreamType DelegateStreamWriteFunction)` and `SetStreamFunctionRead(PIStreamType DelegateStreamReadFunction)`.

by Uwe Lesta, SBS-Softwaresysteme GmbH
Send comments on this topic to Lesta at SBS-Softwaresysteme.de