

Programming the SimaPro COM interface

March 2010



Colophon

Title : Programming the SimaPro COM interface

Written by: PRé Consultants
Chris de Gelder & Michael Moore

Report version: 1.2.1
Date: Mar 2010
Language: English
Availability: PDF file

Copyright: © 2007-2010 PRé Consultants. All rights reserved.
PRé Consultants grants the right to print the PDF version of this manual. Parts of the manual may be reproduced only if a clear reference is made that PRé Consultants is the author.
The manual may not be used for commercial purposes.

Support: phone +31 33 4555022
fax +31 33 4555024
e-mail support@pre.nl
website www.pre.nl

Contents

1	INTRODUCTION	1
2	SUPPORTED ENVIRONMENTS AND OPERATING SYSTEMS	1
3	ARCHITECTURE	1
4	TECHNICAL SAMPLES	2
4.1	VBA (EXCEL / WORD)	2
4.2	DELPHI	3
4.3	PHP	4
4.4	C++	4
4.5	VB.NET SAMPLE	5
4.6	C# SAMPLE	6
5	REFERENCE	8
	INDEX	60

1 Introduction

SimaPro is a versatile LCA tool, but in some occasions you need more than a stand-alone tool. For example, if you want to analyze many data automatically you would like to integrate SimaPro in your business software.

In the SimaPro Developer version a COM-interface is available. This allows the user to control SimaPro from applications such as Excel, .NET applications, Delphi, PHP etc. Practical applications can be:

- Linking with your ERP system
- Analyzing a Bill of Materials from another software tool, such as a CAD/CAM system
- Exporting specific data to Excel
- Create processes from Excel sheets
- Create a website with results from SimaPro
- Create a website where users can enter data into processes in SimaPro.

2 Supported environments and operating systems

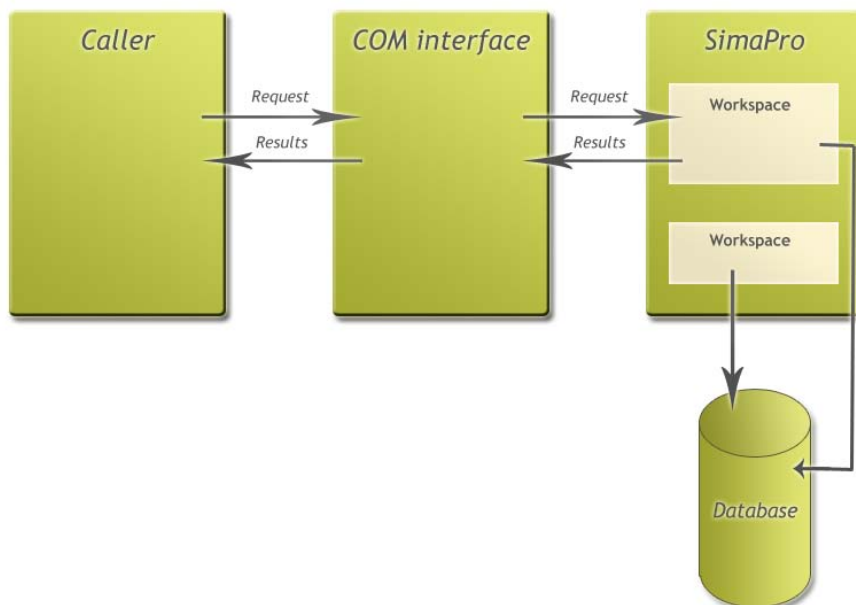
All software that can use the COM interface are supported, for example:

- Excel, Word (VBA)
- Delphi
- C++
- Visual Basic
- PHP
- Visual studio (C#, VB.NET)

Using SimaPro and the COM interface to support a web server requires more on the user rights level. Most easy is to use .NET, but ASP is also possible. COM is typical for Windows so it is only available on Windows Systems.

3 Architecture

Every caller has its own workspace within SimaPro. This workspace contains opened database, project and calculation results of the last calculation.



4 Technical samples

Below you find some simple examples. Note for all samples: adapt the server name, alias, database and project and process names to your own situation. If you use the COM interface with a single user version then adapt the connection properties according to the following guidelines:

- Server is not empty but "local server"
- Alias is the directory where your database files reside
- Login details are ignored, i.e. User name and password, if no password has been set
- The path to use for Alias and the name to use for Database can be found by opening the SimaPro application and selecting the menu option File->Open SimaPro Database
- The list of available projects can be found by opening the SimaPro application and selecting the menu option File->Open Project

4.1 VBA (Excel / Word)

This example creates a substance and 2 processes including parameters. One process gets input from the other.

```

Sub CreateProcess()
  Dim SP As SimaProServer
  Dim PC As Process
  Dim PC2 As Process
  Dim PL As ProcessLine
  Dim Param As ParamLine
  Dim Subs As Substance

  Set SP = New SimaProServer
  SP.Server = "nexusdb@192.168.1.220"
  SP.Alias = "Default"
  SP.Database = "Professional"
  SP.OpenDatabase
  SP.Login "Manager", ""
  SP.OpenProject "Introduction to SimaPro 7", ""

  SP.CreateSubstance "Air", Subs
  Subs.CASNumber = "4-5-13"
  Subs.Name = "Some substance"
  Subs.DefaultUnit = "kg"
  Subs.Update

```

```

SP.CreateProcess ptMaterial, PC
Set PL = PC.AddLine(ppProduct, -1)
PL.ObjectName = "Steel 2"
PL.UnitName = "kg"
PL.Amount = "2"
PL.Comment.Add ("My new created process")
PL.CategoryPath = "Chemicals\inorganic"

PC.Update

' create second material process Case
SP.CreateProcess ptMaterial, PC2
Set PL = PC2.AddLine(ppProducts, 0)
PL.ObjectName = "Case 2"
PL.UnitName = "kg"
PL.Amount = "10"

Set Param = PC2.AddParamLine(ptInputParameter, -1)
Param.Name = "A"
Param.Value = "2,3"

' add input from Steel
Set PL = PC2.AddLine(ppMaterialsFuels, -1)
' input from steel
PL.SetProduct "Introduction to SimaPro 7", ptMaterial, "Steel 2"
PL.Amount = "8"
PL.UnitName = "kg"

Set PL = PC2.AddLine(ppAirborneEmissions, -1)
' input from steel
PL.SetSubstance "Some substance", ""
PL.Amount = "A+1"
PL.UnitName = "kg"

PC2.Update

SP.Logout
SP.CloseDatabase
Set SP = Nothing

```

End Sub

4.2 Delphi

4.2.1 Calculate a single score

This example calculates the single score of a process, using the Eco-indicator 99 H/A method and returns the result.

```

var
  SimaPro: SimaProServer;
begin
  SimaPro := CoSimaProServer.Create;
  SimaPro.Server := 'nexusdb@192.168.1.220';
  SimaPro.Alias := 'Default';
  SimaPro.Database := 'Professional';
  SimaPro.OpenDatabase;
  SimaPro.Login ('Manager', '');
  SimaPro.OpenProject ('Introduction to SimaPro 7', '');
  if SimaPro.Analyse('BUWAL250', ptMaterial, Electricity Netherlands B250 ',
    'Methods', 'Eco-indicator 99 (H)', 'Europe EI 99 H/A') then
  begin
    memol.lines.add('single score = ');
    memol.lines.add(FloatToStr(SimaPro.AnalyseResult(rtSingleScore, 0).Amount));
  end;
end;

```

4.2.2 Create a process

This example creates a process with an input of plastic.

```

var
  SimaPro: SimaProServer;

```

```

PC2: Process;
PL: ProcessLine;
begin
SimaPro := CoSimaProServer.Create;
SimaPro.Server := 'nexusdb@192.168.1.220';
SimaPro.Alias := 'Default';
SimaPro.Database := 'Professional';
    SimaPro.OpenDatabase;
SimaPro.Login ('Manager', '');
SimaPro.OpenProject ('Introduction to SimaPro 7', '');
SimaPro.CreateProcess (ptMaterial, PC2);
PL := PC2.AddLine(ppProducts, 0);
PL.ObjectName := 'Case 2';
PL.UnitName := 'kg';
PL.Amount := '10';
PL := PC2.AddLine(ppMaterialsFuels, -1);
PL.SetProduct ('BUWAL250', ptMaterial, 'PVC B250');
PL.Amount := '8';
PL.UnitName := 'kg';
PC2.Update;
memo1.lines.add('ready');

```

4.3 PHP

This example, for PHP in console mode, prints an overview of all processes and product stages in a project to the console.

```

<?php
$SP = new COM("SimaPro.SimaProServer");
$SP->Server = 'nexusdb@192.168.1.111';
$SP->Alias = 'Default';
$SP->Database = 'Professional';
$SP->OpenDatabase;
$SP->Login ('Manager', '');
$SP->OpenProject ('Introduction to SimaPro 7', '');
print $SP->ProductCount + "\n";
for ($I = 1; $I < $SP->ProductCount; $I = $I + 1) {
    print $SP->ProductName($I) . "\n";
}
$sp = null;

?>

```

4.4 C++

The C++ is more complex due to the memory allocation requirements. Precondition is a form with a memo called memo1. Otherwise replace the "Memo1->Lines-Add" part with more suitable code.

This example below calculates the single score of a process, using the Eco-indicator 99 H/A method and returns the result.

```

BSTR Server = ::SysAllocString( L"nexusdb@192.168.1.220" );
BSTR Alias = ::SysAllocString( L"Default" );
BSTR Database = ::SysAllocString( L"Professional" );
BSTR User = ::SysAllocString( L"Manager" );
BSTR Project = ::SysAllocString( L"Introduction to SimaPro 7" );
BSTR ProcessProject = ::SysAllocString(L"BUWAL250");
BSTR Process = ::SysAllocString( L"PVC B250" );
BSTR MethodLib = ::SysAllocString( L"Methods" );
BSTR Method = ::SysAllocString( L"Eco-indicator 99 (H)" );
BSTR NWSet = ::SysAllocString( L"Europe EI 99 H/A" );

TCOMISimaProServer SimaPro = CoSimaProServer::Create();
SimaPro->Server = Server;
SimaPro->Alias = Alias;
SimaPro->Database = Database;
SimaPro->OpenDatabase();
SimaPro->Login (User, L"");
SimaPro->OpenProject (Project, L"");
if (SimaPro->Analyse(ProcessProject, ptMaterial, Process,
    MethodLib, Method, NWSet))
{

```

```

        Memol->Lines->Add("Single score = ");
        Memol->Lines->Add(FloatToStr(SimaPro->AnalyseResult(rtSingleScore, 0)->Amount));
    }

    ::SysFreeString(Server);
    ::SysFreeString(Alias);
    ::SysFreeString(Database);
    ::SysFreeString(User);
    ::SysFreeString(Project);
    ::SysFreeString(ProcessProject);
    ::SysFreeString(Process);
    ::SysFreeString(MethodLib);
    ::SysFreeString(Method);
    ::SysFreeString(NWSet);

```

4.5 VB.NET Sample

This example will create webpage with a list of all processes in a database and create a new process with a given name.

Create a website or aspx page in Visual Studio with

- 1 Label
- 2 Buttons
- 1 Textbox
- 1 GridView

And use the following code:

```

using System;
using System.Data;
using System.Collections;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using SimaPro;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        SimaProServer SP = new SimaPro.SimaProServer();
        SimaPro.Process PC;
        SP.Server = "nexusdb@192.168.1.220";
        SP.Alias = "Default";
        SP.Database = "Professional";
        SP.OpenDatabase();
        SP.Login("Manager", "");
        SP.OpenProject("Introduction to SimaPro 7", "");
        SP.CreateProcess(TProcessType.ptMaterial, out PC);
        ProcessLine PL = PC.AddLine(TProcessPart.ppProducts, -1);
        PL.ObjectName = TextBox1.Text;
        PL.UnitName = "kg";
        PL.Amount = "2";
        PL.Comment.Add("My new created process");
        PC.Update();
        SP.Logout();
        SP.CloseDatabase();
        Label1.Text = TextBox1.Text + " is created";
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        ArrayList AL = new ArrayList();

```

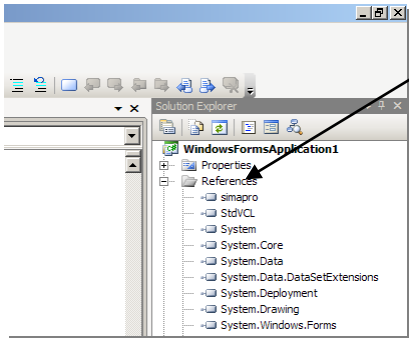
```

SimaProServer SP = new SimaPro.SimaProServer();
SimaPro.Process PC;
SP.Server = "nexusdb@192.168.1.220";
SP.Alias = "Default";
SP.Database = "Professional";
SP.OpenDatabase();
SP.Login("Manager", "");
SP.OpenProject("Introduction to SimaPro 7", "");
for(int I = 0; I < SP.ProductCount; I++)
{
    AL.Add(SP.get_ProductName(I));
};
GridView1.DataSource = AL;
GridView1.DataBind();
SP.Logout();
SP.CloseDatabase();
Labell.Text = TextBox1.Text + " list done";
}
}

```

4.6 C# Sample

In Visual Studio, after creating your application you need to reference to the SimaPro Library. Go to the Solution Explorer and add Reference. Choose the COM tab and select 'SimaPro Library' (Version 2.0).



Create a console application and paste this code snippet into the Main method. Change the server name, alias and database to your own situation. If opening the database in single user mode, be sure to use double '\\\' in the alias, otherwise it will be handled as an escape character.

```

using System;
using SimaPro;

namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            // Declare objects and strings to identify them
            SimaProServer SP;
            Process PCassembly;
            Process PCmaterial;
            ProcessLine PLassembleyName;
            ProcessLine PLassembleyMaterial;
            ProcessLine PLmaterialProperties;
            string assembly = "New Assembly";
            string material = "Steel";
            string project = "Introduction to SimaPro 7";

            Console.WriteLine("Started app");

            // Open database and project
            SP = new SimaProServer();
            SP.Server = "local server"; // "nexusdb@192.168.1.113" if multi user
            SP.Alias = "C:\\Users\\Public\\SimaPro\\data"; // "Default" if multi user
            SP.Database = "Professional";
            SP.OpenDatabase();
            SP.Login("Manager", ""); // values ignored if no password set
            SP.OpenProject(project, "");
        }
    }
}

```

```

Console.WriteLine("Opened database and project");

// Test if assembly already exists
if (SP.FindProcess(project, TProcessType.ptAssembly, assembly, out PCassembly))
{
    Console.WriteLine("Assembly already exists");

    PCassembly.Delete();
    Console.WriteLine("Assembly deleted");
}

// Test if material already exists
if (!SP.FindProcess(project, TProcessType.ptMaterial, material, out PCmaterial))
{
    // Create material
    SP.CreateProcess(TProcessType.ptMaterial, out PCmaterial);
    Console.WriteLine("Created material process");

    PLmaterialProperties = PCmaterial.AddLine(TProcessPart.ppProducts, -1);
    PLmaterialProperties.ObjectName = material;
    PLmaterialProperties.UnitName = "kg";
    PLmaterialProperties.Amount = "2";
    PLmaterialProperties.Comment.Add("My newly created process");
    PCmaterial.Update();
    Console.WriteLine("New material process committed");
}
else
{
    Console.WriteLine("Material " + material + " already exists");
}

// Create new process object of type Assembly
SP.CreateProcess(TProcessType.ptAssembly, out PCassembly);
Console.WriteLine("Created process");

// Get process line of type 'Product' from the new Assembly object.
// For product-stage processes, this line is created automatically.
// This line is needed to set the name, comments and category path.
PLassemblyName = PCassembly.get_Line(TProcessPart.ppProducts, 0);

// Set properties of process line
PLassemblyName.ObjectName = assembly;
PLassemblyName.Comment.Add("My newly created assembly");
PLassemblyName.CategoryPath = "COM demonstration\\C# test";
Console.WriteLine("Created assembly");

// add material to assembly
PLassemblyMaterial = PCassembly.AddLine(TProcessPart.ppAssembliesAndMaterials, -1);
PLassemblyMaterial.SetProduct(project, TProcessType.ptMaterial, material);
PLassemblyMaterial.Amount = "2";
PLassemblyMaterial.UnitName = "kg";
Console.WriteLine("New material line added to assembly");

PCassembly.Update();
Console.WriteLine("New assembly committed");

// Logout and close database
SP.Logout();
SP.CloseDatabase();
SP = null;

Console.WriteLine("Press <Enter> to continue...");
Console.ReadLine();
}
}
}

```

This code creates new assembly and material processes and attaches them in the project. If the assembly already exists, it is deleted then recreated.

5 Reference

Amount Property

Object	Property description
ProcessLine	Amount or percentage of product or substance
SimaProAnalyseResult	Amount of units substance or impact assessment score
SimaProNetworkResult	Amount of the product
SimaProTreeResult	Amount of the product

Cancel Method

Object	Method description
Process	Cancel edit mode and returns to read mode
Substance	Cancel edit mode and returns to read mode

Comment Property

Object	Property description
ParamLine	Comment of the parameter
Process	Comment of the process
ProcessLine	Comment
Substance	Comment

Distribution Property

Object	Property description
ParamLine	Distribution of the input parameter
ProcessLine	Distribution of amount

Edit Method

Object	Method description
Process	Set the object in edit mode
Substance	Set the object in edit mode

LineNumber Property

Object	Property description
ParamLine	Index in the section of the process to which the ProcessLine object belongs
ProcessLine	Index in the section of the process to which the ProcessLine object belongs

MainCompartmentName Property

Object	Property description
SimaProAnalyseResult	
SimaProServer	Name of a main-compartment

Maximum Property

Object	Property description
ParamLine	Maximum value of the input parameter
ProcessLine	Maximum value of amount

Minimum Property

Object	Property description
ParamLine	Minimum value of the input parameter
ProcessLine	Minimum value of amount

Mode Property

Object	Property description
Process	Mode of the object, can be: Read, New or Edit
Substance	Mode of the object, can be: Read, New or Edit

Name Property

Object	Property description
ParamLine	Name of the parameter
Substance	Name of the substance (e.g. 'Carbon dioxide')

ParamLine

Class ParamLine

Represents one parameter in SimaPro

ParamLine properties

ParamLine Legend

- ▶ Comment
- ▶ Distribution
- ▶ Expression
- ▶ Hide
- ▶ LineNumber
- ▶ Maximum
- ▶ Minimum
- ▶ Name
- ▶ ParameterType
- ▶ Process
- ▶ StandardDeviation
- ▶ Value

ParamLine.Comment

Property Comment As IStrings
Comment of the parameter
Member of ParamLine

Example

Add comment to a parameter

```
ParamLine1.Comment.add('estimated value')
```

ParamLine.Distribution

Property Distribution As TDistribution
Distribution of the input parameter
Member of ParamLine

Example

Set the distribution to normal

```
Param1.Distribution := distNormal
```

ParamLine.Expression

Property Expression As String
Expression of the calculated parameter
Member of ParamLine

Example

B is A + 1

```
Param1.Name = 'B'  
param1.ParameterType = ptCalculatedParameter  
Param1.Expression = 'A+1'
```

ParamLine.Hide

Property Hide As Boolean
Only locally visible in process
Member of ParamLine

ParamLine.LineNumber

Property LineNumber As Integer
Index in the section of the process to which the ProcessLine object belongs
Member of ParamLine
Read-Only

ParamLine.Maximum

Property Maximum As Double
Maximum value of the input parameter
Member of ParamLine

ParamLine.Minimum

Property Minimum As Double
Minimum value of the input parameter
Member of ParamLine

ParamLine.Name

Property Name As String
Name of the parameter
Member of ParamLine

Example

B is A + 1

```
Param1.Name = 'B'  
param1.ParameterType = ptCalculatedParameter  
Param1.Expression = 'A+1'
```

ParamLine.ParameterType

Property ParameterType As TParameterType

Section in the process to which the ProcessLine object belongs

Member of ParamLine

Read-Only

Example

B is A + 1

```
Param1.Name = 'B'  
param1.ParameterType = ptCalculatedParameter  
Param1.Expression = 'A+1'
```

ParamLine.Process

Property Process As Process

Process to which the ParamLine object belongs

Member of ParamLine

Read-Only

Can be used to trace the process properties.

ParamLine.StandardDeviation

Property StandardDeviation As Double

Standard deviation of the input parameter

Member of ParamLine

ParamLine.Value

Property Value As Double

Value of the input parameter

Member of ParamLine

Example

Set A to 13

```
if Process.FindParameter('a', Param) then  
    Param.Edit  
else  
    Param = Process.AddParamLine(ptInputParameter, -1)  
    Param.Name = 'a'  
Param.Value = 13  
Param.Update
```

Process

Class Process

Objects comprising inputs and outputs that model the environmental impact of real world activities.

Process methods

Process Legend

- ▶ AddLine
- ▶ AddParamLine
- ▶ Cancel
- ▶ Delete
- ▶ DeleteLine
- ▶ DeleteParamLine
- ▶ Edit
- ▶ FindParameter
- ▶ Update

Process properties

Process Legend

- ▶ Comment
- ▶ Line
- ▶ LineCount
- ▶ Mode
- ▶ ParamLine
- ▶ ParamLineCount
- ▶ ProcessType
- ▶ ProjectName
- ▶ Status

Process.AddLine

Function AddLine(ByVal Part As TProcessPart, ByVal LineNumber As Integer) As ProcessLine

Add a line and returns a ProcessLine object

Member of Process

Parameters	Description
Part	Part within process (ppProducts, ppMaterialsFuels etc)
LineNumber	Linenummer (zero based) -1 means: at the end

Return value

Returns ProcessLine

Example

Add emission and product to a process

```
Line := MyProcess.AddLine(ppEmissionsWater, -1);
Line.SetSubstance('Carbon dioxide', 'Air');
Line.Amount := '34';
Line.UnitName := 'g';
Line := MyProcess.AddLine(ppProduct, -1);
Line.ObjectName := 'My product name',
Line.Amount := '1';
Line.UnitName := 'kg';
Line.WasteType := 'Steel';
```

See also the example at `SimaProServer.CreateProcess`

Process.AddParamLine

Function `AddParamLine(ByVal ParameterType As TParameterType, ByVal LineNumber As Integer) As ParamLine`
 Add a line and returns a `ParamLine` object
 Member of `Process`

Parameters	Description
ParameterType	Type
LineNumber	LineNumber, -1 adds at end

Return value
 Returns `ParamLine`

Example
 Set A to 13

```
if NOT Process.FindParameter('a', Param) then
begin
    Param = Process.AddParamLine(ptInputParameter, -1);
    Param.Name = 'a' ;
end;
Param.Value = 13
```

Process.Cancel

Sub `Cancel()`
 Cancel edit mode and returns to read mode
 Member of `Process`

Example
 Undo all edits

```
ProcessLine := Process.AddLine(ppProcesses, -1);
Process.cancel;
```

Process.Comment

Property `Comment As IStrings`
 Comment of the process
 Member of `Process`

Example
 Set the comment

```
process.comment.add('changed by Joe');
```

Process.Delete

Sub Delete()

Remove a process from the database. An exception error is raised if process has already been deleted or it is being used as a sub-process.

Member of **Process**

Example

PC.Delete

Process.DeleteLine

Sub DeleteLine(ByVal Part As TProcessPart, ByVal LineNumber As Integer)

Delete a line

Member of **Process**

Parameters	Description
Part	Which part of the process
LineNumber	Which line within part (Zero based)

Example

Delete first product

```
Process.DeleteLine(ppProducts, 0)
```

Process.DeleteParamLine

Sub DeleteParamLine(ByVal ParameterType As TParameterType, ByVal LineNumber As Integer)

Delete a parameter line

Member of **Process**

Parameters	Description
ParameterType	Type of parameter
LineNumber	Linenummer (zero based)

Example

Delete second calculated parameter

```
Pc.deleteParamLine(ptCalculatedParameter, 1)
```

Process.Edit

Sub Edit()

Set the object in edit mode

Member of **Process**

Example

Change the waste type of 'Steel NL' to 'Steel'

```
If SimaPro.FindProcess('Sample project', ptMaterial, 'steel', pc) then
  pc.Edit;
  try
    // change waste type of first product
    pc.processline[ppProduct, 0].WasteType := 'Steel';
    pc.Update;
  except
```

```

    pc.Cancel;
end;

```

Process.FindParameter

Function FindParameter(ByVal Name As String, ByRef ParamLine As ParamLine) As Boolean

Find a line and returns a ParamLine object

Member of Process

Parameters	Description
Name	Name of Parameter
ParamLine	Result

Return value

Returns Boolean

Example

Set A to 13

```

if Process.FindParameter('a', Param) then
    Param.Edit
else
    Param = Process.AddParamLine(ptInputParameter, -1)
    Param.Name = 'a'
Param.Value = 13
Param.Update

```

Process.Line

Property Line(ByVal Part As TProcessPart, ByVal LineNumber As Integer) As ProcessLine

Returns a ProcessLine object

Member of Process

Read-Only

Parameters	Description
Part	Part within process (ppProducts, ppMaterialsFuels etc)
LineNumber	Linenumbr (zero based)

Example

Delete the emissions to air of Carbon Dioxide.

```

Pc := SimaPro.FindProcess(ptMaterial, 'steel');
Pc.Edit;
I := 0;
while I < PC.LineCount[ppEmissionsAir] do
begin
    if Pc.Line[ppEmissionsAir, I].ObjectName2 = 'Carbon Dioxide, biogenic' then
        Pc.DeleteLine(ppEmissionsAir, I)
    else
        Inc(I);
end;

```

```
Pc.Update;
```

Process.LineCount

Property LineCount(ByVal Part As TProcessPart) As Integer

Number of lines in a part

Member of Process

Read-Only

Parameters	Description
------------	-------------

Part	Products etc
-------------	--------------

```
if SimaPro.FindProcess('Sample project', ptMaterial, 'steel', pc) then
begin
  Pc.Edit;
  I := PC.LineCount[ppEmissionsAir] - 1;
  while I >= 0 do
  begin
    s := pc.Line[ppEmissionsAir, I].ObjectName2;
    if Pc.Line[ppEmissionsAir, I].ObjectName2 = 'Carbon Dioxide, biogenic' then
      Pc.DeleteLine[ppEmissionsAir, I]
    else
      Dec(I);
    end;
  Pc.Update;
end;
```

Process.Mode

Property Mode As String

Mode of the object, can be: Read, New or Edit

Member of Process

Read-Only

Process.ParamLine

Property ParamLine(ByVal ParameterType As TParameterType, ByVal LineNumber As Integer) As ParamLine

Returns a ParamLine object

Member of Process

Read-Only

Parameters	Description
------------	-------------

ParameterType	ptInputParameter,
----------------------	-------------------

LineNumber	Linenummer where you want to insert. -1 inserts at end
-------------------	--

Process.ParamLineCount

Property ParamLineCount(ByVal ParameterType As TParameterType) As Integer

Number of parameters

Member of Process

Read-Only

Parameters	Description
------------	-------------

ParameterType	ptInputParameter or ptCalculatedParameter
---------------	---

Process.ProcessType

Property ProcessType As TProcessType

Type of the process

Member of Process

Read-Only

Process.ProjectName

Property ProjectName As String

Name of the project to which the process belongs

Member of Process

Read-Only

Process.Status

Property Status As TProcessStatus

Status of the process

Member of Process

Example

Set status to draft

```
Pc.Status := stDraft;
```

Process.Update

Sub Update()

Store the data of the object in the database and switch to read mode

Member of Process

Example

Change the waste type of 'Steel NL' to 'Steel'

```
pc := SimaPro.FindProcess(ptmaterial, 'Steel NL');
pc.Edit;
try
  // change waste type of first product
  pc.processline[ppProduct, 0].WasteType := 'Steel';
```

```
    pc.Update;  
except  
    pc.Cancel;  
end;
```

ProcessLine

Class ProcessLine

A line in a process as seen on screen in SimaPro. Can be either products, inputs or outputs.

ProcessLine methods

ProcessLine Legend

- ▶ SetMaterial
- ▶ SetProduct
- ▶ SetSubstance

ProcessLine properties

ProcessLine Legend

- ▶ Allocation
- ▶ Amount
- ▶ CategoryPath
- ▶ Comment
- ▶ Distribution
- ▶ LineNumber
- ▶ Maximum
- ▶ Minimum
- ▶ ObjectName
- ▶ ObjectName2
- ▶ Part
- ▶ Process
- ▶ ProcessType
- ▶ ProjectName
- ▶ ProjectName2
- ▶ StandardDeviation
- ▶ UnitName
- ▶ WasteType

ProcessLine.Allocation

Property Allocation As String
Allocation percentage of a product
Member of ProcessLine

ProcessLine.Amount

Property Amount As String
Amount or percentage of product or substance
Member of ProcessLine

Example

```
Line.SetProduct('eco invent unit processes', ptEnergy, 'Electricity UCTPE');  
Line.Amount := '10';  
Line.UnitName := 'kWh';
```

See also the example at `SimaProServer.CreateProcess`

ProcessLine.CategoryPath

Property CategoryPath As String
Category path of a product
Member of ProcessLine

Example

Change the category

```
AProcessLine.Category := '\others\chemicals';
```

ProcessLine.Comment

Property Comment As IStrings
Comment
Member of ProcessLine

ProcessLine.Distribution

Property Distribution As TDistribution
Distribution of amount
Member of ProcessLine

ProcessLine.LineNumber

Property LineNumber As Integer

Index in the section of the process to which the ProcessLine object belongs

Member of ProcessLine

Read-Only

ProcessLine.Maximum

Property Maximum As Double

Maximum value of amount

Member of ProcessLine

ProcessLine.Minimum

Property Minimum As Double

Minimum value of amount

Member of ProcessLine

ProcessLine.ObjectName

Property ObjectName As String

Name of the product

Member of ProcessLine

See the example at `SimaProServer.CreateProcess`

ProcessLine.ObjectName2

Property ObjectName2 As String

Sub-compartment or material name

Member of ProcessLine

Read-Only

ProcessLine.Part

Property Part As TProcessPart

Section in the process to which the ProcessLine object belongs

Member of ProcessLine

Read-Only

ProcessLine.Process

Property Process As Process
 Process to which the ProcessLine object belongs
 Member of ProcessLine
 Read-Only

ProcessLine.ProcessType

Property ProcessType As TProcessType
 Type of the product
 Member of ProcessLine
 Read-Only

ProcessLine.ProjectName

Property ProjectName As String
 Name of the project to which the product belongs
 Member of ProcessLine
 Read-Only

ProcessLine.ProjectName2

Property ProjectName2 As String
 Name of the project to which the material belongs
 Member of ProcessLine
 Read-Only

ProcessLine.SetMaterial

Sub SetMaterial(ByVal ProjectName As String, ByVal ProductName As String)
 Select a material and link it to the process; only for waste treatment product and specific waste flow
 Member of ProcessLine

Parameters	Description
ProjectName	Name of project (see SimaproServer.Projects)
ProductName	Name of product Only for advanced disposal modelling. Use SetProduct to link inputs from technosphere.

Example

```
PC.SetMaterial('My project', 'Steel');
```

ProcessLine.SetProduct

Sub SetProduct(ByVal ProjectName As String, ByVal ProcessType As TProcessType, ByVal ProductName As String)

Select a product of process as an input and link it to the process

Member of ProcessLine

Parameters	Description
ProjectName	Name of project
ProcessType	
ProductName	Name of the product

Link inputs from technosphere.

Example

```
Line.SetProduct('eco invent unit processes', ptEnergy, 'Electricity UCTPE');
Line.Amount := '10';
Line.UnitName := 'kWh';
```

ProcessLine.SetSubstance

Sub SetSubstance(ByVal SubstanceName As String, ByVal SubCompartmentName As String)

Select a substance and link it to the process

Member of ProcessLine

Parameters	Description
SubstanceName	Name as in the substance list
SubCompartmentName	Name of the subcompartment

Use this to define the emissions and raw material use of a process

Example

```
Line.SetSubstance('Carbon dioxide', 'Air')
Line.Amount := '11.4';
Line.UnitName := 'kg';
Pc.Update;
```

ProcessLine.StandardDeviation

Property StandardDeviation As Double

Standard deviation of amount

Member of ProcessLine

ProcessLine.UnitName

Property UnitName As String

Name of the unit of amount

Member of ProcessLine

Example

```
Line.SetProduct('eco invent unit processes', ptEnergy, 'Electricity UCTPE');
```

```
Line.Amount := '10';
Line.UnitName := 'kWh';
```

See also the example at `SimaProServer.CreateProcess`

ProcessLine.WasteType

Property WasteType As String

Waste type of a product or specific waste flow

Member of ProcessLine

Process Property

Select one of the available subtopics below to see detailed help on **Process** property

Object	Property description
ParamLine	Process to which the ParamLine object belongs
ProcessLine	Process to which the ProcessLine object belongs

ProcessType Property

Select one of the available subtopics below to see detailed help on **ProcessType** property

Object	Property description
Process	Type of the process
ProcessLine	Type of the product

ProductName Property

Select one of the available subtopics below to see detailed help on **ProductName** property

Object	Property description
SimaProNetworkResult	Name of the product
SimaProServer	Name of a product
SimaProTreeResult	Name of the product

ProjectName Property

Select one of the available subtopics below to see detailed help on **ProjectName** property

Object	Property description
Process	Name of the project to which the process belongs
ProcessLine	Name of the project to which the product belongs

SimaProAnalyseResult

Class SimaProAnalyseResult

Object containing substance list or impact assessment score resulting from the calculation of a process object.

SimaProAnalyseResult properties

SimaProAnalyseResult Legend

- ▶ Amount
- ▶ IndicatorName
- ▶ MainCompartmentName
- ▶ SubCompartmentName
- ▶ UnitName

SimaProAnalyseResult.Amount

Property Amount As Double

Amount of units substance or impact assessment score

Member of **SimaProAnalyseResult**

Example

See `SimaProServer.Analyse`

SimaProAnalyseResult.IndicatorName

Property IndicatorName As String

Name of substance, impact category etc. In case of single score it is empty

Member of **SimaProAnalyseResult**

Example

See `SimaProServer.Analyse`

SimaProAnalyseResult.MainCompartmentName

Property MainCompartmentName As String

Only for inventory results

Member of **SimaProAnalyseResult**

SimaProAnalyseResult.SubCompartmentName

Property SubCompartmentName As String
Only for inventory results
Member of **SimaProAnalyseResult**

SimaProAnalyseResult.UnitName

Property UnitName As String
Unit (e.g. kg, m3)
Member of **SimaProAnalyseResult**

Example

See **SimaProServer.Analyse**

SimaProCalculationError

Class **SimaProCalculationError**
Object containing error details resulting from the calculation of a process object.

SimaProCalculationError properties

SimaProCalculationError Legend

- ▀ AdditionalInfo
 - ▀ ErrorCode
 - ▀ ErrorDescription
-

SimaProCalculationError.AdditionalInfo

Property AdditionalInfo As String

Member of **SimaProCalculationError**

SimaProCalculationError.ErrorCode

Property ErrorCode As Integer
Number of the error
Member of **SimaProCalculationError**

SimaProCalculationError.ErrorDescription

Property ErrorDescription As String
Description of the error
Member of SimaProCalculationError

SimaProNetworkResult

Class SimaProNetworkResult
Object containing network flows resulting from the calculation of a process object.

SimaProNetworkResult properties

SimaProNetworkResult Legend
Amount
ChildProductName
ProductName
UnitName

SimaProNetworkResult.Amount

Property Amount As Double
Amount of the product
Member of SimaProNetworkResult

SimaProNetworkResult.ChildProductName

Property ChildProductName As String
Name of the child-product
Member of SimaProNetworkResult

SimaProNetworkResult.ProductName

Property ProductName As String
Name of the product
Member of SimaProNetworkResult

SimaProNetworkResult.UnitName

Property UnitName As String
Unit of the amount
Member of SimaProNetworkResult

SimaProServer

Class SimaProServer
Object handling connection to database and functions applied to collections of process objects.

SimaProServer methods

SimaProServer Legend

- ▶ Analyse
- ▶ AnalyseResult
- ▶ CalculationError
- ▶ CloseDatabase
- ▶ CloseProject
- ▶ CreateProcess
- ▶ CreateSubstance
- ▶ DeleteParamLine
- ▶ FindParamLine
- ▶ FindProcess
- ▶ FindProcessEx
- ▶ FindSubstance
- ▶ Login
- ▶ Logout
- ▶ Network
- ▶ NetworkCalcScore
- ▶ NetworkResult
- ▶ OpenDatabase
- ▶ OpenProject
- ▶ ParamLine
- ▶ ParamLineCount
- ▶ Product
- ▶ Substance
- ▶ Tree
- ▶ TreeCalcScore
- ▶ TreeResult

SimaProServer properties

SimaProServer Legend

- ▶ Alias
- ▶ Aliases
- ▶ CalculationErrorCount
- ▶ CurrentProject
- ▶ CurrentUser
- ▶ Database
- ▶ DatabaseOpen
- ▶ Databases
- ▶ LoggedIn

- ▶ MainCompartmentCount
- ▶ MainCompartmentName
- ▶ MethodCount
- ▶ MethodName
- ▶ MethodProjectName
- ▶ NetworkChildNodeCount
- ▶ NetworkChildNodeIndex
- ▶ NetworkNodeCount
- ▶ NetworkProductName
- ▶ NetworkTopNodeIndex
- ▶ NWSets
- ▶ ProductCount
- ▶ ProductName
- ▶ ProductProcessType
- ▶ ProductProcessTypeName
- ▶ ProductProjectName
- ▶ ProjectOpen
- ▶ Projects
- ▶ QuantityCount
- ▶ QuantityName
- ▶ ResultCount
- ▶ ResultIndicatorName
- ▶ ResultMainCompartmentName
- ▶ ResultSubCompartmentName
- ▶ Server
- ▶ Servers
- ▶ SubCompartmentCount
- ▶ SubCompartmentName
- ▶ SubstanceCASNumber
- ▶ SubstanceCount
- ▶ SubstanceDefaultUnit
- ▶ SubstanceName
- ▶ TreeChildNodeCount
- ▶ TreeChildNodeIndex
- ▶ TreeNodeCount
- ▶ TreeProductName
- ▶ TreeTopNodeIndex
- ▶ UnitCount
- ▶ UnitDefault
- ▶ UnitFactor
- ▶ UnitMetric
- ▶ UnitName
- ▶ WasteTypeCount
- ▶ WasteTypeName

SimaProServer.AddParamLine

Function AddParamLine(ByVal ParameterType As TParameterType, ByVal ParameterScope As TParameterScope, ByVal LineNumber As Long) As ParamLine

Add a project or database parameter line and returns a ParamLine object

Parameters	Description
ParameterType	Type of parameter: Input or calculated
LineNumber	LineNumber, -1 adds at end
ParameterScope	psDatabase or psProject. psProject means currently open project

Return value

Returns **ParamLine**. If no project is open an exception is raised.

Example

Set project parameter A to 13

```

Param = SimaProServer.FindParameter('OptionA', psDatabase)
if Param is nothing then
begin
    Param = SimaProServer.AddParameter(ptInputParameter, psDatabase, -1)
    Param.Name = 'OptionA'
    Param.Value = 13
End

```

SimaProServer.Alias

Property Alias As String

Currently used alias

Member of SimaProServer

See the example at SimaProServer.CreateProcess

SimaProServer.Aliases

Property Aliases As IStrings

List of available aliases, set Server first

Member of SimaProServer

Read-Only

SimaProServer.Analyse

Function Analyse(ByVal ProjectName As String, ByVal ProcessType As TProcessType, ByVal ProductName As String, ByVal MethodProjectName As String, ByVal MethodName As String, ByVal NWSetName As String) As Boolean

Perform the analyse function for a process or product stage

Member of SimaProServer

Parameters	Description
ProjectName	Name of project
ProcessType	Type of process
ProductName	Name of Product
MethodProjectName	Project name where method resides
MethodName	name of method
NWSetName	Normalisation Weighting set.

Return value

Returns Boolean

Example

Show the inventory

```

SP.Analyse('My project', ptMaterial, 'Steel', 'Methods', 'EI99', 'N/A');
// show inventory

```

```

for I := 0 to SP.ResultCount(rtInventory) - 1 do
begin
    Res := SP.AnalyseResult(rtInventory, I)
    Print Res.Amount, Res.IndicatorName, Res.UnitName;
end;

```

SimaProServer.AnalyseResult

Function AnalyseResult(ByVal AnalyseResultType As TResultType, ByVal I As Integer) As SimaProAnalyseResult

Retrieve the result of the analyse function

Member of SimaProServer

Parameters	Description
AnalyseResultType	rtCharacterisation, rtDamage, rtNormalisation, rtWeighting, rtSingleScore or rtInventory
I	Index

Return value

Returns SimaProAnalyseResult

Example

See SimaProServer.Analyse

SimaProServer.CalculationError

Function CalculationError(ByVal I As Integer) As SimaProCalculationError

Calculation error data

Member of SimaProServer

Parameters	Description
I	Index

Return value

Returns SimaProCalculationError

SimaProServer.CalculationErrorCount

Property CalculationErrorCount As Integer

Number of calculation errors

Member of SimaProServer

Read-Only

SimaProServer.CloseDatabase

Sub CloseDatabase()

Close the currently open database

Member of SimaProServer

SimaProServer.CloseProject

Sub CloseProject()

Close the currently open project

Member of SimaProServer

SimaProServer.CreateProcess

Sub CreateProcess(ByVal ProcessType As TProcessType, ByRef Process As Process)

Creates a new process

Member of SimaProServer

Parameters	Description
ProcessType	ProcessType (ptMaterial, ptEnergy, etc)
Process	Resulting process object

Example

Create 2 processes and link to each other, then link to an assembly (VB)

```

Dim SP As SimaProServer
Dim PC As Process
Dim PC2 As Process
Dim PC3 As Process
Dim PL As ProcessLine

Set SP = New SimaProServer
SP.Server = "nexusdb@192.168.2.113"
SP.Alias = "Default"
SP.Database = "Professional"
SP.OpenDatabase
SP.Login "Manager", ""
SP.OpenProject "A COM DEMO", ""

SP.CreateProcess ptMaterial, PC
Set PL = PC.AddLine(ppProducts, -1)
PL.ObjectName = "Steel"
PL.UnitName = "kg"
PL.Amount = "2"
PL.Comment.Add ("My new created process")
PC.Update

' create second material process Case
SP.CreateProcess ptMaterial, PC2
Set PL = PC2.AddLine(ppProducts, 0)
PL.ObjectName = "Case"
PL.UnitName = "kg"
PL.Amount = "10"

' add input from Steel
Set PL = PC2.AddLine(ppMaterialsFuels, -1)
' input from steel
PL.SetProduct "A COM DEMO", ptMaterial, "Steel"
PL.Amount = "8"
PL.UnitName = "kg"
PC2.Update

```

```

' create Assembly product stage
SP.CreateProcess ptAssembly, PC3
Set PL = PC3.get_Line(ppProducts, 0)
PL.ObjectName = "New Assembly"
PL.Comment.Add ("My newly created assembly")
PL.CategoryPath = "COM demonstration\Create process test"

' add Case material to assembly
Set PL = PC3.AddLine(ppAssembliesAndMaterials, -1)
PL.SetProduct "A COM DEMO", ptMaterial, "Case"
PL.Amount = "2"
PL.UnitName = "kg"
PC3.Update

SP.Logout
SP.CloseDatabase
Set SP = Nothing

```

SimaProServer.CreateSubstance

Sub CreateSubstance(ByVal MainCompartment As String, ByRef Substance As Substance)

Create a new substance

Member of SimaProServer

Parameters	Description
MainCompartment	MainCompartment goes here ('Air', 'Water', 'Soil', etc)
Substance	Resulting substance object

Example

Create a new substance

```

SimaPro.CreateSubstance('Air', Substance)
Substance.Name := 'My new substance'
Substance.UnitName := 'kg';
Substance.Update; // save in database

```

SimaProServer.CurrentProject

Property CurrentProject As String

Name of the currently open project

Member of SimaProServer

Read-Only

SimaProServer.CurrentUser

Property CurrentUser As String

Name of the user that is currently logged in

Member of SimaProServer

Read-Only

SimaProServer.Database

Property Database As String
 Currently used database, see OpenDatabase
 Member of SimaProServer

See the example at SimaProServer.CreateProcess

SimaProServer.DatabaseOpen

Property DatabaseOpen As Boolean
 Indicates if a database is currently open
 Member of SimaProServer
 Read-Only

SimaProServer.Databases

Property Databases As IStrings
 List of available databases, set Server and Alias first
 Member of SimaProServer
 Read-Only

SimaProServer.DeleteParamLine

Sub DeleteParamLine(ByVal ParameterType As TParameterType, ByVal ParameterScope As TParameterScope, ByVal LineNumber As Long)

Deletes a parameter from database or the current project

Member of SimaProServer

Parameters	Description
ParameterType	Type of parameter (input or calculated)
ParameterScope	Scope of parameter (project or database)
LineNumber	Linenummer

The collection of parameters is updated directly so be careful deleting multiple parameters.

SimaProServer.FindParameter

Function FindParameter(ByVal Name As String, ByVal ParameterScope As TParameterScope) As ParamLine

Finds a parameter in the database or current open project

Member of SimaProServer

Parameters	Description
Name	Name of parameter
ParameterScope	Scope of parameter (project or database)

Return value
Returns ParamLine if found

SimaProServer.FindProcess

Function FindProcess(ByVal ProjectName As String, ByVal ProcessType As TProcessType, ByVal ProductName As String, ByRef Process As Process) As Boolean

Looks for a process in the project and libraries
Member of SimaProServer

Parameters	Description
------------	-------------

ProjectName	Name of project
ProcessType	Type of process
ProductName	Name
Process	Result object

Return value
Returns Boolean

Example

Remove lines from process

```
if SimaPro.FindProcess('Sample project', ptMaterial, 'steel', pc) then
begin
  Pc.Edit;
  I := PC.LineCount[ppEmissionsAir] -1 ;
  while I > 0 do
  begin
    s := pc.Line[ppEmissionsAir, I].ObjectName2;
    if Pc.Line[ppEmissionsAir, I].ObjectName2 = 'Carbon Dioxide, biogenic' then
      Pc.DeleteLine[ppEmissionsAir, I]
    else
      Dec(I);
  end;
  Pc.Update;
end;
```

SimaProServer.FindProcessEx

Function FindProcessEx(ByVal ProjectName As String, ByVal ProcessType As TProcessType, ByVal ProductName As String) As Process

FindProcess version that returns a Process
Member of SimaProServer

Parameters	Description
------------	-------------

ProjectName	Name of project
ProcessType	Type of process
ProductName	Name

Return value
Returns Process

Same as FindProcess but returns a process object. Useful in Java, which does not support passing parameters by reference, or if you prefer this style of programming.

SimaProServer.FindSubstance

Function FindSubstance(ByVal MainCompartmentName As String, ByVal SubstanceName As String, ByRef Substance As Substance) As Boolean

Find a substance in the database

Member of SimaProServer

Parameters	Description
MainCompartmentName	'Water', 'Air' etc
SubstanceName	Required substancename
Substance	Substance object

Return value

Returns Boolean

Example

Read CO2 CAS Number

```
PC.FindSubstance('Air', 'Carbon dioxide', Substance);
Print Substance.CASNumber;
```

You can also use

```
PC.SubstanceCASnumber('Air', 'Carbon dioxide')
```

SimaProServer.LoggedIn

Property LoggedIn As Boolean

Indicates if a user is currently logged in

Member of SimaProServer

Read-Only

SimaProServer.Login

Function Login(ByVal UserName As String, ByVal Password As String) As Boolean

Log in the database, not needed for single user if manager-password = empty

Member of SimaProServer

Parameters	Description
UserName	Name of user
Password	Password

Return value

Returns Boolean

See the example at `SimaProServer.CreateProcess`

SimaProServer.Login

Function `Login()` As Boolean
 Log out from the database
 Member of `SimaProServer`

Return value
 Returns Boolean

See also the example at `SimaProServer.CreateProcess`

SimaProServer.MainCompartmentCount

Property `MainCompartmentCount` As Integer
 Number of main-compartments
 Member of `SimaProServer`
 Read-Only

Example

List the maincompartments

```
for I := 0 to SP.MainCompartmentCount - 1 do
  print SP.MainCompartmentName(i)
```

SimaProServer.MainCompartmentName

Property `MainCompartmentName(ByVal I As Integer)` As String
 Name of a main-compartment
 Member of `SimaProServer`
 Read-Only

Parameters	Description
------------	-------------

I	Index
---	-------

Example

See `SimaProServer.MainCompartmentCount`

SimaProServer.MethodCount

Property MethodCount As Integer

Number of impact assessment methods in the currently open project and selected libraries

Member of SimaProServer

Read-Only

SimaProServer.MethodName

Property MethodName(ByVal I As Integer) As String

Name of an impact assessment method

Member of SimaProServer

Read-Only

Parameters	Description
------------	-------------

I	Index
---	-------

SimaProServer.MethodProjectName

Property MethodProjectName(ByVal I As Integer) As String

Name of the project of an impact assessment method

Member of SimaProServer

Read-Only

Parameters	Description
------------	-------------

I	Index
---	-------

SimaProServer.Network

Function Network(ByVal ProjectName As String, ByVal ProcessType As TProcessType, ByVal ProductName As String, ByVal MethodProjectName As String, ByVal MethodName As String, ByVal NWSetName As String) As Boolean

Perform the network function for a process or product stage

Member of SimaProServer

Parameters	Description
------------	-------------

ProjectName	Name of project
ProcessType	ProcessType (ptMaterial, ptEnergy, etc)
ProductName	Name of product
MethodProjectName	Project where methods are stored (often 'Methods')
MethodName	name of the Method
NWSetName	Name of normalisation weighting set

Return value

Returns Boolean

Example

Calculate a network and fetch the results of the top node

```

if SP.Network('My project', 'ptEnergy', 'Electricity', 'Methods', 'ei99', 'N/A') then
begin
  SP.NetworkCalcScore(rtIndicator, '', '', '');
  for I := 0 to SimaPro.NetworkChildNodeCount[SimaPro.NetWorkTopNodeIndex] - 1 do
  begin
    Res := SP.NetworkResult(nrProductAmount,
                           SP.NetworkChildNodeIndex(SimaPro.NetWorkTopNodeIndex, i), 0);
    print Res.ProductName, Res.Amount, Res.UnitName;
  end;
end;
end;

```

SimaProServer.NetworkCalcScore

Function NetworkCalcScore(ByVal ResultType As TResultType, ByVal Param1 As String, ByVal Param2 As String, ByVal Param3 As String) As Boolean

Calculates the node and flow scores of a network.

Member of **SimaProServer**

Parameters	Description		
ResultType	Param1	Param2	Param3
rtCharacterisation	Impact Category	-	-
rtDamage	Damage Category	-	-
rtNormalisation	Damage Category or Impact Category	-	-
rtWeighting	Damage Category or Impact Category	-	-
rtSingleScore	-	-	-
rtlInventory	MainCompartment	Subcompartment	SubstanceName

You can perform multiple NetworkCalcScores calculations on an existing Network.

Return value

Returns Boolean

Example

See SimaProServer.Network

SimaProServer.NetworkChildNodeCount

Property NetworkChildNodeCount(ByVal NodeIndex As Integer) As Integer

Number of child nodes of a network node

Member of **SimaProServer**

Read-Only

Parameters	Description
NodeIndex	Index of the node

All nodes are indexed. This function return the number of children of a certain node.

Example

See `SimaProServer.Network`

SimaProServer.NetworkChildNodeIndex

Property `NetworkChildNodeIndex(ByVal NodeIndex As Integer, ByVal FlowIndex As Integer) As Integer`

Index of a child node of a network node

Member of `SimaProServer`

Read-Only

Parameters	Description
NodeIndex	Node
FlowIndex	Flow of that node

Points to a node, for example get the productname with `NetworkProductName`

SimaProServer.NetworkNodeCount

Property `NetworkNodeCount As Integer`

Number of nodes in the network

Member of `SimaProServer`

Read-Only

SimaProServer.NetworkProductName

Property `NetworkProductName(ByVal NodeIndex As Integer) As String`

Product name of a network node

Member of `SimaProServer`

Read-Only

Parameters	Description
NodeIndex	Node

SimaProServer.NetworkResult

Function `NetworkResult(ByVal NodeResultType As TNodeResultType, ByVal NodeIndex As Integer, ByVal FlowIndex As Integer) As SimaProNetworkResult`

Retrieve the data of a network node

Member of `SimaProServer`

Parameters	Description
NodeResultType	nrProductAmount, nrIndicatorContribution, nrIndicatorTotal, nrFlowIndicator
NodeIndex	Node index see <code>NetworkNodeCount</code>
FlowIndex	Flow index (per node) See <code>NetworkChildNodeCount</code>

Return value

Returns `SimaProNetworkResult`

Example

See SimaProServer.Network

SimaProServer.NetworkTopNodeIndex

Property NetworkTopNodeIndex As Integer

Index of the top node of the network

Member of SimaProServer

Read-Only

Example

See SimaProServer.Network

SimaProServer.NWSets

Property NWSets(ByVal ProjectName As String, ByVal MethodName As String) As IStrings

List of normalisation-weighting sets in a method

Member of SimaProServer

Read-Only

Parameters	Description
ProjectName	Name of Project
MethodName	Name of Method

Example

Show the first NWSet

```
Print SP.NWSets('methods', 'ecoindicator 99')[0]
```

SimaProServer.OpenDatabase

Sub OpenDatabase()

Open a database

Member of SimaProServer

Set Server, Alias and Database first

Example

```
SP.Server := 'local server';
SP.Alias := 'C:\DATA'
SP.Database := 'Professional';
SP.OpenDatabase;
```

See also the example at SimaProServer.CreateProcess

SimaProServer.OpenProject

Sub OpenProject(ByVal ProjectName As String, ByVal Password as String)

Open a project

Member of SimaProServer

Parameters	Description
ProjectName	Project
Password	Only needed if the project is protected

Example

Open a project

```
SP.OpenProject('Introduction into LCA');
```

SimaProServer.ParamLine

Property ParamLine(ByVal ParameterType As TParameterType, ByVal ParameterScope As TParameterScope, ByVal LineNumber As Long) As ParamLine

Member of SimaProServer

Read-Only

Parameters	Description
ParameterType	Type of parameter (input or calculated)
ParameterScope	Scope of parameter (project or database)
LineNumber	Linenummer (zerobase) must be less than ParamLineCount

See process.Paramline and Paramlinecount for an example.

SimaProServer.ParamLineCount

Property ParamLineCount(ByVal ParameterType As TParameterType, ByVal ParameterScope As TParameterScope) As Long

Member of SimaProServer

Read-Only

Parameters	Description
ParameterType	Type of parameter (input or calculated)
ParameterScope	Scope of parameter (psProject or psDatabase)

For psProject a project must be open.

Returns number of parameters

See process.Paramline and Paramlinecount for an example.

SimaProServer.Product

Property Product(ByVal I As Long) As Process

Returns the process directly addressed by an index (see ProductCount)

Member of SimaProServer

Read-Only

Parameters	Description
I	Index of product

Gives direct access to the product listed with productname.

SimaProServer.ProductCategoryPath

Property ProductCategoryPath(ByVal I As Long) As String

Returns the complete category addressed by an index

Member of SimaProServer

Read-Only

Parameters	Description
I	Index

SimaProServer.ProductCount

Property ProductCount As Integer

Number of processes and product-stages in the currently open project and selected libraries

Member of SimaProServer

Read-Only

SimaProServer.ProductName

Property ProductName(ByVal I As Integer) As String

Name of a product

Member of SimaProServer

Read-Only

Parameters	Description
I	Index

Used for listing data

SimaProServer.ProductProcessType

Property ProductProcessType(ByVal I As Integer) As TProcessType

Process type of a product

Member of SimaProServer

Read-Only

Parameters	Description
I	Index

SimaProServer.ProductProcessTypeName

Property ProductProcessTypeName(ByVal I As Integer) As String
 Description of the process type of a product
 Member of SimaProServer
 Read-Only

Parameters	Description
I	Index

SimaProServer.ProductProjectName

Property ProductProjectName(ByVal I As Integer) As String
 Name of the project of a product
 Member of SimaProServer
 Read-Only

Parameters	Description
I	Index

SimaProServer.ProjectOpen

Property ProjectOpen As Boolean
 Indicates if a project is currently open
 Member of SimaProServer
 Read-Only

SimaProServer.Projects

Property Projects As IStrings
 List of available projects, open database and log in first
 Member of SimaProServer
 Read-Only

SimaProServer.QuantityCount

Property QuantityCount As Integer
 Number of quantities
 Member of SimaProServer
 Read-Only

SimaProServer.QuantityName

Property QuantityName(ByVal I As Integer) As String
 Name of a quantity
 Member of SimaProServer
 Read-Only

Parameters	Description
------------	-------------

I	Index
---	-------

Listing the quantities in the database.

SimaProServer.ResultCount

Property ResultCount(ByVal ResultType As TResultType) As Integer
 Number of indicators
 Member of SimaProServer
 Read-Only

Parameters	Description
------------	-------------

ResultType	rtCharacterisation etc,
------------	-------------------------

SimaProServer.ResultIndicatorName

Property ResultIndicatorName(ByVal ResultType As TResultType, ByVal I As Integer) As String
 Name of an indicator
 Member of SimaProServer
 Read-Only

Parameters	Description
------------	-------------

ResultType	rtCharacterisation etc
------------	------------------------

I	Index
---	-------

SimaProServer.ResultMainCompartmentName

Property ResultMainCompartmentName(ByVal I As Integer) As String
 Name of the main-compartment of a substance
 Member of SimaProServer
 Read-Only

Parameters	Description
I	Index

SimaProServer.ResultSubCompartmentName

Property ResultSubCompartmentName(ByVal I As Integer) As String
 Name of the sub-compartment of a substance
 Member of SimaProServer
 Read-Only

Parameters	Description
I	Index

SimaProServer.SaveParameters

Sub SaveParameters()
 Saving the changes in the parameters on database and project level
 Member of SimaProServer

SimaProServer.Server

Property Server As String
 Currently used server, e.g. 'local server' or 'myserver@w.p1.local'
 Member of SimaProServer

SimaProServer.Servers

Property Servers As IStrings
 List of available servers
 Member of SimaProServer
 Read-Only

SimaProServer.SubCompartmentCount

Property SubCompartmentCount(ByVal MainCompartmentName As String) As Integer
 Number of sub-compartments
 Member of SimaProServer
 Read-Only

Parameters	Description
MainCompartmentName	Name of main compartment

SimaProServer.SubCompartmentName

Property `SubCompartmentName(ByVal MainCompartmentName As String, ByVal I As Integer) As String`
 Name of a sub-compartment
 Member of `SimaProServer`
 Read-Only

Parameters	Description
MainCompartmentName	name of maincompartment
I	Index

SimaProServer.SubstanceCASNumber

Property `SubstanceCASNumber(ByVal MainCompartmentName As String, ByVal I As Integer) As String`
 CAS number of a substance
 Member of `SimaProServer`
 Read-Only

Parameters	Description
MainCompartmentName	Maincompartment ('Air', 'Water', etc)
I	Index

Meant for listing the substances. Addressing with index.

Example
 See `FindSubstance`

SimaProServer.Substance

Property `Substance(ByVal MainCompartmentName As String, ByVal I As Long) As Substance`
 Return a substance object addressed by an index (see `SubstanceCount`)
 Member of `SimaProServer`
 Read-Only

Parameters	Description
MainCompartmentName	Maincompartment
I	Index

SimaProServer.SubstanceCount

Property `SubstanceCount(ByVal MainCompartmentName As String) As Integer`
 Number of substances

Member of **SimaProServer**
Read-Only

Parameters	Description
------------	-------------

MainCompartmentName	Maincompartment ('Air', 'Water', etc))
----------------------------	--

Meant for listing the Substances.

Example

List all substances

```
For I := 0 to SimaPro.SubstanceCount('Air') - 1 do
  print SimaPro.SubstanceName('Air', I);
```

SimaProServer.SubstanceDefaultUnit

Property SubstanceDefaultUnit(ByVal MainCompartmentName As String, ByVal I As Integer) As String

Default unit of a substance

Member of **SimaProServer**

Read-Only

Parameters	Description
------------	-------------

MainCompartmentName	Maincompartment ('Air', 'Water', etc))
----------------------------	--

I	Index
----------	-------

Meant for listing the substances. Addressing with index.

SimaProServer.SubstanceName

Property SubstanceName(ByVal MainCompartmentName As String, ByVal I As Integer) As String

Name of a substance

Member of **SimaProServer**

Read-Only

Parameters	Description
------------	-------------

MainCompartmentName	Maincompartment ('Air', 'Water', etc))
----------------------------	--

I	Index
----------	-------

Meant for listing the substances. Addressing with index.

Example

List all substances

```
For I := 0 to SimaPro.SubstanceCount('Air') - 1 do
  print SimaPro.SubstanceName('Air', I);
```

SimaProServer.Tree

Function Tree(ByVal ProjectName As String, ByVal ProcessType As TProcessType, ByVal ProductName As String, ByVal MethodProjectName As String, ByVal MethodName As String, ByVal NWSetName As String) As Boolean

Perform the tree function for a process or product stage

Member of **SimaProServer**

Parameters	Description
ProjectName	Name of project
ProcessType	ProcessType (ptMaterial, ptEnergy, etc)
ProductName	Name of product
MethodProjectName	Name of project where methods is stored (often 'methods')
MethodName	Name of method
NWSetName	Name of normalisation weighting set
Return value	
Returns Boolean	

SimaProServer.TreeCalcScore

Function TreeCalcScore(ByVal ResultType As TResultType, ByVal Param1 As String, ByVal Param2 As String, ByVal Param3 As String) As Boolean

Calculates the node and flow scores of a tree

Member of SimaProServer

Parameters	Description
ResultType	Param1 Param2 Param3
rtCharacterisation	Impact Category - -
rtDamage	Damage Category - -
rtNormalisation	Damage Category - -
	or Impact Category
rtWeighting	Damage Category - -
	or Impact Category
rtSingleScore	- - -
rtInventory	MainCompartment Subcompartment SubstanceName
Return value	
Returns Boolean	

SimaProServer.TreeChildNodeCount

Property TreeChildNodeCount(ByVal NodeIndex As Integer) As Integer

Number of child nodes of a tree node

Member of SimaProServer

Read-Only

Parameters	Description
NodeIndex	Node index

SimaProServer.TreeChildNodeIndex

Property TreeChildNodeIndex(ByVal NodeIndex As Integer, ByVal FlowIndex As Integer) As Integer

Index of a child node of a tree node

Member of SimaProServer

Read-Only

Parameters	Description
------------	-------------

NodeIndex	Nodeindex
------------------	-----------

FlowIndex	Flowindex
------------------	-----------

SimaProServer.TreeNodeCount

Property TreeNodeCount As Integer

Number of nodes in the tree

Member of SimaProServer

Read-Only

SimaProServer.TreeProductName

Property TreeProductName(ByVal NodeIndex As Integer) As String

Product name of a tree node

Member of SimaProServer

Read-Only

Parameters	Description
------------	-------------

NodeIndex	Nodeindex (refers to list of Nodes of network or tree)
------------------	--

SimaProServer.TreeResult

Function TreeResult(ByVal NodeResultType As TNodeResultType, ByVal NodeIndex As Integer) As SimaProTreeResult

Retrieve the data of a tree node

Member of SimaProServer

Parameters	Description
------------	-------------

NodeResultType	nrProductAmount, nrIndicatorContribution, nrIndicatorTotal, nrFlowIndicator
-----------------------	---

NodeIndex	Nodeindex (refers to list of Nodes of network or tree)
------------------	--

Return value

Returns SimaProTreeResult

SimaProServer.TreeTopNodeIndex

Property TreeTopNodeIndex As Integer
 Index of the top node of the tree
 Member of SimaProServer
 Read-Only

SimaProServer.UnitCount

Property UnitCount(ByVal QuantityName As String) As Integer
 Number of units per quantity
 Member of SimaProServer
 Read-Only

Parameters	Description
------------	-------------

QuantityName	Quantity
--------------	----------

Example

Number of 'Mass' units

```
for I := 0 to Sp.unitCount('Mass') - 1 do
  print Sp.UnitName('Mass', I);
```

SimaProServer.UnitDefault

Property UnitDefault(ByVal QuantityName As String) As String
 Default unit of a quantity (factor = 1)
 Member of SimaProServer
 Read-Only

Parameters	Description
------------	-------------

QuantityName	Quantity ('Mass', 'Volume' etc)
--------------	---------------------------------

SimaProServer.UnitFactor

Property UnitFactor(ByVal QuantityName As String, ByVal I As Integer) As Double
 Factor of a unit
 Member of SimaProServer
 Read-Only

Parameters	Description
------------	-------------

QuantityName	Quantity ('Mass', 'Volume' etc)
--------------	---------------------------------

I	Index
---	-------

SimaProServer.UnitMetric

Property UnitMetric(ByVal QuantityName As String, ByVal I As Integer) As Boolean
 Indicates if a unit is metric

Member of **SimaProServer**
Read-Only

Parameters	Description
QuantityName	Quantity ('Mass', 'Volume' etc)
I	Index

SimaProServer.UnitName

Property UnitName(ByVal QuantityName As String, ByVal I As Integer) As String
Name of a unit
Member of **SimaProServer**
Read-Only

Parameters	Description
QuantityName	Quantity ('Mass', 'Volume' etc)
I	Index

Example

Number of 'Mass' units

```
for I := 0 to Sp.unitCount('Mass') - 1 do
    print Sp.UnitName('Mass', I);
```

SimaProServer.WasteTypeCount

Property WasteTypeCount As Integer
Number of waste-types
Member of **SimaProServer**
Read-Only

SimaProServer.WasteTypeName

Property WasteTypeName(ByVal I As Integer) As String
Name of a waste-type
Member of **SimaProServer**
Read-Only

Parameters	Description
I	Index

SimaProTreeResult

Class SimaProTreeResult

Object containing tree flows resulting from the calculation of a process object.

SimaProTreeResult properties

SimaProTreeResult Legend

- ▶ Amount
- ▶ ProductName
- ▶ UnitName
- ▶ Valid

SimaProTreeResult.Amount

Property Amount As Double

Amount of the product

Member of SimaProTreeResult

SimaProTreeResult.ProductName

Property ProductName As String

Name of the product

Member of SimaProTreeResult

SimaProTreeResult.UnitName

Property UnitName As String

Unit of the amount

Member of SimaProTreeResult

SimaProTreeResult.Valid

Property Valid As Boolean

Indicates if the node is part of the tree

Member of SimaProTreeResult

StandardDeviation Property

Select one of the available subtopics below to see detailed help on **StandardDeviation** property

Object	Property description
ParamLine	Standard deviation of the input parameter
ProcessLine	Standard deviation of amount

SubCompartmentName Property

Select one of the available subtopics below to see detailed help on **SubCompartmentName** property

Object	Property description
SimaProAnalyseResult	
SimaProServer	Name of a sub-compartment

Substance

Class Substance

Properties Methods

Represents a substance in SimaPro. Use to edit, find or use substances.

Substance methods

Substance Legend

- ▶ Cancel
- ▶ Edit
- ▶ Update

Substance properties

Substance Legend

- ▶ CASNumber
- ▶ Comment
- ▶ DefaultUnit
- ▶ MainCompartment
- ▶ Mode
- ▶ Name

Substance.Cancel

Sub Cancel()

Cancel edit mode and returns to read mode

Member of **Substance**

Substance.CASNumber

Property CASNumber As String

CAS number

Member of **Substance**

Example

```
Substance.Casnumber := '45-32-45'
```

Substance.Comment

Property Comment As IStrings

Comment

Member of **Substance**

Substance.DefaultUnit

Property DefaultUnit As String

Default unit, defines also the quantity

Member of **Substance**

Substance.Edit

Sub Edit()

Set the object in edit mode

Member of **Substance**

Substance.MainCompartment

Property MainCompartment As String

Main compartment (e.g. 'Airborne emission')

Member of **Substance**

Read-Only

Substance.Mode

Property Mode As String

Mode of the object, can be: Read, New or Edit

Member of **Substance**

Read-Only

Substance.Name

Property Name As String

Name of the substance (e.g. 'Carbon dioxide')

Member of **Substance**

Substance.Update

Sub Update()

Store the data of the object in the database and switch to read mode

Member of **Substance**

Example

Create a new substance

```
SimaPro.CreateSubstance('Air', Substance)
Substance.Name := 'My new substance'
Substance.UnitName := 'kg';
Substance.Update; // save in database
```

TDistribution

Enum TDistribution

Constant	Value	Description
dsUndefined	0	Distribution is not defined
dsLogNormal	1	Lognormal
dsNormal	2	Normal (Gaussian)
dsTriangle	3	Triangle
dsUniform	4	Uniform

TNodeResultType

Enum TNodeResultType

Type of result from a network or tree node

Constant	Value	Description
nrProductAmount	0	Amount of a product
nrIndicatorContribution	1	Contribution of a product to the selected indicator
nrIndicatorTotal	2	Contribution of a product including all sub-processes to the selected indicator
nrFlowIndicator	3	Contribution of a flow to the selected indicator

TParameterType

Enum TParameterType

Constant	Value	Description
ptInputParameter	0	Parameter is a constant value optional with distribution data
ptCalculatedParameter	1	Parameter is an expression

TProcessPart

Enum TProcessPart

Parts of a process

Constant	Value	Description
ppProducts	0	Products (outputs)
ppMaterialsFuels	1	Inputs from technosphere (other processes)
ppElectricityHeat	2	Inputs from technosphere (other processes)
ppAvoidedProducts	3	Avoided product
ppWasteToTreatment	4	Waste
ppRawMaterials	5	Use of resources (raw materials)
ppAirborneEmissions	6	Emissions to air
ppWaterborneEmissions	7	Emissions to water
ppFinalWasteFlows	8	Emissions to waste
ppEmissionsToSoil	9	Emissions to soil
ppNonMaterialEmissions	10	Non material emissions
ppSocialIssues	11	Social issues
ppEconomicIssues	12	Economic issues
ppSpecificWaste	13	Outputs to specific waste
ppRemainingWaste	14	Remaining waste
ppSubAssembly	15	Subassembly (product stages only)
ppReferencedAssembly	16	Referenced assembly (product stages only)
ppAssembliesAndMaterials	17	Assemblies or materials (product stages only)
ppProcesses	18	Process (product stages only)
ppWasteScenarios	19	Waste scenario (product stages only)
ppDisposalScenarios	20	Disposal scenario (product stages only)
ppAdditionalLifeCycles	21	Additional life cycle (Life cycle product stage only)

ppDisassemblies	22	Disassembly (product stages only)
ppReuses	23	Reuse (product stages only)
ppWasteOrDisposalScenario	24	Waste or disposal scenario (product stages only)

TProcessStatus

Enum TProcessStatus

Constant	Value	Description
stEmpty	0	No status
stTemporary	1	Temporary process
stDraft	2	Draft, work to be done
stToBeRevised	3	To be revised
stToBeReviewed	4	To be reviewed
stFinished	5	Finished

TProcessType

Enum TProcessType

Constant	Value	Description
ptMaterial	0	Material process
ptEnergy	1	Energy Process
ptTransport	2	Transport process
ptProcessing	3	Processing process
ptUse	4	Use process
ptWasteScenario	5	Waste scenario
ptWasteTreatment	6	Waste treatment
ptAssembly	7	Assembly product stage
ptLifeCycle	8	Life cycle product stage
ptDisposalScenario	9	Disposal scenario
ptDisassembly	10	Disassembly
ptReuse	11	Reuse

TResultType

Enum TResultType

Type of result

Constant	Value	Description
rtCharacterisation	0	Characterisation score

rtDamage	1	Damage score
rtNormalisation	2	Normalised score
rtWeighting	3	Weighted score
rtSingleScore	4	Single score
rtInventory	5	Inventory results (LCI)

UnitName Property

Select one of the available subtopics below to see detailed help on **UnitName** property

Object	Property description
--------	----------------------

ProcessLine	Name of the unit of amount
SimaProAnalyseResult	Unit (e.g. kg, m3)
SimaProNetworkResult	Unit of the amount
SimaProServer	Name of a unit
SimaProTreeResult	Unit of the amount

Update Method

Select one of the available subtopics below to see detailed help on **Update** method

Object	Method description
--------	--------------------

Process	Store the data of the object in the database and switch to read mode
Substance	Store the data of the object in the database and switch to read mode

Index

- Add, 32
- AdditionalInfo, 26
- AddLine, 13, 32
- AddParamLine, 12, 14, 29
- Alias, 30, 41
- Aliases, 30
- Allocation, 20
- Amount, 8, 20, 25, 27, 53
- Analyse, 30
- AnalyseResult, 30, 31
- CalculationError, 31
- CalculationErrorCount, 31
- Cancel, 8, 14, 15, 54
- CASnumber, 36, 55
- CategoryPath, 20, 32
- ChildProductName, 27
- CloseDatabase, 32
- CloseProject, 32
- Comment, 8, 10, 14, 20, 55
- CreateProcess, 32
- CreateSubstance, 33, 56
- CurrentProject, 33
- CurrentUser, 33
- Database, 34, 41
- DatabaseOpen, 34
- Databases, 34
- DefaultUnit, 55
- Delete, 15
- DeleteLine, 15, 16, 17, 35
- DeleteParamLine, 15, 34
- Distribution, 8, 10, 20, 56
- Edit, 8, 15, 16, 17, 35, 55
- ErrorCode, 26
- ErrorDescription, 27
- Expression, 10
- FindParameter, 12, 14, 16, 34
- FindParamLine, 29
- FindProcess, 15, 16, 17, 18, 35
- FindProcessEx, 35
- FindSubstance, 36
- Hide, 11
- IndicatorName, 25
- Line, 16, 17, 35
- LineCount, 17, 35
- LineNumber, 8, 11, 21
- LoggedIn, 36
- Login, 36
- Logout, 37
- MainCompartment, 55
- MainCompartmentCount, 37
- MainCompartmentName, 9, 25, 37
- Maximum, 9, 11, 21
- MethodCount, 38
- MethodName, 38
- MethodProjectName, 38
- Minimum, 9, 11, 21
- Mode, 9, 17, 56
- Name, 9, 11, 56
- Network, 38
- NetworkCalcScore, 38, 39
- NetworkChildNodeCount, 38, 39
- NetworkChildNodeIndex, 38, 40
- NetworkNodeCount, 40
- NetworkProductName, 40
- NetworkResult, 38, 40
- NetworkTopeNodeIndex, 38
- NetworkTopNodeIndex, 41
- NWsets, 41
- ObjectName, 13, 21, 32
- ObjectName2, 16, 17, 21, 35
- OpenDatabase, 41
- OpenProject, 42
- ParameterType, 12
- ParamLine, 9, 42
 - method, 17
 - properties, 10
- ParamLineCount, 18, 42
- Part Part, 21
- Process, 12, 13, 22, 24
 - methods, 13
 - properties, 13
- ProcessLine, 15, 18, 19
 - methods, 19
 - properties, 19
- ProcessType, 18, 22, 24
- Product, 43
- ProductCategoryPath, 43
- ProductCount, 43
- ProductName, 24, 27, 43, 53
- ProductProcessType, 43
- ProductProcessTypeName, 44
- ProductProjectName, 44
- ProjectName, 18, 22, 24
- ProjectName2, 22
- ProjectOpen, 44
- Projects, 44
- QuantityCount, 45
- QuantityName, 45
- ResultCount, 30, 45
- ResultIndicatorName, 45
- ResultMainCompartmentName, 45
- ResultSubCompartmentName, 46
- SaveParameters, 46
- Server, 41, 46
- Servers, 46
- SetMaterial, 22
- SetProduct, 20, 23, 32
- SetSubstance, 13, 23
- SimaProAnalyseResult, 25
 - properties, 25
- SimaProCalculationError, 26
 - properties, 26
- SimaProNetworkResult, 27
 - properties, 27

SimaProServer, 28
 methods, 28
 properties, 28
SimaProTreeResult, 53
 properties, 53
StandardDeviation, 12, 23, 54
Status Status, 18
SubCompartmentCount, 46
SubCompartmentName, 26, 47, 54
Substance, 47, 54
 methods, 54
 properties, 54
SubstanceCASnumber, 36, 47
SubstanceCount, 47, 48
SubstanceDefaultUnit, 48
SubstanceName, 47, 48
TNodeResultType, 56
TParameterType, 57
TProcessPart, 57
TProcessStatus, 58
TProcessType, 58
Tree, 48
TreeCalcScore, 49
TreeChildNodeCount, 49
TreeChildNodeIndex, 50
TreeNodeCount, 50
TreeProductName, 50
TreeResult, 50
TreeTopNodeIndex, 51
TResultType, 49, 58
Uncertainty, 8, 10, 20, 56
UnitCount, 51, 52
UnitDefault, 51
UnitFactor, 51
UnitMetric, 51
UnitName, 23, 26, 28, 51, 52, 53, 56, 59
Update, 15, 16, 17, 18, 56
 methods, 59
Valid, 53
Value, 12
WasteType, 15, 18, 24
WasteTypeCount, 52
WasteTypeName, 52