## LUSAS

## LUSAS Programmable Interface (LPI)

Developer Guide

### LUSAS Programmable Interface (LPI) Developer Guide

LUSAS Version 18.0 : Issue 2

#### LUSAS

Forge House, 66 High Street, Kingston upon Thames, Surrey, KT1 1HN, United Kingdom

> Tel: +44 (0)20 8541 1999 Fax +44 (0)20 8549 9399 Email: info@lusas.com http://www.lusas.com

Distributors Worldwide

Copyright ©1982-2019 LUSAS All Rights Reserved.

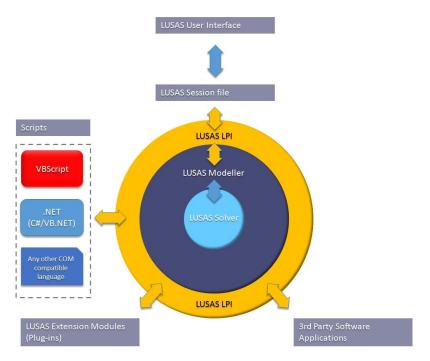
#### **Table of Contents**

Introduction	1
Introduction	
Topics covered in this guide	2
LUSAS Progammable Interface (LPI) Customisation and Automation Guide	2
Creating dialogs using VB.NET	3
Choosing a development environment	3
Downloading and installing Visual Studio Community	4
Creating a LUSAS dialog	4
Module Manager	6
Creating a new module	7
Running Visual Studio	7
Build the project	11
Run the project	
Adding dialog controls	14
Defining a ListBox	17
Defining a Delete Button	18
Defining a Cancel Button	
Handling errors	
General considerations	
Basic dialog design	
Basic dialog controls	
Code design considerations	
Multiple Dialogs in a single module	
Translation considerations	
VB.NET online tutorials	
VB.NET dialog exercise	
VB.NET dialog solution	
LUSAS via COM	33
Component Technology	
Application Example	
Create a new project	
Interfacing to LUSAS using C++	
LUSAS Material Model Interface	43

### Introduction

#### Introduction

LUSAS software is highly customisable. The built-in LUSAS Programmable Interface (LPI) allows the customisation and automation of modelling and results processing tasks and creation of user-defined menu items, dialogs and toolbars as a means to access those user-defined resources. It can also be used for transferring data between LUSAS and other software applications, and to control other programs from within LUSAS Modeller, or control LUSAS Modeller from other programs.



With LPI, any user can automate the creation of complete structures, either in LUSAS or from third-party software, carrying out design checks, optimising members and outputting graphs, spreadsheets of results and custom reports. Because everything carried out by a user is recorded in a LUSAS Modeller session file, anything that LUSAS can do, can also be controlled by another application via the LUSAS Programmable Interface. This means that you can view and edit a recorded session,

parameterise those commands, turn them into sub-routines, add loops and other functions to the scripts and create a totally different application or program - using the proven core technology of LUSAS.

In addition to the accessing and customising LUSAS Modeller via the LUSAS Programmable Interface, user-defined material models (written in Fortran) can be compiled and built into a customised LUSAS Solver executable by using the LUSAS Material Model Interface (LUSAS MMI).

#### **Topics covered in this guide**

The aim of this guide is to help you use the more advanced facilities available to customise LUSAS and interface with other applications. The guide covers:

- **Creating dialogs using VB.NET**
- □ LUSAS via COM
- LUSAS Material Model Interface

#### LUSAS Progammable Interface (LPI) Customisation and Automation Guide

A separate LUSAS Programmable Interface (LPI) Customisation and Automation Guide is also available covering more basic topics:

- Getting started with the LUSAS Programmable Interface (LPI)
- **Identifying LPI Functions**
- **Customising the interface**
- **Getting started with VBS**
- □ A simple example script
- **Creating your own menus**

# Creating dialogs using VB.NET

A dialog is a window that appears on the display screen to neatly present information or request input from the user. Creating dialogs, and some other advanced features, such as object oriented programming, is a more advanced topic, and some programming skills are required.

VBScript belongs to the group of interpreted languages, meaning that there is no need for compilers to be used. This is because the implementations execute directly, line by line. Examples of other interpreted languages are JavaScript, Python and Perl.

To create a dialog you will need to use a compiled programming language. Examples of compiled languages are Visual Basic .NET (VB.NET), C#, Java, C++. For this example VB.NET will be used, because although it is a different language to VBScript, it uses the same syntax, and you should be already familiar with it. Also, and more importantly, you do not need to be familiar with COM programming, as you would if C++ was used instead.

Advantages of VB.NET over VBScript include

- Better user-interface creation
- Debugging with break points
- More extensible. Allows Object Oriented Programming (OOP)
- Multiple functions saved into one .dll file. Easier to share across the company than multiple .vbs script files
- Easier to do testing

#### **Choosing a development environment**

To use VB.NET a compiler needs to be installed. Both Visual Studio Express and Visual Studio Community are free to use and could be used but the **Visual Studio Community** edition, which is a fully featured and extensible Integrated Development Environment (IDE) is the application recommended by LUSAS and the application that is documented in this guide. This is used in preference to the more limited Visual

Studio Express version, which is a closed IDE that cannot use any Visual Studio extensions.

Visual Studio Community is free for individual developers, open source projects, academic research, education, and small professional teams. Users should be aware of the licensing terms under which it is supplied.

#### **Downloading and installing Visual Studio Community**

Visual Studio Community is free to use and can be downloaded from:

https://www.visualstudio.com/downloads



**Note.** Problems can be experienced when installing Visual Studio Community from a downloaded exe file (relating to the installation being unable to locate the package source as a result of you attempting to install the software when you are part of a company network). To overcome this issue, download the .iso file instead and burn it onto a DVD. Then install it from the DVD (or alternatively virtually mount the iso image).

Visual Studio Community can take a long time to install.

#### **Creating a LUSAS dialog**

To create a LUSAS dialog, you will need to add a new **Module** to LUSAS. Modules are like plugins.

- 1. Run LUSAS.
- 2. All the installed modules can be seen by selecting the menu item Help > About LUSAS Modeller and checking "Show all installed components"

bout LUSAS Modeller					X
🛃 LUSAS				Copyright 2015 Finite Element Analys	sis Ltd
for 32-bit Windows XF	P, Vista, 7,	8, 8.1, 10			
LUSAS	Sales			Support	
66, High Street	Tel: -	+44 (0)20 8541 199		Tel: +44 (0)20 8549 9229	
Kingston Upon Thames Surrey, KT1 1HN, UK	Eax:	+44 (0)20 8549 939	9	Fax: +44 (0)20 8549 9399	
www.lusas.com		: info@lusas.com	-		
www.iusas.com	email	: mo@usas.com		email: support@lusas.com	
it version: 15.1-2c1	Install	date: 16/12/2015			
nstalled components	1150011	0000. 10/12/2010			
File Name		Version	Descri	otion	
🐏 LusasRes2		15.1.1617.22304	LUSAS	res2 Dynamic Link Library	
Modules					1
😼 Arbitrary Section Property	Calculator	15.1.1617.22304	Arbitra	ary Section Property Calculator - LU.	•
🐿 Composite Library		15.1.1617.22304	Compo	osite Library - LUSAS Extension Modul	e
😼 Construction Table		15.1.1617.22304	Constr	ruction Table - LUSAS Extension Mo	
🐏 Crack Widths		15.1.1617.22304	Crack	Widths - LUSAS Extension Module	-
Show all installed component	ts			Copy to clipbo	bard
				Key info	•
Currently using 1043.6MB of a p	oossible 40	95.9MB (3052.3MB)	free)	Build info	
<b>D</b> . I	v			3rd party licen	ses
Release reserved memor	,				

#### **Module Manager**

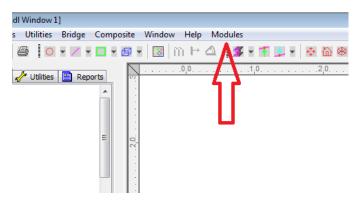
LUSAS Modeller's Modules are controller by the Module Manager and the Module Manager dialog can be displayed by following the steps below.

- 1. Open the file C:\<LUSAS Installation Folder>\Programs\Modules\LUSAS.lml into a text file editor.
- 2. Search for the text **ModuleManager**
- 3. Enable it by changing **false** to **true** as shown below

D Ope	en Burn New folder		-	1.00	dit Search View Encoding Language Settings Macro
	en Burn New folder			l 🔓 🖻	) 🗄 🛍 💪 🕼 🎝   🎸 🗅 🛍   Ə 🖒 🗰 🍢   🤏
, i	Name	Date modified	Туре	😑 LUS	iAS.Iml
p	ApproximatePeriodCalculation.dll	03/06/2019 10:40	Applicat	486	<identity></identity>
bads	ArbitrarySectionPropertyCalculator.dll	03/06/2019 11:01	Applicat	487	<friendlyname>Load Combination Wiza</friendlyname>
Places	Building.dll	03/06/2019 10:40	Applicat	488	<description>Load Combination Wizar</description>
Flaces	CompositeLibrary.dll	03/06/2019 11:01	Applicat	489	
	ConstructionTable.dll	03/06/2019 11:01	Applicat	490	<keys></keys>
	CrackWidths.dll	03/06/2019 11:01	Applicat	491	<verified></verified>
ents :	DesignAttributes.dll	03/06/2019 11:01	Applicat	492	<version>15.2</version>
	DesignFactors.dll	03/06/2019 11:01	Applicat	493	
5	DesignReport.dll	03/06/2019 11:01	Applicat	494	<options></options>
ion	EN1993.dll	03/06/2019 11:01	Applicat	495	<decrypt>true</decrypt> <exact>true</exact>
	Sectors State	03/06/2019 11:01	Applicat	496	<pre><exact>true</exact> </pre>
	Grillage.dll	03/06/2019 11:01	Applicat	498	
r	MDPlus dll			499	<module name="ModuleManager"></module>
		03/06/2019 11:01	Applicat	500	<assembly></assembly>
	LUSAS.Iml	26/06/2019 11:45	LML File	501	<path>%interopbindir%</path>
RDP	MaterialLibrary.dll	03/06/2019 11:01	Applicat	502	<file>ModuleController.dll</file>
0	MovingLoadGenerator.dll	03/06/2019 11:01	Applicat	503	<class>Lusas.ModuleController.Modul</class>
	PedestrianMovingLoad.dll	03/06/2019 11:01	Applicat	504	
01	PlasticHinges.dll	03/06/2019 11:01	Applicat	505	<identity></identity>
03	PrecastSection.dll	03/06/2019 11:01	Applicat	506	<friendlyname>Module Manager</friendlyname>
91	PrestressSS.dll	03/06/2019 11:01	Applicat	507	<description>Module Manager - LUSAS</description>
EST	Pushover.dll	03/06/2019 11:01	Applicat	508	
LT01	Renumbering.dll	03/06/2019 11:01	Applicat	509 510	<keys></keys>
LT02	ResponseSpectra.dll	03/06/2019 11:01	Applicat	510	<verified></verified> <options></options>
P00				511	<force>true</force>
50B	SeismicStoreyLoading.dll	03/06/2019 11:01	Applicat	513	<pre><enabled>true</enabled></pre>
	SlabDesigner.dll	03/06/2019 11:01	Applicat	514	
	SliceResultants.dll	03/06/2019 11:01	Applicat	515	
L	StoreyLoading.dll	03/06/2019 11:01	Applicat	516	<module name="Lusas.Utils"></module>
LD03	TemperatureProfiles.dll	03/06/2019 11:01	Applicat	517	<assembly></assembly>
S	TimeManagement.dll	03/06/2019 11:01	Applicat	518	<pre><path>%interopbindir%</path></pre>

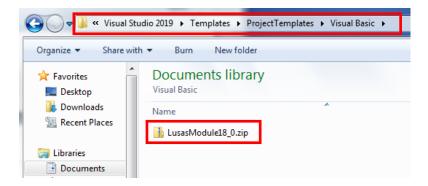
4. Save the file.

From now on, when you run LUSAS Modeller you will see the Modules menu:



#### **Creating a new module**

 Copy the LusasModule<version\_number>.zip file from the LUSAS installation directory C:\<LUSAS Installation Folder>\Programs to the Visual Studio project template folder. For instance: Libraries\Documents\Visual Studio 2019\Templates\ProjectTemplates\Visual Basic



#### **Running Visual Studio**

- Open Visual Studio and select File>Create a new Project
- In Visual Studio scroll to the bottom and select the LUSAS Module<version number> template for your version of LUSAS) from the available list>Next.
- The dialog example that will be covered in this guide creates a new module to delete loadcases, therefore enter the name **DeleteLoadcases** (without any space) and click **OK**.



**Note.** It is important to use a sensible name as this propagates throughout the automatically generated code.

**Note.** Spaces can be used within Module names, but LUSAS generally avoids the use of spaces in its own scripted file names.

Configure your new project		
LUSAS Module 18.0		
Project name	_	
DeleteLoadCases		
Location		
C:\Users\matth\source\repos		
Solution name 👔		
DeleteLoadCases		
Place solution and project in the same directory		



**Note.** If more than one version of LUSAS is installed on your machine you should check the version number of LUSAS that is being referenced by Visual Studio and ensure that this number is what is required. This can be done by checking the path to the LusasInterop dll.

• If this is required, on the **Solution Explorer** double-click on **My Project** and visually check the path for LusasInterop.dll

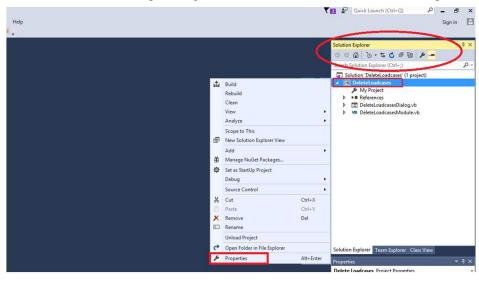
Serve	DeleteLoadCases* -= ×						Solution Explorer
ĝ	Application	Configurations N/A			Platform:	VA	0048.0.
lore	Compile						Search Solution Explorer (Ct
7	Debug					Reference Paths	Solution 'DeleteLoadC
₽¢	References	References:					My Project
8	Resources				Copy Local		References
Par 1	Services			18.0.1637.31413	False	C:\Program Files (x86)\LUSAS180\Programs\Lusas.Interop.dll	DeleteLoadCase
ŝ				4.0.0.0	False	C/\Program Files (x80)\Keterence Assemblies\Microsoft\FrameworkNETFramework\v4.0\System.dll	VB DeleteLoadCase
N.	Settings				False	C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0\System.Drawing.dll	
- 84	Signing	System.Windows.Forms	.NET	4.0.0.0	Faise	C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0\System.Windows.Forms.dll	
	My Extensions						
	Code Analysis						Solution Explorer Team Ex
							Properties
							Properties
							않 및4 🔑
						Add Remove Update	
						Add - Remove Update	

• If the wrong version of LUSAS is being referenced, click **Add** and search for the Lusas.Interop.dll file in the right version path i.e. :\<LUSAS Installation

Select the files to reference						23
💽 🗢 📕 « OS (C:) 🕨	Program Files (x86) 🕨 LU	USAS180 🕨 Programs 🕽	•	✓ Search Programs		٩
Organize 🔻 New folde	r			:==	•	0
Microsoft Visual St	Name		Date modified	Туре	Size	*
repos	🚳 Isapiw32.dll		25/04/2019 17:48	Application extens	3,009 KB	
	Isinit32.dll		25/04/2019 17:48	Application extens	1,158 KB	
🛠 Favorites	🗟 Lusas.Interop.dll		03/06/2019 10:12	Application extens	1,949 KB	
Folder>\P	rograms\Lusas	s.Interop.dll a	nd click add			

#### Carrying on:

• In the Solution Explorer right-click on DeleteLoadcases and choose Properties



• On the Application page set the Target framework to .NET Framework 4

DeleteLoadCases* + 2	
Compile	Configuration: N/A   Platform: N/A
Debug	Assembly name: Root namespace:
References	DeleteLoadCases Lusas.DeleteLoadCases
Resources	Target framework: Application type:
Services	.NET Framework 4   Class Library
Settings	Auto-generate binding redirects
Signing	Startup object: Icon:
My Extensions	(None)
Code Analysis	
	Assembly Information View Windows Settings
	☑ Enable application framework



**Note.** For LUSAS v18, .NET Framework 4 is needed. Later versions of LUSAS will require a different target framework to be set.

 On the Compile page, browse and change the build output path to Libraries\Documents\LUSAS180 folder>\Modules as it is a user-made module dialog.

Serv	Report20190626-1449.dia	gsession DeleteLoadCases + X	-
Server Explorer	Application	Configuration: Active (Debug)   Platform: Active (Any CPU)	ł
lore	Compile		I.
	Debug	Build output path:	I.
Toolbox	References	C:\Users\matth\Documents\LUSAS180\Modules\ Browse	I.
	Resources	Compile Options:	I.
Data Sources	Services	Option explicit: Option strict:	I.
purci	Settings	On V Off	I.
CR	Signing	Option compare: Option infer:	I.
	My Extensions	Binary V On	I.
	Code Analysis	Unitive Contraction Contractio	

• On the **Debug** page, browse and change the external program path to C:\<LUSAS Installation Folder>\Programs\lusas\_m.exe

Serv	DeleteLoadCases* 😕 🔾	< <u> </u>	Solution Explorer
Server Explorer	Application	Configuration: Active (Debug)	○ ○ ☆ ∺
lore	Compile		Search Solution Ex
	Debug*	Start action	Solution 'DeleteLo
Toolbox	References	Start project	My Pro
	Resources	Start external program: C:\Program Files (x86)\LUSAS180\Programs\lusas_m.exe     Browse	▷ ■ ■ Refere
)ata	Services	Start external program: C:\Program Files (x86)\LUSAS180\Programs\lusas_m.exe     Browse	Delete
Data Sources	Settings	Start browser with URL:	Þ <b>v</b> B Delete
ß	Signing	Start options	
	My Extensions	Command line arguments:	
	Code Analysis		

#### **Build the project**

• In the Solution Explorer right-click on the **DeleteLoadcases** entry and select **Build**.



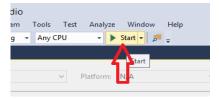


Note. In Visual Studio you can also press F7 to build the project.

Details of the build will appear in the output window.

#### **Run the project**

• Press the F5 Key in Visual Studio to run (or click on start button)



LUSAS Modeller should start.

- Open a LUSAS model or create a new one.
- Open the Module Manager

ite Window Help	_			_	_			
🗗 🕶   🖾   în F		Modu	ile Mana	ger	¢	合《	₿ -	N
20.0 , -18.0		-16.	.0 ,	-14.0		-	12.0	

• Click Add new module

Name	Description	Version	State	
Approximate Period Calcul	Approximate Period Calcul	18.0.1637.31413	Running	
Arbitrary Section Property	Arbitrary Section Property	18.0.1637.31413	Running	E
Building	Building - LUSAS Extensio	18.0.1637.31413	Running	
CCFModule	CCFModule - LUSAS Exte	18.0.1637.31413	Running	
Composite Library	Composite Library - LUSAS	18.0.1637.31413	Running	
Construction Table	Construction Table - LUSA	18.0.1637.31413	Running	
cp27b effective length	cp27b effective length - L	18.0.1637.31413	Running	
Crack Widths	Crack Widths - LUSAS Ext	18.0.1637.31413	Running	
Crane Load Wizard	Crane Load Wizard - LUS	18.0.1637.31413	Running	
Define Member Type	Define Member Type - LU	18.0.1637.31413	Running	
Design Attributes For China	Design Attributes For Chin	18.0.1637.31413	Running	
Design Attributes	Design Attributes - LUSAS	18.0.1637.31413	Running	
Design Factors	Design Factors - LUSAS	18.0.1637.31413	Running	-
Send command	Configuration	Module	Push verbosit	ty
Send command addTableData	Configuration			Č.,

 Browse on the Add Module dialog for Libraries\Documents\LUSAS180\Modules\DeleteLoadcases.dll and click OK

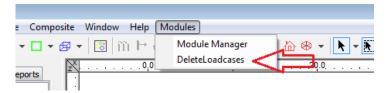
Add Module					23
Look in:	Modules		•	G 🤌 📂 🛄 -	
Recent Places	Name	ases.dll		Date modified 26/06/2019 14:56	Type Applicatic
Desktop					
Libraries					
Computer					
Network	File name:				ок
	Files of type:	Library Files		<b>•</b>	Cancel

The LUSAS Module Configuration Editor dialog will appear. This permits (amongst many other features) restricting the Module to only work with a particular LUSAS version or licence key. Change the **assembly path** to match the build output path in the user directory by typing in **%ModulesUserDir%**.

lodule assembly		Module controller options	
Assembly path	%ModulesUserDir¦%	Load with Modeller (required for event subscription)	
Assembly filename	DeleteLoadCases.dll	Use encryption to protect code	
nstance class	Lusas.DeleteLoadCases.DeleteLoad	Allow advanced debugging of runtime errors	
COM class CLSID		Report verbose/debug information in text window	
		Enable module	
Module manager identi	ty	Use IClassFactory support for unmanaged code	
riendly name	Delete Load Cases	Load as library (without ILusasModule interface)	
Short description	Delete Load Cases - LUSAS Exten:	Disable forward/backward interop compatability	
Required licences	Verified interop versions	Attempt to load regardless of interop compatability	
AdvGeometry	<ul> <li>15.0</li> </ul>	Search assembly path for missing references	
Analyst Autol oader	15.1	Module configuration	
Bridge	=	Module identifier DeleteLoadCases	
BridgeOrCivil CADToolkit			
Civil Composites		Company name DeleteLoadCases.AutoConfig	
FastSolver		Configuration version 1.0.0.0	
Geotechnical Hydration			
IMDPlus	-	Save Cancel	

• Press the **Save** button and then **Close** 

From now on, you will see the Delete Loadcases item in the modules menu.



• Choose the **DeleteLoadcases** menu item. A blank window is displayed, because you have not yet added any controls or written any code.

ù ⊢ i	dules Module Manager DeleteLoadcases	<b>B</b>
	DeleteLoadcasesDialog	

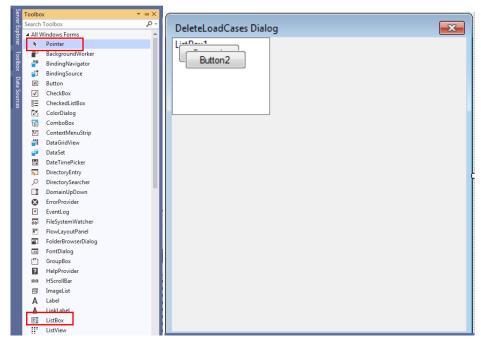
• Click on **Stop Debugging** in Visual Studio to close LUSAS Modeller and amend the project.

indows Desktop	
TEST WINDOW	HELP
Mixed Platforms +	B → G G G K → G G K → G G K → G G K → G G K → G G K → G G K → G G K → G G K → G K

#### **Adding dialog controls**

• In Visual Studio double-click on **DeleteLoadcasesDialog.vb** to open a blank dialog.

	oolbox 🔹 🖙 🗙				Solution Explorer -
er Explor	earch Toolbox 🖉 🗸				0 0 🕁 🗄 + 10 + 5 🖒 🖉 🕼 💠 🌶
plor	> All Windows Forms			Search Error List	Search Solution Evolorer (Ctrl+1)
		1 Warning 🛛 🚺 0 of 1 Message 🛛 🏋 🛛 Build			a. Solution 'DeleteLoadCases' (1 of 1 project)
	Containers	Project	File	Line Suppression St 🔻	▲ VE DeleteLoadCases
N N	Menus & Toolbars				🔎 My Project
	Data Components				References
	Printing				DeleteLoadCasesDialog.vb
	Dialogs				VB DeleteLoadCasesModule.vb
	WPF Interoperability				
	4 General				
					Solution Explorer Team Explorer
	There are no usable controls in this group. Drag an				Properties -
	item onto this text to add it to the toolbox.				DeleteLoadCasesDialog Lusas.Module.LusasModule



• Click on Toolbox, all Windows Forms and add a ListBox and two Buttons by double clicking them

• Select each of the controls to modify its properties (Name, location, size...)

			Solution Explorer	
Delete loadcase Dialog				0000 20
			Search Solution Explore	r (Ctrl+;)
Q	Do		Solution 'DeleteLo	
ListBox1			▲ MB DeleteLoadcas	
			<ul> <li>My Project</li> <li>DeleteLoad</li> </ul>	
				casesDialog.vb casesModule.vb
			V VB DeleteLoad	casesiviodule.vb
0			Properties	
			ListBox1 System.Wind	lows.Forms.ListBox
			≣ ∰↓ 🖓 <b>F</b> 🖉	
		L	E Design (Name)	ListBox1
			GenerateMember	True
			Locked	False
0	Ö		Modifiers	Friend
Delete Cano	el		E Focus	Thenu
Contro		, i i i i i i i i i i i i i i i i i i i	CausesValidation	True
				True
		- 1		
		- E	∃ Layout	Ten Left
		(	Anchor	Top, Left
DefaultErrorProvider	Source		Anchor Dock	None
DefaultErrorProvider	Source	ε	Anchor	

• Click on Button1in the dialog and define its Name to be **btnDelete** with Text: **Delete** 

P	roperties		···· <b></b>
ł	tnDelete System.Wind	lows.Forms.Buttor	n +
2	🗄 💱 💭 🗲 🏓		
	Tag		
E	Design		
	(Name)	btnDelete	
	GenerateMember	True	
	Locked	False	
	Modifiers	Friend	
E	Focus		
	CausesValidation	True	
E	Layout		
	Anchor	Top, Left	-

- For Button 2 define its Name to be **btnCancel** with Text: **Cancel**
- For the ListBox define the Selection Mode to be **MultiExtended** (This will allow selection of multiple items in the listbox).

Properties and and and a		• ¶ ×
ListBox1 System.Winde	ows.Forms.ListBox	*
🔡 💱 🔎 🗲 🎤		
IntegralHeight	True	-
ItemHeight	13	
MultiColumn	False	
ScrollAlwaysVisible	False	
SelectionMode	MultiExtended	
Sorted	False	
TabIndex	3	
TabStop	True	
UseTabStops	True	
Visible	True	

#### **Defining a ListBox**

All the loadsets (Loadcases, Combinations and Envelopes) are to be listed in the ListBox

• Double-click on the form (i.e the dialog itself) to create an event handler for the form load event

Modify the code as shown below:



**Tip.** Open the PDF file for this guide and copy and paste the code. Take care to ensure that any unwanted line breaks are removed.

```
'Add loadsets to the listbox
Dim LoadsetsArray =
moduleObject.Modeller.db.getLoadsets("All", "All")
For i = 0 To UBound(LoadsetsArray)
ListBox1.Items.Add(LoadsetsArray(i).getName())
Next
End Sub
```

#### **Defining a Delete Button**

The selected loadsets of the list box need to be deleted when pressing this button, so:

- Double-click on the button **Delete** to create an event handler for this button's click event.
- Modify it by typing the following:



**Tip.** Rather than type the lines of VB required, open the PDF file for this guide and for the remainder of this section of the Guide copy and paste the code where needed. Take care to ensure that any unwanted line breaks are removed.

```
Private Sub btnDelete_Click(sender As Object, e As EventArgs)
Handles btnDelete.Click
'Delete loadsets that are selected in the listbox
For Each Item In ListBox1.SelectedItems
Call moduleObject.Modeller.database.deleteLoadset(Item)
Next
Call PopulateListBox()
End Sub
```

#### **Defining a Cancel Button**

To cause the dialog to close:

• Double-click on the button **Cancel** to create an event handler for this buttons click even and modify it to:

```
Private Sub btnCancel_Click(sender As Object, e As EventArgs)
Handles btnCancel.Click
Close()
End Sub
```

- Now build the project by pressing the **F7** key and run it by pressing the **F5** key.
- In LUSAS Modeller select the menu item **Modules> DeleteLoadcases**, then select which loadcase to delete and press the **Delete** button.

	s Analyses Utilities Bridge Composite	
) 🖨 🖬 🖆 🐂 📾 🕷 🗙 🗅	· · · / · · · · · · · · · · · · · · · ·	+ 🐻 III I→ Module Manage
		Dulutul and an
Delete Loadcases.md     Delete Loadcases.md	DeleteLoadcase: Dialog	
Analysis 1		
Geometric		
(B) (B) Material		
1:Loadcase 1	Loadcase 1 Loadcase 1	^
(i) 2:Loadcase 2	Loadcase 2	=
(i)-(i) 3:Loadcase 3	Loadcase 3	E.
E-(E) 4:Loadcase 4	Loadcase 4	
H S:Loadcase 5	Loadcase 5	
ap . And	Loadcase 6	
⊕-(È) 6:Loadcase 6	Loadcase 7 Loadcase 8	
E-C 7:Loadcase 7	Loadcase 9	
B-C 8:Loadcase 8	Loadcase 10	
⊕	Loadcase 11	
(E) 10:Loadcase 10	Loadcase 12	
Delta 11:Loadcase 11	Loadcase 13	
(b) 12:Loadcase 12	Loadcase 14 Loadcase 15	
⊕-(E) 13:Loadcase 13	Loadcase 15	
⊕- ( 14:Loadcase 14	Loadcase 17	
De 15:Loadcase 15	Loadcase 18	
E-C 16:Loadcase 16	Loadcase 19	
17:Loadcase 17		
18:Loadcase 18	Delete	Cancel
19:Loadcase 19	Delete	Cancer
⊕ (E) 20:Loadcase 20		
E-(E) 21:1 oadcase 21		

You may have noticed that if you try to delete all loadcases you get the following:



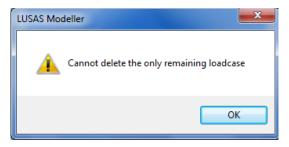
This is because Modeller has raised an exception as there must always be at least one loadcase. You can handle or catch the exception so you get a meaningful message saying "Cannot delete the only remaining loadcase"

#### **Handling errors**

The code will be modified to handle this situation. Add the following code at the beginning of the **btnDelete\_Click** function.

```
Private Sub btnDelete_Click(sender As Object, e As EventArgs)
Handles btnDelete.Click
Dim loadcaseArray =
moduleObject.Modeller.database.getLoadsets("Loadcase", "All")
Try
    'Delete loadsets that are selected in the listbox
    For Each Item In ListBox1.SelectedItems
        Call moduleObject.Modeller.database.deleteLoadset(Item)
        Next
Catch ex As Exception
        Call moduleObject.Modeller.AfxMsgBox(ex.Message())
End Try
    Call PopulateListBox()
End Sub
```

Now, if you try to delete all loadcases you get the following 'cleaner' message:



#### **General considerations**

#### **Basic dialog design**

- Use the ToolBox to create controls
- Double-click on the control in the toolbox to create the control at standard size
- Drag and drop to desired position using grid lines to line up with other controls
- Name the controls in Properties using standard naming convention (See basic dialog controls prefixes below e.g. txt, btn, opt, chk)
- Set FormBorderStyle = FixedDialog
- Set Localizable = True

#### **Basic dialog controls**

- TextBox (**txt**) string of input or output
- Button (**btn**) activate an event
- ComboBox (cbo) choice of preset input
- CheckBox (**chk**) true or false
- RadioButton (opt) choice of a number of options
- NumericUpDown (**spn**) Integer or decimal within specified range
- PictureBox (img) display images on dialog
- GroupBox (**grp**) groups radio buttons
- Label (**lbl**) add text to dialog
- Panel (**pnl**)- invisible group, enable/disable multiple controls

#### **Code design considerations**

- Only code relating to the dialog should be contained in the dialog class <projectname>Dialog.vb - e.g. Events, data check functions, change label text etc. (Access to the module code is achieved by using moduleObject.<function>)
- Place all worker code in module class <projectnameModule.vb> (access to Modeller LPI functions is achieved using Modeller.<LPIfunction>")

- All variables must be allocated a Type. Modeller has a number of predefined data types e.g. IFPoint, IFLine. All Modeller data types start with IF and a full list is automatically displayed in Visual Studio when the type is being declared.
- Use access modifiers to restrict the scope of functions as much as possible. Only make the function public if it is required outside the module.
- Private only available to routines in this module
- Protected only available within class and derived classes
- Friend only available within the assembly (Dialog to module)
- Public visible globally and outside of the assembly
- It is a good idea to comment all functions, classes, modules etc using the standard XML comment blocks. Note: Typing three quotes (''') on the line immediately above the function name will automatically present the standard html template with the parameters included.

#### **Multiple Dialogs in a single module**

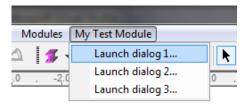
By default the LUSAS module template is set up to handle a single dialog. It is possible for a module to provide multiple dialogs and provide multiple menu entries. It is good practice to keep all related functionality in a single module.

When a module creates a new menu item, Modeller will return a unique id for that menu item, these id's should be stored in "member" variables within the module. Modules can create new menu items in the onRefreshMainMenu function as shown below:

```
Private m_dialog1_ID As Integer
Private m_dialog2_ID As Integer
Private m_dialog3_ID As Integer
''' <summary>
''' Called when Modeller is redrawing the Main Menu.
''' </summary>
''' <remarks>
''' Allows custom modules to append and maintain their own menus.
''' </remarks>
Protected Overrides Sub onRefreshMainMenu()
    Dim myModuleMenu As IFMenu = rootMenu.appendMenu("My Test Module")
    m_dialog1_ID = myModuleMenu.appendItem("Launch dialog 1...",
"textwin.writeLine(""Test Dialog 1"")")
    m_dialog2_ID = myModuleMenu.appendItem("Launch dialog 2...",
```

```
"textwin.writeLine(""Test Dialog 2"")")
    m_dialog3_ID = myModuleMenu.appendItem("Launch dialog 3...",
"textwin.writeLine(""Test Dialog 3"")")
End Sub
```

When the user clicks these menu items all modules will be called with the id of the menu item. It is the responsibility of the module to listen for any menu ids it creates and respond accordingly.



#### Notes:

- The modules are called via the **onMenuClick** function
- The module should respond when it is called with the correct menu ID.
- For each menu id the moduleDialog should be set to a new instance of the correct dialog before the runModule function is called. This way the correct dialog will be shown, as below.

```
''' <summary>
''' Called when the user clicks on a menu entry.
''' </summary>
''' <param name="menuID">ID of the menu that has been clicked.</param>
''' <param name="edittingObj">Object that is being edited (nothing when
creating a new object).</param>
''' <param name="clientData">Data that was provided to Modeller when
defining edittingObj.</param>
''' <returns>true if the click event was handled by this
Module.</returns>
''' <remarks>
''' LUSAS expects the a Module handling the event to execute itself
(typically using runModule()).
''' </remarks>
Function onMenuClick (ByVal menuID As Integer, ByVal edittingObj As
Object, Optional ByVal clientData As Object = Nothing) As Boolean
    If (m dialog1 ID = menuID) Then
```

```
moduleDialog = New myDialog1(Me)
        runModule()
        Return True
    End If
    If (m dialog2 ID = menuID) Then
        moduleDialog = New myDialog2(Me)
        runModule()
        Return True
   End If
    If (m dialog3 ID = menuID) Then
       moduleDialog = New myDialog3(Me)
        runModule()
        Return True
    End If
    Return False
End Function
```

#### • Menu items maybe enabled or disabled in the **onMenuUpdate** event

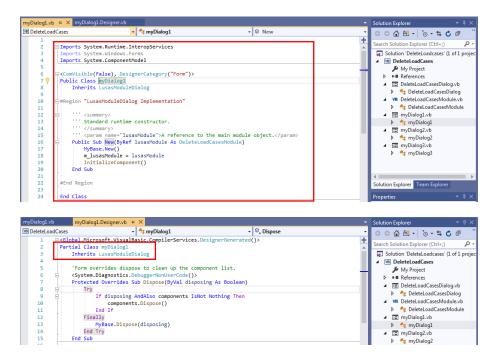
```
''' <summarv>
''' Called when a menu entry needs to be drawn.
''' Allows the Module to specify whether the menu item should be
disabled or checked.
''' </summary>
''' <param name="menuID">ID of the menu that has been clicked.</param>
''' <param name="edittingObj">Object that is being edited (nothing when
creating a new object).</param>
''' <param name="enable">Set to true to enable the menu item.</param>
''' <param name="checked">
''' Set to 0 to show an 'off' tickbox next to the menu.
''' Set to 1 to show an 'on' tick mark by the side of the menu.
''' Set to 2 to show an indeterminate check.
''' Set to 3 to show no tick at all.
''' </param>
''' <param name="clientData">Data that was provided to Modeller when
defining edittingObj.</param>
''' <returns>true if the update event was handled by this
Module.</returns>
''' <remarks>
''' Only when a Module handles an menu update event are the changed
values of enable/checked respected.
''' </remarks>
Function onMenuUpdate (ByVal menuID As Integer, ByVal edittingObj As
Object, ByRef enable As Boolean, ByRef checked As Integer, Optional
ByVal clientData As Object = Nothing) As Boolean
    If (m dialog1 ID = menuID) Then
```

```
enable = (Modeller.db.countSurfaces() > 0)
    Return True
End If
If (m_dialog2_ID = menuID) Then
    enable = True
    Return True
End If
Return False
End Function
```



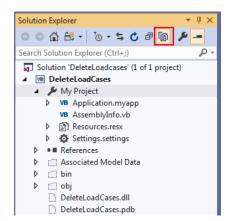
•

**Note.** All dialogs <u>must</u> inherit from LusasModuleDialog. When adding a new dialog you should change the code in the dialog designer to inherit from LusasModuleDialog rather than System.Windows.Forms.Form



#### **Translation considerations**

• The default language should always be **English**. All strings should be defined in the Resources.resx file. To access the Resources.resx file pick the **Show All Files** button in the Solution Explorer.



Double clicking on the Resources.resx will display a window to name and define the strings.

R	esourc	es.resx 🕫 🗙			-
Į	sbe Strin	ngs 👻 🏠 Add Resour	rce 👻 Remove Resource	🖅 👻 Access Modifier: Friend	•
		Name	🔺 Value		Comment
	**	String1			



**Note.** If languages other than English are to be supported the dialog property **Localizable** property should be set to be **True and the Language** should be changed to the translation language, as for example for Chinese (Simplified) This will automatically create a new resource file for the dialog where the translated string should be defined and allow the labels to be translated and the size and position of the controls to be customised. Changing the **Language** back to **Default** will display the English labels with the controls set to in their original size and position.

9		
∃ Design		DeleteLoadCases.xml
(Name)	DeleteLoadCasesDialog	DeleteLoadCasesDialog.vb
Language	Chinese (Simplified)	DeleteLoadCasesDialog.Designer.vb
Localizable	True	DeleteLoadCasesDialog.resx
Locked	False	DeleteLoadCasesDialog.zh.resx
∃ Focus		Magnetic DeleteLoadCasesDialog
CausesValidation	True	VB DeleteLoadCasesModule.vb

By using this approach any strings which do not have a translation will be displayed in English and an Language for which translation is not supported will show English labels and strings.

#### **VB.NET** online tutorials

VB.NET online tutorials are widely available. Here are some examples:

English:

https://www.youtube.com/watch?v=hkcO\_M9gcNw&index=1&list=PL42055376AE25 291E

English:

https://www.youtube.com/watch?v=AJpTbPasJqI&list=PLS1QulWo1RIYLpgVN\_Cp XbkOQoYJTItzg

Chinese: <u>https://channel9.msdn.com/Series/Visual-Basic-Fundamentals-for-Absolute-Beginners/01</u>

#### **VB.NET dialog exercise**

Colour Line		X
Colour Index	0	
ОК	Cancel	Apply

The preceding dialog is required to allow a user to change the colour of all Lines in a model. The dialog should be activated from the menu item **My Menu> Colour Line** 

Write the code to enable this to take place.

The solution is shown on the next page.

#### **VB.NET** dialog solution

1. In Dialog Class:

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnOK.Click
    Call btnApply_Click(sender, e)
    Call btnCancel_Click(sender, e)
    End Sub
    Private Sub btnApply_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnApply.Click
    moduleObject.ColourLines(spnColourIndex.Value)
    End Sub
    Private Sub btnCancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCancel.Click
    Me.Close()
    End Sub
```

2. In Module Class, in existing function onRefreshMainMenu change:

```
If rootMenu.exists("Modules") Then
    modMenu = rootMenu.getSubMenu("Modules")
Else
    modMenu = rootMenu.appendMenu("Modules")
End If
to:
```

```
Dim menuName As String = "My Menu"
If rootMenu.exists(menuName) Then
    modMenu = rootMenu.getSubMenu(menuName)
Else
    modMenu = rootMenu.appendMenu(menuName)
End If
```

#### 3. Add function

```
''' <summary>
''' Routine to colour lines
''' </summary>
''' <param name="colour">colour index</param>
''' <param name="colour">colour
''' <param name="colour">colour index</param>
''' <param name="colour">colour
''' Public Sub ColourLines(ByVal colour)
Dim lines As Object = Modeller.database.getObjects("Lines",
"All")
For Each line As IFLine In lines
line.setPen(colour)
Next
End Sub
```

# **LUSAS via COM**

# **Component Technology**

The LUSAS Programmable Interface allows interfacing with other compatible Windows programs through a Component Object Model (COM) interface. This defines a set of rules by which two programs can communicate and allows controlling those programs as if they were part of LUSAS Modeller. LUSAS can also be used as a component of another system (running transparently if required) providing modelling capabilities, analysis solutions and results viewing and processing options for that application.

In order to drive LUSAS from a standalone application via COM (Component Object Model), LUSAS must be installed and licenced. When creating a COM instance of LUSAS a licence will be used. The licence will be in use for the lifetime of the instance and must be properly disposed of to release the licence.

## **Application Example**

To illustrate the process involved, a stand-alone application called SimpleBeam will be created. The application will accept two parameters, length and load. The application will use LUSAS to analyse the beam and return the results for the maximum bending moment.

#### Create a new project

• In Visual Studio create a new Windows Form App (.NET Framework) in Visual Basic called SimpleBeam.

Create a new project	Search for project templates P - Language - Platform - Project type -
Recent project templates	Filtering by: Visual Basic Clear filter
LUSAS Module 18.0         Visual Basic           WPF App (.NET Framework)         Visual Basic           WPF App (.NET Framework)         Ce           Blank Solution         Ce	Console App (NET Framework) A project for creating a command-line application Visual Basic Windows Console WPF App (NET Framework) Windows Presentation Foundation client application Visual Basic Windows Desktop
	Class Library (NET Standard) A project for creating a class library that targets .NET Standard. Visual Basic Android IOS Linux macOS Windows Library
	Windows Forms App (,NET Framework) A project for creating an application with a Windows Forms (WinForms) user interface Vsual Basic Windows Desktop
	Class Library (NET Framework) A project for creating a VB class library (dll) Visual Basic Windows Library
	Unit Test Project (NET Framework) A project that contains unit tests.
Configure your new p	roject
Windows Forms App (.NET Framework)	Visual Basic Windows Desktop
Project name	
SimpleBeam	
Location	
C:\Users\matth\source\repos	•
Solution	
Create new solution	•
Solution name ()	
SimpleBeam	
Place solution and project in the same directory	
Framework	
.NET Framework 4.7.2	-
	Back Create

- Add a reference to LUSAS Modeller.
- In Solution Explorer, click Show All Files
- Right-click on **References** and select **Add reference**
- In the COM tab select LUSAS Modeller ActiveX Script Language 16.0 Version 18.0

(Note that later versions of LUSAS will require a different selection)

- Solution Explorer Ψ× G O 🟠 🛗 - 🔞 - 与 🖒 🖓 Search Solution Explorer (Ctrl+;) ۰ م Solution 'SimpleBeam' (1 of 1 project) ▲ VB SimpleBeam 👂 🏓 My Project ▲ ■■ References Analyzers LusasM18\_0 ■■ System ■■ System.Core ■■ System.Data System.Data.DataSetExtensions ■■ System.Deployment
- Create the following dialog using the Toolbox as shown in the previous example:

Form1		
Beam length	10.0	
Uniform loading	5.0	
Maximum Bending =		
	Quit Calculate	

For simplicity all code is placed in the dialog as follows, by double clicking the form window.

```
Imports LusasM18_0
Public Class Form1
Private m_lusas As LusasM18_0.LusasWinApp' Reference to Lusas
Modeller
Private Sub btnCalculate_Click(sender As System.Object, e As
System.EventArgs) Handles btnCalculate.Click
analyseBeam()
End Sub
```

```
Private Sub analyseBeam()
    ' Get the params
   Dim length As Double = Double.Parse(txtLength.Text)
   Dim loading As Double = Double.Parse(txtLoading.Text)
    ' Create an instance of modeller
   m lusas = New LusasM18 0.LusasWinApp
    ' Create a new model
   m lusas.newDatabase()
    ' Set the vertical axis
   m lusas.db.setLogicalUpAxis("Z")
    ' Set the unit system
   m lusas.db.setModelUnits("kN,m,t,s,C")
    ' *** Create a line ***
    ' Get the geometry data object
    Dim geomData As IFGeometryData = m lusas.geometryData()
    ' Set the defaults
    geomData.setAllDefaults()
    ' Set the coordinates of the first point
    geomData.addCoords(0, 0, 0)
    ' Set the coordinates of the second point
   geomData.addCoords(length, 0, 0)
    ' Create the line object
    Dim linesDBop As IFObjectSet = m lusas.db.createLine(geomData)
    ' Get the lines
    Dim lines()As Object = linesDBop.getObjects("Lines", "All")
    ' Get a reference to the created line
```

```
Dim beamLine As IFLine = lines(0)
        ' *** Create a mesh attribute ***
        Dim meshAttr As IFMeshLine = m lusas.db.createMeshLine("Beam
Mesh")
        ' Set the element type and number of elements (1m elements here)
        meshAttr.setNumber("BMS3", length)
        ' *** Create a geometric attribute ***
        Dim geomAttr As IFGeometricLine =
m lusas.db.createGeometricLine("Beam Geometry")
        ' Set the element type
        geomAttr.setValue("elementType", "3D Thick Beam")
        ' Set the beam properties
        geomAttr.setBeam(0.0125, 0.0004573, 0.00002347, 0.0, 0.00000121,
0.00532608, 0.00755776, 0.0, 0.0, 0)
        ' *** Create a material attribute ***
        Dim materialAttr As IFMaterialIsotropic =
m lusas.db.createIsotropicMaterial("Steel", 209000000.0, 0.3, 7.8)
        ' *** Create a support attribute (fixed) ***
        Dim fixedSupport As IFSupportStructural =
m lusas.db.createSupportStructural("Fixed")
        ' set the freedoms
        fixedSupport.setStructural("R", "R", "R", "F", "F", "F", "F",
"F", "F")
        ' *** Create a support attribute (pinned) ***
        Dim pinnedSupport As IFSupportStructural =
m lusas.db.createSupportStructural("Pinned")
        ' set the freedoms
```

```
pinnedSupport.setStructural("F", "R", "R", "F", "F", "F", "F",
"F", "F")
        ' *** Create a load attribute ***
        Dim loadAttr As IFLoadingGlobalDistributed =
m lusas.db.createLoadingGlobalDistributed("UDL")
        ' Set the parameters
        loadAttr.setGlobalDistributed("Length", 0.0, 0.0, -loading, 0.0,
0.0, 0.0, 0.0, 0.0)
        ' *** Assign the attributes to the geometry ***
        ' get the assignment object
        Dim assignment As IFAssignment = m lusas.assignment()
        ' set the defaults
        assignment.setAllDefaults()
        ' Assign the mesh
        meshAttr.assignTo(beamLine, assignment)
        ' Assign the geometry
        geomAttr.assignTo(beamLine, assignment)
        ' Assign the material
        materialAttr.assignTo(beamLine, assignment)
        ' Assign the loading
        loadAttr.assignTo(beamLine, assignment)
        ' Assign the supports to the points of the line
        ' get the points - Lower Order Features
        Dim pointsArray()As Object = beamLine.getLOFs()
        ' Assign the fixed support to the first point
        fixedSupport.assignTo(pointsArray(0), assignment)
        ' Assign the pinned support to the last point
        pinnedSupport.assignTo(pointsArray(1), assignment)
        ' Set the mesh
```

```
m lusas.db.updateMesh()
        ' The model is ready to be solved - get the temporary file path
        Dim tempFilePath As String = System.IO.Path.GetTempPath()
        ' Get the solverOptions object and set the defaults
        Dim solverOptions As IFLusasRunOptionsObj =
m lusas.solverOptions()
        solverOptions.setAllDefaults()
        ' Get the exporter object that will export the model to solver
        Dim solverExport As IFTabulateDataObj = m lusas.solverExport()
        ' Set the defaults
        solverExport.setAllDefaults()
        ' Set a filename
        solverExport.setFilename(tempFilePath &
"beam.dat").setSearchAreaFileOn()
        ' Export the model as a solver data file (.dat)
        Dim returnCode As Integer = m lusas.db.exportSolver(solverExport,
solverOptions)
        If (returnCode <> 0) Then
            ' If the export fails we cannot run the analysis
            MsgBox("The tabulation failed")
            Return
        End If
        ' Save the model before solving
        m lusas.db.saveAs(tempFilePath & "beam.mdl")
        ' If we sucessfully exported the .dat file we can run the
analysis
        returnCode = m lusas.solve(tempFilePath & "beam.dat",
solverOptions)
        If (returnCode <> 0) Then
            ' If the analysis fails we cannot access the results
```

```
MsgBox("The analysis failed")
            Return
        Else
            ' if the analysis is successful load the results
            m lusas.database.openResults(tempFilePath & "beam.mys")
            m lusas.database.getResultsCache().calculateNow()
        End If
        ' *** Successful analysis - Process the results to determine the
max bending ***
        Dim maxMom As Double
        Dim nodeNum As Integer
        ' Get the results at each node to determine the max
        For Each element As IFElement In beamLine.getElements()
            For Each node As IFNode In element.getNodes()
                ' Extract the nodal result for the required Entity and
Component
                Dim my As Double = node.getResults("Force/Moment - Thick
3D Beam", "My")
                ' Save the minimum (sagging) moment
                If my < maxMom Then
                    maxMom = my
                    nodeNum = node.getID()
                End If
            Next
        Next
        ' Get the units of the current model for display
        Dim forceUnit As String =
m lusas.db.getModelUnits().getForceShortName()
        Dim lengthUnit As String =
```

```
m lusas.db.getModelUnits().getLengthShortName()
        ' Set the dialog label
        lblMaxMom.Text = m lusas.convertToString(maxMom) & forceUnit &
lengthUnit
        ' Quit the application and free the licence
        m lusas.quit()
    End Sub
    Private Sub btnClose Click(sender As System.Object, e As
System.EventArgs) Handles btnQuit.Click
        For Each p As Process In
System.Diagnostics.Process.GetProcessesByName("Lusas m")
            Try
                p.Kill()
                p.WaitForExit()
            Catch ex As Exception
            End Try
        Next
        Me.Close()
    End Sub
End Class
```

#### Interfacing to LUSAS using C++

Generally, LUSAS recommends that you use VB or any other language that natively supports COM interfaces. C++ does not natively support COM interfaces, thus COM programming in C++ is much more complex, and results in code which is more likely to contain bugs and is harder to read. However, it is possible for experienced C++ programmers to interface to Modeller. A simple example follows:.

```
#import "C:\LUSAS152\programs\Lusas_m.exe"
// create a modeller
    pModeller = IFModellerPtr("LUSAS.Modeller.18.0");
// create and return a database
    IFDatabasePtr db = pModeller->newDatabase();
// create and return a line
    IFLinePtr 1 = db->createLineByCoordinates(0, 0, 0, 5, 5, 5);
// calculate line length
    double len = l->getLineLength();
```



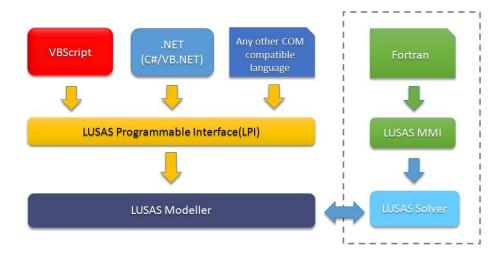
**Note.** The LPI functions often return a base class pointer which often needs to be downcast to the desired type (e.g. attribute -> material). VB will do this for you, but C++ will not. Therefore you must explicitly cast, and catch any exceptions that may result



**Note.** LPI functions often have VARIANT inputs and outputs. VB will handle conversion between simple data types (integers, strings, objects) and VARIANTs, but C++ will not. Therefore you must be familiar with the use of the VARIANT type. If in doubt, consult Microsoft documentation.

## **LUSAS Material Model Interface**

In addition to the accessing and customising LUSAS Modeller via the LUSAS Programmable Interface, user-defined material models (written in Fortran) can be compiled and built into a customised LUSAS Solver executable by using the LUSAS Material Model Interface (LUSAS MMI).



The use of LUSAS MMI is beyond the scope of this manual. Please contact LUSAS Technical Support for more information.

# LUSAS

TITI

LUSAS, Forge House, 66 High Street, Kingston upon Thames, Surrey, KT1 1HN, UK Tel: +44 (0)20 8541 1999 | Fax: +44 (0)20 8549 9399 | info@lusas.com | www.lusas.com