Bentley®

Automation Manual

Maxsurf Modeler 2025
MOSES Modeler 2025
24 September 2025

License & Copyright

Table of Contents

Nomenclature

Array	An ordered collection of items of a particular data type.
Automation	(Formerly OLE Automation) is a feature of the Component Object Model (COM), an industry-standard technology that applications use to expose their objects to development tools, macro languages, and other applications that support Automation
Binding	Process of creating a link between two programs, such as linking Maxsurf Modeler to Excel.
Boolean	Data type, contains a True or False variable
Collection	A group containing all of a type of items in the design, such as all the markers
COM	Component Object Model, a common interface used to communicate between applications.
Decimal	Data Type, Largest available numeric data type, can hold a positive or negative number up to 29 digits.
Double	Data Type, a single precision floating point value between 4.094E-324 and 1.797E308 and the opposite in negatives.
Integer	Data type containing a whole number between –32768 and 32767
List	A specified group of one or more members of a data type, such as several surfaces in a design. A list is a subset of a collection.
Long	Data type containing a whole number between –2,147,483,648 and 2,147,483,647
Macro	Code written in VBA to perform an action in an application such as Microsoft Excel
Method	Part of the Object hierarchy, methods are actions to do something in the design.
Object	Part of automation interface hierarchy, containing Properties and Methods, or other objects
Procedure	A procedure tells the application how to perform a specific task
Property	Part of the Object hierarchy, properties contain information regarding the design
Script	Another name for a procedure
Single	Data type, a single precision floating point value between 1.401298E-45 and 3.402823E38 and the opposite in negatives.
String	Data Type, containing textual information comprising of any ASCII characters
Variant	Data type, Containing any other data type including arrays
VBA	Visual Basic for Applications. The programming language used in this manual.

About this Manual

This manual describes the Maxsurf Modeler automation interface which lets you write macros, scripts and programs to access data from Maxsurf Modeler or to create and analyse models in Maxsurf Modeler. The manual provides a description of the objects, properties and methods contained in the Maxsurf Modeler Automation Interface. VBA code examples have been used throughout the manual to demonstrate how best to use these objects.

Chapter 1 Introduction; This chapter provides an introduction to automation and many of the general concepts needed to get started using VBA.

Chapter 2 Getting Started; Gets you started writing a simple script to automate Maxsurf Modeler. It is a simple series of step-by-step tutorials designed to introduce you to most of the objects within the Maxsurf Modeler Automation model.

Chapter 3 Basic Maxsurf Modeler Automation; Details some of the basic operations for automating Maxsurf Modeler, like opening and saving designs. It describes the main objects of the Maxsurf Modeler Automation Model and provides code examples of their use.

Chapter 4 Advanced Maxsurf Modeler Automation; This chapter outlines the use of collections and lists to manage groups of objects. Information included in this chapter is aimed at increasing your understanding of collections and lists and detailing how to use lists to increase the speed of execution. This chapter can be omitted without missing any Maxsurf Modeler Automation functionality.

Chapter 5 User Interface; This chapter describes which items in the Maxsurf Modeler user interface can be controlled using Automation.

Chapter 6: Automation macros in Microstation.

Chapter 7 Examples; A number of examples are presented, demonstrating the use of Microsoft Excel to control aspects of the Maxsurf Modeler Automation Interface.

Note

The aim of this manual is to provide information regarding the use of Maxsurf Modeler Automation and the Maxsurf Modeler Object Library. It is *not* the aim of this manual to teach programming in VBA or the use of Maxsurf Modeler.
The context of this manual assumes an in depth knowledge of Maxsurf Modeler and some experience in programming VBA. The Further Reading section on page 15 lists some resources on VBA and Maxsurf Modeler that may be useful in conjunction with this manual.

Chapter 1 Introduction

This chapter provides an introduction to automation, VBA and object models and their use in writing macros for Maxsurf Modeler. It discusses the applications that can be used to access and control Maxsurf Modeler and presents examples of how this can be done. Aspects of Maxsurf Modeler that can and cannot be automated are also discussed in this chapter.

This chapter is divided up into two sections:

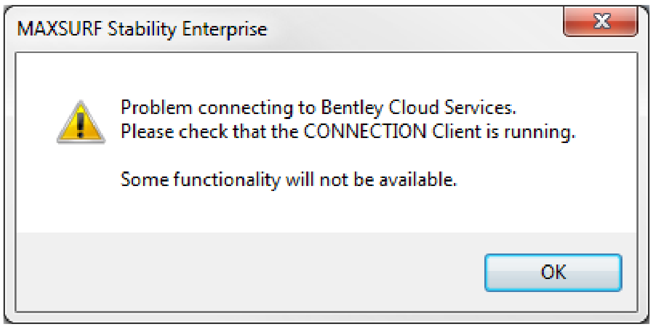
- Automation
- The Maxsurf Modeler Object Model

Disabling dialogs during startup

Dialogs shown when the application starts up can cause problems with COM. These dialogs can be suppressed as described below.

Disable CONNECTION Client warning dialog

When running an application through the COM automation interface, it is desirable to suppress dialogs which require user intervention. When the CONNECTION Client is not running the application will normally show you a warning dialog on startup:

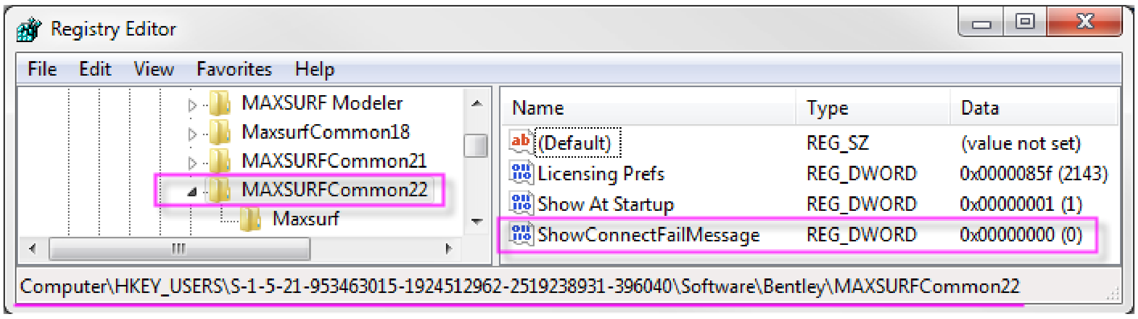


Problem connecting to Bentley Cloud Services.
Please check that the CONNECTION Client is running.

Some functionality will not be available.

CONNECTION Client warning dialog

To turn off this dialog change the “ShowConnectFailMessage” registry entry to 0. Warning: do not edit the Registry unless you are confident about doing so.

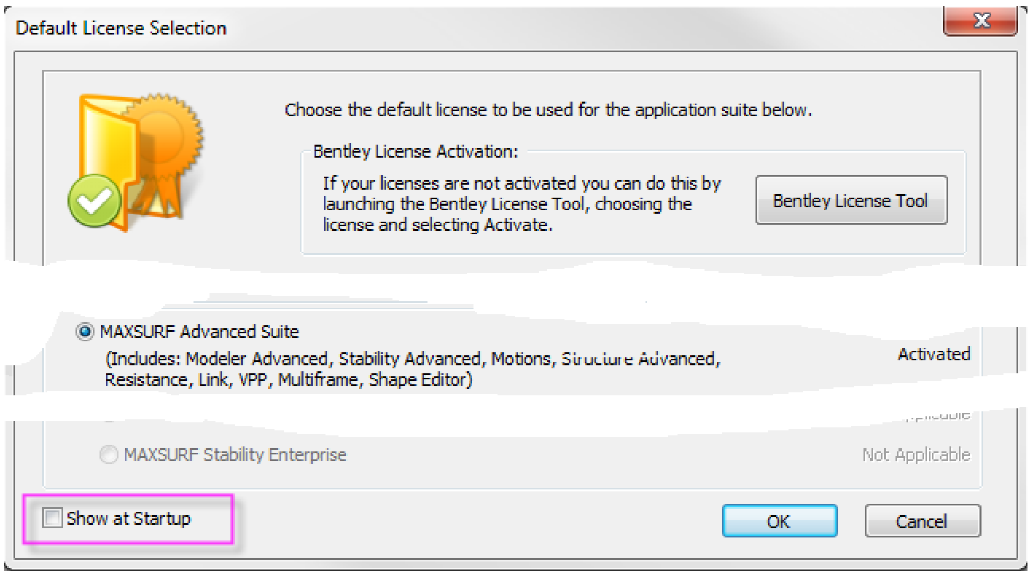


Name	Type	Data
(Default)	REG_SZ	(value not set)
Licensing Prefs	REG_DWORD	0x0000085f (2143)
Show At Startup	REG_DWORD	0x00000001 (1)
ShowConnectFailMessage	REG_DWORD	0x00000000 (0)

Computer\HKEY_USERS\S-1-5-21-953463015-1924512962-2519238931-396040\Software\Bentley\MAXSURFCommon22

Turn off CONNECTION Client warning dialog using the Registry Editor

It is also advisable when running via COM to have pre-selected the desired license and then *not* to show the license dialog at startup:


Default License Selection

Choose the default license to be used for the application suite below.

Bentley License Activation:

If your licenses are not activated you can do this by launching the Bentley License Tool, choosing the license and selecting Activate.

(● MAXSURF Advanced Suite) Activated
(Includes: Modeler Advanced, Stability Advanced, Motions, Structure Advanced, Resistance, Link, VPP, Multiframe, Shape Editor)

○ MAXSURF Stability Enterprise — Not Applicable

☐ Show at Startup

It is recommended to disable "Show at Startup" when running via COM

RSS News feed

The RSS news feed dialog should also be tunred off at startup. This can be done through the Help | News dialog.

Automation

Automation is a term used to describe the ability of one application to control or access data from another. Automation is a common feature in many Microsoft applications such as Word and Excel. In fact, the macros in each of these applications are written using automation. The automation interface in these applications gives the user access to a range of *objects* that can be used to control the application and its data. For example, Microsoft Word contains paragraph, word and font objects. In a similar way, Maxsurf Modeler's automation interface contains surface, control point and marker objects.

Maxsurf Modeler provides support for automation via an interface that allows the user to create, modify and analyse a design. While the Maxsurf Modeler application does not incorporate a facility for writing or recording macros directly, the automation interface will allow users to develop macros for Maxsurf Modeler from other applications. Maxsurf Modeler's automation interface enables it to interact with many other applications that support automation. This is very easily achieved in applications that provide a suitable VBA macro-programming environment such as

- Microsoft Excel
- Microsoft Word
- Microsoft Access
- Bentley MicroStation

Automation can also be used via many programming languages such as Visual Basic, Visual C++, Java and Compaq Visual Fortran. It is also supported by the Windows Scripting Host, which can be used to automate applications directly from the Windows environment.

For the technically inclined...

The core technology behind Maxsurf Modeler Automation is COM, Microsoft's Component Object Model. If you are familiar with COM you can use Maxsurf Modeler's COM interface as you would the COM interface of any other program. It can be accessed using VB, C++, C, Java or any other COM compatible language.

In this section:

- VBA

VBA

Visual Basic for Applications, or simply VBA, is Microsoft's application scripting language. It is the language used to write macros within the entire Microsoft Office suite and in other Microsoft products. It is also used within applications such as MicroStation and MathCAD. If you have experience in writing macros within any of these products then you should be able to quickly adapt to writing macros for Maxsurf Modeler.

VBA is the most readily available platform in which to write Maxsurf Modeler macros as most engineers have access to Microsoft Excel or Microsoft Word. It is also a relatively easy environment in which to develop scripts, macros or small programs that exploit automation. As such, this manual will concentrate of the use of VBA for the development of macros for Maxsurf Modeler. All examples presented in the manual will be coded using VBA.

VBA Compatibility

The VBA language provided in Microsoft Office 2000 and later contains a number of improvements to the version used within the Office 97 suite of products. A significant difference is in the use of enumerated types. The Maxsurf Modeler automation interface uses many enumerated types to make programming scripts much simpler when using the latest versions of VBA. However, when programming using an older version of VBA, such as used in Office 97, the use of enumerated types is not supported and the enumerated constants must be replaced with their integer value. The enumerated types and their values are summarised in Appendix A of this document.

Object Models

The key to the use of VBA within all these different applications is that VBA is simply a language for manipulating objects. Each application that uses VBA for scripting has its own object model that is a unique set of objects that can be manipulated by the language. If you are familiar with VBA then learning to write macros for another application only involves learning that application's object model.

Uses of Automation

Automation is a powerful tool that can be used to write anything from a simple macro to a full Windows application. Automation allows Maxsurf Modeler to directly communicate data with Word, Excel and MicroStation to automatically alter a design in Maxsurf Modeler, or obtain details of the design in Excel. Some examples of how this technology could be employed are:

- A macro to manipulate the geometry of a design.
- Writing a VBA program to export a design to IGES file format
- Writing a script to find surfaces or markers with particular properties
- Writing Excel macros to automatically access design data from Maxsurf Modeler, such as Hydrostatic Calculations
- Genetic Algorithms for Ship Design
- Manipulation of Design Control Points from Microsoft Excel
- Generating customised reports in Microsoft Word

Each of these examples could be designed as a simple macro or with a sophisticated user interface. Once familiar with VBA, it is an easy task to add dialogs and menus to your automation scripts.

What can't I automate?

The following parts of the Maxsurf Modeler application and data structures cannot be accessed using the current Maxsurf Modeler object model. This may change in the future.

- The Maxsurf Modeler user interface (i.e. Windows, menus, toolbars etc.) except for selections in the individual windows.
- Exporting of file formats other than IGES
- Controlling of the views, such as displaying lights, rendering and toggling section lines
- Surface Operations, such as align and trim. Other surface operations such as flip and size are not implemented, but the process can be replicated by editing the control points.

Speed

An important issue in automation programming is speed; automated scripts or macros can be relatively slow when they require making calls between applications. There are a number of techniques to improve their performance, the most important of which is to minimize the number of calls between applications. The Maxsurf Modeler automation interface has been designed to allow the advanced user to use a number of techniques to minimize the number of calls across the interface (between applications). These include using collections and lists to add or modify a group of items at once. Examples of these are given in Chapter 4 Advanced Maxsurf Modeler Automation and in Chapter 6 Microstation Automation.

Code Samples

This manual features many example scripts to demonstrate how the objects, properties and methods of the Maxsurf Modeler Automation interface are used. These scripts are all written in VBA and were developed using Microsoft Excel, Microsoft Word or MicroStation. Most of the scripts in the manual can be executed by inserting the code between the dashed lines in the following macro.

```
Sub                                                         Test()
  'definition       of       app       and       Design       objects
    Dim msApp As New BentleyModeler.Application
    Dim msDesign As New BentleyModeler.Design
    Set             msDesign            =            msApp.Design

  'Insert              sample              code              here!!!
  '----------------------------



  '----------------------------
End Sub
```

Note:

Many of the scripts will require a suitable design to be open within the Maxsurf Modeler application as they refer directly to particular control points, markers or grid lines.

Errors generated by code procedures may be related to having no design or an inappropriate design loaded into Maxsurf Modeler.

All the example code presented in this manual was written using the VBA editor provided within Microsoft Office 2000 or MicroStation. This code uses some features not available in the version of VBA provided with versions of Microsoft Office prior to Office 2000. One significant difference is that the older version of VBA does not support the use of enumerated data types, which have been used throughout the Maxsurf Modeler Object model. For this reason we recommend the use of Microsoft Office 2000 or later products when writing VBA scripts. However, scripts can still be written using Office 97 in which case enumerated values must be replaced by their integer value. All the enumerated types, and their values, used by the Maxsurf Modeler Automation Model are listed in Appendix A of this document.

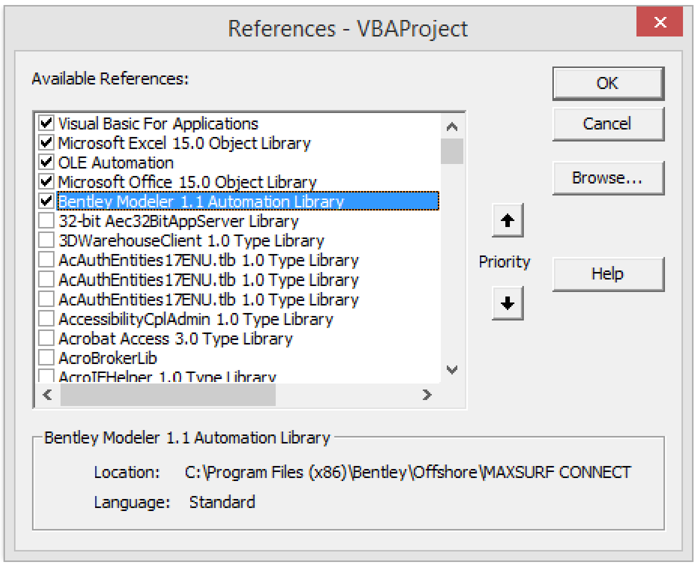
Early and Late Binding

To manipulate Maxsurf Modeler using VBA scripting in another application it is necessary to first create an instance of the Maxsurf Modeler Application object. This can be done using either Early Binding or Late Binding.

Binding refers to making a link between two programs. To be able to use Maxsurf Modeler from within Excel, we must create a link, referencing Maxsurf Modeler from Excel. When we bind Maxsurf Modeler to Excel, the functionality of Maxsurf Modeler becomes available in Excel.
Late binding uses the CreateObject method to create an instance of the Application object. For example, to create a new instance of the Maxsurf Modeler application object using late binding:

```
    Dim msApp As Object
    Set msApp = CreateObject("BentleyModeler.Application")
```

However, when programming with VBA we recommend that you use Early Binding. It has many advantages over late binding in that the code will execute faster, coding errors will be detected at compile time and the BentleyModeler object model will be incorporated into the intellisense features of the VB editor. To use early binding a reference to the BentleyModeler object library must be added to your project. This is done using the Tools | References menu, which brings up the following dialog:



- [✓] Visual Basic For Applications
- [✓] Microsoft Excel 15.0 Object Library
- [✓] OLE Automation
- [✓] Microsoft Office 15.0 Object Library
- [✓] Bentley Modeler 1.1 Automation Library
- [ ] 32-bit Aec32BitAppServer Library
- [ ] 3DWarehouseClient 1.0 Type Library
- [ ] AcAuthEntities17ENU.tlb 1.0 Type Library
- [ ] AcAuthEntities17ENU.tlb 1.0 Type Library
- [ ] AcAuthEntities17ENU.tlb 1.0 Type Library
- [ ] AccessibilityCplAdmin 1.0 Type Library
- [ ] Acrobat Access 3.0 Type Library
- [ ] AcroBrokerLib
- [ ] AcroIEHelper 1.0 Type Library

Bentley Modeler 1.1 Automation Library

Location: C:\Program Files (x86)\Bentley\Offshore\MAXSURF CONNECT

Language: Standard

- **Scroll down the list of available references until you find the BentleyModeler Automation Library**
- **Select this item by clicking in the box to it's left**
- **Click OK.**

A new instance of Maxsurf Modeler can now be created using early binding:

```
   Dim msApp As BentleyModeler.Application
   Set msApp = New BentleyModeler.Application
```

or more simply using the line

```
          Dim msApp as New BentleyModeler.Application
```

Maxsurf Modeler Does not Appear in the References Dialog

It is possible that the BentleyModeler Automation Library has not been registered in Windows. If this is the case then we will need to update the regserver to include BentleyModeler, using the following steps. You will only need to do this process once.

- Open a command prompt with administrator privileges (from start menu type command prompt and select "Run as administrator").
- Change directory to the location of the Maxsurf executable, normally located in: C:\Program Files\Bentley\Offshore\Maxsurf CONNECT Edition VXX\

Where XX is the version number
Eg type "cd C:\Program Files\Bentley\Offshore\Maxsurf CONNECT Edition V23\"

- At the prompt type the name of the executable appended by " /unregserver" (note space)
  Usually "MAXSURFModeler64.exe /unregserver"
- The prompt will state whether the unregserver was successful or not.
- Repeat but this time appending " /regserver" (note space)
  Usually "MAXSURFModeler64.exe /regserver"
- The prompt will state whether the regserver was successful or not.

This method can also be used to update to a new version of the Maxsurf Modeler Application that has not been installed by the installer.

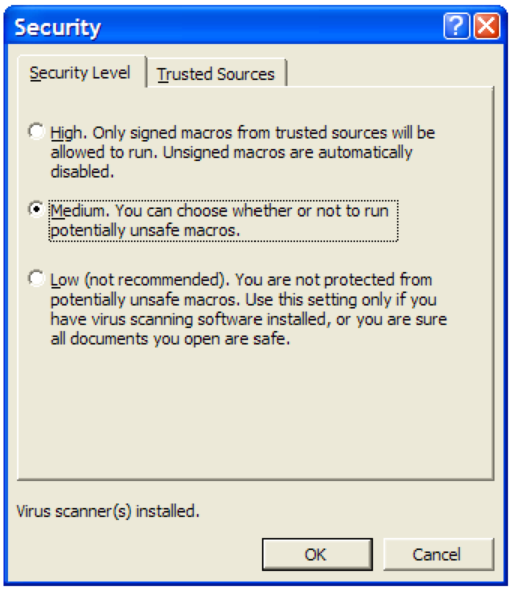
Initial Settings

This section contains some initial settings you may have to change for the examples in this manual to work.

Enabling Macros in Excel and Word

Macro's can contain viruses. Hence Microsoft applications have a default security setting to prevent Macro's from running. If you know a macro is from a trusted source, you can allow macros by setting the security setting to medium.

In most Microsoft applications, such as Excel, PowerPoint and Word, the security setting can be changed in the tools | macros menu. We recommend setting the security to medium, so that you can choose whether or not to enable the running of macros.



High. Only signed macros from trusted sources will be allowed to run. Unsigned macros are automatically disabled.

[✓ Medium. You can choose whether or not to run potentially unsafe macros.]

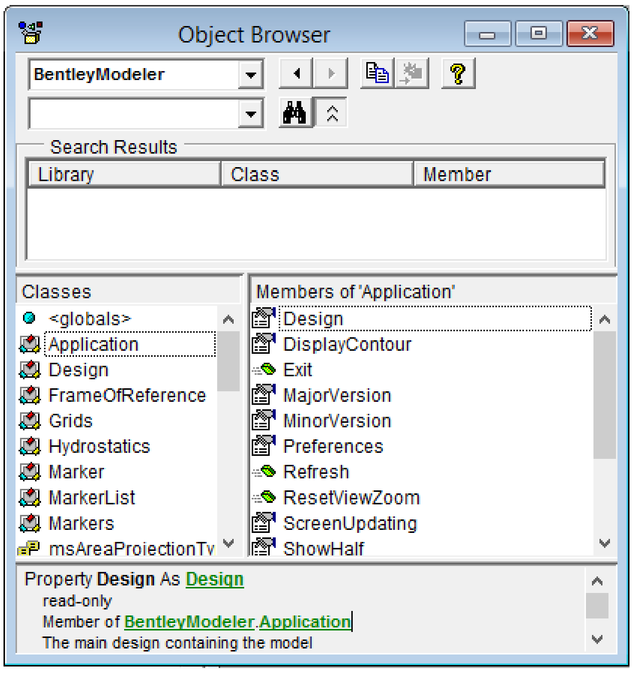
Low (not recommended). You are not protected from potentially unsafe macros. Use this setting only if you have virus scanning software installed, or you are sure all documents you open are safe.

Virus scanner(s) installed.

You will have to restart the host application for these changes to become effective.

Object Browser

Another advantage of using Early Binding is that you can use the Object Browser (select View | Object Browser menu in the VB Editor), shown below, to examine the names, properties and methods of the objects in the Maxsurf Modeler Object library.



Property **Design** As Design
read-only
Member of BentleyModeler.Application
The main design containing the model

Further Reading

The aim of this manual is to describe the Maxsurf Modeler automation interface and its use. The manual does not aim to teach the reader how to program in VBA or any other language. It is assumed that the reader is familiar with Visual Basic or VBA programming or that you have access to materials on this topic. There are many books available on VBA, particularly in relation to the Microsoft Office suite of products. Some references and other resources that may be useful in helping you to learn VBA and automation are:

- "Excel 2003 Power Programming with VBA" by John Walkenbach, John Wiley and Sons Inc. 2004
- A range of reference books on VBA programming with Word 2003 and Excel 2003 are published by Apress, www.apress.com
- https://msdn.microsoft.com/en-us/office
- News groups

  microsoft.public.word.vba.beginners
  microsoft.public.word.vba.general
  microsoft.public.office.developer.vba
  microsoft.public.excel.programming

The Maxsurf Modeler Object Model

This chapter provides an overview of the Maxsurf Modeler object model describing its structure and the purpose of each object within the model. Most of the objects within the object model are used to describe Surfaces, Markers or the Grid, or to find hydrostatic data from within Maxsurf Modeler.

In this section:

- Collections Objects and Lists
- Frame of Reference
- Grids
- Hydrostatics
- Surface
- Markers
- Preferences Object

Collections Objects and Lists

When programming using VBA you manipulate *objects* that represent different aspects of an application as defined by its object model. Schematic diagrams showing the entire Maxsurf Modeler object model are displayed in Figure 1 through to Figure 7 on the following pages. It divides the Maxsurf Modeler application and the design model into several distinct *objects*. These *objects* may represent a single component of a model such as a marker, surface or grid line. Other objects within the model do not represent a physical part of the model but are used to store settings or data such as the hydrostatic data.

Each *Object* contains *Properties* and *Methods*:

- Properties are variables that store details of the object such as the name, height or offset. The actual type and name of the properties are specific to each type of object.
- *Methods* are functions that you can use to manipulate the object, such as Add, Delete or modify variables or aspects of the object.

A special type of object is a *collection*, a container storing an ordered set of objects of the same type. For example, the Markers collection in the Design object contains all of the markers in the design. Collections within the Maxsurf Modeler object model are named as the plural of the objects they contain. Collections in Maxsurf Modeler are initially empty.

Note

All of the collections in Maxsurf Modeler are 1-based i.e. the first item in the list has an index of one (1). Some care must be taken when programming in VBA, as it can use zero based collections and arrays in which the first item in the list has an index of zero (0).

Application Object

The root of the Maxsurf Modeler object model is the Application object. All other objects within the object model can be accessed either directly or indirectly via this object. The Application object provides access to the Design object and six Maxsurf Modeler properties as shown in Figure 1. There are also two methods in this object.

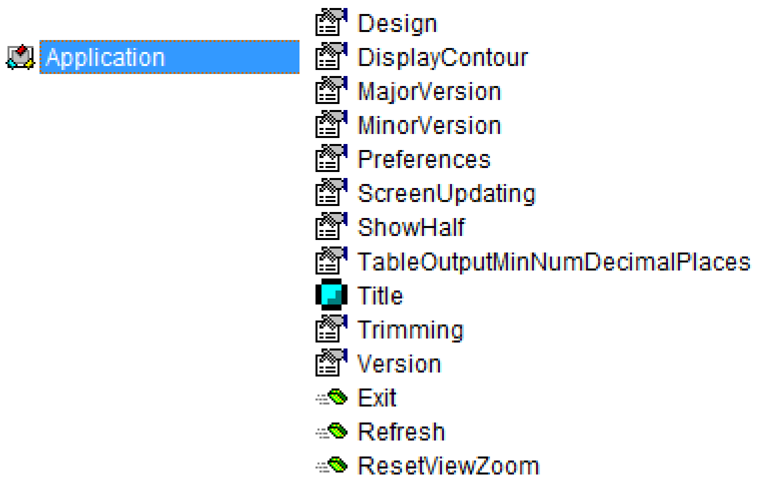
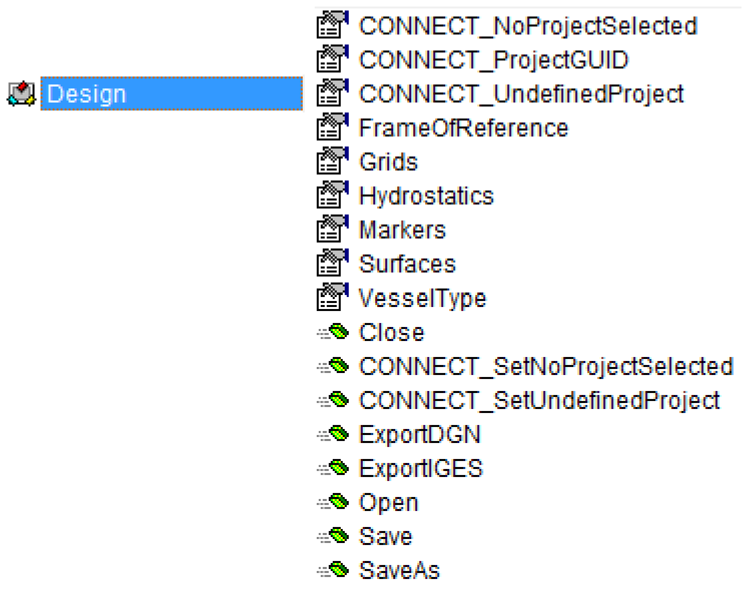


Figure 1 Application Object Hierarchy

The objects, properties and methods located within the application object refer to global settings for Maxsurf Modeler. Properties of the current design are located within the Design object.

Design Object

The Design object is used to describe all aspects of the current design. There are several objects within the Design object and several methods. The objects within the Design object are described in the remainder of the chapter. Uses of the methods in the Design object can be found in Chapter 3 Basic Maxsurf Modeler Automation.



Design

CONNECT_NoProjectSelected
CONNECT_ProjectGUID
CONNECT_UndefinedProject
FrameOfReference
Grids
Hydrostatics
Markers
Surfaces
VesselType
Close
CONNECT_SetNoProjectSelected
CONNECT_SetUndefinedProject
ExportDGN
ExportIGES
Open
Save
SaveAs

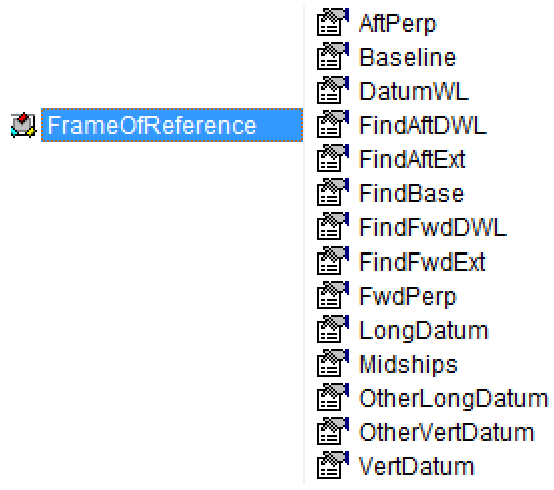
Figure 2 The Design Object Hierarchy

Sample code to access these objects and methods:

```
msApp.Design.Save

i = msApp.Design.Markers(1).Height
```

Frame of Reference

The FrameOfReference object is a read only object, used to get the properties of the Frame of Reference used in the design. The five properties will return the values shown in the Frame of Reference dialog within Maxsurf Modeler (Data | Frame of Reference . .) and the midships location.



FrameOfReference

AftPerp
Baseline
DatumWL
FindAftDWL
FindAftExt
FindBase
FindFwdDWL
FindFwdExt
FwdPerp
LongDatum
Midships
OtherLongDatum
OtherVertDatum
VertDatum

Figure 3 The Frame of Reference Object Hierarchy

Sample code to access these objects and method:

```
    i = msApp.Design.FrameOfReference.aftPerp

    j = msApp.Design.FrameOfReference.DatumWL
```

See Also:
Frame of Reference Object on page 32

Grids

The Grids object can be used to create, get or edit the grid in Maxsurf Modeler. It does not support the space function (to distribute grid lines) that can be used from inside of Maxsurf Modeler, but does support all other aspects.

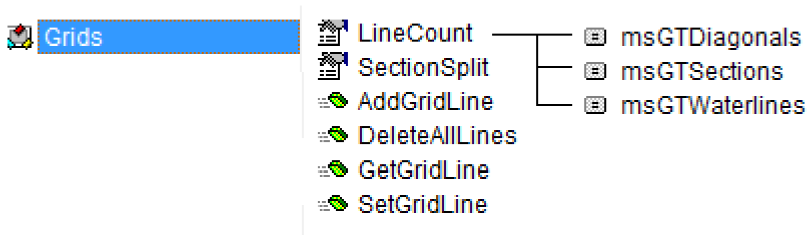


Figure 4 Grids Object Hierarchy

Sample code to access these objects and method:

```
    msApp.Design.Grids.AddGridLine msGTButtocklines, Butt_01, 4, 0

    i=msApp.Design.Grids.LineCount
```

See Also:
Tutorial Part 3: Creating a Grid on page 26
Grids Object on page 33

Hydrostatics

The Hydrostatics object can be used to get all of the data available in the 'Hydrostatics at DWL' dialog (Data | Calculate Hydrostatics...).

The Hydrostatics object also provides methods to calculate the hydrostatic data and to perform a parametric transformation, these methods are discussed further in Hydrostatics Object on page 34 and in the Example file Creating a Systematic Series on page 57.

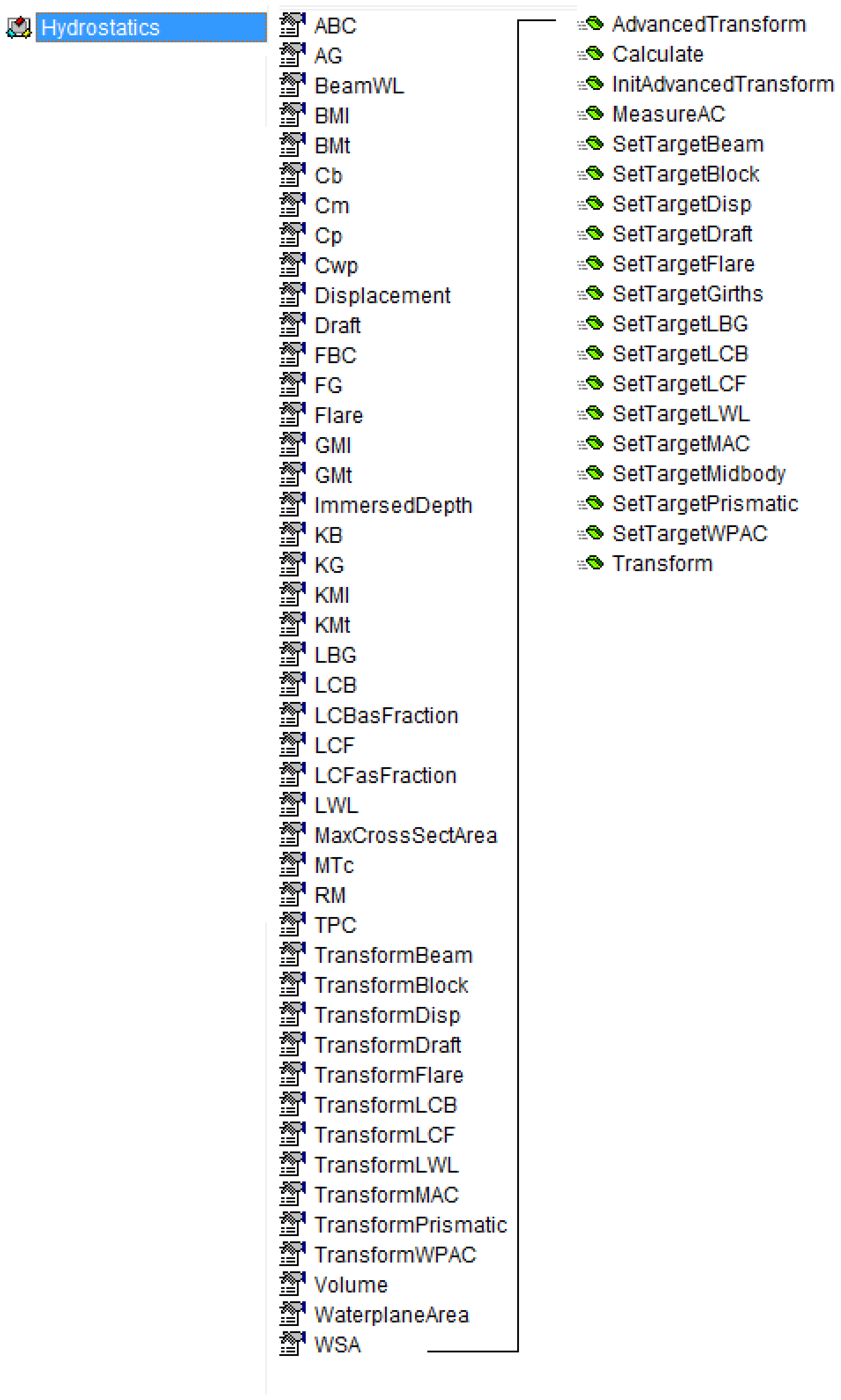


Figure 5 Hydrostatics Object Hierarchy

Sample code to access these objects and methods:

```
    i = msApp.Design.Hydrostatics.Cb
    msApp.Design.Hydrostatics.Calculate 1.025, 2
```

See Also:

Tutorial Part 4: Calculating the Hydrostatics on page 26
Hydrostatics Object on page 34

Surface

The Surface object is used to control aspects of the design that would be controlled from the surfaces window, within Maxsurf Modeler. The objects within the Surface object have control over all the same features as appear in that window. The object hierarchy shown in Figure 6 denotes the objects available for the Surface Object

In addition to the Surface Object, there are also the Surfaces and SurfaceList Objects within the Application Object Hierarchy. The Item method of the Surfaces or SurfaceList objects is used to reference a particular surface, which can then be used to access the Surface object.

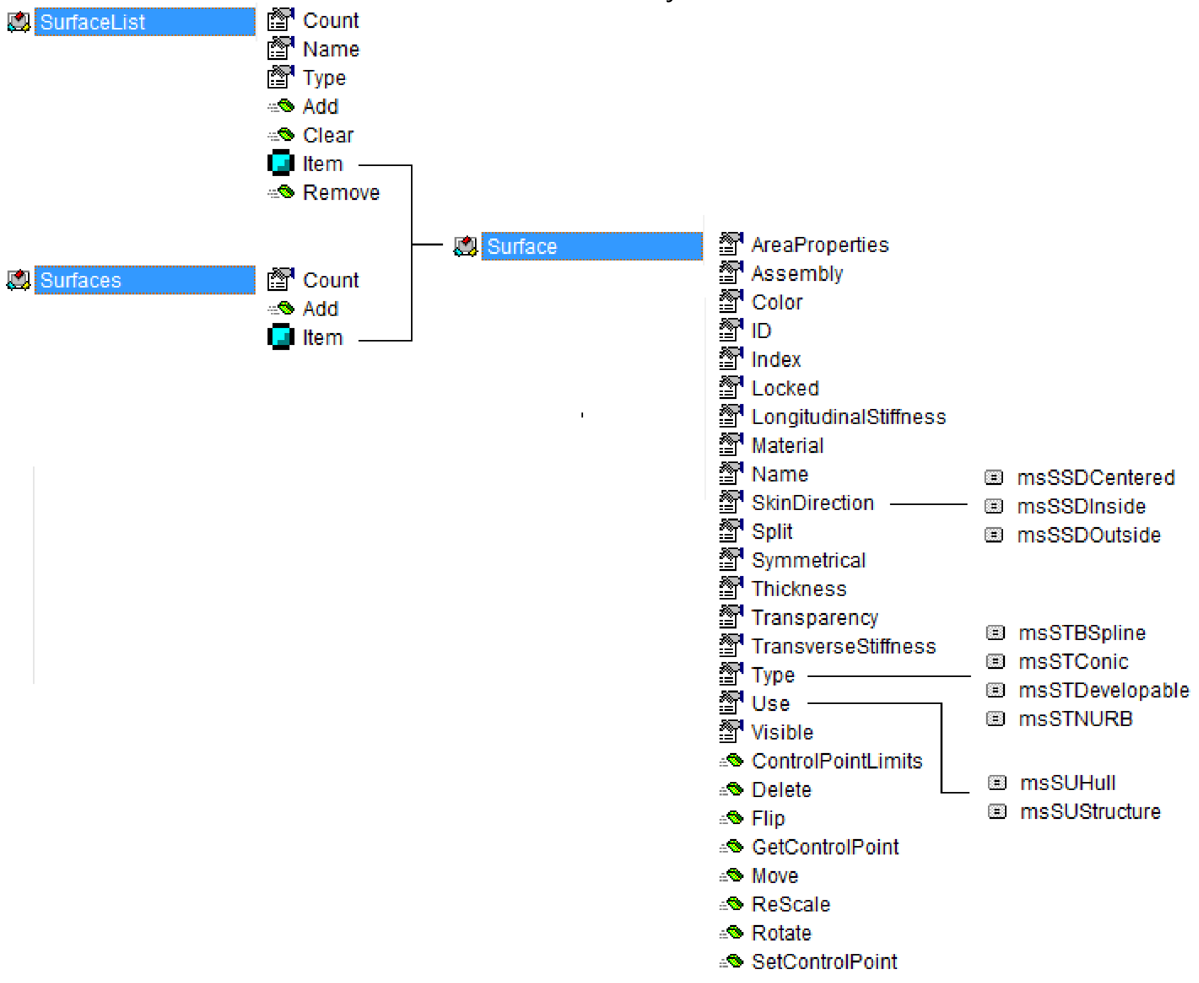


Figure 6 Surface Object Hierarchy

Accessing Surfaces

The Surface object cannot be accessed directly through the interface, and must be accessed through either the Surfaces or SurfaceList objects to define which particular surface is being accessed.

This can be done either with:

msApp.Design.Surfaces.Item(i).*property*

or

msApp.Design.Surfaces(i).*property*

Sample code to access these objects and method:

```
    msApp.Design.Surfaces(i).Move 5, 2, 1
    i = msApp.Design.Surfaces(i).Color
```

See Also:

Tutorial Part 1: A Basic Maxsurf Modeler Script on page 24
Tutorial Part 2: Moving the Surface Control Points on page 25
Surface Object on page 37

Markers

The Marker object allows access to the properties that would be accessed from the Markers window within Maxsurf Modeler, this object can be used to set or get the Position, Offset or Height of the markers, relating to the (x, y, z) coordinates each marker. The Marker Object can also obtain properties for the marker station, the surface it is associated with and the type of marker (its location within the surface).

The Objects Markers and MarkerList are used to refer to the Marker object in the same way as the Surfaces and Surface objects. It is not possible to use the Marker object directly through the interface. The Marker Object needs to be accessed either through the Markers or MarkerList objects so as to define which marker is being accessed.

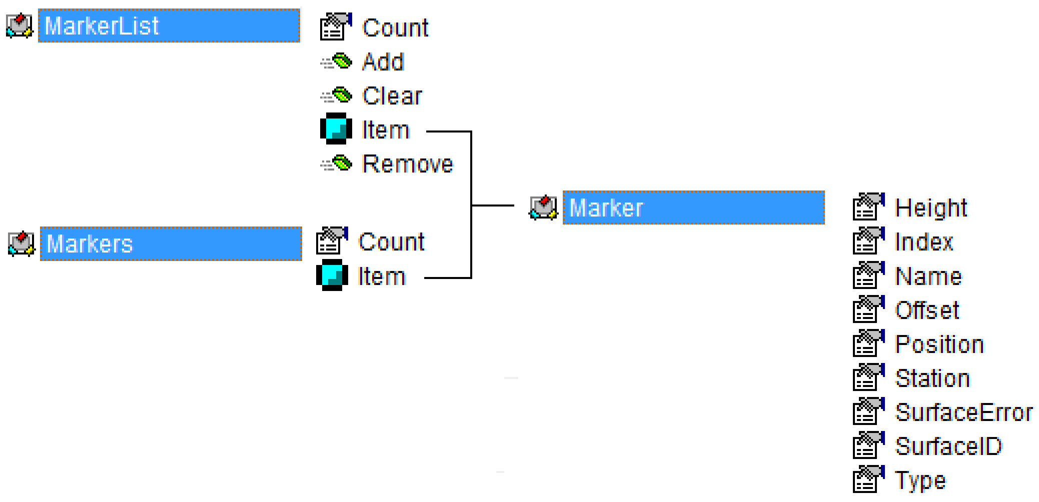


Figure 7 Marker Object Hierarchy

Sample code to access these objects and methods:

```
    i = msApp.Design.Markers(i).Height
    j = msApp.Design.Markers.Count
```

See Also:

Marker Object on page 36

Preferences Object

The Preferences object provides access to settings within the Maxsurf Modeler application that control how data is presented inside of Maxsurf Modeler. The Preferences object contains objects to set the units of measurement for dimensions and weight inside of Maxsurf Modeler. It also provides method to set the precision of Maxsurf Modeler surfaces.

Note

All methods and properties in the BentleyModeler Automation Interface use the SI-units of **metres** and **kilograms**. Changing the preference units only changes the units within the Maxsurf Modeler user interface.
Maintaining consistent units between the BentleyModeler Automation and Maxsurf Modeler interfaces can be useful to avoid any confusion when working with both environments.

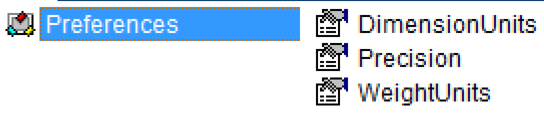


Figure 8 Marker Object Hierarchy

Sample code to access these properties:

```
    msApp.Preferences.DimensionUnits = msDUCentimeters
    msApp.Preferences.Precision = msSPLow
```

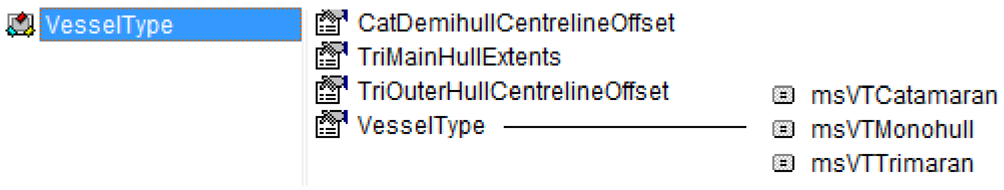
The Enumerated values for the types available in the preferences object are listed in Enumerated Types on page 73.

See Also:

Preferences and Units on page 39
Preferences on page 46

VesselType Object

The VesselType object allows access to properties pertaining to the type of vessel being modelled.



Sample code to access these properties:

msApp.VesselType.VesselType = msVTCatamaran
The Enumerated values for the types available in the preferences object are listed in Enumerated Types on page 73.

Chapter 2 Getting Started

In chapter 1 we have discussed the configuration of the Maxsurf Modeler Object Model and what it can and cannot do. This chapter presents tutorials that provide an introduction to writing VBA scripts that interact with Maxsurf Modeler. We will start with an example of a simple Macro and then continue with a tutorial that develops a script to design a simple hull form in Maxsurf Modeler.

Example of a Simple Macro

Maxsurf Modeler does not include facilities for its own Visual Basic Editor, writing scripts is done externally through the Visual Basic Editors in programs such as Excel and MicroStation. In the Microsoft Office products this is performed via the Tools | Macro | Visual Basic Editor command from the main menu.

Open the Visual Basic Editor and add the following text to the content of a file open within this window.

```
Sub Hello()

  MsgBox "Hello World"

End Sub
```

To run this macro, locate the cursor within the code and select the command Run | Run Sub/Userform from the main menu. You will see a simple dialog saying "Hello World".
For more information on the basics of creating macros, see the Further Reading section on page 15.

Tutorial: Creating a Simple Hull Form

The remainder of this chapter concentrates on the development of a very simple model hull form. From a default surface, the control points will be rearranged to form the hull shape shown in Figure 9. The following sections describe how to add a surface and modify the geometry of the surface by moving the controls that make up the surface. The successive sections then extend the script by adding a new piece of code to define the Grid lines and determine the Hydrostatics of the hull.

- Tutorial Part 1: A Basic Maxsurf Modeler Script
- Tutorial Part 2: Moving the Surface Control Points
- Tutorial Part 3: Creating a Grid
- Tutorial Part 4: Calculating the Hydrostatics
- Tutorial Part 5: Combining the Code Segments

Each of these sections begins by introducing a small part of the Maxsurf Modeler object model. It then uses this in the development of a script for creating the hull form. The entire script for generating the hull form is listed at the end of this chapter.

The final result of this tutorial will look like this in Maxsurf Modeler:

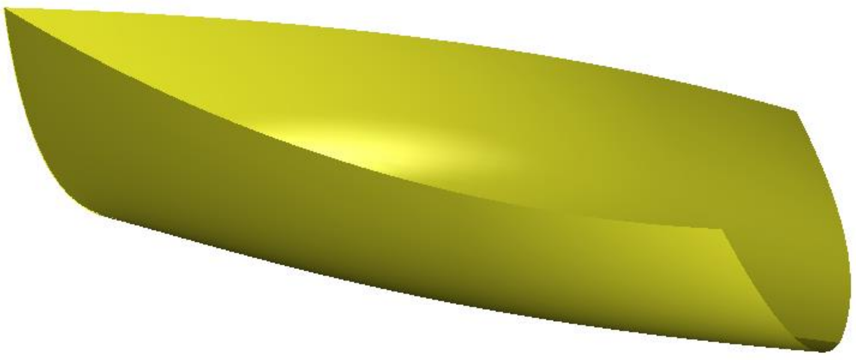
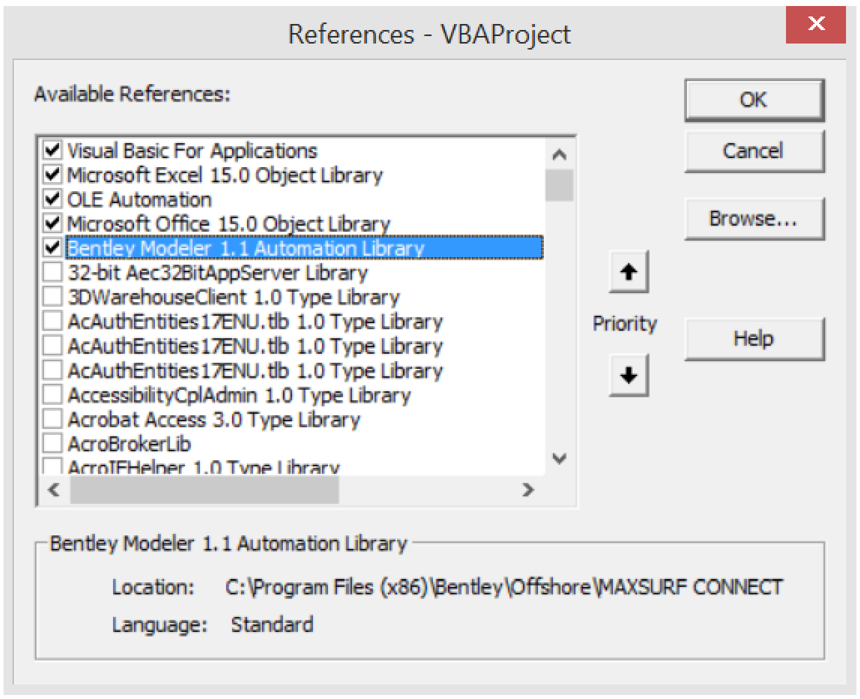


Figure 9 The Surface Hull Shape, created by moving the control points from the default Maxsurf Modeler surface.

Tutorial Part 1: A Basic Maxsurf Modeler Script

In this tutorial, as for all the code in this manual, we will use early binding. To enable Maxsurf Modeler within your VBA script using early binding you must first reference the BentleyModeler Automastion Library in the Visual Basic environment. To do this, select the Tools | References command from the main menu.



References - VBAProject

Available References:

- [✓] Visual Basic For Applications
- [✓] Microsoft Excel 15.0 Object Library
- [✓] OLE Automation
- [✓] Microsoft Office 15.0 Object Library
- [✓] Bentley Modeler 1.1 Automation Library
- [ ] 32-bit Aec32BitAppServer Library
- [ ] 3DWarehouseClient 1.0 Type Library
- [ ] AcAuthEntities17ENU.tlb 1.0 Type Library
- [ ] AcAuthEntities17ENU.tlb 1.0 Type Library
- [ ] AcAuthEntities17ENU.tlb 1.0 Type Library
- [ ] AccessibilityCplAdmin 1.0 Type Library
- [ ] Acrobat Access 3.0 Type Library
- [ ] AcroBrokerLib
- [ ] AcroIEHelper 1.0 Type Library

Bentley Modeler 1.1 Automation Library

Location: C:\Program Files (x86)\Bentley\Offshore\MAXSURF CONNECT

Language: Standard

Figure 10 The references dialog. Accessed via Tools | References, from the main menu.

Search the list of items displayed in the resulting dialog and find the BentleyModeler 1.1 Automation Library entry. To enable Maxsurf Modeler, simply click in the check box to its left and then press OK to exit the dialog. If the reference is set up correctly you will get the benefits of automatic assistance in the editor as you write your macros or scripts. Amongst other things, this will automatically list the properties and methods of objects as you write VBA code (intellisense).

Troubleshooting:

If BentleyModeler Does not Appear in the References Dialog

See Maxsurf Modeler Does not Appear in the References Dialogon page 13 for the procedure to register the BentleyModeler Automation Library with Windows.

The VBA subroutine below is a simple script that uses Maxsurf Modeler automation. Type this script into the VBA programming environment.

```
Dim msApp As New BentleyModeler.Application
Public Sub Tutorial_1()
    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    'Creates a surface in Maxsurf Modeler
    msDesign.Surfaces.Add msSLDefault
End Sub
```

Before running this script, make sure Maxsurf Modeler is running and a new design has been started. Now run this script from the VBA environment, it creates a new surface in the current Maxsurf Modeler design. To check this, you may want to turn on the control point net, or rendering.

Tutorial Part 2: Moving the Surface Control Points

Once we have the surface in the design, we can edit the locations of the control points to create the desired hull shape. The control points are moved using the SetControlPoint method. The method takes in 5 variables in the following order: row, column, position offset and height;

```
Public Sub Tutorial_2()

    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    i = msDesign.Surfaces.Count 'This finds the number
                                'of surfaces in the design

    'Move all the control Points to form a vessel shape.
    'All these dimensions are in metres,
        'regardless the units set inside of Maxsurf Modeler
    msDesign.Surfaces(i).SetControlPoint 1, 1, -10, 0, -2
    msDesign.Surfaces(i).SetControlPoint 2, 1, -10, 3, -1.5
    msDesign.Surfaces(i).SetControlPoint 3, 1, -8,  3,  2
    msDesign.Surfaces(i).SetControlPoint 1, 2,  0,  0, -2.5
    msDesign.Surfaces(i).SetControlPoint 2, 2,  0,  5, -2
    msDesign.Surfaces(i).SetControlPoint 3, 2,  0,  5,  2
    msDesign.Surfaces(i).SetControlPoint 1, 3, 7.5, 0, -2
    msDesign.Surfaces(i).SetControlPoint 2, 3, 9.5, 0, -2
    msDesign.Surfaces(i).SetControlPoint 3, 3,  10, 0,  3

    'To  update  the  screens  in  Maxsurf  Modeler,  use  the  refresh
command
    msApp.Refresh

End Sub
```

This script also uses the Surfaces.Count method. This finds the total number of surfaces in the design. This is then used by the Surfaces object, to ensure that the SetControlPoints method is acting on the most recently created surface.

The code looks cluttered with all the control point coordinate data in it. Instead of including all the data in the code, automation allows us to place the data in an Excel spreadsheet and make references to this from the code. This way, changes to the design can be easily implemented in the spreadsheet, rather than in the code. This is where the power of automation comes in. Examples using this are included in Chapter 6 Microstation Automation.

See Also:

Surface on page 20

Surface Object on page 37

Tutorial Part 3: Creating a Grid

Grid lines can be added to a design using the AddGridLines method of the Grids object. The following section of code adds waterlines, section lines and buttock lines to the current design. The variables for adding grid lines are in the following order: grid line type, name, location, and angle*

- * = The grid line angle is ignored except for diagonal gridlines.

```
Public Sub Tutorial_3()
    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    'Create a Grid
    'Create waterlines
    msDesign.Grids.AddGridLine msGTWaterlines, WL1, -2, 0
    msDesign.Grids.AddGridLine msGTWaterlines, WL2, -1, 0
    msDesign.Grids.AddGridLine msGTWaterlines, WL3, 1, 0
    msDesign.Grids.AddGridLine msGTWaterlines, WL4, 2, 0
    'Create Section lines
    msDesign.Grids.AddGridLine msGTSections, Sec1, -9, 0
    msDesign.Grids.AddGridLine msGTSections, Sec2, -5, 0
    msDesign.Grids.AddGridLine msGTSections, Sec3, -1, 0
    msDesign.Grids.AddGridLine msGTSections, Sec4, 1, 0
    msDesign.Grids.AddGridLine msGTSections, Sec5, 5, 0
    msDesign.Grids.AddGridLine msGTSections, Sec6, 9, 0
    'Create Buttock Lines
    msDesign.Grids.AddGridLine msGTButtocklines, B0, 0, 0
    msDesign.Grids.AddGridLine msGTButtocklines, B1, 1.5, 0
    msDesign.Grids.AddGridLine msGTButtocklines, B2, 3, 0

    msApp.Refresh

End Sub
```

This code creates new grid lines, regardless of existing grid lines. Running this code twice will create two sets of grid lines on top of each other. To avoid this situation, all the grid lines can be deleted, prior to creating the new grid lines by using the following code inserted before the grid is created in tutorial_3()

```
    'Remove any existing Grid
    msDesign.Grids.DeleteAllLines msGTWaterlines
    msDesign.Grids.DeleteAllLines msGTSections
    msDesign.Grids.DeleteAllLines msGTButtocklines
```

Running the code now will remove all existing grid lines, prior to creating the new ones.

See Also:

Grids on page 18

Modifying Grid Lines in Excel on page 51

Tutorial Part 4: Calculating the Hydrostatics

The Maxsurf Modeler automation interface allows calculation and reading of Hydrostatic data for a design.

For this tutorial exercise, we will read some of the hydrostatic data and display it in a message box. Although this code only takes in a small amount of data, all of the data that is available through the Hydrostatics window inside Maxsurf Modeler is available through the automation interface.
The code is comprised of four sections:

- Defining variables
- Calculating the hydrostatics
- Reading in hydrostatic data and
- Displaying the data in message boxes.

```
Public Sub Tutorial_4()

    Dim msDesign As Maxsurf Modeler.Design
    Set msDesign = msApp.Design

    'Define Variables for use in the code
    Dim Displacement As Long
    Dim Draft As Long
    Dim Beam As Long

    'Calculate the Hydrostatic Data for the Hull
    msDesign.Hydrostatics.Calculate 1025, -2
    'Get the Hydrostatics of the Hull
    Displacement = msDesign.Hydrostatics.Displacement
    Draft = msDesign.Hydrostatics.ImmersedDepth
    Beam = msDesign.Hydrostatics.BeamWL
    'This is only a selection of all the hydrostatic data
 'available

    'Display the Hydrostatic Data in Message Boxes
    MsgBox "The Displacement is " & Displacement & " kg"
    MsgBox "The Draft is " & Draft & " m"
    MsgBox "The Beam is " & Beam & " m"

End Sub
```

The Calculate method reads in two inputs, these are the water density (kg/m^3) and the vertical centre of gravity (m). These are the same two options as the Hydrostatics dialog box inside Maxsurf Modeler.

Other hydrostatic data could be displayed by:

- Defining another variable name
  Eg Dim WSA As Long
- Setting the variable to equal a data amount
  Eg WSA = msDesign.Hydrostatics.WSA
- Creating another message box to display the amount
  Eg MsgBox “The Surface Area is ” & WSA & ” m^2”

See Also:

Hydrostatics on page 18
Calculating the Hydrostatics on page 60

Tutorial Part 5: Combining the Code Segments

The previous four sections of code can be combined into one single piece of code that can be executed as one program. Alternatively, if you have programmed each section individually, the four components can be executed using calls from a fifth program.

The following section of code is the complete combined tutorial:

```
Dim msApp As New BentleyModeler.Application
```

```
Public Sub Tutorial()

    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    'Creates a surface in Maxsurf Modeler
    msDesign.Surfaces.Add msSLDefault

    i = msDesign.Surfaces.Count
        'This finds the number of surfaces in the design

    msDesign.Surfaces(i).Visible = False

    'Move all the control Points to form a vessel shape.
    'All these dimensions are in metres,
        'regardless the units set inside of Maxsurf Modeler

    msDesign.Surfaces(i).SetControlPoint 1, 1, -10, 0, -2
    msDesign.Surfaces(i).SetControlPoint 2, 1, -10, 3, -1.5
    msDesign.Surfaces(i).SetControlPoint 3, 1, -8,  3,  2
    msDesign.Surfaces(i).SetControlPoint 1, 2,  0,  0, -2.5
    msDesign.Surfaces(i).SetControlPoint 2, 2,  0,  5, -2
    msDesign.Surfaces(i).SetControlPoint 3, 2,  0,  5,  2
    msDesign.Surfaces(i).SetControlPoint 1, 3, 7.5, 0, -2
    msDesign.Surfaces(i).SetControlPoint 2, 3, 9.5, 0, -2
    msDesign.Surfaces(i).SetControlPoint 3, 3, 10,  0,  3

    'Create a Grid
    'Create waterlines
    msDesign.Grids.AddGridLine msGTWaterlines, WL1, -2, 0
    msDesign.Grids.AddGridLine msGTWaterlines, WL2, -1, 0
    msDesign.Grids.AddGridLine msGTWaterlines, WL3, 1, 0
    msDesign.Grids.AddGridLine msGTWaterlines, WL4, 2, 0
    'Create Section lines
    msDesign.Grids.AddGridLine msGTSections, Sec1, -9, 0
    msDesign.Grids.AddGridLine msGTSections, Sec2, -5, 0
    msDesign.Grids.AddGridLine msGTSections, Sec3, -1, 0
    msDesign.Grids.AddGridLine msGTSections, Sec4, 1, 0
    msDesign.Grids.AddGridLine msGTSections, Sec5, 5, 0
    msDesign.Grids.AddGridLine msGTSections, Sec6, 9, 0
    'Create Buttock Lines
    msDesign.Grids.AddGridLine msGTButtocklines, B0, 0, 0
    msDesign.Grids.AddGridLine msGTButtocklines, B1, 1.5, 0
    msDesign.Grids.AddGridLine msGTButtocklines, B2, 3, 0

    'Get the Hydrostatics of the Hull
    Dim Displacement As Long
    Dim Draft As Long
    Dim Beam As Long

    msDesign.Hydrostatics.Calculate 1025, -2
    Displacement = msDesign.Hydrostatics.Displacement
    LWL = msDesign.Hydrostatics.LWL
    Draft = msDesign.Hydrostatics.Draft
    Beam = msDesign.Hydrostatics.BeamWL

    MsgBox "The Displacement is " & Displacement & " tonnes"
    MsgBox "The Draft is " & Draft & " m"

    msDesign.Surfaces(i).Visible = True
    'To  update  the  screens  in  Maxsurf  Modeler,  use  the  refresh
command
    msApp.refresh
End Sub
```

The alternative to combining all of this code into one procedure is to use *calls*. The advantage of using calls is that multiple procedures can call on one procedure. For example, a procedure to determine the hydrostatics could be called by several other procedures, and hence only needs to be written once.

For this tutorial example, the procedure to call all the tutorial segments looks like;

```
Public Sub Tutorial_RunAll()

    Call Tutorial_1
    Call Tutorial_2
    Call Tutorial_3
    Call Tutorial_4

End Sub
```

This script so far has not shown anything particularly useful. The power of automation in Maxsurf Modeler comes from being able to manipulate and refine data through integration with other programs such as Microsoft Excel and MicroStation. This tutorial has provided a basis of programming with the Maxsurf Modeler object model; subsequent chapters will look at more powerful uses of Maxsurf Modeler automation and integration of Maxsurf Modeler with other programs.

Chapter 3 Basic Maxsurf Modeler Automation

There are several operations that are common to many automation scripts, such as opening and closing designs, saving and exporting designs and redrawing the Maxsurf Modeler screen. These objects and methods will be discussed in this chapter to provide a background and a starting point for other processes.
In this section;

- Basic Operations
- Working With The Design
- Preferences and Units

Basic Operations

There are several operations that are used in most scripts, such as file handling and screen updating. These processes are covered in the following sections.
In this section:

- Opening and Closing a Design
- Saving and Exporting DesignsScreen Updating and Refresh

Opening and Closing a Design

The following section of code contains three separate procedures, the first two are different methods of opening a design from a file and the final procedure closes the design.

The difference between the first two procedures is in how they open the design. The first procedure has the name of the file to be opened written into the code. This will open the same specified file every time.

The second procedure opens a dialog box in Excel, allowing the user to select the file they wish to open. Which procedure is most suitable depends on the situation.

```
Dim msApp As New BentleyModeler.Application

Sub OpenFile()
    'Opens the named File

    Dim FileName As String
        'Creates a Variable for the File name

    FileName  =  "C:\Program  Files\Bentley\Offshore\Maxsurf  CONNECT
Edition V21.1\Sample Designs\Modeler Sample_Trawler.msd"
        'Defines the FileName Variable. Changing this
        'will change the file that is loaded
    msApp.Design.Open FileName, False, False

End Sub

Sub OpenFileDialog()
    'Opens a file selected in the Dialog Box
    Dim Filter As String
    Dim FileName As String

    Filter = "Maxsurf Modeler Design File (*.msd), *msd"
    'The Filter only allows certain file types to be loaded
    FileName  =  Application.GetOpenFilename(Filter,  ,  "Open  Maxsurf
Modeler File", , False)

    msApp.Design.Open FileName, False, False

End Sub

Sub CloseDesign()
```

```
    msApp.Design.Close False

End Sub
```

The two Maxsurf Modeler Objects being used in the script are

```
msApp.Design.Open FileName, False, False
```

and

```
msApp.Design.Close False
```

The options available when opening a file are to specify the filename (as a String), whether you would like to merge the new design with the current design (Boolean) and whether you would like to save the current design (Boolean).

The close method provides an option to save the file before closing (Boolean)

Saving and Exporting Designs

The Maxsurf Modeler Automation interface provides facilities to save designs and export IGES file types. The following code samples will run if placed in the generic code shown on page 12.

To save a file under the same name, use;

```
    msApp.Design.Save
```

To save a design under a different file name, use the SaveAs method, specifying the path and filename and whether the file is to overwrite an existing file name (Boolean).

```
    msApp.Design.SaveAs  "C:\Documents  and  Settings\UserName\My
Documents\TestSave.msd", True
```

To export a design in the IGES format, use the ExportIGES method, again specifying the file and pathname, as for the save as method.

```
    msApp.Design.ExportIGES  "C:\Documents  and  Settings\UserName\My
Documents\IGESTest.igs"
```

Screen Updating and Refresh

When running large procedures, disabling screen updates in Maxsurf Modeler can reduce execution times. This can be done by setting the ScreenUpdating property to False at the start of the procedure, then returning it to True at the end.

```
    msApp.ScreenUpdating = False
    'To turn it off

    '--------------------
    'Procedure Code goes here
    '--------------------

    msApp.ScreenUpdating = True
    'and to turn it back on again
```

Other times, changes will be made in Maxsurf Modeler through the automation interface that won't be updated on the screen. At the end of a procedure or a loop in a procedure the Refresh method can be given, to redraw the screen and update

```
    msApp.Refresh
```

This updates the screen to show the most current settings.

Working With The Design

This section describes the objects that are used to develop the design through Maxsurf Modeler automation. These include objects for describing the Frame of Reference, Grids, Hydrostatics, Markers and Surfaces. These objects form part of a hierarchy contained within the Design object that encapsulates the objects describing an entire design and its associated properties.

In this section;
- Frame of Reference ObjectGrids Object
- Hydrostatics Object
- Marker Object
- Surface Object

Example Code Usage for Maxsurf Modeler Design Objects;
- Frame of Reference Example
- Grids Example
- Hydrostatics Example
- Marker Example
- Surface Example

Frame of Reference Object

Property	Type	Description
aftPerp	Double	Read Only. Get the Aft Perpendicular Location
Midships	Double	Read Only. Get the Midships Location
Baseline	Double	Read Only. Get the Baseline Location
DatumWL	Double	Read Only. Get the Datum Waterline Location
fwdPerp	Double	Read Only. Get the Forward Perpendicular Location

The FrameOfReference object is used to get the locations of the Fore and Aft perpendiculars, Midships, Baseline and the Design Waterline.

See Also:

Frame of Reference on page 17

Frame of Reference Example

This information could be very useful if you were trying to find the LCB as a proportion of the LWL, aft of the Forward Perpendicular (as used by the parametric transformation method). Make sure an instance of Maxsurf Modeler is open with a valid design before running this macro.

```
Dim msApp As New BentleyModeler.Application

Sub FindLCBPercent()

    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    Dim AftPerp As Double
    Dim FwdPerp As Double
    Dim LCB As Double
    msDesign.Hydrostatics.Calculate

    AftPerp = msDesign.FrameOfReference.AftPerp
    FwdPerp = msDesign.FrameOfReference.FwdPerp
    LCB = msDesign.Hydrostatics.LCB

    LCB_percent = (LCB - FwdPerp) / (AftPerp - FwdPerp)

    MsgBox "LCB from fwd perp " & LCB_percent

End Sub
```

Grids Object

The Sections, Buttocks, Diagonals and Waterlines are specified in Maxsurf Modeler using a grid. The Grids object is used to create, edit and remove these grid lines. The Properties and Methods available are shown in the following tables:

Property	Type	Description
LineCount	msGridType	Gets the number of Lines of a particular type in the design
SectionSplit	Long	Set/Get the index of the split section line.

Method	Type	Description
AddGridLine	gridType As msGridType, label As String, pos As Double, angle As Double	Add a Grid line to the design
DeleteAllLines	msGridType	Delete all of a type of Grid line
GetGridLine	gridType As msGridType, Index As Long, pLabel As String, pPos As Double, pAngle As Double	Reads in the type of Grid line and the index, returns the Label, Position and Angle (where applicable) of that grid line
SetGridLine	gridType As msGridType, Index As Long, label As String, pos As Double, angle As Double	Sets a grid line of type msGridType and a given index to a new Label, position and angle (where applicable)

See Also:

Grids on page 18
Tutorial Part 3: Creating a Grid on page 26.

Grids Example

To count the number of section lines currently in the design, use the LineCount property.

```
    i = msDesign.Grids.LineCount(msGTSections)
```

To add another Sectionline, with a title of “Section 6” (If the previous number of sections was 5) and a longitudinal position of 4m,

```
msDesign.Grids.AddGridLine msGTSections, "Section " & i + 1,
                                                         4, 0
```

Hydrostatics Object

The hydrostatics object can be used to read the hydrostatics data for a design in Maxsurf Modeler and to perform parametric transformations of the design. The following tables of Properties and Methods show all available possibilities for the Hydrostatics Object.

Property	**Type**	**Description**
BeamWL	Double	Read Only. Gets the waterline beam
BMl	Double	Read Only. Gets the distance from the centre of buoyancy to the longitudinal metacentric height
BMt	Double	Read Only. Gets the distance from the centre of buoyancy to the transverse metacentric height
Cb	Double	Read Only. Gets the block coefficient
Cm	Double	Read Only. Gets the midships coefficient
Cp	Double	Read Only. Gets the prismatic coefficient
Cwp	Double	Read Only. Gets the waterplane coefficient
Displacement	Double	Read Only. Gets the displacement
Draft	Double	Read Only. Gets the height of the DWL from the baseline
GMl	Double	Read Only. Gets the longitudinal metacentric height
GMt	Double	Read Only. Gets the transverse metacentric height
ImmersedDepth	Double	Read Only. Gets the immersed depth of the design
KB	Double	Read Only. Gets the height of the centre of buoyancy from the baseline
KG	Double	Read Only. Gets the height of the centre of gravity from the baseline
KMl	Double	Read Only. Gets the height of the longitudinal metacentre from the baseline
KMt	Double	Read Only. Gets the height of the transverse metacentre from the baseline
LCB	Double	Read Only. Gets the longitudinal centre of buoyancy relative to the datum
LCF	Double	Read Only. Gets the longitudinal centre of floatation relative to the datum
LWL	Double	Read Only. Gets the waterline length
MaxCrossSectArea	Double	Read Only. Gets the maximum cross sectional area
MTc	Double	Read Only. Gets the moment required to change the trim by one centimetre
RM	Double	Read Only. Gets the Righting Moment
TPC	Double	Read Only. Gets the Tonnes per centimetre Immersion for the design.
Volume	Double	Read Only. Gets the immersed volume of the design

Property	Type	Description
WaterplaneArea	Double	Read Only. Gets the waterplane area
WSA	Double	Read Only. Gets the wetted surface area

Method	Type	Description
Calculate	Density as Double, VCG as Double	Calculates / Recalculates the hydrostatic data
Transform	TargetLCB As Double, TargetCoeff As Double, TargetMac As Double, TargetDisp As Double, TargetImmersedDepth As Double, TargetBeam As Double, TargetLWL As Double, DoConstrainDisp As Boolean, DoConstrainLWL As Boolean, DoConstrainBeam As Boolean, DoConstrainImmersedDepth As Boolean, OptimiseBlockCoeff As Boolean	Performs a Parametric Transformation of the hull as per Maxsurf Modeler's parametric transformation (Data | Parametric Transformation)
InitAdvancedTransform		Initialise Transformation
SetTargetDisp	DispValue as Double	Set the Target Displacement
SetTargetLWL	LWLValue as Double	Set the Target LWL
SetTargetLBG	LBGValue as Double	Set the Target Length Between Girths
SetTargetGirths	AGValue as Double, FGValue as Double	Set the Target Aft and Forward Girths
SetTargetBeam	BeamValue as Double	Set the Target Waterline BEam
SetTargetDraft	DraftValue as Double	Set the Target Draft
SetTargetLCB	LCBValue as Double	Set the Target Longitudinal Center of Buoyancy
SetTargetBlock	BlockValue as Double	Set the Target Block Coefficient
SetTargetPrismatic	PrismaticValue as Double	Set the Target Prismatic Coefficient
SetTargetMAC	MACValue as Double	Set the Target Midship Area Coefficient
SetTargetFlare	FlareValue as Double	Set the Target Topside Flare
SetTargetLCF	LCFValue as Double	Set the Target Longituinal Center of Flotation (not currently enabled)

Method	Type	Description
SetTargetMidbody	TargetMidbody1 as Double, TargetMidbody2 as Double	Set the Target Aft and Forward Midbody positions
SetTargetWPAC	WPACValue as Double	Set the Target Waterplane Area Coefficient
AdvancedTransform		Perform the transformation

The parametric transformation reads in a large number of variables in order to provide the correct constraints on the transformation. An example showing the use of parametric transformation in the Automation interface is in the Example Creating a Systematic Series on page 60 and Blending Hull Forms on page 64.

The Advanced Parametric Transformation is an alternative transformtion function that permits a far larger range of constraints to be specified. To use this function, first call InitAdvancedTransform, then call whichever SetTarget functions you require to specify and constrain your transformation. Once the constraints have been set, call Advanced Transform to complete the transformation.

See Also:

Hydrostatics on page 18
Tutorial Part 4: Calculating the Hydrostatics on page 26

Hydrostatics Example

To calculate the current hydrostatic data use the Calculate method, with the options for density and VCG

```
    msDesign.Hydrostatics.Calculate 1025, 2
```

This will set the contents of Cell "B5" to be the vessel displacement

```
    Range("B5") = msDesign.Hydrostatics.Displacement
```

Marker Object

Markers can be created in Maxsurf Modeler to mark locations on the hull surface. Normally these markers are located along section lines and hence are associated with sections.

There are several Marker Properties as listed:

Property	Type	Description
Height	Double	Set/Get the height (y coord) of the Marker
Index	Long	The reference index for that particular marker
Name	String	The Name for that Marker
Offset	Double	Set/Get the Offset (z coord) of the Marker
Position	Double	Set/Get the Longitudinal Position (x coord) of the Marker
Station	Long	Set/Get the Station that a Marker is associated with
SurfaceID	Long	Set/Get the surface that a Marker is associated with
Type	msMarkerType	Define the type of Marker that a Marker is (Interior, top edge etc.) see page 74 for the complete list.

The Marker object can be used to set or get the properties of existing Markers. The object could be used to export a Marker table into Excel, edit or alter the points, and then return them to Maxsurf Modeler.

See Also:

Markers on page 21

Marker Example

To set the contents of the Cell E*i* (where *i* is an integer) to the height value of the *ith* marker

```
        Range("E" & i) = msDesign.Markers(i).Height
```

To set the type of the *i*th marker to be msMTBottomLeft

```
        msDesign.Markers(i).Type = msMTBottomLeft
```

Surface Object

The Surface object is a very important object as it contains all the objects, properties and methods that describe the surfaces currently in the Maxsurf Modeler application. This includes all the properties available in the surfaces window in Maxsurf Modeler and methods for moving and altering control points and surfaces.

A summary of all the properties and methods of a Surface object are listed in the table below.

Property	Type	Description
Assembly	Long	Get the ID of the assembly this surface belongs to
Color	OLE_COLOR	Set/Get the Color of the surface
ID	Long	Read Only. Returns the ID of the surface.
Index	Long	Read Only. Returns the Index of the surface
Locked	Boolean	Set/Get if the surface is locked for modification
LongitudinalStiffness	Long	Set/Get the Surfaces Longitudinal Stiffness
Material	Long	Set/Get the surface material
Name	String	Set/Get the descriptive name for a surface
SkinDirection	msSurfaceSkinDirection	Set/Get the surfaces skin direction
Split	Boolean	Set/Get if a surface is split in the body plan view
Symmetrical	Boolean	Set/Get if the surface is symmetrical about a Centreline Plane
Thickness	Double	Set/Get the surface thickness
Transparency	Long	Set/Get the percentage transparency for the surface
TransverseStiffness	Long	Set/Get the Transverse Stiffness of the Surface
Type	msSurfaceType	Set/Get the surface type
Use	msSurfaceUse	Set/Get the use of the surface (hull/internal)
Visible	Boolean	Set/Get if a surface is Visible

Method	Type	Description
ControlPointLimits	iRows As Long, iColumns As Long	Read Only. Returns the number of Rows and Columns in a surface.

Method	Type	Description
Delete	-	Deletes a Surface
GetControlPoints	iRow As Long, iColumn As Long, xVal As Double, yVal As Double, zVal As Double	Gets the x,y,z coordinates for a control point, given the row and column indices
Move	X As Double, Y As Double, Z As Double	Moves a surface
Rotate	dRoll As Double, dPitch As Double, dYaw As Double, dLongCentre As Double, dTransCentre As Double, dVertCentre As Double	Rotates a Surface
SetControlPoint	iRow As Long, iColumn As Long, xVal As Double, yVal As Double, zVal As Double	Sets the x,y,z coordinates for a control point, given the row and column indices

OLE_COLOR

The OLE_COLOR data type represents a color as a BGR (Blue, Green, Red) value. The OLE_COLOR value of a color specified by its red, green and blue components (each of which has a value from 0 - 255) is determined using the following expression:

OLE_COLOR value = red + (green x 256) + (blue x 256^2)

The OLE_COLOR values of some common colors are as follows:

Black	0
Dark Grey	4210752
Grey	8421504
Light Grey	12632256
White	16777215
Red	255
Green	65280
Blue	16711680
Magenta	16711935
Cyan	16776960

See Also:

Surface on page 20
Tutorial Part 1: A Basic Maxsurf Modeler Script on page 24

Surface Example

To determine the OLE_COLOR of the surface i, Use the Color property. This will return a value as described above.

```
MsgBox msDesign.Surfaces(i).Color
```

Alternatively, we could set a cell color in Excel to be the same as the surface color in Maxsurf Modeler using the code;

```
  Range("E13").Interior.Color = msApp.Design.Surfaces(1).Color
```

To determine the number of Rows and Columns of Control Points in the *ith* surface;

```
    Dim NumRows As Long
    Dim NumCols As Long

    msDesign.Surfaces(i).ControlPointLimits NumRows, NumCols
    MsgBox "There are " & NumRows & " rows and " & NumCols & " 
columns"
```

Preferences and Units

The Preferences object provides access to settings that control the units used to display quantities in the Maxsurf Modeler user interface. The properties of the preference object are summarised in the following table. The enumerated values for the various types are listed under Enumerated Types on page 73.

- Units
- Precision

Property	Type	Description
DimensionUnits	msDimensionUnits	Gets/Sets the Dimension units for the application
Precision	msSurfacePrecision	Set/Get the surface precision for the application
WeightUnits	msWeightUnits	Set/Get the Weight unit for the application

NB: This does not effect the units used in the Automation Interface, which are always in kilograms and metres

Units

The Length and Weight units for Maxsurf Modeler are identified through the DimensionUnits and WeightUnits objects respectively. These two objects represent the choices available in the Units Dialog Box (Data | Units) within Maxsurf Modeler. The objects use enumerated values to define the unit types. These values are shown on page 73 under Enumerated Types.

An example of setting the units within Maxsurf Modeler to millimeters and kilograms is;

```
    msApp.Preferences.DimensionUnits = msDUMillimeters
    msApp.Preferences.WeightUnits = msWUKilograms
```

The Units objects can also be used to get the dimension units from Maxsurf Modeler.

Precision

The surface precision in Maxsurf Modeler can be changed through the Automation interface using the Precision object of the Preferences object. The precision can be set to 5 stages between 'lowest' and 'highest'. The precision object uses enumerated values of msSurfacePrecision type as defined on page 73 in the section Enumerated Types.

In code, the Precision object is used as:

```
    msApp.Preferences.Precision = msSPMedium
```

The precision object can also be used to get the current surface precision from Maxsurf Modeler.

Chapter 4 Advanced Maxsurf Modeler Automation

This chapter describes the use of collection and list objects defined within the Maxsurf Modeler Automation Interface. List objects are an important class of object as they provide a means of rapidly accessing and modifying data for a large group of objects with only a small number of subroutine calls between Maxsurf Modeler and the host. A good understanding of the behaviour and use of list objects will ensure that scripts execute efficiently. The first two sections of this chapter describe the properties and methods of collection and list objects.

Figure 11 Shows the relation between Items, Collections and Lists. An item represents a single entity, such as a Marker or a Surface. A Collection is the group defining all of the Items. Within the collection it is possible to define a sub-set of the collection, known as a List. Using lists can be advantageous as they can improve the execution speed of an automation procedure.

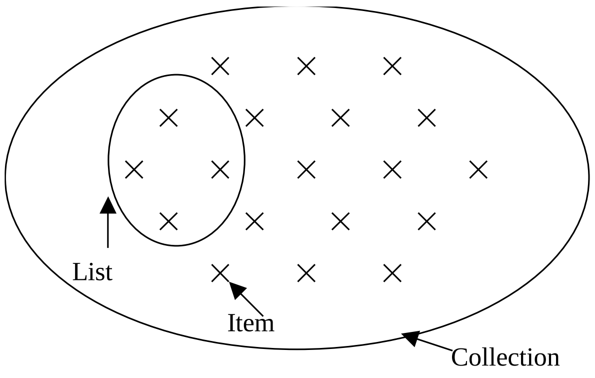


Figure 11 Relations between Items, Collections and Lists

- Using Collections
- Using Lists

Using Collections

A collection within the Maxsurf Modeler object model is a container for storing an ordered group of objects. It is essentially the equivalent of an array within VBA but its implementation has been encapsulated within an object. As such, it provides some methods and properties for accessing the items within the collection. The content of a collection is not arbitrary; in fact its contents correspond directly to the components of the model. As such, adding an item to a collection adds a corresponding component to the design. For example, all the surfaces representing a design are contained within a Surfaces collection object stored within the Design object. Adding a new surface to the design is performed by adding a new surface to the Surfaces collection.

All collection objects have a number of common properties and methods for manipulating their content. These are summarised in the tables below.

Property	**Type**	**Description**
Count	Long	Read Only. Returns the number of items in list.

Method	**Returns**	**Description**
Add		Add objects to list
Item(*int*)	Object	Returns the ith item in list.

Collection Properties

Collection objects have a single property, Count, which returns the number of objects contained within the collections.

```
Dim                 nMarkers                   as                  long

'Get       number       of       markers       in       the       design
nMarkers                   =                   msApp.Design.Markers.Count

'Tell                                                               user
MsgBox "Design contains " & nMarkers & " Markers."
```

Collection Methods

An object within the collection is returned via the Item method.

```
    Dim sName As String

    'Get the name of the 1st Surface
    sName = msApp.Design.Surfaces.Item(1).Name

    MsgBox sName
```

The above code returns the name of the first surface in the design. All of the surface and marker properties can be accessed through the Item property of their relevant Collection object.

The Surfaces collection object has a method for adding new items to the design. A new surface is added to the design using the Add method of the Surfaces collection. The Add method for the surfaces collection has been used already in this manual in the tutorial on page 23

Using Lists

A drawback of using automation is that calls between the applications are relatively slow. To ensure that your scripts run quickly it is important to reduce the number of calls between the applications. Maxsurf Modeler uses list objects, which provide a means of accessing and manipulating the properties of a number of objects in a single statement. List objects are similar to collections in that they store a number of objects in an array. However, list objects can store an arbitrary group of objects while collections store all the instances of a particular type of object. An example of this is the Surfaces collection, which provides a reference to all the Surfaces in a design. An arbitrary subset of these surfaces can be stored using the SurfaceList object.

Both the Markers and Surfaces collections in Maxsurf Modeler have corresponding list objects, MarkerList and SurfaceList. List objects have a number of common properties and methods for manipulating their content. These are summarised in the tables below.

Property	**Type**	**Description**
Count	Long	Read Only. Returns a number of items in list.
Name (surfaceList only)	Variant	Set/Get the Names of Surfaces in the list
Type (surfaceList only)	msSurfaceType	Set/Get the type of surface for surfaces in the list

Method	**Returns**	**Description**
Add		Add items to the list.
Clear		Remove all items from the list.

Method	Returns	Description
Item	Object	Returns item in list.
Remove		Remove items from the list.

The behaviour and use of these properties are described in detail in the following sections. Each list object also has a number of properties that it inherits from the objects it contains.

These properties are either accessed individually through the Item method of the List, or as a complete list, through properties native to that list, such as the Name property of SurfaceList. For example, the SurfaceList object can be used to access all the surface types using the Type property. Alternatively, the type of a single surface can be returned using the item(i) method.

```
    sList.Type
    sList.Item(1).Type
```

All the properties of a Surface or Marker can be accessed individually, using the Item method, through the SurfaceList or MarkerList objects respectively.

Further details regarding lists are included in the following sections of this chapter.

- Declaring Lists
- List Properties
- List Methods

Declaring Lists

List objects differ from the other objects and collections in Maxsurf Modeler object model, as they do not have a direct representation within the Maxsurf Modeler user interface. To declare a list object, the new instance must be created by declaring the object using the New keyword. For example:

```
Dim       sList       as       New       BentleyModeler.SurfaceList
Dim mList as New BentleyModeler.MarkerList
```

List Properties

The list objects have one property that is common to all list objects. The Count property of a list object returns the number of items in the list.

```
    i = mList.Count
    j = sList.Count
```

Lists have some of their own properties that will store data for all the items in the list. Lists can also be used to access all the properties of their respective objects through the item method. Properties of the lists themselves do not return a single value but a variant containing an array of values, each array entry corresponding to the value of the property for each item within the list.

For example, the SurfaceList object has the Name property describing the name of each of the surfaces in the list. The following code will display the name of the first surface in the collection;

```
MsgBox sList.Name(1)
```

Alternatively, we could store the names of all the surfaces into a variant array, then display one of those names, using this section of code.

```
    sName = sList.Name
'This stores all the surface names in a variant called sName
    MsgBox sName(1)
```

In one call to Maxsurf Modeler, this second method has retrieved the names of all the surfaces, and stored them into an array for further use. Using this method can significantly increase the speed of the script, especially when you are using more than one piece of data (one name) or using the same data more than once.

List data is read/write in the same way as collection data. Data can either be set individually to a specific new value, using a For Next statement, or can all be set to one value. Both are demonstrated in the following code segment. The code appends the surface name for all surfaces in the list with a number, according to the surfaces position in the list. The code then changes all the surface types to be developable surfaces (type 3)

```
Dim msApp As New BentleyModeler.Application

Sub ChangesUsingLists()

    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    Dim sList As New BentleyModeler.SurfaceList
    Dim sName As Variant

    sList.Add msDesign.Surfaces
    sName = sList.Name

    For i = 1 To UBound(sName)
        sName(i) = sName(i) & " " & i
        'MsgBox sName(i)
    Next

    sList.Name = sName

    'Set all surfaces to Developable
    sList.Type = 3

    msApp.Refresh
End Sub
```

This script executes much faster than one that loops over each surface in the design and sets each attribute separately, as it significantly reduces the number of subroutine calls made between Maxsurf Modeler and the host application. This script requires only three (3) inter-application calls to retrieve and set the data for the surfaces. When compared to the two (2) calls per surface required when looping over the individual surfaces in the design, the savings in time are significant.

List Methods

An item is added to a list using the Add method. By adding an item to a list we are adding a reference to that item in the list. This should not be confused with the Add methods of the collection object that create new items within the design. To complement this method, the Remove method is used to remove items from the list.

The Marker and Surface lists require a variant parameter to define the reference of the items being added or removed from the list. There are several ways to define which items are being added to the list. These are shown in the following code examples.

```
Dim msApp As New BentleyModeler.Application
Sub AddingItems()

    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    Dim mList As New BentleyModeler.MarkerList

    'Adding a marker already in the list will create an error
    'This line will handle that error and move on
    On Error Resume Next

    'Add marker 1
    mList.Add 1

    'Add Markers 2,3,4,5
    mList.Add "2,3,4,5"

    'Add Markers 6 through 10
    mList.Add "6-10"

    'Add Marker 11 using the markers object
    mList.Add msDesign.Markers(11)

    'Clear the entire List
    mList.Clear

    'Add every marker in the collection
    mList.Add msDesign.Markers

End Sub
```

The clear method is also used in the above code, to remove all markers from the list.

Items within the list can be accessed in the same way as items within a collection, using an item number; the following code will store all the height values for the marker list into the C column of the current Excel sheet.

```
Dim msApp As New BentleyModeler.Application
Sub AccessItems()

    Dim msDesign As BentleyModeler.Design
    Set msDesign = msApp.Design

    Dim mList As New BentleyModeler.MarkerList

    mList.Add msDesign.Markers

    For i = 1 To mList.Count
        Cells(i + 10, 3) = mList(i).Height
    Next

End Sub
```

List and Collection Item Numbers

Care needs to be taken when accessing elements in a List. The item reference used in the list is for that list. The MarkerList element 6 will be the 6th marker in the list, not the 6th marker in the collection.

Chapter 5 User Interface

The Maxsurf Modeler Automation interface is designed to operate externally from Maxsurf Modeler, with Maxsurf Modeler running in the background. It is therefore not important to be able to control the Maxsurf Modeler user interface settings through the Automation interface. Maxsurf Modeler does however provide some control over settings within the user interface.

- Change Title
- Screen updating
- Refresh
- Trimming
- Preferences

Change Title

The title of the drawing, shown on the title bar across the top of the Maxsurf Modeler Window can be set through the Automation interface using the Title property.

```
Sub ChangeTitle()

    msApp.title = "New Hull Design"

End Sub
```

Running this code will change the title to read “New Hull Design”. This property could be used to show the current settings being implemented by a code, for instance;

```
Sub AlterDesign()

    For i = 1 To 100
        '-----------------
        'Code to iterate through a process
        '-----------------

        msApp.Refresh
        msApp.title = "Iteration " & i
    Next i

End Sub
```

This code will show the current hull design, and will specify the iteration number of that hull design in the title bar, allowing the user to follow the progress on the Maxsurf Modeler screen. Saving overwrites the design title, so the Title property cannot be used to make a permanent change to the title.

Screen updating

To improve the speed of execution of code, screen updating can be disabled. This means that Maxsurf Modeler will not redraw the screen every time a change is made, reducing the amount of work being done as the code executes. Screen updating can be disabled at the beginning of the code and then enabled at the end of the code;

```
Sub ScreenUpdating()
    msApp.ScreenUpdating = False
    '--------------
    'Code Goes here
    '--------------
    msApp.ScreenUpdating = True
End Sub
```

This code will make the Maxsurf Modeler user interface dormant while the code executes, then display the altered settings once completed

Refresh

Many changes made through the automation interface may not be updated immediately in the user interface. To force the screen to update, use the Refresh method. It is good practice to include this command at the end of every code section.

```
    msApp.Refresh
```

Trimming

Trimming affects both the graphics in the user interface and the hydrostatics calculations. It is good practice to ensure that trimming is turned on before performing any hydrostatic calculations. Trimming is accessed through the application object.

```
    msApp.Trimming = True 'Sets trim invisible
    msApp.Trimming = False 'Sets trim Visible
```

Preferences

Some control over preferences for Weight and Dimension units is given within Maxsurf Modeler automation. The preferences only affect the units within Maxsurf Modeler. Units within the automation interface always work in metres and kilograms. These preferences are described in Preferences Object on page 21 and again on page 39 in the Units section.

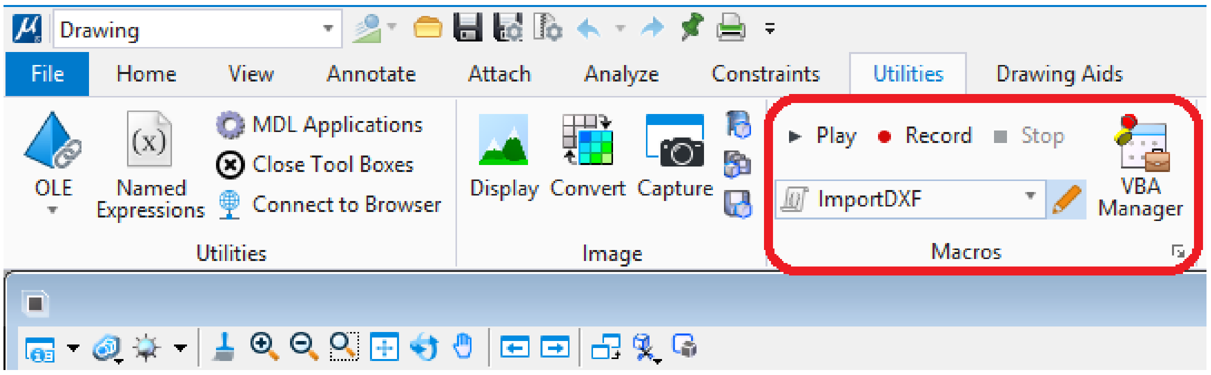
There is also a preference for the surface precision. Changing the surface precision within Maxsurf Modeler (Surfaces | Precision | . .) will change the number of stations used to calculate parameters of the hull. This can be seen by turning parametrics on, (Display | Net | Show Net). The surface precision determines the number of stations used to calculate the hydrostatics, which will affect the accuracy of the hydrostatic calculations.
The Precision property is described in detail in the **Precision** section on page 39.

Chapter 6 Microstation Automation

In this chapter a simple brief overview of writing VBA macros in Microstation will be given.

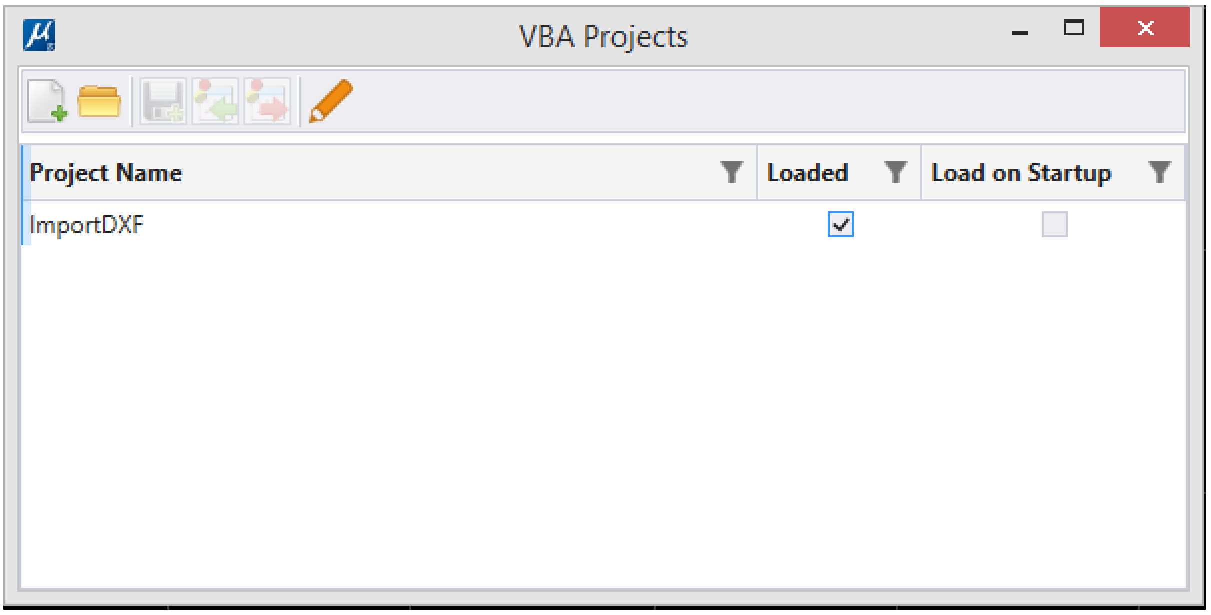
Writing Macros in Microstation

Microstation has its own VBA editor, similar to the Microsoft Office Suite VBA editors.
The Macro functionality can be accessed from the Utilities Ribbon:



To run a Macro choose the Marco from the drop down list and press Play. A user can record Macros without writing any VBA code simply by clicking on the record button, performing the steps of the required functionality manually and then clicking stop. Once stopped Microstation will ask for a name for the Macro.

Alternatively Macros can be written in VBA code directly into the VBA editor supplied with Microstation. To do this click on the VBA Manager button. This will activate the VBA Projects window:

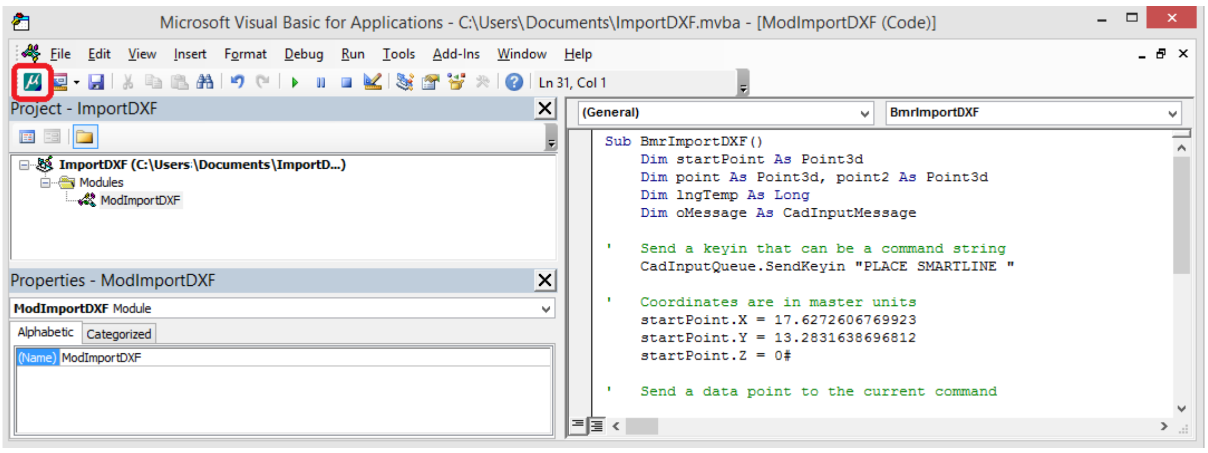


Project Name	Loaded	Load on Startup
ImportDXF	✓	☐

This will list all the VBA projects that Microstation knows about. Microstation VBA projects have the file extension ".mvba". To load a macro for this Microstation session simply click on the "loaded" check box. If you use a macro regularaly and would like to have it automatically loaded each time you start a Microstation session check the "Load on Startup "option.

You can add new Macros by clicking on the new button ( ) or load existing ones by clicking on the open folder ( ). Once a project is highlighted you can go to the code in the Microstation VBA editor by clicking on the edit button( ).

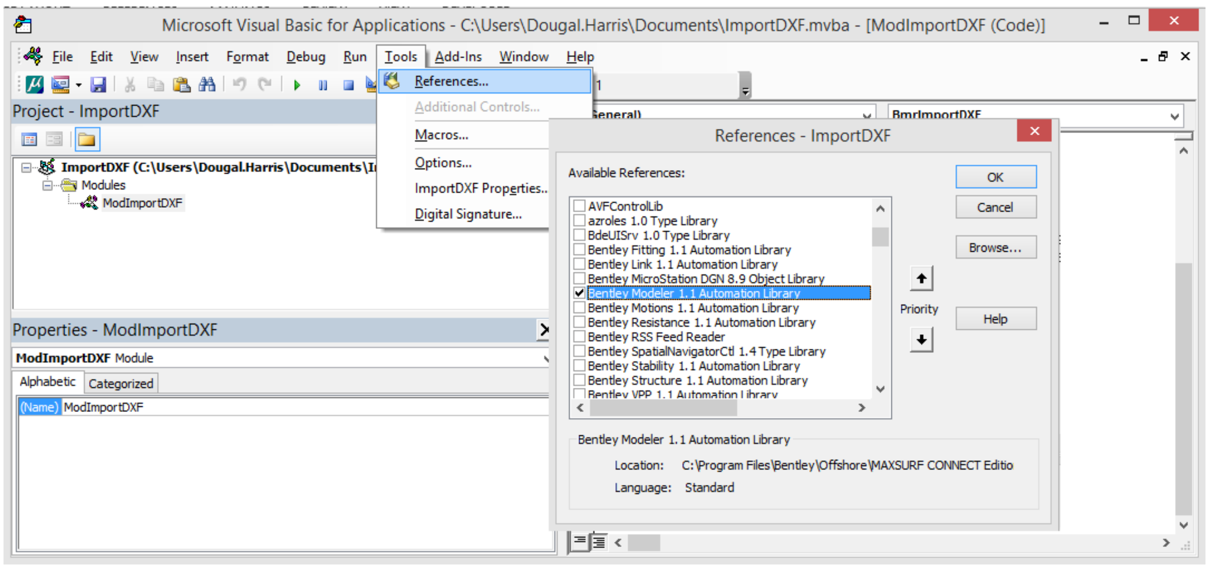
This will bring up the VBA editor. The editor is the same one as supplied with Microsoft applications, the only difference being a button in the top left hand corner for switching back to Microstation:


Microsoft Visual Basic for Applications - C:\Users\Documents\ImportDXF.mvba - [ModImportDXF (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 31, Col 1
Project - ImportDXF
ImportDXF (C:\Users\Documents\ImportD...)
Modules
ModImportDXF
Properties - ModImportDXF
ModImportDXF Module
Alphabetic
Categorized
(Name) ModImportDXF
(General)
BmrImportDXF

```
Sub BmrImportDXF()
    Dim startPoint As Point3d
    Dim point As Point3d, point2 As Point3d
    Dim lngTemp As Long
    Dim oMessage As CadInputMessage

'   Send a keyin that can be a command string
    CadInputQueue.SendKeyin "PLACE SMARTLINE "

'   Coordinates are in master units
    startPoint.X = 17.6272606769923
    startPoint.Y = 13.2831638696812
    startPoint.Z = 0#

'   Send a data point to the current command
```

Linking Microstation and Maxsurf Modeler

As with any automation code using COM, a reference to Bentley Modeler Automation library needs to be made:


Microsoft Visual Basic for Applications - C:\Users\Dougal.Harris\Documents\ImportDXF.mvba - [ModImportDXF (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
References...
Additional Controls...
Macros...
Options...
ImportDXF Properties...
Digital Signature...
Project - ImportDXF
ImportDXF (C:\Users\Dougal.Harris\Documents\I
Modules
ModImportDXF
Properties - ModImportDXF
ModImportDXF Module
Alphabetic
Categorized
(Name) ModImportDXF
References - ImportDXF
OK
Cancel
Browse...
Priority
Help

Available References:

- [ ] AVFControlLib
- [ ] azroles 1.0 Type Library
- [ ] BdeUISrv 1.0 Type Library
- [ ] Bentley Fitting 1.1 Automation Library
- [ ] Bentley Link 1.1 Automation Library
- [ ] Bentley MicroStation DGN 8.9 Object Library
- [✓] Bentley Modeler 1.1 Automation Library
- [ ] Bentley Motions 1.1 Automation Library
- [ ] Bentley Resistance 1.1 Automation Library
- [ ] Bentley RSS Feed Reader
- [ ] Bentley SpatialNavigatorCtl 1.4 Type Library
- [ ] Bentley Stability 1.1 Automation Library
- [ ] Bentley Structure 1.1 Automation Library
- [ ] Bentley VPP 1.1 Automation Library

Bentley Modeler 1.1 Automation Library

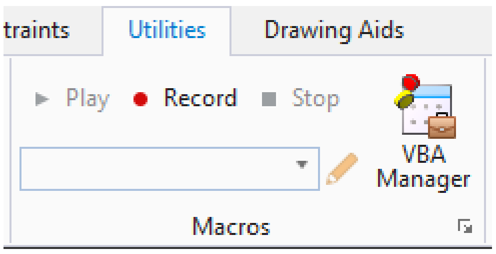
Location: C:\Program Files\Bentley\Offshore\MAXSURF CONNECT Editio

Language: Standard

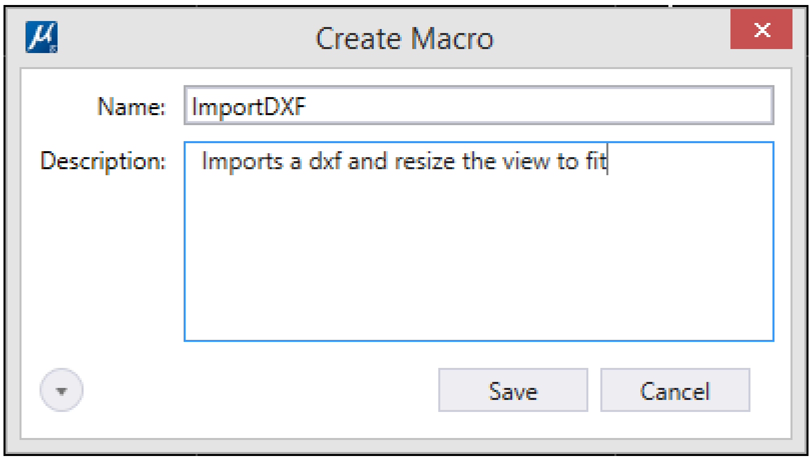
Once the reference has been established all the BentleyModeler COM objects and functionality will be available from within the Microstation VBA editor.

‘Promoting’ to VBA code

If you are unsure of the exact Microstation VBA command for certain functions, you can record a macro and then promote the macro to VBA code for viewing. Recording Macros in Microstation is similar to recording marcos in Microsoft Office applications such as Excel. Simply hit the Record button in the Macros group of the Utilities tab:



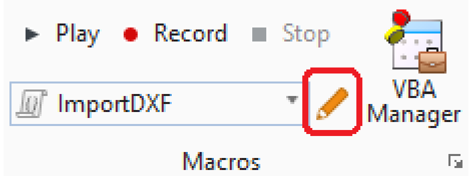
Once you have finished the required steps, hit the stop button, you will be asked to name the macro:

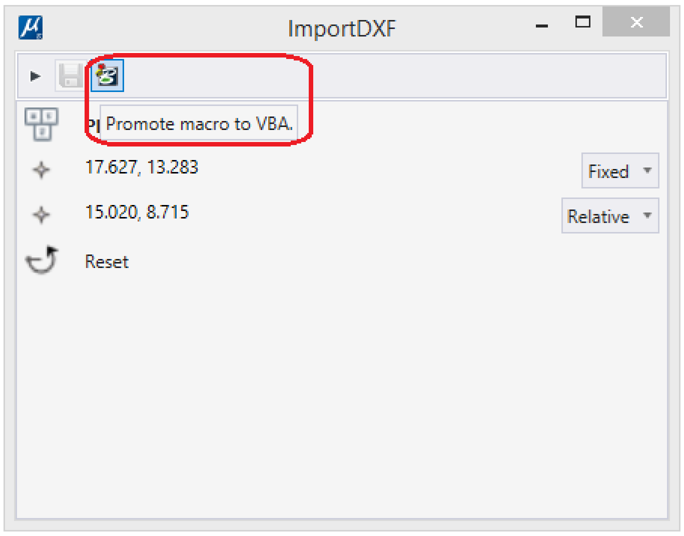


Name: ImportDXF

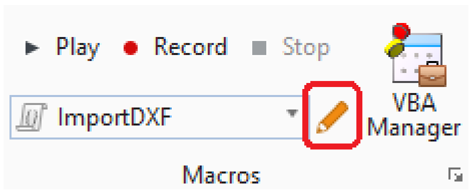
Description: Imports a dxf and resize the view to fit

Give it a meaningful name and description. You can then edit the macro :

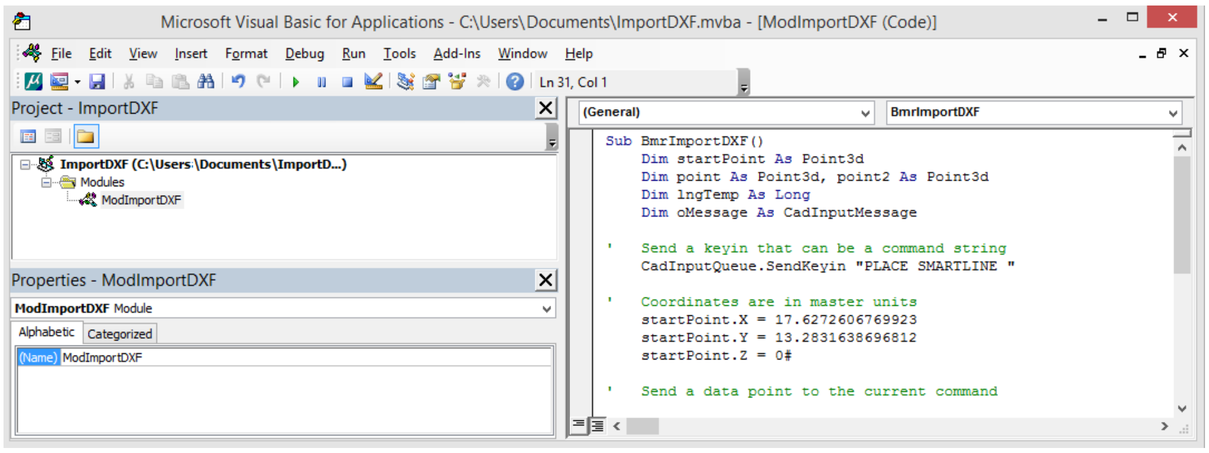




From here the marco can be 'promoted' to VBA code. Once 'promoted' click on the edit button next to the macro again:



To bring up the Microstation VBA editor and view the code:


Microsoft Visual Basic for Applications - C:\Users\Documents\ImportDXF.mvba - [ModImportDXF (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 31, Col 1
Project - ImportDXF
ImportDXF (C:\Users\Documents\ImportD...)
Modules
ModImportDXF
Properties - ModImportDXF
ModImportDXF Module
Alphabetic
Categorized
(Name) ModImportDXF
(General)
BmrImportDXF

```
Sub BmrImportDXF()
    Dim startPoint As Point3d
    Dim point As Point3d, point2 As Point3d
    Dim lngTemp As Long
    Dim oMessage As CadInputMessage

'   Send a keyin that can be a command string
    CadInputQueue.SendKeyin "PLACE SMARTLINE "

'   Coordinates are in master units
    startPoint.X = 17.6272606769923
    startPoint.Y = 13.2831638696812
    startPoint.Z = 0#

'   Send a data point to the current command
```

Further Microstation VBA resources

Microstation has quite an extensive library of VBA and Macro resources available on their Bentley Communities, Microstation Wiki, Developing Code in VBA webpage (http://communities.bentley.com/products/microstation/w/microstation__wiki/developing-code-in-vba).

Chapter 7 Examples

This chapter presents a number of example VBA-scripts to show what you can do with Maxsurf Modeler automation. The first three examples show the use of various objects, properties and methods of the Maxsurf Modeler Object Model, such as editing grid lines, performing parametric transformations and moving control points. The next example also uses list objects to edit surfaces in a chined hull vessel. Finally Microstation macros are used to control Maxsurf Modeller and load a 2D lines plan.

- Modifying Grid Lines in Excel

  Using the Grids object to create, delete and move gridlines through Excel.

- Creating a Systematic Series

  Performing parametric transformations, hydrostatic calculations and file handling through Excel to create a series of hull forms with systematically varying parameters.

- Blending Hull Forms

  Moving Control Points and performing parametric transformations to create a hull, blended from two other hull forms.

- Creating a Chined Hull Vessel

  Altering the size and shape of a simple chined hull according to parameters specified in an Excel sheet. The Example could be used as an effective starting point for designs.

- Using Microstation (VBA) Macros to control Maxsurf Modeler

  Use Microstation automation to open a design in Maxsurf Modeler, export of the orthogonal views to a dxf file. Import the dxf files into Microstation and arrage in lines plan format.

Note

If you have skipped directly to this section, please ensure that your macro security settings are not set to high (disabling macros). And if there are any problems please review Maxsurf Modeler Does not Appear in the References Dialogon page 13.

Exercises:

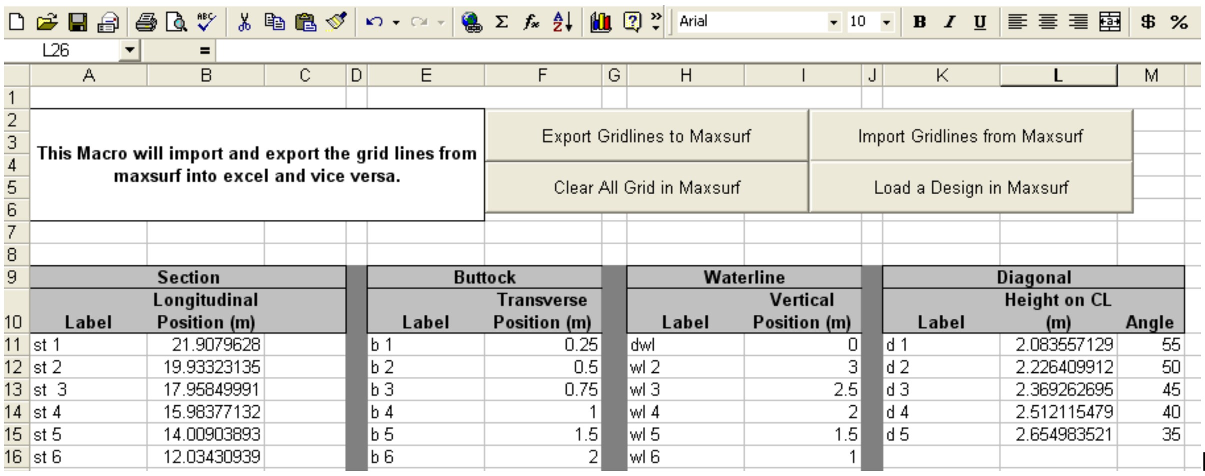
Some examples also contain exercises to show other uses of Automation. The exercises are designed to build on the example file, by extending its uses or altering the file to another use. These exercises have not been worked through for this manual, but all are possible. The exercises are:

- Exercise 1 - Inserting Section Lines for Hydromax
- Exercise 2– Optimising Code for Faster Execution
- Exercise 3 – Placement of Containers in a Container Ship
- Exercise 4 – Generic Code for Hull Blending
- Exercise 5 – Finding the Displacement and Immersion for the Chined Hull

Modifying Grid Lines in Excel

Gridlines are useful to show the shape of hull forms. They can be added, deleted and moved from inside of Maxsurf Modeler through the Grid Space Dialog (Main Menu | Data | Grid Spacing . .). Grid lines can also be moved in through the Automation interface. This example shows how to replicate the Grid Space Dialog in Excel so that the Grid can be edited from Excel. The Excel file detailed in this example can be found in C:\Program Files\Bentley\Offshore\Maxsurf CONNECT Edition VXX\Automation Samples\Modeler\Editing Gridlines.xls (where XX is the version number), or in the Maxsurf Modeler install directory.

- Open the Excel spreadsheet


Arial
10
L26
Export Gridlines to Maxsurf
Import Gridlines from Maxsurf
Clear All Grid in Maxsurf
Load a Design in Maxsurf

This Macro will import and export the grid lines from maxsurf into excel and vice versa.

Section		Buttock		Waterline		Diagonal		
Label	Longitudinal Position (m)	Label	Transverse Position (m)	Label	Vertical Position (m)	Label	Height on CL (m)	Angle
st 1	21.9079628	b 1	0.25	dwl	0	d 1	2.083557129	55
st 2	19.93323135	b 2	0.5	wl 2	3	d 2	2.226409912	50
st 3	17.95849991	b 3	0.75	wl 3	2.5	d 3	2.369262695	45
st 4	15.98377132	b 4	1	wl 4	2	d 4	2.512115479	40
st 5	14.00903893	b 5	1.5	wl 5	1.5	d 5	2.654983521	35
st 6	12.03430939	b 6	2	wl 6	1			

Figure 12 Layout For the Editing grid lines Example, Command Buttons will import, export and clear the grid lines.

There are four procedures in this Spreadsheet; linked to the four Command buttons. The buttons will:

- Load a Design into Maxsurf Modeler
- Import the current grid lines from Maxsurf Modeler
- Clear all the grid lines in Maxsurf Modeler
- Export the grid lines in Excel, back into Maxsurf Modeler

The first procedure has been used already in this manual to open files. The procedure is described on page 30 and won't be described again now.

- Open a sample design, using the button on the Excel spreadsheet

If the Maxsurf Modeler Application does not start, or the design does not load, refer to Maxsurf Modeler Does not Appear in the References Dialogon page 13.

Get Grid Lines from Maxsurf Modeler

This procedure uses a number of loops (one for each set of grid lines) to determine the label, position and angle (appropriate only for diagonals) of each grid line.
The GetGridLine method reads in the grid line type and the index, it returns the label, position and angle (where appropriate):

```
        msApp.Design.Grids.GetGridLine msGTSections, s, Sect_Label,
sVal, sAngle
```

The Label and position values can then be written into Excel using the Range or Cells properties:

```
        Range("A" & s + 10) = Sect_Label
        Range("B" & s + 10) = sVal
```

When placed in a For Next statement with s values from 1 to the number of sections (using LineCount property), this will get all the Grid lines in the section direction.

We also need to determine the split section line, this is the point where section lines aft of this one, will show on the portside of the model, forward will show on starboard in body plan view. It is found in two lines of code:

```
    i = msApp.Design.Grids.SectionSplit
    Range("C" & i + 10) = "Split"
```

This could be further shortened to:

```
    Range("C" & msApp.Design.Grids.SectionSplit + 10) = "Split"
```

Combining four loops to find the grid line locations and the code to find the split line gives this completed code to get the grid lines.

```
Private Sub GetGridLines_Click()
'Get the locations of the grid lines from Maxsurf Modeler into Excel


    Dim Sect_Label As String
    Dim sVal As Double
    Dim sAngle As Double
    Dim Buttock_Label As String
    Dim bVal As Double
    Dim bAngle As Double
    Dim Waterline_Label As String
    Dim wVal As Double
    Dim wAngle As Double
    Dim Diagonal_Label As String
    Dim dVal As Double
    Dim dAngle As Double

    'This loop gets the section grid line locations
    For s = 1 To msApp.Design.Grids.LineCount(msGTSections)

        'The following line gets the section line data.
        'The input information is msGTSections (get sections) and s
(the index value)
        'The   method   returns   Sect_Label   (the   section   Label),
longitudinal position and the angle
        msApp.Design.Grids.GetGridLine msGTSections,  s,  Sect_Label,
sVal, sAngle
        Range("A" & s + 10) = Sect_Label
        Range("B" & s + 10) = sVal
    Next s

    'Write Split in the Split Section Line
    i = msApp.Design.Grids.SectionSplit
    Range("C" & i + 10) = "Split"



    'This loop gets the buttock line locations
    For B = 1 To msApp.Design.Grids.LineCount(msGTButtocklines)
        msApp.Design.Grids.GetGridLine     msGTButtocklines,     B,
Buttock_Label, bVal, bAngle
        Range("E" & B + 10) = Buttock_Label
        Range("F" & B + 10) = bVal
    Next B

    'This loop gets the Waterline locations
    For w = 1 To msApp.Design.Grids.LineCount(msGTWaterlines)
        msApp.Design.Grids.GetGridLine      msGTWaterlines,      w,
Waterline_Label, wVal, wAngle
        Range("H" & w + 10) = Waterline_Label
        Range("I" & w + 10) = wVal
    Next w

    'This loop gets the diagonal line locations
    For D = 1 To msApp.Design.Grids.LineCount(msGTDiagonals)
        msApp.Design.Grids.GetGridLine      msGTDiagonals,       D,
Diagonal_Label, dVal, dAngle
        Range("K" & D + 10) = Diagonal_Label
        Range("L" & D + 10) = dVal
        Range("M" & D + 10) = dAngle
    Next D

    msApp.Refresh
```

```
End Sub
```

Set the Grid Lines in Maxsurf Modeler

The concept of exporting grid lines into Maxsurf Modeler is the same as importing them. However, the method you have to use is reasonably different. We no longer have a definitive number of entries, so we cannot do a For Next statement through all of the data. Instead we need to use a Do While Loop, testing each time whether the target cell is empty or not.

There are two methods for defining grid line locations, SetGridLines and AddGridLines. Adding grid lines will create new grid lines, where setting grid lines will update a current grid line to new data.

When exporting the grid lines to Maxsurf Modeler we need to use the SetGridLines method to move all existing gridlines to new locations, then if there are insufficient grid lines, we need to add the remaining grid lines using the AddGridLines method.
The loop for creating the section lines is:

```
s = 1
Do While Range("B" & s + 10) <> "" 'While the position column has an
entry in it
   Sect_Label = Range("A" & s + 10) 'Set the values for the label,
position and angle
   sVal = Range("B" & s + 10)

   'This If statement will overwrite existing sections, or add more
if there aren't enough
   If s <= msApp.Design.Grids.LineCount(msGTSections) Then
     'All 5 variables are inputs here, they are the same definition
as in GetGridLines
     msApp.Design.Grids.SetGridLine  msGTSections,  s,  Sect_Label,
sVal, 0
   Else
      msApp.Design.Grids.AddGridLine msGTSections, Sect_Label, sVal,
0
   End If

   If Range("C" & s + 10) = "Split" Then
      msApp.Design.Grids.SectionSplit = s
   End If

   s = s + 1
Loop
```

This shows the If Then Else statement testing if s (the section index number) is less than or equal to the number of section lines. If it is, it will set the section lines to a new value, if it isn't, it will add new section lines.

The complete code for setting all the grid lines is as follows;

```
Private Sub SetGridLines_Click()
'Set the locations of the grid lines in Maxsurf Modeler from Excel

    Dim Sect_Label As String
    Dim sVal As Double
    Dim Buttock_Label As String
    Dim bVal As Double
    Dim Waterline_Label As String
    Dim wVal As Double
    Dim Diagonal_Label As String
    Dim dVal As Double
    Dim dAngle As Double
    'This Do While loop sets the section lines in Maxsurf Modeler
    s = 1
    Do While Range("B" & s + 10) <> "" 'While the position column
has an entry in it
        Sect_Label = Range("A" & s + 10) 'Set the values for the
label, position and angle
        sVal = Range("B" & s + 10)

        'This if loop will overwrite existing sections, or add more
if there aren't enough
        If s <= msApp.Design.Grids.LineCount(msGTSections) Then
            'All 5 variables are inputs here, they are the same
definition as in GetGridLines
            msApp.Design.Grids.SetGridLine     msGTSections,     s,
Sect_Label, sVal, 0
        Else
            msApp.Design.Grids.AddGridLine msGTSections, Sect_Label,
sVal, 0
        End If

        If Range("C" & s + 10) = "Split" Then
            msApp.Design.Grids.SectionSplit = s
        End If

        s = s + 1
    Loop

    'This loop gets the buttock lines
    B = 1
    Do While Range("F" & B + 10) <> ""
        Buttock_Label = Range("E" & B + 10)
        bVal = Range("F" & B + 10)

        If B <= msApp.Design.Grids.LineCount(msGTButtocklines) Then
            msApp.Design.Grids.SetGridLine   msGTButtocklines,   B,
Buttock_Label, bVal, 0
        Else
            msApp.Design.Grids.AddGridLine        msGTButtocklines,
Buttock_Label, bVal, 0
        End If
        B = B + 1
    Loop

    'This loop gets the waterlines
    w = 1
    Do While Range("I" & w + 10) <> ""
        Waterline_Label = Range("H" & w + 10)
        wVal = Range("I" & w + 10)
        If w <= msApp.Design.Grids.LineCount(msGTWaterlines) Then
            msApp.Design.Grids.SetGridLine    msGTWaterlines,    w,
Waterline_Label, wVal, 0
        Else
            msApp.Design.Grids.AddGridLine          msGTWaterlines,
Waterline_Label, wVal, 0
        End If
        w = w + 1
    Loop

    'This loop gets the Diagonal Lines
    D = 1
    Do While Range("L" & D + 10) <> ""
        Diagonal_Label = Range("K" & D + 10)
        dVal = Range("L" & D + 10)
        dAngle = Range("M" & D + 10)
        If D <= msApp.Design.Grids.LineCount(msGTDiagonals) Then
            msApp.Design.Grids.SetGridLine    msGTDiagonals,     D,
Diagonal_Label, dVal, dAngle
```

```
        Else
            msApp.Design.Grids.AddGridLine            msGTDiagonals,
Diagonal_Label, dVal, dAngle
        End If
        D = D + 1
    Loop

    msApp.Refresh
End Sub
```

Clear all Grid Lines in Maxsurf Modeler

This procedure simply deletes all the grid lines in Maxsurf Modeler; it uses one method, applied to each type of grid line.

The code reads:

```
Private Sub ClearAllGridinMS_Click()
    'This deletes all existing grid lines in Maxsurf Modeler when
the "Clear All Grid in Maxsurf Modeler" button is Clicked
    'Each Grid direction is deleted individually
    msApp.Design.Grids.DeleteAllLines (msGTButtocklines)
    msApp.Design.Grids.DeleteAllLines (msGTDiagonals)
    msApp.Design.Grids.DeleteAllLines (msGTSections)
    msApp.Design.Grids.DeleteAllLines (msGTWaterlines)
    'Refresh must be included to update the views in Maxsurf Modeler
to show no grid lines
    msApp.Refresh

End Sub
```

Exercise 1 - Inserting Section Lines for Hydromax

The aim of this exercise is to create an automation program to add section lines to a design according to the arrangement of control points. The section lines are to be more densely located around areas with greater curvature and more spaced around less curved areas.

Background Information

When opening a Maxsurf Modeler design in Hydromax, there is an option for using the Maxsurf Modeler section lines instead of evenly spaced section lines for its calculations. The advantage of this is that extra section lines can be added around areas with sudden change in section, such as around bow thrusters and bulbous bows.

Process

- Start by creating 50 evenly spaced section lines, between the aft most and forward most control point.
- To test for sudden changes in shape, find the 3 dimensional distance to the next nearest control point, compared to the two dimensional distance. If the difference is above a certain multiple (√2 gives 45 degrees), then add a section line either side of the control points.

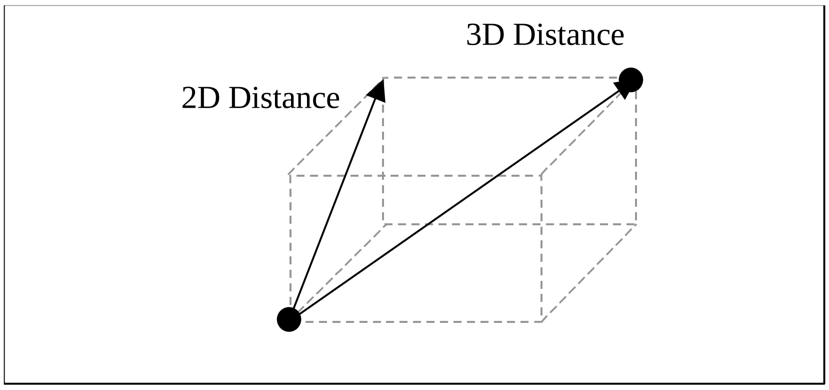


Figure 13 Two and Three Dimensional Distances

- Test if there is a Sectionline already present within a tolerance of where the new line will go. If there is, do not add it, or move it else where

Constraints

- Hydromax can handle a Maximum of 200 section lines, therefore you need to test that the number of lines created is less than this.

Hints:

Instead of placing the Section line inbetween the control points, it is better to place a point either side of the control points. This way, Hydromax will have a definite location either side of the sudden change where it knows that the hull will pass through.

Creating a Systematic Series

Systematic series are useful when optimising a hull form. Maxsurf Modeler includes a powerful tool for parametrically transforming a hull form to have slightly different parameters. In this example we develop a program to create a series of eight hull forms with varying parameters. The hull forms could then be imported into Seakeeper or Hullspeed for analysis and comparison. In the future, automation will also be available for Seakeeper and Hullspeed. This will allow you to fully exploit the advantages of using automation.

This example has been designed to work for any design, however, parametric transformations work best on plain hull forms, rather than designs with appendages. For best results with designs including appendages, turn the appendage surfaces off before transforming.

The example is divided into several sub procedures, some of which are activated from buttons in the Excel sheet; some are activated by calls from other procedures. An overview of the process used by this macro is shown in Figure 14

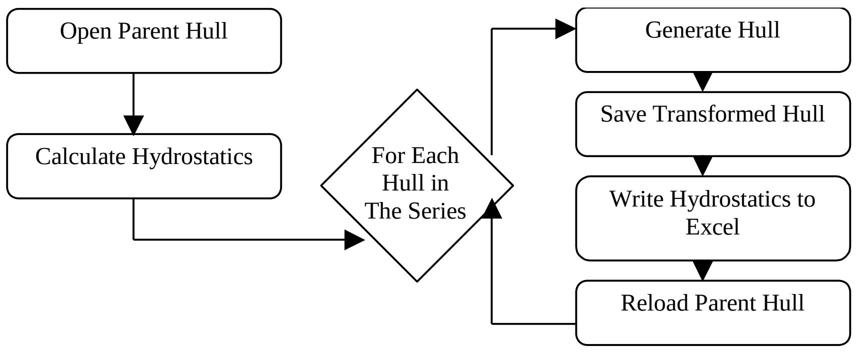


Figure 14 The Process for Creating A Systematic Series

Resources
The Excel File with this example is located in C:\Program Files\Maxsurf Modeler\ Automation Samples\Maxsurf Modeler\SystematicSeries.xls, or in the Maxsurf Modeler install directory.

The Excel file has some headings and formatting included already, creating the layout will not be discussed in this Manual.

Opening the Parent Hull Form

The Parent hull form is loaded into Maxsurf Modeler using a dialog box to allow the user to select the file. The dialog box is part of the Excel method Application.GetOpenFilename. The procedure is activated using a button in the spreadsheet.

To create buttons, use the Command Button on the Control Toolbox Menu bar. The most left hand button toggles between *design mode* (for editing buttons) and *exit design mode*, for executing buttons.

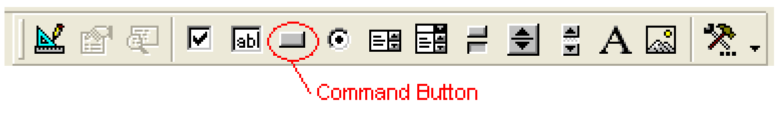


Figure 15 The Command Button on the Control Toolbox Menu Bar.

The Command Button is linked to a procedure by changing the name to something appropriate, and then creating a procedure of the same name followed by _Click(). For Example, the Button to Load the parent hull form is named **GetParentHull**, the procedure is called

```
Sub GetParentHull_Click()
```

The name of the Command button can be set in Excel using the properties dialog, accessed from the Control Toolbox menu. The properties can also be set from the VB editor.

The Open File Dialog box is brought up using a similar code to that shown on page 30, Basic Operations. The code used in this example writes the filename to a cell in Excel, so that other procedures can easily access the file name.

```
    Dim Filter As String
    Dim FileName As String

    Filter = "Maxsurf Modeler Design File (*.msd), *msd"
    'The Filter only allows certain file types to be loaded
    FileName = Application.GetOpenFilename(Filter, , "Open Maxsurf
Modeler File", , False)
```

The filter defines the visible file types in the Open File dialog box. This code will store the name of the File to be opened in the Variable FileName. If the Cancel button is pressed in the dialog, the word “FALSE” will be stored in the variable FileName.

Before we open the File, we need to check for an error (For example, when the cancel button has been clicked). If it has, we can’t open the file. The following section of code will load the design if one has been specified, or exit if not.

```
    If FileName = "False" Then
        Exit Sub
    Else
        Range("D7") = FileName
        msApp.Design.Open FileName, False, False
        msApp.Refresh
    End If

End Sub
```

These segments of code, when combined, will open a Maxsurf Modeler File, according to what is selected in the Dialog box. This is a useful piece of generic code that can be applied to most automation files.

Creating a Systematic Series

To create a systematic series, we want to take the current hydrostatics and transform the hull to have slightly different parameters. So to create the series, we need to read in the current Hydrostatic Data, define which parameters we would like to change and how much we would like to change them and then perform the transformation.

The macro that creates the series uses a set of nested For Next loops to test all the options. In the middle of the nested For Next loops is a call to ParaTransform. This sub procedure reads in the target Cb, LWL and Draft values and then performs the parametric transformation. The Call to ParaTransform will take place 8 times, creating the eight different hull forms.

```
Sub CreateSeries_Click()

    Dim msDesign As Maxsurf Modeler.Design
    Set msDesign = msApp.Design

    Dim Cb As Double
    Dim LWL As Double
    Dim ImmersedDepth As Double

    Call OpenFile
    Call CalcHydrostatics("L10")

    'Chr(67) represents the letter C
    z = 67

    For i = 0 To 1
        Cb = Cells(12, 12) + Cells(11, 8 + i)
        For j = 0 To 1
            LWL = Cells(11, 12) + Cells(12, 8 + j)
            For k = 0 To 1
                ImmersedDepth = Cells(15, 12) + Cells(13, 8 + k)
                Call ParaTransform(Cb, LWL, ImmersedDepth)
                Call CalcHydrostatics(Chr(z) & "28")
                z = z + 1
            Next
        Next
    Next
    MsgBox "Done"
End Sub
```

The cells referenced in this code are for the Hydrostatic data for the parent hull form, and the amount by which the hydrostatic data will change. The code makes calls to three different procedures, namely:

- OpenFile
- CalcHydrostatics
- ParaTransform

The call to CalcHydrostatics is made in two different locations. The call reads in a variable that specifies a cell in the worksheet. This cell is the upper most cell for the list of hydrostatic data to be written in.

This code could easily be modified to have more parameters or more hull forms in the series than one at each corner of the parametric space.

Note:

The hydrostatic calculations and parametric transformation include visible surfaces only. After the call to open the Parent File, a line of code could be included, to turn visibility of certain surfaces off.

Calculating the Hydrostatics

The use of Hydrostatic Data has been described in this manual already, for more information on using it, see Tutorial Part 4: Calculating the Hydrostatics on page 26. The code in example has been designed to be applied anywhere in the spreadsheet, in this example it will be executed nine times.

The code makes a particular cell active (as specified by the input string), and then writes data to all the other cells relative to the active cell. The code reads as follows:

```
Sub CalcHydrostatics(StartCell As String)
    'Calculate the Parent Hydrostatics and Display in the page

    Dim msDesign As Maxsurf Modeler.Design
    Set msDesign = msApp.Design

    msApp.Trimming = True
    With msDesign.Hydrostatics
        msDesign.Hydrostatics.Calculate 1025, 2

        'Select the top Cell of the row
        Range((StartCell)).Activate
        'Write Contents into Cells, relative to the active cell
        'Giving relative positions makes alterations to the layout
simple
        ActiveCell = .Displacement
        ActiveCell.Offset(1, 0) = .LWL
        ActiveCell.Offset(2, 0) = .Cb
        ActiveCell.Offset(3, 0) = .BeamWL
        ActiveCell.Offset(4, 0) = .Draft
        ActiveCell.Offset(5, 0) = .ImmersedDepth
        ActiveCell.Offset(6, 0) = .Cm
        ActiveCell.Offset(7, 0) = .Cp
        ActiveCell.Offset(8, 0) = .Cwp
        ActiveCell.Offset(9, 0) = _
            (msDesign.FrameOfReference.fwdPerp - .LCB) / .LWL
        ActiveCell.Offset(10, 0) = _
            (msDesign.FrameOfReference.fwdPerp - .LCB) / .LWL
        ActiveCell.Offset(11, 0) = .WSA


    End With

End Sub
```

Because all cell references are relative, it can be used for any location in the work sheet. The code will also not require updating if changes are made to the worksheet layout.

Performing the Parametric Transformation,

The parametric transformation is done in a sub procedure that requires three variable inputs, the Block Coefficient, the Waterline Length and the Immersed Depth. The call to this procedure requires these three variables, as specified in the name of the transformation procedure:

```
Call ParaTransform(Cb, LWL, Draft)
```

The sub procedure name includes the three variable inputs and their type in the brackets.

```
Private Sub ParaTransform(Cb As Double, LWL As Double, Draft As
Double)
```

The code that executes the transformation reads as follows:

```
Private Sub ParaTransform(Cb As Double, LWL As Double, Draft As
Double)
```

```
    Dim msDesign As Maxsurf Modeler.Design
    Set msDesign = msApp.Design

    Call OpenFile

    msDesign.Hydrostatics.Transform _
        Range("L18"), _
        Cb, _
        Range("L15"), _
        Range("L10"), _
        Draft, _
        Range("L13"), _
        LWL, _
        True, _
        True, _
        False, _
        False, _
        True
```

The Range properties refer to the cells containing the Hydrostatic data in the Excel sheet. The second half of this procedure saves the transformed hull into a specified directory.

Saving the Transformed Hull

To save the file into a specified directory, we need to have that directory specified and have a different name for each different hull. The obvious name for each hull in the series is the values of the parameters that have been modified.

If the save directory has not been specified, a call is made to the sub procedure GetSaveFolder_click. This procedure will specify the directory to be saved into. Otherwise, the hull will be saved into the specified directory.

```
    'Check if save directory exists.
    'Separate the File Name from the end
    k = InStrRev(Range("D8"), "\", -1, vbTextCompare)
    If k < 1 Then 'No directory is specified
        Call GetSaveFolder_click
        k = InStrRev(Range("D8"), "\", -1, vbTextCompare)
    End If

    FolderPath = Left$(Range("D8"), k - 1)
    'check if the save folder exists, if not, get a new one
    If Dir(FolderPath, vbDirectory) = "" Then
        Call GetSaveFolder_click
    End If

    msDesign.SaveAs  Range("D8")  &  "_"  &  Round(Cb,  1)  &  "_"  &
Round(LWL, 1) & "_" & Round(ImmersedDepth, 1) & ".msd", True
    'msDesign.ExportIGES  Range("D8")  &  Round(Cb,  1)  &  "_"  &
Round(LWL, 1) & "_" & Round(Draft, 1) & ".igs"

    msApp.Refresh

End Sub
```

The code for exporting IGES files has been included but commented out. Depending on the purpose of the files, IGES files may be more appropriate.

Reloading the Parent Hull

If a valid parent hull form has already been selected then it is unnecessary to bring up the Dialog box again. This procedure checks for the existence of a valid file by testing if the file name is present in the specified directory.

```
Sub OpenFile()
    'Opens the named File

    Dim FilePathName As Variant
    Dim FilesInFolder As Variant
    Dim Path As Variant
    Dim PathSegments As Variant

    FilePathName = Range("D7")

    'Path Segments represent each folder name of the Path and File
Name
    PathSegments = Split(FilePathName, "\", , vbTextCompare)

    'If there is are no "\"
    If UBound(PathSegments) < 1 Then
        Call GetParentHull_Click
        Exit Sub
    End If

    'Path is the Folder Path
    Path = PathSegments
        ReDim Preserve Path(UBound(Path) - 1)
    Path = Join(Path, "\")
    'Join rejoins all the split segments.

    FileName = PathSegments(UBound(PathSegments))

    'All the design files in the Path Folder
    FilesInFolder = Dir(Path & "\" & "*.msd")

    'This loops through the names of all the files in the folder
    'and compares to the FileName to be opened
    While FilesInFolder <> ""
        If FileName = FilesInFolder Then
            'If the file is in the folder, it is opened
            msApp.Design.Open FilePathName, False, False
            Exit Sub
        End If
        FilesInFolder = Dir()
    Wend

    Call GetParentHull_Click

End Sub
```

If the parent hull file is not in the specified folder, this procedure makes a Call to another procedure, namely GetParentHull_Click. The Call method will run the named procedure and then return to this procedure.

Exercise 2– Optimising Code for Faster Execution

Speed of execution is often important in computer-processed tasks. The Current file has many inefficiencies that could be improved for faster execution. One of the main inefficiencies is the reloading of the parent hull form for each iteration.

Try improving the execution speed by making these alterations.

- Modify the code so that the control point data for the parent hull is read into an array or a set excel cells. After each iteration move the control points back to their original locations, using data stored in the control point array.
- Create a surface list and access the surfaces using the list object.

Hints

To find how long each section of code is taking to execute, use the VBA.Timer property. An example is shown below.

```
Sub UsingTimer()
Dim TimeIn As Long

TimeIn = VBA.Timer

'-------------
'Code Segment Here
'-------------

MsgBox "Execution Time was " & VBA.Timer - TimeIn

End Sub
```

Exercise 3 – Placement of Containers in a Container Ship

A large number of cargo vessels need to carry the cargo in standard sized units, such as fish boxes or sea containers. An early design decision is to determine the stacking arrangement, the number of bays, rows and tiers, for the cargo to be arranged in.

Taking a container ship as an example, we can increase the vessel width by a one container, and then reduce the length accordingly to maintain the same number of container bays. Varying the container arrangement will alter the initial vessel cost, lightship weight, resistance, stability and more.

The aim of this exercise is to create a series of vessels of similar hull shape, with different breadths and widths to carry different arrangements of containers.

Background Information

- Use the Sample Design of a container ship as a base boat. C:\Program Files\Maxsurf Modeler\Sample Designs\Ships\Containership_1Surface.msd
- The vessel is designed to carry 640 containers in its midbody, arranged in 8 tiers, 8 bays along and 10 rows across.
- For a standard 40 ft container allow dimensions of 12.9(L) x 2.52(W) x 2.6m(H)

Assumptions

- Assume that the vessel displacement is independent of the dimension. This could be accounted for at a later stage by using a displacement, non-dimensionalised by the length x breadth x height.
- Only account for the number of containers in and above the midbody. Assume the number of containers elsewhere remains constant.

Process

- Open a parent design
- Create a series of container arrangements, maintaining the same number of containers, but varying the number of rows, bays and tiers.
- For each container arrangement;

  Scale the vessel breadth to suit the number of rows

  Scale the vessel length to the number of bays

  Parametrically transform the hull, altering the depth and coefficients to return to the original displacement.

  Save the hull with an appropriate file name.

Reload the parent design.

- The design can then be analysed for resistance, sea keeping, cost and stability, so to find an optimised design.

The process could be made far more complicated, accounting for the change in displacement according to the dimensions of the hull, propeller immersion with changes in resistance and more.

Blending Hull Forms

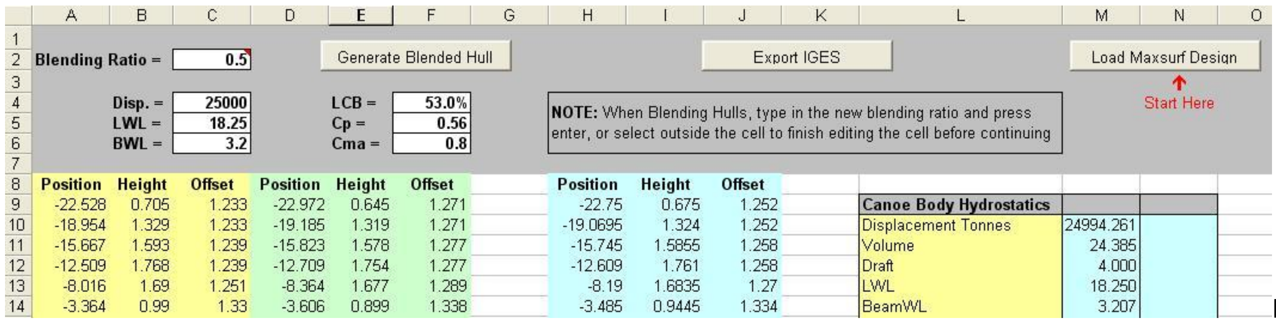
Maxsurf Modeler provides a function to parametrically modify a design to a have a different length, draft or block coefficient. This maintains the same body shape, but alters the dimensions. Through Maxsurf Modeler Automation, we can do the opposite and alter a hull shape while maintaining the existing parameters.

Resources

The spreadsheet associated with this example is located in C:\Program Files\Maxsurf Modeler 14\Automation Samples\Maxsurf Modeler\Blending Hulls.xls or in the directory in which Maxsurf Modeler is installed. The example is based around the racing yacht hull form in C:\Program Files\Maxsurf Modeler\Sample Designs\SailingYachts\ AC hull_7Surface.msd.

Using the Example File

Opening up the spreadsheet shows three lists of coordinates, these are the control point locations for the original hull (yellow), the modified hull shape (green) and the blended hull (blue). The blended hull control point coordinates are proportional between the two other coordinate sets, in the ratio specified by the blending ratio (Cell C2), ranging from 0 to 1 (entirely modified shape, to entirely original shape)



Blending Ratio = 0.5

Disp. =	25000
LWL =	18.25
BWL =	3.2
LCB =	53.0%
Cp =	0.56
Cma =	0.8

NOTE: When Blending Hulls, type in the new blending ratio and press enter, or select outside the cell to finish editing the cell before continuing

Position	Height	Offset	Position	Height	Offset	Position	Height	Offset
-22.528	0.705	1.233	-22.972	0.645	1.271	-22.75	0.675	1.252
-18.954	1.329	1.233	-19.185	1.319	1.271	-19.0695	1.324	1.252
-15.667	1.593	1.239	-15.823	1.578	1.277	-15.745	1.5855	1.258
-12.509	1.768	1.239	-12.709	1.754	1.277	-12.609	1.761	1.258
-8.016	1.69	1.251	-8.364	1.677	1.289	-8.19	1.6835	1.27
-3.364	0.99	1.33	-3.606	0.899	1.338	-3.485	0.9445	1.334

Canoe Body Hydrostatics	
Displacement Tonnes	24994.261
Volume	24.385
Draft	4.000
LWL	18.250
BeamWL	3.207

F
igure 16 The Blending Hulls Example Spreadsheet. The three columns of coordinates are control points for the two parent hulls (left) and the blended hull (right)

To create a blended hull form;

- Load the design into Maxsurf Modeler if it is not already loaded, by clicking on the "Load Maxsurf Modeler Design" button.
- Type in a blending ratio between 0 and 1 into Cell C2 (0 is a good starting point, to show the extreme shape)
- Press enter or select another cell to finish editing cell C2
- Select the "Generate Blended Hull" Button.

View the modified hull form in Maxsurf Modeler. Creating Hulls at the two extremes (0 and 1) gives the following two hull shapes

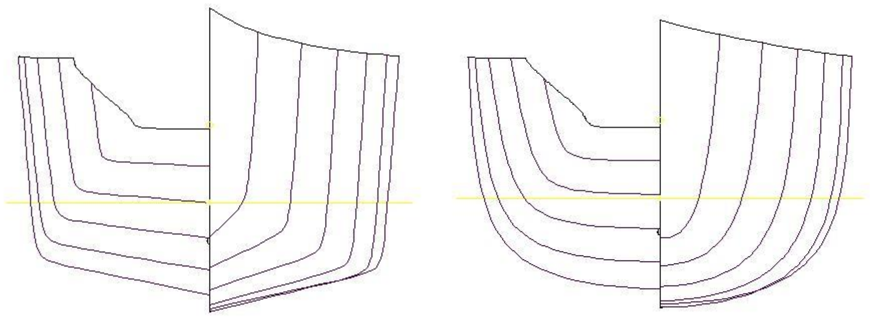


Figure 17 The two extremes for the blended hulls, Blending ratio 0 on the Left and blending ratio 1 on the Right

Code in the Example File

The majority of the code in this example has been described previously in this manual, so only the code regarding the blending of hulls will be described below.

Most of the work in this example takes place outside of Macros. The values for the blended hulls' control points are calculated using formulas in the Excel cells (select one of the cells to see the formula). The control point data for the two parent hull forms has been manually inserted into Excel using cut and paste from Maxsurf Modeler's control point's window, this process could have been automated too.

The following code makes up the cmdMoveControlPoints_Click() procedure that is run when the "Generate Blended Hulls" command button is clicked. The code has been broken up and annotated throughout.

```
Sub cmdMoveControlPoints_Click()
    Dim NumRows As Long
    Dim NumCols As Long
    Dim TheSurf As Surface

    For i = 2 To msApp.Design.Surfaces.Count
        msApp.Design.Surfaces(i).Visible = False
    Next i
```

Turning all surfaces except the hull surface is a prelude to calculating the Hydrostatics. The hydrostatic data is calculated only on the visible surfaces.

```

    msApp.Design.Surfaces(1).ControlPointLimits NumRows, NumCols
    n = 9
    For R = 1 To NumRows
        For C = 1 To NumCols
            msApp.Design.Surfaces(1).SetControlPoint R, C, Cells(n,
8), Cells(n, 9), Cells(n, 10)
            n = n + 1
        Next C
    Next R
```

The section above sets the new control point locations, the first set of coordinates is in row 9, hence the n = 9 before the For Next statement. Each successive loop increases n by 1, to move to the next set of coordinates.

```
    msApp.Trimming = True
    msApp.Design.Hydrostatics.Transform Cells(4, 6), Cells(5, 6),
Cells(6, 6), Cells(4, 3), 0.953, Cells(6, 3), Cells(5, 3), True,
True, True, False, False
```

This method performs a parametric transformation on the hull; this ensures that the blended hull form has the same characteristics as the parent hulls. Trimming has been turned on so that the hydrostatics will not be not affected. The following statement calculates the new hull's hydrostatics and stores them in column 13 (cells M10 to M34)

```
    msApp.Design.Hydrostatics.Calculate 1025, 0

    Cells(10, 13) = msApp.Design.Hydrostatics.Displacement
    Cells(11, 13) = msApp.Design.Hydrostatics.Volume
    Cells(12, 13) = msApp.Design.Hydrostatics.Draft
    Cells(13, 13) = msApp.Design.Hydrostatics.LWL
    Cells(14, 13) = msApp.Design.Hydrostatics.BeamWL
    Cells(15, 13) = msApp.Design.Hydrostatics.WSA
    Cells(16, 13) = msApp.Design.Hydrostatics.MaxCrossSectArea
    Cells(17, 13) = msApp.Design.Hydrostatics.WaterplaneArea
    Cells(18, 13) = msApp.Design.Hydrostatics.Cp
    Cells(19, 13) = msApp.Design.Hydrostatics.Cb
    Cells(20, 13) = msApp.Design.Hydrostatics.Cm
    Cells(21, 13) = msApp.Design.Hydrostatics.Cwp
    Cells(22, 13) = msApp.Design.Hydrostatics.LCB
    Cells(23, 13) = msApp.Design.Hydrostatics.LCF
    Cells(24, 13) = msApp.Design.Hydrostatics.KB
    Cells(25, 13) = msApp.Design.Hydrostatics.KG
    Cells(26, 13) = msApp.Design.Hydrostatics.BMt
    Cells(27, 13) = msApp.Design.Hydrostatics.BMl
    Cells(28, 13) = msApp.Design.Hydrostatics.GMt
    Cells(29, 13) = msApp.Design.Hydrostatics.GMl
    Cells(30, 13) = msApp.Design.Hydrostatics.KMt
    Cells(31, 13) = msApp.Design.Hydrostatics.KMl
    Cells(32, 13) = msApp.Design.Hydrostatics.ImmersedDepth
    Cells(33, 13) = msApp.Design.Hydrostatics.MTc
    Cells(34, 13) = msApp.Design.Hydrostatics.RM

    For i = 2 To msApp.Design.Surfaces.Count
        msApp.Design.Surfaces(i).Visible = True
    Next i


    msApp.Refresh
End Sub
```

This example shows some of the potential for refining and optimising a hull form. This process could easily be applied to other hull forms, giving the potential to perform non-parametric transformation, as well as parametric transformations.

Handling Errors

When writing any program it is a good practice to elegantly handle any errors that may arise during the execution of the code. When writing scripts in VBA the On Error command is used to specify the action to be taken when an error occurs.
In the subroutines presented above, the On Error command is used to specify that in the event of an error, execution of the script will be redirect to the label ErrorHandler. This label is at the very bottom of the subroutines and is followed by three lines of code that firstly determine if an error occurred and then report the error message to the user using the standard message box dialog. It is important to test for an error, as these lines will be executed whenever the subroutines are executed irrespective of whether an error occurred or not.

Exercise 4 – Generic Code for Hull Blending

The current example file has been designed to work with a particular design. The file would be more useful if it was adaptable to any given design. To do this would require the following steps to be taken.

- Import the control points from Maxsurf Modeler, into their relevant Excel cells.
- Manually modify the hull form in Maxsurf Modeler to a new hull shape.
- Perform a parametric transformation of the hull so that some parameters remain the same between the two hulls, such as LWL and Displacement.
- Import the second set of control points to their relevant cells in Excel

For this exercise, try creating macros to perform the above tasks so that any design can be used in the program.

Hint:

Start by manually performing the process of importing another hull form into the program, this will give a greater understanding of the processes that need to be coded

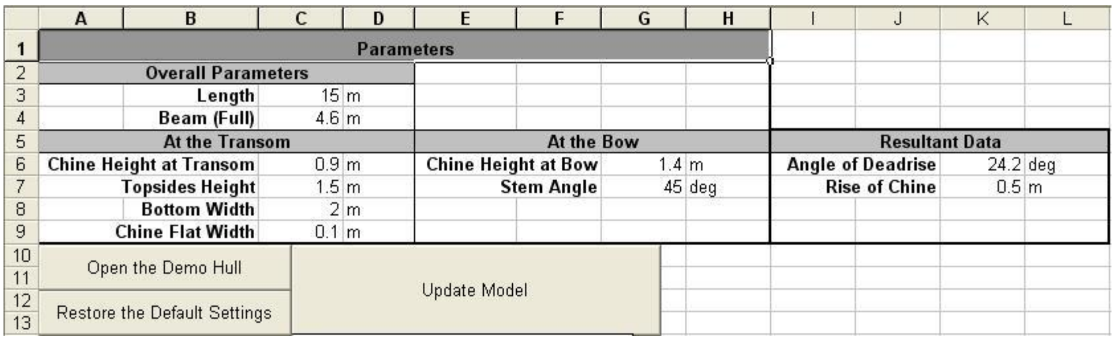
Creating a Chined Hull Vessel

Often the hardest part of creating a design is to find a suitable starting point. This example file modifies a parent hull to fit specified parameters, allowing a vessel to be quickly transformed into a variety of different parameter hulls.

The sample file uses a very simple three surface planing hull and allows the user to specify various parameters for the hull as shown in the screen shot below.

Resources

The Excel Workbook containing this file is stored in C:\Program Files\Maxsurf Modeler XX\Automation Samples\Maxsurf Modeler\Chined Hull Example.xls. The file uses the Maxsurf Modeler design "Chined Hull Example.msd" in that same folder.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Parameters											
2	Overall Parameters											
3		Length	15	m								
4		Beam (Full)	4.6	m								
5	At the Transom				At the Bow				Resultant Data			
6	Chine Height at Transom		0.9	m	Chine Height at Bow		1.4	m	Angle of Deadrise		24.2	deg
7		Topsides Height	1.5	m		Stem Angle	45	deg		Rise of Chine	0.5	m
8		Bottom Width	2	m								
9		Chine Flat Width	0.1	m								
10												
11												
12												
13												

Figure 18 Chined Hull Example Spreadsheet. The sheet contains parameters for the hull and macro buttons to execute changes.

If a designer designs a large number of vessels of a similar style, then a program like this could be developed to create a hull from the correct initial parameters. The hull form will be mostly faired already and will save a lot of the initial design time.

The procedures executed by the command buttons are separated into several small procedures. Each procedure moves a specific set of control points, such as the inboard chine control points or the deck edge, topside surface control points. As each procedure is somewhat dependant on other procedures, some procedures are run more than once.

For example, scaling the length of the vessel changes the stem angle, but changing the stem angle will change the vessel length, so the processes must be run more than once to ensure each ends up with the correct values.

The code for this procedure is quite lengthy and won't be included in full. The following code segment has been included, showing the procedure for setting the chine to the correct height and beam locations. This code moves three separate sets of control points.

- The Inboard set of Chine Surface Control Points
- The Outboard/Upper set of Bottom surface control points
- The Bottom surface control points defining the rise of keel up to the chine line at the bow.

Note that the use of a surface list to access the surface control points and the use of arrays for storing rows of control points

```
Sub SetChineHeightandBottomWidth()

Dim sList As New Maxsurf Modeler.SurfaceList
sList.Add msApp.Design.Surfaces

Dim CPx(9) As Double
Dim CPy(9) As Double
Dim CPz(9) As Double

Dim ChineHeight As Double
Dim MoveDistZ As Double
Dim BottomWidth As Double
Dim ScaleY As Double
Dim ChineAtBow As Double
Dim ScaleZ As Double


ChineHeight = Range("C6")
If ChineHeight < 0.05 Then
    ChineHeight = 0.05
    Range("C6") = ChineHeight
End If

BottomWidth = Range("C8")
If BottomWidth < 0.05 Then
    BottomWidth = 0.05
    Range("C8") = BottomWidth
End If

'Set the ChineHeight and Bottom Width
For m = 1 To 9
    sList(3).GetControlPoint 1, m, CPx(m), CPy(m), CPz(m)
Next

ScaleY = BottomWidth / CPy(1)
MoveDistZ = ChineHeight - CPz(1)

For m = 1 To 9
    'Chine Surface Inboard CPs
    sList(3).SetControlPoint 1, m, CPx(m), CPy(m) * ScaleY, CPz(m) +
MoveDistZ
    'Bottom Surface Outer/Upper CPs
    sList(2).SetControlPoint 1, m, CPx(m), CPy(m) * ScaleY, CPz(m) +
MoveDistZ
Next

'Scale the keel line curve between zero and the forward chine point
ChineAtBow = CPz(9)
For m = 1 To 9
    'Bottom keel line CPs
    sList(2).GetControlPoint 2, m, CPx(m), CPy(m), CPz(m)
Next
ScaleZ = 0.5 * ChineAtBow / CPz(9)

For m = 1 To 9
```

```
    sList(2).SetControlPoint 2, m, CPx(m), CPy(m), CPz(m) * ScaleZ
Next

msApp.Refresh

End Sub
```

None of the procedures in this example file allow input variables, such as the bottom width, of zero. If the bottom width was set to zero, all the y coordinates for the chine and outer bottom edge would sit on the vessel centre line. If we then tried to make the vessel wider by scaling the control points outwards, all the control points would remain in a straight line. By keeping the bottom width slightly above zero, we maintain the curved shape (even if it is extremely flat) in the hull and can therefore reverse any process performed, returning the bottom width out to its previous size.

To see the remained of the code in this example file, view it through the VBA editor in Excel.

Exercise 5 – Finding the Displacement and Immersion for the Chined Hull

Knowing the displacement of the vessel created would be very useful. Add another procedure to the program that will move the surfaces so that the chine at the transom is on the waterline, then calculate the displacement.

Alternatively, for a given displacement, move the vessel upwards or downwards to find the correct displacement, then report back the resulting immersed depth and the chine immersion at the transom

Hints

To move surfaces, use the move method;

```
Sub MoveSurfaces()

Dim sList As New Maxsurf_Modeler.SurfaceList
sList.Add msApp.Design.Surfaces

For i = 1 To sList.Count
    sList(i).Move 0, 0, zDist
Next

End Sub
```

To move the surfaces to a given displacement, you will need to iterate until the displacement is within a tolerance. Don't forget to account for the vessel sinking, you can do this by testing if the waterline is between the highest and lowest control points.

Using Microstation (VBA) Macros to control Maxsurf Modeler

In this section we wil use Microstation automation to open a design in Maxsurf Modeler and export the orthogonal views to a dxf file. The dxf files will then be imported into Microstation and arranged into a lines plan format. It is recommended that Chapter 6 Microstation Automation be reviewed prior to continuing.

Resources
The Microstation Macro containing this VBA code is stored in C:\Program Files\Maxsurf Modeler XX\Automation Samples\Maxsurf Modeler\ Microstation_Modeler_Create_Lines_Plan.mvba. The file uses the Maxsurf Modeler sample design "Modeler Sample_Trawler.msd".

Open microstation with a new file. Open the microstation VBA file listed in Resources above in the microstation VBA macro editor (if you are not sure on how to do this please refer to Chapter 6 Microstation Automation).

The only public Subroutine is the Main Sub:

```
Sub Main()

Maxsurf_ModelerOpenMSD
'Open the .msd file in Maxsurf Modeler
```

```
Maxsurf_ModelerExportDXFFiles
'Export the thre orthogonal views to dxf files

Microstation_CreateLinePlan
'Open the dxf files and arrange into lines plan format

End Sub
```

Ensure that the BentleyModeler object library is referenced and Maxsurf Modeler is running. The Main sub calls the three private Subs. The first two use the BentleyModeler object library to 1) load the "Modeler Sample_Trawler.msd" file then 2) export the three orthogonal views to dxf files:

```
Private Sub Maxsurf_ModelerOpenMSD()

    Dim FileName As String
    FileName  =  "C:\Program  Files\Bentley\Offshore\Maxsurf  CONNECT
Edition V21.1\Sample Designs\Modeler Sample_Trawler.msd"

'set file to be opened must have path and extension

    msApp.Design.Open FileName, False, False
'tell Maxsurf Modeler to open the file

    msApp.DisplayContour(msCTSections) = True
'turn on sections
    msApp.DisplayContour(msCTButtocklines) = True
'turn on buttocks
    msApp.DisplayContour(msCTWaterlines) = True
'turn on waterlines

End Sub

Private Sub Maxsurf_ModelerExportDXFFiles()

    Dim DXFBodyOutName As String
    DXFBodyOutName                                                    =
"C:\Users\Public\Documents\Maxsurf\Trawler_Body.dxf"
    msApp.Design.ExportOrthogonalViewDXF  DXFBodyOutName,  2  'Export
body plan

    Dim DXFBodyOutName2 As String
    DXFBodyOutName2                                                   =
"C:\Users\Public\Documents\Maxsurf\Trawler_Profile.dxf"
    msApp.Design.ExportOrthogonalViewDXF  DXFBodyOutName2,  3  'Export
Profile view

    Dim DXFBodyOutName3 As String
    DXFBodyOutName3                                                   =
"C:\Users\Public\Documents\Maxsurf\Trawler_Plan.dxf"

    msApp.Design.ExportOrthogonalViewDXF DXFBodyOutName3, 4
'Export Plan view

End Sub
```

The Microstation_CreateLinePlan sub does not use the BentleyModeler object library, rather the Bentley Microstation object library to open the 3 dxf orthogonal views and move them relative to one another to create the lines plan in the desired format:

```
Declare    PtrSafe    Function    mdlRefFile_attachCoincident    Lib
"stdmdlbltin.dll" (ByRef outModelRefP As Long, _
ByVal   FileName   As   String,   ByVal   logical   As   LongPtr,   ByVal
description As LongPtr, _
ByVal  levelDisplayFlag  As  Long,  ByVal  snapLock  As  Long,  ByVal
locateLock As Long) As Long
```

```
Private  Sub  attachDxfFile(ByVal  FileName  As  String,  ByVal  sp1  As
String, ByVal sp2 As String)
    Dim status As Long
    Dim outModelRefP As Long

    status   =   mdlRefFile_attachCoincident(outModelRefP,   FileName,
StrPtr(sp1), StrPtr(sp2), 1, 1, 1)
'used to open a dxf file
End Sub

Private Sub Microstation_CreateLinePlan()

'Dim oAttachment      As Attachment
'Dim oAttachments     As Attachments

Call
attachDxfFile("C:\Users\Public\Documents\Maxsurf\Trawler_Body.dxf",
("Trawler_Body"), ("Trawler Body DXF"))
'import the body plan dxf
ActiveDesignFile.SaveAs
"C:\Users\Public\Documents\Maxsurf\temp1.dxf",                 True,
msdDesignFileFormatDXF
'save

CadInputQueue.SendKeyin "choose all"
'select all using microstation key-in functions

Dim dataPoint As Point3d
dataPoint.X = 0
dataPoint.Y = 0
dataPoint.Z = 0

Dim data2Point As Point3d
data2Point.X = 10
data2Point.Y = 10
data2Point.Z = 0

CadInputQueue.SendKeyin "move"
CadInputQueue.SendDataPoint dataPoint
CadInputQueue.SendDataPoint data2Point
CadInputQueue.SendReset
'move the body plan to the right by 10m and vertically by 10m


Call
attachDxfFile("C:\Users\Public\Documents\Maxsurf\Trawler_Profile.dxf
", ("Trawler_Profile"), ("Trawler Profile DXF"))
'import the profile dxf
ActiveDesignFile.SaveAs
"C:\Users\Public\Documents\Maxsurf\temp2.dxf",                 True,
msdDesignFileFormatDXF
'save file

CadInputQueue.SendKeyin "choose all"
'select all

Dim data3Point As Point3d
data3Point.X = 0
data3Point.Y = 10
data3Point.Z = 0

CadInputQueue.SendKeyin "move"
CadInputQueue.SendDataPoint dataPoint
CadInputQueue.SendDataPoint data3Point
CadInputQueue.SendReset
'move the body plan and profile up by 10m
```

```
Call
attachDxfFile("C:\Users\Public\Documents\Maxsurf\Trawler_Plan.dxf",
("Trawler_Plan"), ("Trawler Plan DXF"))
 'import plan dxf

ActiveDesignFile.SaveAs     "C:\Users\Public\Documents\Maxsurf\Lines
Plan.dxf", True, msdDesignFileFormatDXF
'save file

CadInputQueue.SendKeyin "fit view extended"
'zoom to fit window

CadInputQueue.SendKeyin "choose none"
'deselect all

CommandState.StartDefaultCommand

End Sub
```

Note:

When microstation opens older version dxf files (as those generated by Maxsurf Modeler) it opens them in read only mode. To enable editing of the dxf objects the file must be saved using the SaveAs command.

Appendix A Object Model Summary

This Appendix contains a listing of all the objects and constants defined in the Maxsurf Modeler Automation interface.

Objects

The objects defined with the Maxsurf Modeler object model are summarised in the following tables. The first table lists the collection and list objects for the Marker and Surface objects. The second table shows the parent-child hierarchy for the Object model.

Object	**Collection**	**List**
Marker	Markers	MarkerList
Surface	Surfaces	SurfaceList

Object	**Parent**	**Children**
Application	-	Design
Design	Application	FrameOfReference Grids Hydrostatics Markers Surfaces
FrameOfReference	Design	-
Grids	Design	-
Hydrostatics	Design	-
Marker	Design	-
MarkerList	Design, Application	Marker
Markers	Design	Marker
Preferences	Design	-
Surface	Design	-
SurfaceList	Design, Application	Surface
Surfaces	Design	Surface

Enumerated Types

All the enumerated types defined with the Maxsurf Modeler Object Model and their values are summarised below.

msDimensionUnits

msDUMeters	1
msDUCentimeters	2
msDUMillimeters	3
msDUFeet	4
msDUInches	5
msDUFeetAndInches	6

msGridType

msGTSections	1
msGTWaterlines	2
msGTButtocklines	3

msGTDiagonals	4

msMarkerType

msMTInternal	1	msMTTopRight	6
msMTTopEdge	2	msMTTopLeft	7
msMTBottomEdge	3	msMTBottomRight	8
msMTLeftEdge	4	msMTBottomLeft	9
msMTRightEdge	5		

msSurfaceLibrary

msSLDefault	1	msSLSphere	7
msSLCylinder	2	msSLCone	8
msSLCylinder4	3	msSLLongPlane	9
msSLCylinder6	4	msSLTransPlane	10
msSLBox	5	msSLHorzPlane	11
msSLPyramid	6	msSLNACA0010	12

msSurfacePrecision

msSPLowest	1
msSPLow	2
msSPMedium	3
msSPHigh	4
msSPHighest	5

msSurfaceSkinDirection

msSSDOutside	1
msSSDCentered	2
msSSDInside	3

msSurfaceType

msSTBSpline	1
msSTNURB	2
msSTDevelopable	3
msSTConic	4

msSurfaceUse

msSUHull	1
msSUStructure	2

msWeightUnits

msWUKilograms	1
msWUPounds	2
msWUTonnes	3
msWUTons	4

Index