The Straus7 Application Programming Interface (API) allows programmers to interface their code to Straus7. This makes it possible to create a program that can access geometric and result data from Straus7 models. Data obtained can then be used by the program for display or further processing.

The Straus7 API consists of a Dynamic Link Library (DLL) file (St7API.dll) and a number of header and include files. The DLL file contains functions that can be used to: read Straus7 finite element data; modify or create Straus7 finite element data; launch the Straus7 solvers; and read Straus7 result data.

The header files allow external programs to communicate with St7API.dll. They define all the constants used and the function calling conventions for each language supported (all functions in the Straus7 API use the Windows calling convention "stdcall"). A different set of header files is needed for each language (e.g. Delphi, C++, Fortran, etc). Note that in some cases, header files are even compiler product dependent - e.g. the header files for Visual Fortran will be different to the header files for Lahey Fortran. Release 2.4.6 comes with header files for Delphi, C/C++, Compaq/Intel Visual Fortran, Lahey Fortran, Microsoft Visual Basic (including VBA), Microsoft Visual C# and Matlab. New header files are being added to meet user requirements – please contact us if you need header files for a different language.

The majority of this documentation is devoted to describing each of the functions in the Straus7 API. The C syntax for the available functions is given, along with the input and output parameters and example code.

The remainder of the documentation lists error codes and conventions and types for property information, attributes and results.

For compiler specific information, see the *Using the Straus7 API* section.

The Straus7 API file `St7API.dll` must be located in a directory where it can be found by the calling program. This means that `St7API.dll` must be in a directory that is within the Windows search path. Alternatively, it is possible to specify where the DLL is located via the Windows API function `LOADLIBRARY`. See the Win32 API for more information about this.

To call the functions in the API, an interface file that declares the exported function calls in `St7API.dll` is needed. This file is provided in the Straus7 API Toolkit and its name is dependent on the compiler:

`St7APICall.pas`	for Delphi
`St7APICall.h`	for C/C++ and Matlab
`St7APICall.vb`	for Microsoft Visual Basic
`St7APICall.bas`	for Microsoft Visual Basic 6 and VBA
`St7API.cs`	for Microsoft Visual C#
`St7APICall.f90`	for Fortran

As most of the API functions employ pre-defined constants, these are conveniently defined within an external file in the Straus7 API Toolkit. It is not essential that you use this file, especially if you prefer to declare your arrays as 1-based instead of the 0-based approach used. The name of the constants file is dependent on the compiler:

`St7APIConst.pas`	for Delphi
`St7APIConst.h`	for C/C++
`St7APIConst.vb`	for Microsoft Visual Basic

Linking to the API with Visual Basic

There are two source files included in the API Toolkit – these are `St7APICall.vb` and `St7APIConst.vb` as described above. To use these files add them to your project.

API Strings and Visual Basic

The Straus7 API uses null-terminated strings. These are always declared as `ByVal StringName As String`. To pass a string to the API, declare it as `Dim StringName As String` and assign it a value, Visual Basic will ensure that the string is null-terminated when you pass it as an argument. When you need to get a string value back from the API, the string must be pre-allocated and this is no longer possible in Visual Basic without assigning it a value. It is therefore necessary to assign the string a value with a length longer than the specified string length prior to passing to a function that writes to it. When the string is returned it is also necessary to discard all characters from the first `CHAR=0` to the end of the string.

API Arrays and Visual Basic

Many Straus7 API functions use arrays of longint or double as parameters. These are always passed by reference and declared as `ByRef LongArray As Long` or `ByRef DoubleArray As Double`. The array passing syntax `LongArray() As Long` or `DoubleArray() As Double` should not be used with the Straus7 API. The arrays to be passed should be declared as `Dim LongArray(n) As Long` or `Dim DoubleArray(n) As Double`, where `n` is some integer value. When passing these arrays to a Straus7 API function via Visual Basic, it is essential that the first index of the array be passed. The following example further illustrates the correct procedure:

```
function declaration:
Declare Function St7GetNodeXYZ& Lib "St7API.DLL"(ByVal uID
As Long, ByVal NodeNum As Long, ByRef XYZ As Double)

variable declaration:
Dim XYZ(2) As Double

function call:
ErrorCode = St7GetNodeXYZ(1, NodeNumber, XYZ(0))
```

API Boolean and Visual Basic

Many Straus7 API functions use `boolean` or arrays of `boolean` as parameters. These should always be passed as `Byte` in Visual Basic, (both by value and by reference). This is necessary because the Straus7 API uses single byte boolean representation, which is compatible with the Visual Basic `Byte` type. The Visual Basic `Boolean` type is two bytes long, therefore not compatible. True boolean values will therefore be represented by `Byte=1` and False boolean values will be represented by `Byte=0`.